Université de Tunis
Institut Supérieur de Gestion

A FRAMEWORK FOR ADAPTIVE RISK-DRIVEN INTRUSION DETECTION AND RESPONSE SYSTEMS

THESE

En vue de l'obtention du Doctorat

En Informatique de Gestion

Présentée et soutenue publiquement par:

Chaker KATAR

Le 18 Octobre 2014

Membres du Jury:

♦*Mme Rim FAIZ*	*Professeur*	*Université de Carthage*	*:Président*
♦*M. Mhamed Ali ELAROUI*	*Professeur*	*Université de Carthage*	*:Directeur de thèse*
♦*M. Jalel AKAICHI*	*Professeur*	*Université de Tunis*	*:Rapporteur*
♦*M. Lamjed BEN SAID*	*Professeur*	*Université de Tunis*	*:Rapporteur*
♦*M. Talel LATHARI*	*Professeur*	*Université de Tunis*	*:Membre*
♦*M. Belgacem RAGGAD*	*Professeur*	*Pace University NY*	*:Membre Expert*

Année Universitaire 2014-2015

Abstract:

Organizations providing services to their customers through a public infrastructure such as the Internet have excessive security requirements. Existing intrusion detection and response systems partially meet essential requirements of appropriate detection and effective countering of increased attacks. In this thesis, we propose an adaptive cost-effective intrusion detection and response (idrs) framework. The selective combination of detection models ensures adaptiveness of the analysis component in this framework. Within the improved process of this component, the selection step dynamically determines appropriate detection model combinations using an integrated criterion (performance and data dependent factors). The combination step also uses an improved evidential fusion method to aggregate participating detection models at the decision level. Furthermore, the designed cost-effective response component of our idrs framework relies on compliant risk management model to information security standards. The proposed risk model integrates two main and interdependent parts, namely assessment and treatment parts. The former quantitatively assesses inflicted damages by detected threats; while the latter determines cost-effective defense strategy against these, using optimization techniques. Additionally, our idrs framework proposes both structural and functional enhancements for future idrs systems through respectively its improved CIDF inspired architecture and the idrs life cycle. Moreover, it is expandable and can be easily integrated into any computing environment or any risk-driven information security management system. Detection results of our prototyped analysis function are remarkably better than those of the KDD winner; the best one exceeds 93% of detection. Initial results generated by risk driven response component also show great decision support to information security management.

Keywords:

Intrusion detection and response, Adaptive analysis, Risk driven response, dynamic selection, evidential fusion, risk assessment, risk treatment.

DEDICATION

I dedicate this thesis to the memory of my dear father ***Mohamed Chtioui*** *and my lovely sister* ***Rim****, who are always a great source of inspiration for me,*

ACKNOWLEDGMENTS

This doctoral thesis cannot be successfully completed without the support of many people. I would like first to gratefully and sincerely thank my advisors Pr. Mhamed Ali Elaroui and Pr. Belgacem Raggad for their precious counsels, understandings and patience. I would like to express my gratitude to Pr Jalel Akaichi and Pr. Lamjed Bensaid for reviewing and Pr. Rim Faiz and Pr. Talel Lathari for evaluating this thesis.

I would also like to thank Pr. Zied Elouedi and Pr. Abderrazek Jemai for their constructive comments and suggestions on improving this work. My special thanks to Karim Aroua, Tarak Selmi, Haythem Elmir, Nabil Hasni and the Tunisian NSA technical staff who advised me specifically in conducting different experimental tests. I would like to thank Nizar Masour and Bechir Ouederni for their valuable remarks about multiple sections in this thesis.

I am also grateful to all my professors and colleagues and the staff at the ISG. I am heartily thankful to my parents for their constant support and encouragement. Very special thanks go to my wife Wafa and my adorable angels Omar and Ahmed Khalili for their immense love, patience, and for making my life filled with joy and happiness.

ABSTRACT

Organizations providing services to their customers through a public infrastructure such as the Internet have excessive security requirements. Existing intrusion detection and response systems partially meet essential requirements of appropriate detection and effective countering of increased attacks. In this thesis, we propose an adaptive cost-effective intrusion detection and response (idrs) framework. The selective combination of detection models ensures adaptiveness of the analysis component in this framework. Within the improved process of this component, the selection step dynamically determines appropriate detection model combinations using an integrated criterion (performance and data dependent factors). The combination step also uses an improved evidential fusion method to aggregate participating detection models at the decision level. Furthermore, the designed cost-effective response component of our idrs framework relies on compliant risk management model to information security standards. The proposed risk model integrates two main and interdependent parts, namely assessment and treatment parts. The former quantitatively assesses inflicted damages by detected threats; while the latter determines cost-effective defense strategy against these, using optimization techniques. Additionally, our idrs framework proposes both structural and functional enhancements for future idrs systems through respectively its improved CIDF inspired architecture and the idrs life cycle. Moreover, it is expandable and can be easily integrated into any computing environment or any risk-driven information security management system. Detection results of our prototyped analysis function are remarkably better than those of the KDD winner; the best one exceeds 93% of detection. Initial results generated by risk driven response component also show great decision support to information security management.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

ACC	Accuracy
ALE	Annual Loss Expectancy
AMS	Assets Management System
ANN	Artificial Neural Network(s)
API	Applicative Programming Interface
BDM	Base Detection Model(s)
BER	Balanced Error Rate
CERT	Computer Emergency Response Team
CIDF	Common Intrusion Detection Framework
CISA	Certified Information Security Auditor
CISSP	Certified Information System Security Professional
CSI	Computer Security Institute
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
DARPA	Defense Advanced Research Project Agency
DoD	Department of Defense
DOS	Denial Of Service attack
DT	Decision Trees
EEC	Exponential Error Count
FIPS	Federal Information Processing Standards
FIRST	Forum for Incident Response and Security Team
IDMEF	Intrusion Detection Message Exchange Format
idrs	Intrusion Detection and Response System(s)
ids	Intrusion Detection System(s)
IDWG	Intrusion Detection Work Group
IETF	Internet Engineering Task Force
ISMS	Information Security Management System
JDK	Java Development Kit
JGA	Java Genetic Algorithm
KDD	Knowledge Discovery and Data Mining
NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database
OSVDB	Open Source Vulnerability Database
R2L	Remote to Local attack
ROSI	Return On Security Investment
SRR	Success Rate Ratio
SSO	Site Security Officer
SVM	Support Vector Machines
U2R	User to Root or privilege elevation attack
WECC	Weighted Error and Correct Count

LIST OF TABLES

LIST OF FIGURES

GENERAL INTRODUCTION

1 Overview

With the rapid growth of the Internet, multiple new business opportunities, such as service providers, and carriers have emerged. These modern businesses have entirely founded their activities on the public infrastructure. They take advantage of public IP networks to reduce their infrastructure-related cost. Moreover, they can reach a large number of consumers across the world and serve them easier, faster and more efficiently. However, networks and applications which are business critical and essential part of their strategies are becoming increasingly exposed to various security threats.

Security threats have the potential to cause harm to organization's infrastructure and information assets. Threats may be initiated by multiple internal and external sources. They may be caused by employees' actions that unwillingly targeted organization's networks or applications. Different environmental or natural events can also cause various losses to an organization. However, the most harmful threats to the organization's assets are those initiated by malicious entities, which benefit from universal connectivity offered by current public infrastructure identically to modern businesses. Deliberate actions carried by these entities have multiple implications on business critical components and information assets of an organization.

The latest class of threats targeting information assets includes most prominent security breaches. Intrusive actions or attacks of this class are mounted by malevolent entities that share same benefits of public infrastructures with modern businesses. Intruders or malicious entities have the ability to carry out multiple attacks on selected target relying on the universal connectivity and the open environment of the Internet. To achieve their objectives, they use several attacking methods and tools widely available and readily reachable on the Internet. Moreover, they exploit various inherited vulnerabilities or weaknesses in business critical applications and public IP based networks to perform their malicious actions.

Attacks on organization assets have many sources. Organized criminals, individual spies, disgruntled employees and hackers are the most untrustworthy entities who are able to mount multiple attacks on business infrastructure. These groups have several motivations. They are

able to cause various damages to target assets. Organized crime attacks seek business critical information to ensure financial gain for intruder. The most common organized attacks are fraud and theft which are perpetuated in majority by legitimate and authorized system users. Industrial espionage aims at collecting proprietary data of an organization for the benefit of another. Industrial spies seek to improve competitive advantages of their organization relying on stolen manufacturing or product development information. Disgruntled employees are the most familiar with organization's applications. Multiple events trigger destructive behaviour of these entities such organization downsizing which leads them to create mischief and sabotage the organization infrastructure. Malicious hackers are insider or outsider entities to the organization. They use different means to gain unauthorized access to organization networks and compromise integrity and confidentiality of business critical information.

Intruders use numerous means to reach their objectives. Scan or probe is the most used attack category for exploration and information collection on the target computer or network. Sniffing is another passive threat that allows intruder to gain unauthorized access to the target using information contained in sniffed traffic. Spoofing and masquerade are mounted by intruder to break into the target by pretending to have the identity either of legitimate computers or users. Hackers can actively affect victim resources. They can use more prevalent and harmful active attack such as denial of service and malicious code. Denial of service attacks flood business critical resources by an overwhelming traffic and make them unavailable. Malicious code attacks have the potential to interrupt services, destroy data and use resources of the target. Simple implementation of these attacks and rapid propagation of their behaviors make them an appropriate support and convenient tool to realize hacker objectives.

Intruder attacks exploit multiple inherited and supported vulnerabilities in the target systems. These security faults lead to various exploits and inflict variable losses to the organization. The consequences or impacts of exploiting these weaknesses range from simple interruption to complete damage of the target resources. They can be simple to quantify such as business operations interruption or identity theft but difficult to estimate their associated losses. Multiple institutions, organizations and government agencies such as CSI/FBI (Computer Security Institute/Federal Bureau Investigation) and Mi2g have estimated, based on a firm sample, financial losses of computer and network attacks.

The tenth CSI/FBI computer crime and security survey, in 2005, is among the most referenced and detailed security reports. It was conducted on 699 security practitioners from

multiple organizations in different activity sectors. The total loss of these organizations due to attacks on their information technology resources was estimated to 131 millions dollars. Mobile code attacks remain the main source of the greatest loss with 32% of the overall reported losses. Attacks on proprietary information and unauthorized access have comparable effects on surveyed organizations. The inflicted damage by each of these attacks was estimated by nearly 24% of the overall loss. In this survey, websites attacks have also shown an exponential increase. In fact, 95% of surveyed organizations have reported more than 10 incidents on their websites. However, 5% only of organizations have experienced more than 10 websites incidents, as stated in 2004 survey [141], [142]. In a recent survey, CSI institute has reported that mobile code infection remains the most common threat. Furthermore, web attacks, including phishing, and different variants of denial of service attacks were respectively ranked in this report among most experienced threats by surveyed organizations. Additionally, the cost of cyber crime report of Ponemon institute, conducted on about 2000 companies in different countries, states that the number of attacks was increased by 42% from 70 to 204 attacks. Moreover, 50% of mounted attacks per week are successful. In this report also, variable estimates of cyber crime costs were presented depending on countries. It ranges from 53.3 million dollars, in the United Kingdom, to 58.9 million dollars, in the United States [304], [324][1].

The worldwide financial losses of digital attacks were estimated to more than 225 billion dollars in 2004 by the British company Mi2g. In the same year, attacks by malicious code have caused financial losses that exceeded 17 billion dollars as stated by Computer Economics. However, financial damage of the same attack type was evaluated to more than 14 billion dollars in 2005 [83]. Moreover, according to [68], 9.3 million persons across the world were victim of identity theft in 2005. The global financial loss of this attack type was more than 52 billion dollars.

The success of these attacks depends on which flaws are exploited. The intruder attacking process starts each time by locating the most appropriate holes in the target system to infiltrate within. In fact, various vulnerabilities are supported in computer or networks software or hardware. Design or specification flaws are readily exploitable by the intruder because they persist even if hardware or software is perfectly implemented. They represent the most

[1] 2010/2011 CSI survey and Ponemon Cost of cyber crime study of 2012 are respectively available at : www.GoCSI.com and www.ponemon.org/data-security

common issues of multiple attacks. Coding and configuration errors instead induce other types of vulnerabilities respectively at development and use stages of software or hardware. With the new business environment and expansion use of web applications, CERT (Computer Emergency Response Team/ Coordination Center of Carnegie Mellon University, Pittsburgh) has reported more than six million exploitable vulnerabilities in 2005. This number was nearly the double of reported and documented vulnerabilities of 2004 [70], [386].

In addition to design, coding and configuration vulnerabilities, there are other serious flaws that help intruders to achieve their objectives. Actually, lack of appropriate security controls, failure to implement good security practices and complete reliance on primitive security measures of operating systems are most common origins of reported dramatic financial losses of organizations connected to public networks. Moreover, low information security budget (13% of information technology budget according to PriceWaterHouseCoopers survey [386]), increased sophistication of attack tools and wide availability of information to hackers may lead to unauthorized access, fraud, identity theft and more devastating attacks.

To thwart emergent threats, modern business firms have to implement the most appropriate security controls to insure confidentiality of customer information, preserve their market place and guarantee continuity of their business operations. The security solutions required for these should address the challenges and opportunities of universal connectivity insured by today public information technology infrastructure. Moreover, it should be double-folded to prevent and detect attacks on organization assets. The integrated security solution for today's organizations should combine multiple complementary security mechanisms. It should use security principles such as defense-in-depth and allow numerous services at each level within information technology infrastructure. An integrated security solution includes also various prevention mechanisms against attacks such as authentication, encryption, access control and filtering. However, these security technologies are not sufficient to rule out increased and highly sophisticated today's attacks. Thus, integrated security solutions rely on attack detection mechanisms to reinforce security of organizations and allow them a holistic security system to protect against variety of threats and reduce their risks.

2 Motivations

Intrusion detection is a critical dimension of any integrated security solution. Intrusion detection systems (ids) can prevent, detect and counteract mounted attacks. Available ids systems suffer from various weaknesses at different levels including design, implementation

and deployment. Considerable efforts of research groups in this field focus on ids design in order to ensure an appropriate structure on which rely other activities or system development life cycle. These efforts were rewarded by an appealing organization of ids components in the case of CIDF working group, as discussed in chapter 1. Such structure is known as the Common Intrusion Detection Framework (CIDF). The latter was based on a functional criterion to identify different idrs (intrusion detection and response system) components, namely event collection, analysis, response and database components. It has also proposed the common architecture of an idrs relying on these components and links between them to resolve overlapping functions in existing ids.

Recently, with increasingly complicated detection environments and sophisticated attacks, idrs systems, even those CIDF based, require core improvements to meet new requirements. Existing detection environments impose structural enhancements to ids. Besides, detection and response against mounted attacks on these environments need multiple revisions of the corresponding components and their adopted methods.

CIDF based ids systems lack model generation component. Such component is required to satisfy requests of A-boxes, logs analysis component in CIDF framework, and Site Security Officer (SSO) that concern detection models of an ids. It solely focuses on construction, evaluation and validation of detection models to be included in log data analysis tasks. Additionally, when new datasets on expected behavior of the monitored system or intrusive actions of attackers become available, this component is useful also to update built detection models.

Additionally, CIDF based ids systems use static knowledge. Indeed, knowledge about attacks, vulnerabilities and others entities are directly encoded within ids processes. Ids knowledge in this context are neither revised by considering recently available information nor stored to be shared with other ids components or security systems. Therefore, knowledge base component is required to save and maintain ids knowledge up-to-date. Moreover, it ensures knowledge availability to idrs components and other security systems.

Ids components, including event collection, analysis, response and database, were investigated in previous researches. However, log analysis component has gained considerable attention. In fact, the analysis and detection problem of ids systems was studied for years in intrusion detection field. Initial solutions to this problem were based on the single detection model approach. The latter was deemed insufficient to detect anomalous actions of intruders or identify normal activities of monitored system or both. Additionally, detection mechanisms based on this approach are ineffective specifically when dealing with various

attack types due to confusing nature of intrusive event sequences, as stated in [201]. Few recent works have also discussed the multimodel approach for log analysis [102], [135], [139] [204], [222], [254], [288], [330]. This approach has given rise to other difficulties closely related to detection models, their construction, selection and combination. Integrated analysis mechanisms based on this approach require model generation component, as discussed before. These mechanisms often rely on static subsets of detection models. Additionally, they include detection models built using the same technique [102], [135], [254]. Therefore, they inherit same drawbacks of single model based components. Indeed, for both cases, a single facet of normal or intrusive events catches sight of the considered learning or datamining technique, while other aspects (sequence, association, nonlinearity …) are left uncovered or discarded by generated detection models. This increases the sensitivity of generated detection models to slight changes in normal or intrusive behaviors, hence affects performances of ids systems. To overcome these shortcomings, multiple heterogeneous techniques may be involved in generating detection models [139], [204], [222], [288], [330]. The resultant integrated components are capable to capture various aspects that concern normal and intrusive events. Additionally, they are suitable to boost ids accuracy and precision.

In existing detection environments, collected log data are in flux and consequently reported events. These environments allow also various log types that differently trace intrusive and normal events. Log types are identified relying on their sources as well specificities of logging facilities. They may be issued from several sources including network interfaces, hosts, routers, and firewalls. They may also include variable subsets of features depending on specificities of sensors, as discussed in [221]. Unfortunately, the majority of existing integrated components take account a single log data type, most often network or host logs.

Although, they use static combinations of detection models, even heterogeneous, they are ineffective to existing environments. Thus, a dynamic and adaptive component seems well adapted to these. It focuses on several log types that differently report security state of the monitored system. Furthermore, it dynamically selects best combinations of detection models to analyze the current state. Such approach to design adaptive analysis components is useful to decrease confusion and chance of eluding malicious events. Additional improvements are also required by this to meet objectives of high security systems and ensure accurate and precise decisions to other ids components including the response component.

Despite its importance for idrs systems, response components have attracted few research works in this field. The majority of previously designed ids are passive and lack response component. Existing active ids allow various responses against detected attempts or attacks.

Their responses range from simple alert to complex reactions. Notification components merely alert the SSO about detected attacks. They may also provide lists of applicable security controls regarding detected attacks. These response components ensure increased flexibility to the SSO in order to choose and implement appropriate controls, whenever required. Additional improvements have been conducted on notification components by integrating expert systems. These systems save encoded expert knowledge and possible control lists. They provide the SSO with appropriate control combinations depending on detected attacks. Different proposals have been also presented in order to improve reactions of these components. They have devised sophisticated response components, R-boxes, capable to automatically select and implement security controls [290], [306].

Response components discussed above are based on encoded expert knowledge. They form different groups relying on consistency of represented knowledge and their automation levels. However, these components fail to appropriately meet requirements of existing detection environments for two main reasons. On one hand, they partially use information forwarded by analysis components. They solely focus on output labels, even though analysis component decisions include additional details such as confidence levels or likelihoods of detected malicious events. On the other hand, their reactions are similarly implemented to counteract attacks mounted on critical servers or personal printer regardless of a realistic damage appraisal or appropriate ranking of computing environment assets.

Recent enhancements of expert system based responses have devised highly intelligent components. These components are cost based. They use several cost factors including response, operation and penalty costs, to design appropriate responses with respect to the mounted attacks and target system. Additionally, they involve determined costs in assessing appropriateness degrees of designed responses. Moreover, the decisions to select and implement most appropriate responses in these components depend exclusively on trade-offs. Damage-response, disruption-effectiveness or cost-benefit trade-offs are commonly adopted by cost sensitive response components [225], [369], [388]. Although, these components dynamically generate appropriate responses against detected attacks, they use constant costs such as penalty and damage costs. Furthermore, they require additional improvements that concern decision criteria and damage cost assessment. Therefore, risk based response may be a promising approach to overcome existing component failure. Such approach is useful to conduct thorough and realistic assessment of damages inflicted by detected attacks. It allows also the integration of environment dependent parameters that concern assets, vulnerabilities and security controls, to the risk model and hence the designed response component

3 Problem statement

Existing ids are facing different problems that are crucial to their future success. Ids problems including previously discussed lead to several structural and functional shortcoming in designing these security systems. Within an integrated security solution, ids systems are supposed to interact with included systems but their structure is not designed so. Existing ids, even those based on CIDF framework, lack required component to use or share knowledge with other security systems such as vulnerability assessment or security management systems. Moreover, as architectures of ids systems support their functionalities, preparation tasks are integrally neglected by ids designers. Such tasks concern detection model generation and updating. They express new requirements of ids systems, thus they are extremely useful specifically for those using multiple detection models. Although, detection model generation is critical to ids process, similarly to data preprocessing and analysis, a dedicated component to this task was commonly unforeseen in the majority of existing ids architectures, including CIDF.

Additionally, the multimodel approach has recently gained an increased insight in designing analysis and detection components for ids systems. The proposed analysis components using such approach rely on subsets of detection models to cope with underlying failure of those single model based. However, they, in turn, support many shortcomings that concern two main problems. On one hand, they exclusively focus on single log data, even though existing detection environments allow different useful log types. On the other hand, these components use static subsets of detection models, while processed event sequences are continually changing. Additionally, they consider neither conflicts nor uncertain decisions of detection models.

Recently, the idea of intelligent response has attracted the attention of ids designers. More research efforts have been devoted to improve this critical component to idrs systems. These works have devised cost based response components. The latter rely on trade-offs between involved cost factors to design and select appropriate reactions. However, many of cost factors included by these components are assumed constant which is inconsistent and ineffective to existing detection environments. Additionally, intelligent response components require explicit and detailed processes to assess different costs, design responses and select appropriate ones.

To overcome these drawbacks, associated to ids design and deployment, it is essential to enhance existing ids architectures by additional components to meet new requirements of

future ids as well as their deployment environments. Moreover, analysis and detection components of ids require several improvements to dynamically and adaptively process collected log data and assess current security state of the monitored system. Furthermore, response components involve more advanced enhancements to design risk driven reactions capable to counteract effects of detected attacks regarding target assets. Thus, in this thesis, we address the issues of adaptive analysis and risk driven response in idrs systems.

4 Main contributions

This research thesis contributes to intrusion detection and security management fields at different level. Its main contributions that fall under three essential areas are summarized by:

- Expandable CIDF inspired idrs architecture: the proposed idrs architecture extends that of CIDF framework by two additional components. The generation component puts emphasis on the necessity of preparation tasks, including detection models construction and updating, for future idrs systems. The knowledge base component instead provides idrs components with required knowledge about generated detection models, security controls, supported vulnerabilities and assets of the monitored system. Besides, it reinforces cooperation and knowledge sharing between idrs and other security systems. Moreover, the proposed idrs life cycle explicitly presents processing steps of different components within the idrs architecture. The designed architecture and life cycle serve as the foundation of our idrs framework that complements research and normalization efforts in this field.
- Adaptive analysis and detection component: the designed analysis component takes account of several log types that differently report current security state of the monitored system. It dynamically selects best combinations of detection models to conduct a thorough analysis of current state. Afterward, it hierarchically fuses selected models to derive the combined detection decision. The selection and fusion steps of the adaptive component represent the core extensions of the typical analysis process. The former uses an integrated criterion that includes data and model dependent factors, to select appropriate combinations. The latter relies on an enhanced evidential fusion method to combine selected detection models at the decision level.
- Risk driven response component: the designed response component is solely based on a risk management model. The proposed risk model complies with security standards and guidelines including ISO 27005, FIPS 65 and NIST SP800-30. Additionally, its two parts of risk assessment and risk mitigation are based on a quantitative approach.

The assessment part identifies and determines risk parameters including exposure, severity of supported flaws and effectiveness of deployed controls. Then, it estimates the basic risk of the computing environment due to mounted attacks. Afterward, the mitigation part treats reached risk relying on an appropriate minimization program. It selects the best security strategy to mitigate current risk from those incrementally and iteratively designed using return on security investment criterion. The designed and selected security strategies meet both objectives of minimizing inflected risk by mounted attacks to an acceptable level and reducing security investment cost to an imposed level

5 Thesis outline

The thesis is structured as follows. Chapter 1 and Chapter 2 review information security literature. Chapter 1 presents main security concepts, principles and controls. Chapter 2 solely focuses on ids, their components, taxonomies, normalization efforts and data modeling methods experimented in previous works. Chapter 3 introduces our idrs framework and briefly discusses proposed CIDF improvements. Chapter 4 details multimodel analysis component proposed within the idrs framework. Chapter 5 presents our information security risk management model on which relies the response component of the idrs framework. Chapter 6 presents a detailed illustrative example that stresses validity and effectiveness of our idrs framework. Finally, the conclusion enumerates main findings of this work and discusses future directions of researches in this field and others within the information security domain.

CHAPTER 1

BACKGROUND OF INFORMATION SYSTEM SECURITY AND INTRUSION DETECTION

1.1 Introduction

Internet services have submerged all activity domains of today's organizations. Moreover, multiple organizations have entirely based their business activities on these infrastructures which expose their information systems to a high risk level. The risks are related to various weaknesses in designing or bugs in implementing or configuring these services. Organizations should identify and mitigate these risks to protect their critical information systems and thus preserve their business continuity. Moreover, they should implement appropriate security controls or measures both to detect known attacks on their resources and avoid others.

1.2 Terminology

- Asset: Assets "generally include information, hardware, software, and people. Asset values are determined based on the impact to the organization if the asset is lost. Critical assets are those that are essential to meeting an organization's mission and business objectives" [20].

- Security policy :"is some statement about what kind of events are allowed or not allowed in the system. An explicit policy consists of rules that are documented (but not necessarily correctly enforced), while an implicit policy encompasses the undocumented and assumed rules which exist for many systems" [233].

- Vulnerability:"A flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy" [20].

- Threat:"is defined as anything that may compromise an asset. This could be a person, such as an employee or a hacker, or it could be a competitor or anyone else with deliberate intent to compromise an asset. Threats also include anything that results in accidental disruption to an

asset (such as a natural disaster), the means of access to do so, or any outcome or consequence that results in an unwanted effect such as disclosure, modification, destruction, loss, or interruption" [20].

- Attack: "connotes an action conducted by an adversary, the attacker, on a potential victim. From the perspective of the administrator responsible for maintaining a system, an attack is a set of one or more events that has one or more security consequences. From the perspective of a neutral observer, the attack can either be successful-an intrusion-or unsuccessful-an attempted or failed intrusion. From the perspective of an intruder, an attack is a mechanism to fulfill an objective" [20].

- Intrusion: "refers to an actual illegal or undesired entry into an information system. Intrusion includes the act of violating the security policy or legal protections that pertain to an information system" [20]. An intrusion defined in [233] as "is a successful event from the attacker's point of view and consists of:

1) An attack in which vulnerability is exploited, resulting in

2) A breach which is a violation of the explicit or implicit security policy of the system" [233].

- Incident:"is a collection of data representing one or more related attacks. Attacks may be related by attacker, type of attack, objectives, sites, or timing" [20].

1.3 Information security services

Security consists of processes, mechanisms and controls designed to protect private information and critical resources of an organization. Information security main mission is to fulfill the following three goals:

- Confidentiality: limits the use and disclosure of information to legitimate and authorized entities whose job can't be performed without private information.
- Integrity: preserves accuracy and reliability of information. Any modification of information should be performed by authorized entities.
- Availability: ensures accessibility to resources when needed. Resources should be available to legitimate and authorized users.

Additional services can be included such as authenticity, non-repudiation and access control to reinforce organization's security. Such services are required for protecting different exploitation environments (computing systems, E-commerce, etc.).

1.4 Attack taxonomies

Systems, networks and sensitive information of an organization are exposed to multiple, natural, accidental and intentional threats. Flooding, quakes or other natural threats have various implications on availability of organization resources. Accidental threats are generally caused by employees or insider entities to the organization. Data entry clerks, system operators and programmers make unintentional errors which, directly or indirectly, induce various security problems. Intentional threats are based on multiple accidental errors or vulnerabilities of the target system. They are mounted by malevolent insider or outsider entities. These malicious entities have various objectives and use different means or attacks to achieve them.

Computer and network attacks are related sequences of actions mounted by malicious entities with clear goals. The initial goal of an intruder, who mounts such intrusive actions, is the violation of the security policy of the victim. These malicious adversaries have multiple objectives that vary from simple information on specific resources to complete destruction of these. They can reach fixed goals relying on different attack models or scenarios. Attack models focus on supported design, implementation or configuration flaws of the hardware or software of the victim environment. They depend as well on skills of attackers.

Attacks targeting computing environment resources have been structured into different classes. The proposed attack taxonomies have been based on several criteria including exploited vulnerabilities and attackers' objectives. Furthermore, these taxonomies have to meet multiple requirements such as exhaustiveness and simplicity. Following principles are most relevant to discriminate between different taxonomies [164], [322]:

- Mutually exclusive: identified classes should not overlap
- Exhaustive: determined classes should cover all possible cases.
- Unambiguous: clear and precise classes regardless of who is classifying.
- Repeatable: reproduces expected results in the same classification regardless of who is classifying

- Useful: should be useful to the security industry and security solution development.
- Simple: simple to understand either by security experts or others
- Determinism: classification procedure should be clearly defined and provide results in a finite time

These criteria and others [158] impose clear definitions of used terms and well structured decision processes of the designed taxonomies.

Various attack taxonomies have been proposed in [11], [158], [164], [195], [232], [322]. Many of these taxonomies partially meet these principles. They mainly focus on vulnerabilities, attacker objectives and defense mechanisms in determining different attack classes. Vulnerabilities supported by the target victim environment were considered in Landwehr et al.' work as security flaws in computer programs [211]. The proposed taxonomy by Landwehr and his colleagues aims at identifying problematic aspects in system designing process. Thus, it seems to be designed to assist the system designer and programmer to develop more secure systems [195]. In this taxonomy, security flaws are categorized according to three dimensions as stated by Lindqvist et al.[232]: genesis (how the fault was introduced?), time of introduction (when the fault was introduced?) and location (where the fault manifest?) [18], [232].These dimensions have been extended in Krsul and Bishop works [49], [199] by other features that characterize vulnerabilities in order to classify security flaws associated with different abstraction levels.

Attack-centric taxonomies are the most used in security literatures. They concentrate on intruder or the entity who mounts the attack. Implicitly, they thought of exploited security flaws in conjunction with intruder's methods in defining attack categories. Multiple attacks categories have been defined by these taxonomies based upon attack techniques, results, sources and processes.

Defense-centric taxonomies focus on defender goals rather than attacker objectives. They address to the other side of security problem and try to categorize attacks based on means to defend against them. These taxonomies focus on required security controls for a defender side. Moreover, they offer useful tools that help defender to predict performance of implemented detectors.

1.4.1 Vulnerability classifications

Based on three dimensions proposed by Landwehr et al., Aslam's taxonomy of faults defines three main classes of operation, environmental and coding faults. Coding faults are introduced during the software development. Operational faults are associated to improper software installation or misconfiguration. Environmental faults manifest when software is deployed in an environment which is inappropriate. Operation and coding faults, in turn, are subdivided into other classes as depicted by the figure 1.1. Detailed descriptions of the identified fault classes can be found in [27], [211].

1) Operation faults (Configuration errors)
    1a) Object installed with incorrect permissions
    1b) Utility installed in the wrong place
    1c) Utility installed in incorrect setup parameters
2) Environmental faults
3) Coding faults
    3a) Condition validation error
        3a1) Failure to handle exceptions
        3a2) Input validation error
            3a2a) Field value correlation error
            3a2b) Syntax error
            3a2c) Type and number of input fields
            3a2d) Missing input
            3a2e) Extraneous input
        3a3) Origin validation error
        3a4) Access rights validation error
        3a5) Boundary condition error
    3b) Synchronization error
        3b1) Improper or inadequate serialization error
        3b2) Race condition error

Figure 1.1 : Aslam's Taxonomy

Bishop and Krsul have analyzed Aslam's taxonomy [49], [199]. Bishop has illustrated by a sample of UNIX operating system vulnerabilities that Aslam taxonomy supports overlapping classes and thus taken flaws can be assigned to multiple classes at the same time. Moreover, he states that this taxonomy does not meet neither well defined decision process nor uniqueness. Bishop shows also that Aslam's taxonomy focuses solely on vulnerabilities at the implementation level and therefore, lacks high-level categories to classify design faults [49]. He has developed a vulnerability taxonomy that focuses on the underlying causes rather than descriptions of flaws. The proposed taxonomy uses six main axes or dimensions for classifying vulnerabilities:

- Nature of vulnerability or the condition that causes it (validation, synchronization, naming, implementation)
- Time of introduction
- Exploitation domain: gains through the exploitation of vulnerability (access level, …):
- Effect domain: the effect of vulnerability
- Minimum number: minimum number of components required for exploiting the vulnerability.
- Source: the source of identification of vulnerability.

Krsul has constructed his own attack taxonomy based on security fault of Aslam and critical points of Bishop. This taxonomy uses extend dimensions compared to those presented by Aslam to classify software vulnerabilities. It includes dimensions such as cause (which cause has induced the fault?), type (which operation type is faulty?), removal (process steps to remove the fault) and threat (potential threat of the fault). Krsul's taxonomy distinguishes four categories of vulnerabilities [18], [199]:

- Design: focuses on vulnerabilities at high level classes.
- Environmental assumption: vulnerabilities associated with programmer assumptions.
- Coding fault: assumptions on variable length
- Configuration: focuses on error

Krsul and Bishop taxonomies main goal consists of providing useful descriptions of possible exploits for intrusion detection. Moreover, these taxonomies aim at helping software designers by presenting different techniques both to find and avoid various weaknesses at multiple steps of software life cycle [49].

1.4.2 Attack-centric taxonomies

Attack centric taxonomies are based on attacking techniques, processes, results or other attack dependent criteria in identifying different classes. Involved categorization criteria within these taxonomies and determined attack classes are discussed in the following sections.

1.4.2.1 Attack techniques

One of the earliest attack taxonomy was proposed by Neumann and Parker of the SRI laboratory [274]. It was based on about three thousand incidents collected over a period of twenty years. Neumann and Parker' taxonomy categorizes misuse techniques into nine categories. The table 1.1 presents different attack categories according misuse techniques defined by Neumann and Parker [232].

Neumann and Parker stated that their classes don't meet the uniqueness requirement because the majority of recently detected computer misuses involve techniques from different categories. In [164] and [232], Neumann and Parker taxonomy was criticized because its classes are not logical and intuitive. Moreover, relationships between classes are not explained either by an additional structure which may lead to difficult acceptance and limited use of this taxonomy. However, Neumann and Parker's taxonomy is suitable to classify a large number of attacks because it relies on an extended data sample.

1.4.2.2 Attack results

Several taxonomies have been proposed based on attack results or attacker targets. Cohen categorizes attack objectives into three categories [314]:

- Corruption: unauthorized modification of information
- Leakage: unauthorized divulgation of information
- Denial: computer or network services cannot be accessed by authorized and legitimate users.

This taxonomy focuses on security disruptions from the point of view of the three main security services (integrity, confidentiality and availability). Other works use same categories or define new ones in terms of different security services such as secrecy, accuracy and authenticity [164].

Table 1.1: Neumann and Parker misuse taxonomy [232]

Class	Description
NP1 External misuse	Generally non technological and unobserved, physically separate from computer and communication facilities, for example visual spying.
NP2 Hardware misuse	a) Passive, with no (immediate) side effects. b) Active, with side effects
NP3 Masquerading	Impersonation; playback and spoofing attacks etc.
NP4 Setting up subsequent misuse	Planting and arming malicious software.
NP5 Bypassing intended controls	Circumvention of existing controls or improper acquisition of otherwise denied authority.
NP6 Active misuse of resources	Misuse of (apparently) conferred authority that alters the system or its data.
NP7 Passive misuse of resources	Misuse of (apparently) conferred reading authority.
NP8 Misuse resulting from inaction	Failure to avert a potential problem in a timely fashion, or an error of omission, for example.
NP9 Use as an indirect aid in committing other misuse	a) As a tool in planning computer misuse etc. b) As a tool in planning criminal and/or unethical activity.

DARPA (Defense Advanced Research Projects Agency) and Luke have proposed two different taxonomies of attack results. These taxonomies focus respectively on first and the end goal of an attacker. DARPA's taxonomy is simple and widely used in the intrusion detection field. It was adopted in simulation of data sets for intrusion detection systems evaluation because defined attack categories are convenient for specifying system capabilities. DARPA attack categories are depicted in Table 1.2 [90], [154], [190]. The first four classes form DARPA reduced taxonomy which is the most adopted in intrusion detection literature and also used in this report.

Luke has proposed attack implementation taxonomy based on attacks primary objectives. As he has stated: "an attack implementation is a specific way that an act is done" [322]. Luke's taxonomy distinguishes eight categories of attack objectives. These categories are summarized by the table 1.3 [322].

Table 1.2: DARPA Attack Taxonomy

Main category	Description	Sub-category	Description
1. DOS	Denial-of –service attacks have a goal of limiting or denying services to authorized entities (user, computer, network)	1.1 crashing	Using a single malicious event (or a few packets) to crash a system, e.g. the teardrop attack
		1.2 consumption	Using a large number of events to exhaust network bandwidth or system resource, e.g. synflood
2. U2R	User-to-Root allow attacker who had previously a user-access to gain a super-user access on system or computer	2.1 local	By first logging in as a legitimate user on a local system, e.g., buffer overflow on local system programs such as eject
		2.2 remote	From remote host, e.g. buffer overflow of some daemon running suid root
3. R2L	Remote-to-Local attacks aim at illegally gaining local access to a computer or network by an attacker who had had only remote access	3.1 single	A single event, e.g. guessing passwords
		3.2 Multiple	Multiple events, hosts, or days, e.g. the multiple attack
4. PROBE	Illegally, gaining knowledge on the existence or configuration of a computer system or network	4.1 simple	Many of probe within a short period of time, e.g. fast port scan
		4.2 stealth Probe	Events are distributed sparsely across long time windows, e.g. slow port scan
Data	Data attacks allow an attacker access to some piece of information (file or directory) that was impossible according to stated security policy. These attacks are generally classified as U2R and R2L but are considered as data attacks.		

Table 1.3: First objective based attack taxonomy

Main category	Description
Gain unauthorized use	Gain unauthorized control or use of target resources
Access unauthorized data	Allow attackers to read or obtain unauthorized data from the target
Masquerade as a normal software but to have abnormal uses	Programs that hide their true identities to perform unusual and unseen activities in order to violate the security policy: backdoors or torjan horse programs
Deny of service	Intentional impair of normal function of target resources
Propagate malicious code	Unauthorized spreading of harmful code
Solicit a user	Concerns unwanted attack designed to convince system users to perform harmful actions.(spam, virus, social engineering)
Improperly gain information that could be used for further attacks	Describes any attack that collects potentially useful information for further harmful attacks
Violate a local account restrictions	Unauthorized break of limitations placed upon a local account

1.4.2.3 Attack process

Howard's taxonomy focuses on the operational sequence of tools, access and results that links between the attackers and their objectives. In his process-based taxonomy, Howard has defined multiple categories of attackers, tools, access levels, results and objectives. He has divided entities carrying attacks into six categories [164]:

- Hackers: they break into computers primarily for the challenge and status of obtaining access.
- Spies: they break into computers primarily for information which can be used for political gain.
- Terrorists: they break into computers primarily to cause fear which will aid in achieving political gain.

- Corporate raiders: employees of one company break into computers of competitors for financial gain.
- Professional Criminals: they break into computers for personal financial gain (not as a corporate raider).
- Vandals: they break into computers primarily to cause damage.

When these intruders mount an attack, their primary motivation, as stated by Howard, is one of the following:

- Challenge, status
- Political gain
- Financial gain
- Damage

Different means are used by attackers to gain access to the target system. Howard has defined six possible classes of attacking tools:

- User command: typed shell commands or using a graphical interface
- Script or programs: Scripts or Trojan horse programs or cracking programs triggered by an intruder to exploit different vulnerabilities.
- Autonomous agent: initiated by attackers to exploit vulnerabilities independently to who use the system (computer virus or worms).
- Toolkit: Software package of commands, scripts and autonomous agents
- Distributed tools: tools associated with different hosts by an attacker to mount a coordinated attack.
- Data tap: physical attack tool which focuses on electromagnetic devices such as computer memory or network cables to reveal saved or exchanged data.

Intruders use one or more of these tools to exploit design, implementation or configuration vulnerabilities and achieve their goals. They can gain unauthorized access to target resources and control processes, files and exchanged data with the victim. Moreover, they are able to implement other security disruptions.

The complete attack taxonomy that uses five stage process is presented by figure 1.2 [164].

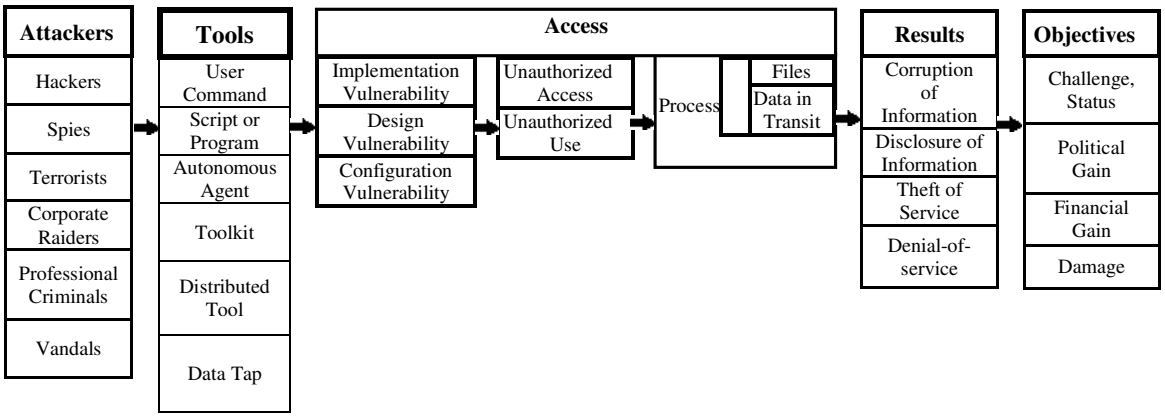


Figure 1.2 : Howard's attack taxonomy

Howard's taxonomy was largely criticized in several works. It was considered as an attempt to derive a process-based taxonomy because it focuses solely on attack rather than attacking stages. Howard's taxonomy fails also to define mutually exclusive classes. This is the case of script or program and toolkit classes of tools categories. Moreover, different other classes of attackers are indistinguishable, from the point of view of the user, such as terrorist and vandals, spy and professional criminal and other entities [158].

Howard's taxonomy was then refined to develop Sandia attack taxonomy of Sandia laboratory [164]. Howard has revised attack stages and proposed new terms such as action and target to describe them. However, all weaknesses of the first are supported in the improved version of the Sandia taxonomy as stated in [158].

Another attack taxonomy based on attacking process was proposed by the Department of Defense (DoD). DoD taxonomy divides attacks into four categories according to attack models. An attack model defines a scenario that combines several entities (people, data, knowledge, software and hardware) to achieve intruder objectives. The four attack models considered in DoD taxonomy are: probe, infrastructure, authorized and factory model [314].

- Probe model: Probe attack model focuses on information collection or gathering attacks to identify opportunities of potential attack that can damage target system.
- Infrastructure model: Infrastructure attack model concentrates on attack designed to persuade entities in target systems to cause harm that affects infrastructure attributes.
- Authorized access models: Authorized access models are concerned with insider entities which have authorizations to access system resources.

- Factory model: Factory attack model focuses on entities in the target systems (hardware and software) that can indirectly cause damage. Tools designed to mount such attacks have always an embedded malicious code.

1.4.2.4 Other attack-centric taxonomies

Multiple attack taxonomies have been proposed based on abuse methods, types of disruption and other criteria. Some of these taxonomies are summarized in following sections

a) Lindquist and Jonsson's Taxonomy

Lindquist and Jonsson have refined SRI computer abuse methods model of Neumann and Parker. They have proposed two dimensions based attack taxonomy that includes techniques and results dimensions. Lindqvist and Jonsson's taxonomy uses misuse techniques proposed at the origin by Neumann and Parker, specifically, NP5, NP6 and NP7 technique categories given in table 1.1. These categories were subdivided into different classes in Lindqvist and Jonson's taxonomy as summarized by the table 1.4 [232].

Table 1.4: First dimension of intrusion techniques

<table>
<tr><th>Misuse technique categories</th><th colspan="2">Lindqvist and Jonsson classes</th></tr>
<tr><td rowspan="3">NP5
Bypassing intended controls</td><td rowspan="2">Password attacks</td><td>Capture</td></tr>
<tr><td>Guessing</td></tr>
<tr><td colspan="2">Spoofing privileged programs</td></tr>
<tr><td colspan="2">Utilizing weak authentication</td></tr>
<tr><td rowspan="2">NP6
Active misuse of resources</td><td colspan="2">Exploiting inadvertent write permission</td></tr>
<tr><td colspan="2">Resources exhaustion</td></tr>
<tr><td rowspan="3">NP7
Passive misuse of resources</td><td colspan="2">Manual browsing</td></tr>
<tr><td rowspan="2">Automated searching</td><td>Using a personal tool</td></tr>
<tr><td>Using a publicly available tool</td></tr>
</table>

Lindqvist and Jonsson's taxonomy includes different intrusion results that mainly concern confidentiality, integrity and availability services. According to this taxonomy, attack objectives are divided into three categories of exposure, DOS and erroneous outputs [232]. These categories and their sub-classes are presented in Table 1.5.

Table 1.5: Second dimension of intrusion results

Results Categories	Classes	
Exposure	Disclosure of confidential information	Only user information disclosed
		System (and user) information disclosed
	Service to unauthorized entities	Access as an ordinary user account
		Access as a special system account
		Access as client root
		Access as server root
DOS	Selective	Affects a single user at a time
		Affects a group of users
	Unselective	Affects all users of the system
	Transmitted	Affects users of other systems
Erroneous outputs	Selective	Affects a single user at a time
		Affects a group of users
	Unselective	Affects all users of the system
	Transmitted	Affects users of other systems

b) DARPA extended taxonomy

Kendall has proposed an attack taxonomy that was reduced to five classes by DARPA team, as previously discussed, to develop a testbed of intrusion detection systems evaluation. Kendall's taxonomy focuses on two main aspects in attacks: transition between privilege levels and actions performed by attacks. The first dimension distinguishes five main classes of privilege or access levels:

- Remote network access (R): minimal access to the target via interconnected networks.
- Local network access (L): ability to read and write to local network where the target is located.
- User access (U): normal access and use of the system.

- Root or super-user access (S): allows super-user or administrator total control of system software.
- Physical access (P): allows operator to manipulate physical component of the systems (hard disk, etc.).

In addition, Kendall has defined five possible ways or means of transition between access levels, these are:

- Masquerading (m): the attacker hides his true identity and convinces the target to believe him as a legitimate user with higher privilege.
- Abuse of features (a): causing a system failure either by making its resources too busy or by imitating privileged actions to gain a higher access level
- Implementation bug (b): exploitation of bugs in trusted programs
- System misconfiguration (c): exploitation of error in security policy configuration
- Social engineering (s): attacker uses indirect means to let the human operator of the system reveals secret information.

Different actions can be performed in each privilege level. For the second dimension, Kendall has used five main categories of actions. Table 1.6 presents actions categories and their descriptions [190].

Kendall has illustrated how to classify and describe different attacks in DARPA database using a string representation of attack privilege, transition and action performed. For example, password cracking attack is represented by the string "***U-use (intrusion)"*** according Kendall's taxonomy. The string representation of password cracking attack states that "user with a local account (U) uses a program which attempts to decrypt an entry in password file". Attacks descriptions and classification process of Kendall's taxonomy are illustrated by figure 1.3 below [190].

Table 1.6: Action categories in Kendall' taxonomy

Action category	Specific type	Description
Probe	Probe(Machines)	Determine types and numbers of machines on a network
	Probe(Services)	Determine the services of a particular system supports
	Probe(Users)	Determine the names or other information about users with accounts on a given system
Deny	Deny(Temporary)	Temporary Denial of Service with automatic recovery
	Deny(Administrative)	Denial of Service requiring administrative intervention
	Deny(Permanent)	Permanent alteration of a system such that a particular service is no longer available
Intercept	Intercept(Files)	Intercept files on a system
	Intercept(Network)	Intercept traffic on a network
	Intercept(Keystrokes)	Intercept keystrokes pressed by a user
Alter	Alter(Data)	Alteration of stored data
	Alter(Intrusion-Traces)	Removal of hint of an intrusion, such as entries in log files
Use	Use(Recreational)	Use of the system for enjoyment, such as playing games or bragging on IRC
	Use(Intrusion-Related)	Use of the system as a staging area/entry point for future attacks

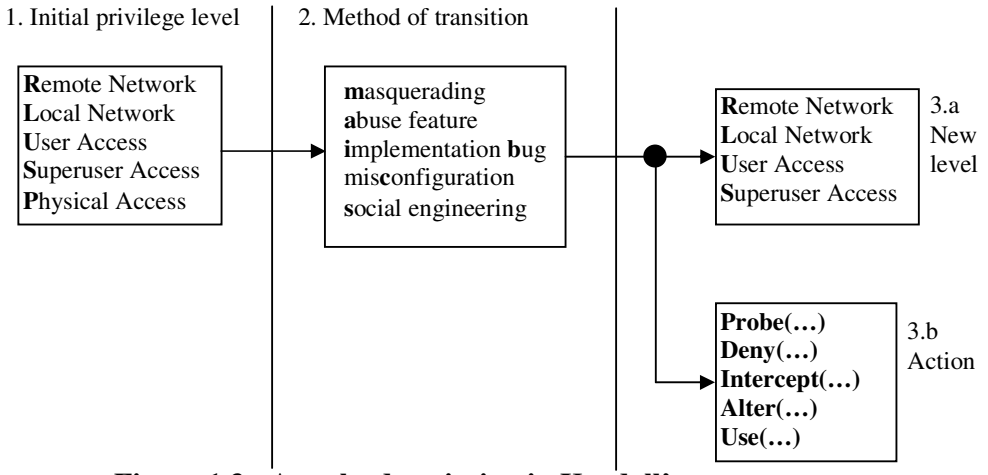


Figure 1.3 : Attacks description in Kendall's taxonomy

c) Raggad's Taxonomy

Raggad's taxonomy relies on three main axes to classify attacks on organization resources. Each attack is assigned to one of the proposed categories based on its source, implication on the target and attack models to carry out intrusive actions.

Raggad uses five types of entities initially identified by Whitten, Bentley and Barlow [408] in system analysis and design. The following entities are considered in Raggad's taxonomy as the origin of any attack on information technology resources:

- People: people in the target system (input data, receive output …).
- Activities: sequences of steps (data flows, security controls …) defined for the target system.
- Technology: technology evolution (impact on target system).
- Data: data and information flows in target system.
- Networks: network requirements at different locations in the target system.

These entities have the ability to cause different types of harms to the target system. Raggad's taxonomy uses three types of disruptions, those introduced by Cohen [81]. Figure 1.4 illustrates causal links between the origin and impact of an attack.

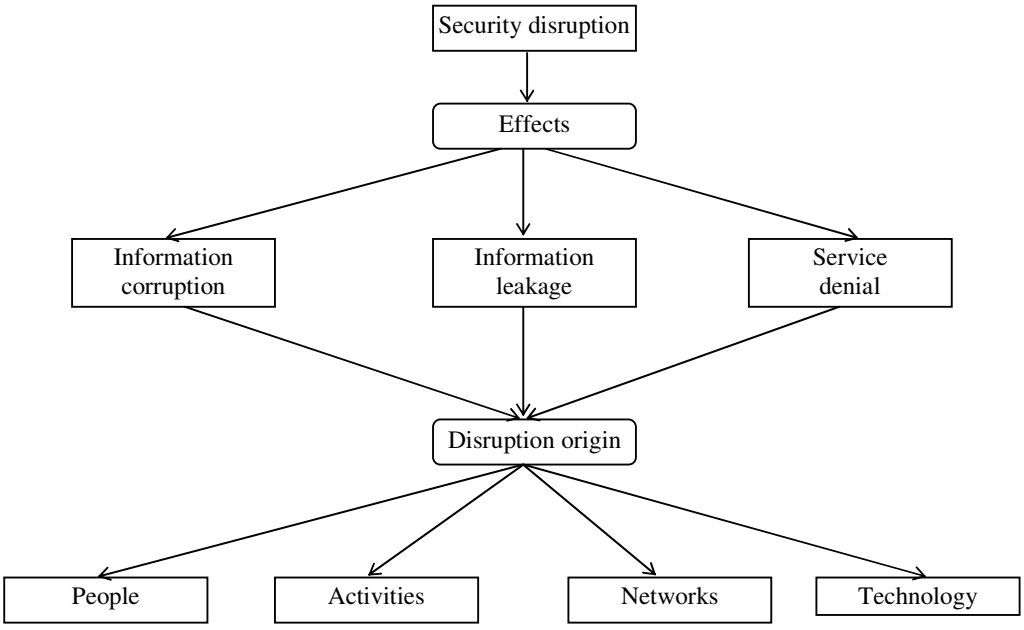


Figure 1.4 : Security disruption classes by (impact, origin) pairs [314]

An attack induces an entity to cause one of security disruptions. DoD attack models form the third dimension of Raggad's taxonomy to describe the attacks. The proposed taxonomy adopts a cube form to represent attack where each axis corresponds to one of the included dimensions.

Raggad has proposed a security and information assurance framework relying on this taxonomy. Based on the three dimensions, he has identified sixteen different information security systems to rule out attacks on confidentiality, integrity and availability of target system resources [314].

d) Multi-dimension attack taxonomy

Hansman [158] has proposed a multidimensional computer and network attack taxonomy. He has fixed the first four dimensions of his taxonomy, which are:

- Attack vector: it provides descriptions of attacks and simplifies their classification.
- Target of attack: it defines different hardware and software components either of computer or network that can be targeted by an attack.
- Vulnerabilities: it presents the implementation and configuration of vulnerabilities that can be exploited by an attack.
- Attack payload: it defines possible results of an attack.

For each of these dimensions, Hansman has defined different classes. Complete taxonomy and detailed classes associated with each dimension can be found in [158]. Furthermore, Hansman taxonomy is extensible and can support additional dimensions such as damage, cost, propagation and defense to reduce classification ambiguity and define precise subclasses.

1.4.3 Defense-centric taxonomy

Defense-centric taxonomy is based on defender goals rather than attacker objectives. In fact, Killourhy and his colleagues have agreed that the attack-centric taxonomies are useful for defender because they inform him on groups of attacks that can be ruled out together. However, defense-centric taxonomies are more helpful from the point of view of defender because they allow him to know whether his detectors have correctly thwarted given attack or not. The authors emphasize that defense-centric taxonomies are much more useful than attack-centric when defender is faced with new attacks. In this situation, is more useful, from the defender and information owner viewpoints, to trigger the most competent detector, if the

new attack falls into one of known attack categories, than analyzing the goal and tools used by intruders.

In their work, Killourhy et al. have performed two main experiments. The first experiment concentrates on attacker behaviors. Twenty five attacks have been carefully selected for this experiment to exploit different vulnerabilities supported in used testbed. During experimentation of these attacks, sensors implemented in the testbed log all sequences of system calls generated either by intrusive actions or vulnerable programs. The second experiment focuses on system normal behavior where sensors collect system calls invoked by system expected activities. Intrusive and normal sequences of system calls are analyzed by authors to identify attacks manifestations. These specific sequences of system calls or manifestations resume attacks performed actions to exploit inherited vulnerabilities. Killourhy and co-works have organized the set of identified attack manifestations into four types:

- Foreign symbol: a manifestation contains a system call that never appears in normal sequences.
- Minimal foreign sequence: a manifestation contains a specific sequence that never appears in normal records but all of its proper sub-sequences are included in normal sequences.
- Dormant sequences: a manifestation contains a sequence which partially matches normal sequences.
- Non anomalous sequence: sequence in attack manifestation that fully matches normal sequences.

Defense-centric taxonomy of Killourhy et al. was based on the four types of manifestation sequences. It defines following mutually exclusive attack classes:

- Class 1 (FS): attack manifestation that contains one or more foreign symbols.
- Class 2 (MFS): attack manifestation that contains no foreign symbol but supports one or more minimal foreign sequences.
- Class 3 (DS): attack manifestation that supports neither foreign symbols nor sequences but contains a dormant sequence.
- Class 4 (MNA): attack manifestation entirely similar to normal sequence and contains no foreign symbol or sequence and no dormant sequence.

Killourhy et al. have shown that their taxonomy meets requirements of acceptable taxonomy. Moreover, they state that defense-centric taxonomy was successful in predicting whether detector is capable to detect a given attack or not, based on its classification results. They consider that defense-centric taxonomy is an accurate predictor and more useful than attack-centric taxonomy for defenders [195]. However, defense-centric taxonomy is neither intuitive nor simple to use. It requires highly skilled and experimented users to classify attacks. In addition, it allows insufficient results when new unseen attacks are classified because implementation of new detectors to defend against them requires detailed information on intrusions sources, tools, results and exploited vulnerabilities. The most important shortcoming of this taxonomy is the complete reliance on a single type of log data, system calls, that induces different problems in identification of manifestation using other types of audit data.

1.5 Attack trends

Attacks are continually evolving as a result of rapid changes in the technology environment. They become real threats to private corporations and governmental services. Moreover, entities carrying attacks have shown, last years, many changes in their malicious activities due to multiple factors such as increased vulnerabilities of new technologies, dependence of critical infrastructures to public network security and wide availability of automated and sophisticated attack tools.

The migration from private to public network has reduced costs and improved the market place of modern businesses. However, this shift from proprietary to standardized systems with common vulnerabilities has dramatically increased the number of attacks. Moreover, new vulnerabilities are increasingly discovered due to rapid changes in information technology resources (wireless network, web applications …). According to CERT, new discovered vulnerabilities are doubling each year [69], [70]. In 2005, Symantec report states that 59% of discovered vulnerabilities are associated with web applications and 73% of the overall are easily exploitable [379]. In addition, attackers have an enormous capacity to exploit either newly discovered flaw in reduced time. The time-to-exploit, the period between discovering and exploiting the vulnerability by an intruder, was estimated to six days in Symantec report. Whereas, system developers report vulnerabilities and their patches after one year to internet security coordination centers such CERT and CVE (Common Vulnerability Exposure).

Increased reliance on the Internet has been propagated to public administrations. Multiple services associated with transportation, financial transactions, healthcare and others are managed over public networks. Security of the critical infrastructures of these governmental services and the Internet systems are becoming interdependent. Any attack that targeted Internet systems can cause dramatic damage to society critical services. The wide availability of automated and highly sophisticated attack management tools provide hackers more opportunities to launch various prevalent attacks that flood networks and tie up mission-critical resources. The most known form of advanced infrastructure attacks is distributed denial of service (DDOS). DDOS uses an array of connected systems, either private corporate or governmental systems, to cause enormous damage to single or multiple sites. Recently, university's networks and Asymmetric Digital Subscriber Line (ADSL) address blocks are becoming the most attractive sites to launch these attacks because they are simple to compromise and easy to remotely control them.

Attack tools are becoming more and more automated and sophisticated. Automated probing tools are able to scan the Internet for vulnerable systems in reduced time. Moreover, they can use the discovered flaws to speed up their propagation and implement more harmful patterns. Attack tools are also capable to self-initiate additional attacks with different behaviors each time. In addition, they allow skilled or beginner hackers to easily launch distributed attacks and readily compromise vulnerable systems on different platforms. Available tools such as a hacker toolkit allow intruders to manage and coordinate a large number of attack tools implemented on multiple systems across the world.

Attack tool developers use sophisticated techniques. These techniques focus on stealthy behavior or anti-forensic feature that makes hacker's intrusive activities difficult to discover and track. Stealthy attacks leave signs indistinguishable to system normal activities in log data which induces multiple ambiguities in forensic analysis and increases the complexity of the security expert mission. Attack tools for mounting such confusing attacks are upgradeable. They can self-initiate varying patterns attacks. Behaviors of performed attacks can change either randomly or according to its decision process or intruder's instructions [70].

As depicted in figure 1.5, attack tools evolution has influenced attacker skill. First attacks were manually implemented. Based on their expertise, attackers develop their own methodology which allows them to compromise tens to hundreds of systems. Twenty years after, widespread automated and sophisticated attack tools assist them to mount devastating attacks on thousands of sites across multiple platforms.

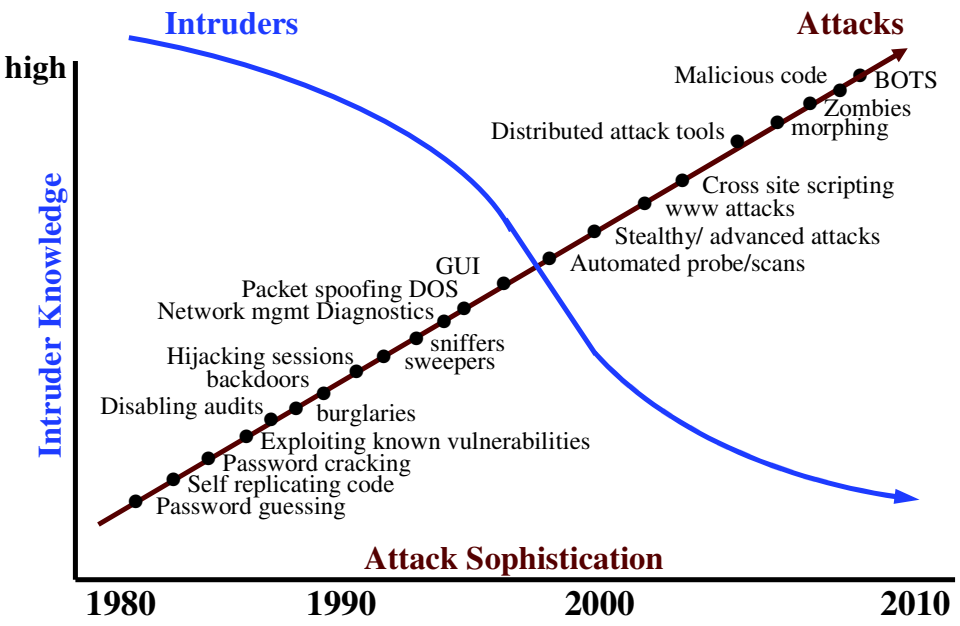


Figure 1.5 : Attacks sophistication versus attacker's technical knowledge [11]

To thwart sophisticated attacks and reduce risks of organizations, multiple security controls should be simultaneously deployed. Various security countermeasures either to avoid or detect attacks on organization resources are presented in the following sections.

1.6 Attack prevention

Various security techniques are integrated in the organization's security package to prevent multiple attacks on confidentiality, integrity and availability of assets. Encryption mechanisms preserve confidentiality of the exchanged data. Authentication techniques establish identities of entities. They ensure the divulgation of exchanged encrypted messages and the usage of system resources only by legitimate users. Authentication allows basic requirements for others security techniques such as access control. Access control mechanisms organize and control access and use of system resources by users with respect to privilege levels and authorizations assigned to them. An extended list of security controls may be found in [148] and the annex A of the ISO 27001 [173]. Some of these controls and their usefulness are discussed in the next sections

1.6.1 Authentication

Authentication is a critical component in preventive part of any integrated security solution. Authentication technique is the first mean of the Site Security Officer (SSO) to discriminate between legitimate and malicious entities that try to access to system resources. It focuses on users, computers, processes and also messages. Authentication techniques use something known, possessed or inherited by entities to authenticate them. The following sections present different authentication techniques which concentrate on system users.

1.6.1.1 Password authentication

Password authentication is the most common and widely used technique. Authentication systems based on password have multiple significant problems. Well-known vulnerabilities associated to password selection and usage are at the origin of many shortcomings of these systems. System users often select weak passwords that can be easily divulgated or stolen by an intruder. Moreover, when users interact with authentication server anyone has the ability to snoop these passwords when keying it in. This is one form of social engineering attack tool.

Two other intrinsic problems are usually associated with password-based authentication systems including password sharing and management. In password authentication systems, the same information is shared between users and the authentication server. Authentication server administrator or SSO can use these passwords because he has a full access to the password file. Moreover, if passwords are saved or exchanged in plaintext any third party is able to intercept the conversation between clients and the authentication server by a simple sniffer and divulgate them. Password management, selection, modification and protection are other serious security problems of password-based authentication systems.

Multiple solutions have been proposed to improve robustness of authentication system based on simple and static password. The majority of actual systems never use manually generated password but they support a password generator. The latter provides extended and difficult to divulgate passwords. These passwords are randomly generated and correctly tested before assigning them to users.

Another solution has been proposed to rule out password reuse and replay attacks. This solution is based on dynamic password or one-time password (OTP). OTP technique ensures password change at each authentication. It is based on shared password list, between user and server, or saved last session password to compute the next one. Cryptographic techniques are

also used to develop strong password authentication protocols. Systems based on these protocols focus on secret information protection either in transit or when saved on the authentication servers [186].

1.6.1.2 Token-based authentication

Tokens are physical devices used by system users as authentication support or authenticator. They can be USB tokens, smart cards or password generating tokens. Authentication tokens are usually used by two factors-based authentication systems. In two-factor authentication, a token has a unique identity in addition to user PID (personal identifier) or a static password. This information cannot be divulgated because user has no permission neither to access nor to modify them. Moreover, if the user tries improperly to modify this information, the token becomes unusable for further authentication trial.

Two-factor authentication is performed in two steps. In the first step, user introduced PID is compared with information saved on the token. In the second step, token identifier is verified using the one saved by the authentication server. These two steps are interdependent; if the first fails the second is dropped.

The majority of token-based authentication systems implement these authentication steps either in the given or the inverse order. However, they can be distinguished by inherited features of used devices. For instance, USB token has the ability to save digital certificates, whereas, password generating token allows random generation of password required in a second step. Moreover, smart card can store and process data using its microprocessor. It can be programmed to perform different strong verifications needed for high-level security systems.

Token based authentication defends against multiple attacks initially possible with static password. However, other weaknesses of static passwords persist also in token based method such as the usage of static PID and loss of token. Furthermore, this solution ensures token authentication rather than user possessing it. Any entity that disposes of a stolen token and knows the associated PID can authenticate himself as the owner of the token [186], [334].

1.6.1.3 Biometrics

Biometrics focuses on physical or behavioral characteristics of an individual person that distinguishes him among others. These unique characteristics associated with each person are

adopted in biometric authentication system to authenticate him. Physical and behavioral features used in biometric authentication include fingerprints, retina pattern, hand geometry, handwritten signature and voice pattern. Biometric authentication systems dispose of databases of users' reference profiles. A reference profile collects biometric information of a user according to desired physical or behavioral feature and serves as his authenticator. When a user initiates a new session on the target system, biometric feature scanner extracts user profile. Extracted and reference profiles are compared and if they completely mach one another, the user is allowed to access to the target.

Biometric authentication has circumvented many inherited weaknesses of previous authentication systems. An important advantage of biometrics is that physical and behavioral features are neither transformed nor stolen. Biometrics ensure that user identity is never forgotten. However, efficiency of biometric authentication depends solely on selected feature. Other technical difficulties such as imperfection of selected feature and profiles extraction or comparison affect the performance of biometric authentication systems. In addition, different feature patterns change, under natural or psychological conditions like speech and face patterns, indirectly affect the efficiency of biometric systems [186].

1.6.2 Access control

Multi-user environment requires the implementation of access control systems to organize access and defend against unauthorized use of shared resources. Multiple approaches can be adopted to control access to system resources. They are based on different access models and use several policies. At a lower level, access mechanisms such as access control lists, ACL, or capabilities lists are used to implement access control systems.

1.6.2.1 Access control models

Access control models provide a basic framework for resource protection. They allow basic components required to the formal definition of access control policies.

1.6.2.1.1 Access control matrix model

Access matrix model was proposed by Lampson. It was improved by Denning et al. This model allows generalized descriptions of access control mechanisms associated with operating systems. Denning et al. improved model was based on three main components organized as a matrix structure [143], [334]:

- Passive objects: a set of resources to be protected and their types (file, host.....). For each object type is associated a set of allowed operations
- Active subject: a subject is an entity wishing to access and perform an operation on an object. Active subjects, users or processes, are also considered as objects that require protection.
- Access rules: the set of access rules defines allowed operations for each pair (subject, object). These rules state subjects' access rights or allowed operations on each object.

1.6.2.1.2 Lattice security model

The lattice security model is an extension of the access matrix proposed in [143]. It is inspired by the military classification system and its structured security levels. The lattice based model focuses on the information flow between security classes in the computer system. Specifically, it concentrates on confidentiality classes. Lattice based model defines three main concepts:

- Set of security classes: objects, users and processes are assigned to security classes.
- Classes-combination operator: an associative and commutative binary operator that determines the result class for any binary function on two operand classes.
- Flow relation: relation defined on two security classes. It specifies permitted information flows between these classes.

In lattice based model, Denning has added the following extensions to the matrix model:

- Each object is assigned a security class (unclassified, confidential, secret and top secret). The security class of an object concerns its content.
- Each subject is assigned a clearance that specifies allowed security classes to be accessed.
- Each pair (object, subject) is labeled with allowed classes for the subject and information content classification for the object.

Other extensions have been proposed for the lattice model such as the BLP model (Bell and LaPadulla model) that aims at the reinforcement of access control policies by concentrating on mandatory access rules. Additional access control models have been developed. They define new concepts or focus on different environments or security services. Role-based and SPM (Schematic Protection Model) are two access control models proposed by Shandhu

[334]. The role-based model focuses on other concepts rather than subject and object, such as task, action and responsibility. SPM model was designed for organizing access in distributed systems. The Biba access control model uses BLP basic concepts, but it concentrates on data integrity rather than information confidentiality [143], [186].

1.6.2.2 Access control policies

Access control policies are based on model components to define different rules for controlling who can perform which operation on what object. Access control rules determine access decision when dealing with authority delegation or access right revocation. Access control policy allows different security levels. The suitable access control policy should satisfy security requirements of the target system. Within the same system, different security policies can be integrated. However, they ensure required security level only if all conflicts between them are resolved.

The most known and used access control policies are discretionary and mandatory policies.

1.6.2.2.1 Discretionary policy

Discretionary access control policy organizes access to system resources on the basis of users' identities and their authorizations. Authorizations are explicitly attributed by objects or resource owners to each user or group of users. For each access request to a specific object, user authorizations are checked to grant or deny his demand.

Discretionary access control policy is widely used in industrial and commercial systems because it offers required flexibilities (policy can be easily modified by object owner). However, it supports multiple shortcomings. It doesn't ensure any consistency with the local policy because each owner has the flexibility to define and modify access control policy on his resources. Moreover, it imposes any constraints on information flow within the system. Therefore, authorized users have the ability to copy information form one object to another and reuse the copy without owner authorization.

1.6.2.2.2 Mandatory policy

Access to resources in mandatory policy is organized based on subject and object classifications. Each object in the system is assigned to a security level that reflects the sensitivity of its content. The clearance associated with each user reflects his trustworthiness to not divulgate sensitive information to the low clearance subject. Security classes in

mandatory policy can be unclassified, confidential, secret or top-secret. In this hierarchy, each class dominates itself and all other classes below it.

Each subject is authorized to access an object depending on access model (read, write), his clearance and the classification of the object. Each access should satisfy following mandatory rules:

- Read down: The subject's clearance should dominate the object's security level in reading
- Write up: the subject's clearance should be dominated by the object's security level in writing.

These two rules allow mandatory policy to prevent information flows from high level sensitivity classes to low levels. Moreover, they ensure information flows with higher levels or within the same level of the hierarchy.

Discretionary or mandatory policies are recognized as standards of the DoD's orange book. Both have been integrated in role-based policy to define access control rules that concentrate on roles rather than objects and subjects.

Access control policies can be implemented using different mechanisms such as ACL or capabilities lists. These access control mechanisms focus respectively on objects or subjects. They allow respectively a list of authorized operations for each subject or a list of capabilities on each object. Other access control mechanisms not included in this work use different data structures to implement access control rules (authorization relation, etc.) [143].

1.6.3 Cryptography

Cryptography is the only known security technique to defend against passive attacks such as message release and traffic analysis. Associated with other security protocols, cryptography can be used as a countermeasure against active attacks. Encryption controls only ensure confidentiality of data and information flows. They can support other security services such as integrity, authenticity and non-repudiation. Cryptography is based on two primitive operations as illustrated in figure 1.6: encryption and decryption [157].

- Encryption: is the operation that consists of performing mathematical transformation or different computations on readable message (M) using secret or public information (e) to generate meaningless message or a cryptogram(C).

– Decryption: is the operation of recovering the initial message using its cryptogram. It consists of applying the inverse transformation or computations on the first unintelligible message (C) using the same secret or other private information (d) to regenerate the initial intelligible message (M).

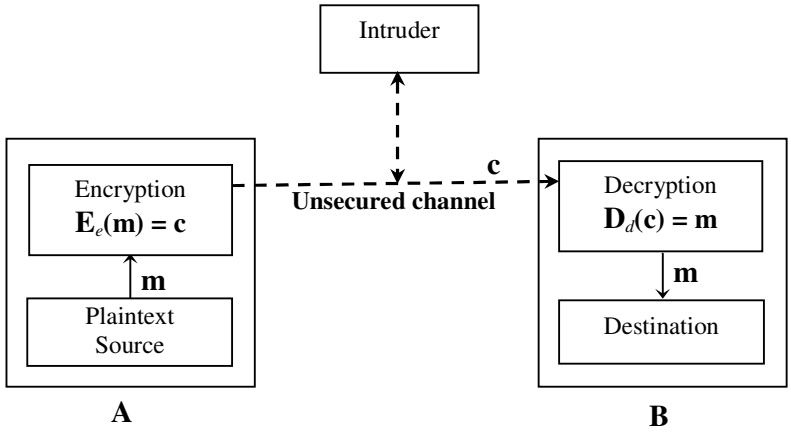


Figure 1.6: Two party communication using encryption [157]

Encryption system based on these two operations can be divided into secret key or public key cryptosystems. Symmetric encryption algorithms allow communicating entities to share the same secret key as depicted by figure 1.7. The shared secret key in symmetric cryptosystems is used both for encryption and decryption.

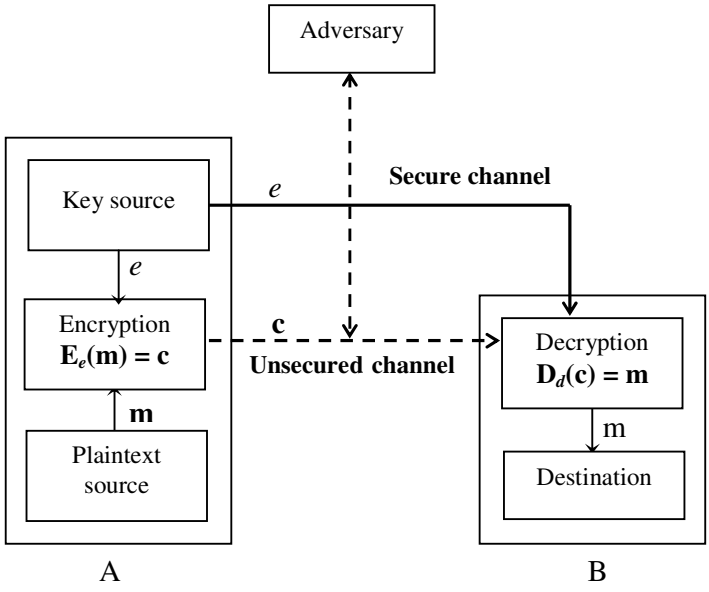


Figure 1.7: Two-party communication using symmetric encryption [157]

Symmetric cryptosystems are widely used because they are simple and efficient in terms of computation time. However, they support multiple weaknesses. They require a secure channel

for exchanging secret key between entities. In addition, secret key should be kept secure to preserve confidentiality of previously exchanged messages.

Public key encryption avoids all previous problems as illustrated by figure 1.8. It involves a pair of key (public and private keys (*e,d*)) associated to each entity. The public key is published to everyone who wants to communicate with the entity. The private key remains secret and not revealed to any other entity. Data are generally encrypted under the control of the public key and never decrypted only using the corresponding private key. In asymmetric cryptosystem, it is computationally unfeasible to deduce the private key from the associated public key. Moreover, each entity has a key pair different and independent to other entities' pairs.

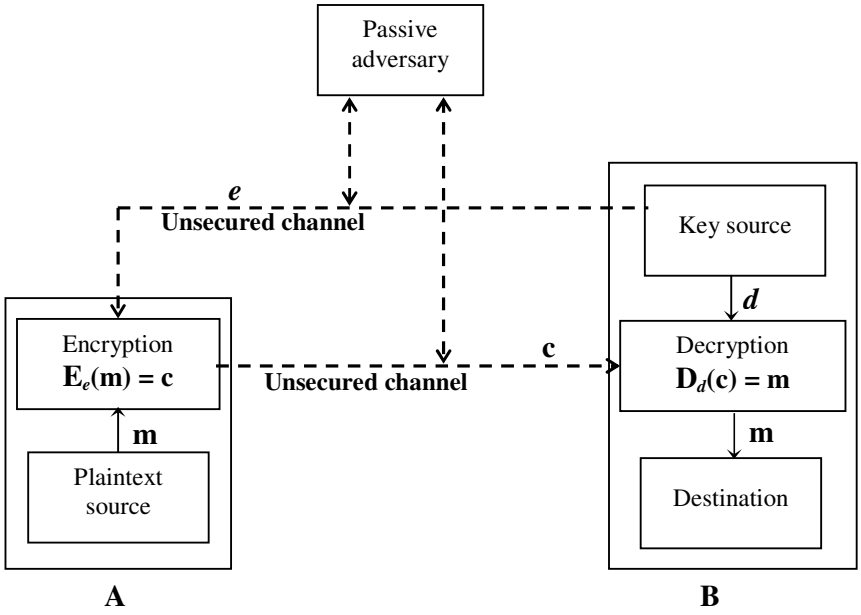


Figure 1.8: Encryption using public-key cryptosystem [157]

Compared to symmetric encryption, public key cryptosystems are more secure and appropriate for safeguarding and exchanging confidential information. However, they are not always effective such as for exchanging large amount of data; they are more costly than symmetric systems in terms of computation and communication loads. Asymmetric cryptosystems have induced other difficulties than those associated to symmetric cryptosystems. Public key authenticity and revocation are among the most known shortcomings of using asymmetric encryption.

The most known and used cryptosystems are DES (Data Encryption Standard) and RSA (Rivest, Shamir and Adleman). These algorithms are recognized as standards of respectively symmetric and asymmetric cryptography. They use keys of different lengths; 64 bits for DES

and 512 bits or more for RSA. DES key length was extended to 128 bits or more in the AES standard (Advanced Encryption Standard). Encryption algorithms of two classes, secret and public key based, have been adopted in multiple authentication systems and e-commerce protocols such as SSL/TLS [186].

1.7 Intrusion detection

With the increased reliance and wide connection to public networks, classical security measures fail to satisfy assurance requirements of modern business environments. They try to determine legitimate users abusing their privileges. Additionally, they focus on minority of known breaches to the target system. Multiple other weaknesses in security policy, software or hardware can be exploited by inside or outside entities to increase their privileges or divulgate sensitive information. Preventive security controls can't thwart attacks mounted by these entities. Intrusion avoidance mechanisms are required to monitor and detect the increase of privileges and abnormal behaviors of system users.

An intrusion detection system (ids) tries to detect attackers' breaches to the monitored systems and legitimate users misusing their privileges. It focuses on known and potential security faults that can be exploited by an intruder. Its main goal is to discover any violation of the security policy not prevented by classical security measures. An intrusion detection and response system (idrs) has the ability to passively or actively respond to detected attacks. It also serves as a quality controller that highlights possible flaws in security design or management of the target system. Moreover, they report detected attacks and allow useful information about them. This information is helpful for the SSO in revising both organization's security policy and configuration [147], [176].

The National Institute of Standard and Technology (NIST) defines intrusion detection as "the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions" which represent any "attempts to compromise the confidentiality, integrity, availability or to bypass the security mechanisms of a computer or network" [176]. An ids is defined as a system that attempts to identify "individuals who are using a computer system without authorization (i.e.: crackers) or those who have legitimate access to the system but abusing their privileges (i.e.: insider threat)" and "any attempts of these". It is based on three main components (data gathering, data processing and response unit) to detect "any set of actions attempting to compromise confidentiality, integrity or availability of a resource" [33]. To achieve these objectives, an ids is based upon a multi-step

process. Furthermore, it has to meet several requirements. Many of these requirements and ids generic process are discussed in the next two sections

1.7.1 Desired characteristics of intrusion detection systems

As stated by Miller et al. and Spafford et al. [160], [366], an ids has to satisfy different characteristics. These desired features of an ids are related to its detection principles, design, performance and environment. The most common characteristics are the following:

- Continuously running with minimum human assistance: Administrator or SSO should be able to monitor ids status.
- Fault tolerance: an ids must recover its previous state when accidental or intentional system crash occurs.
- Resistance to subversion: an ids has to rule out malicious activities against its components and periodically verify their integrity.
- Minimal overhead when running: an ids should not affect other normal applications performances.
- Configurability: an ids must be highly configurable to ensure simple and easy implementation of monitored system security policies.
- Adaptability to changes as well as in system or user behavior: stored patterns also must be regularly updated.
- Scalability: in an extended network, an ids must monitor all hosts' activities and detecting different attacks against them.
- Graceful degradation: ids components must be independent, autonomous and cooperative to eliminate complete system breakdown when an intruder targeted one of them.
- Dynamic reconfiguration: when an administrator modifies his system architecture, the ids must be adaptable and able to implement configuration changes.

Additional features such as visualization of system activities, tracking and tracing of attacks can be appended to this list and others discussed in [72], [93], [192], [217], [315]. These requirements concern a single or multiple components that implement different steps of an ids process. The latter and its main processing steps are discussed in the following section.

1.7.2 Intrusion detection and response process

An idrs process consists of the following three main steps:

- Data collection and preprocessing: gathering log data from different sources (router, firewall, application...) and within multiple monitoring levels in the system (networks, host, application). The collected data are preprocessed and formatted as required for analysis components. This step relies on different types of sensors.
- Data analysis: in this step, data forwarded by sensors are processed by analysis engine or detector to capture intrusive events based on saved patterns. These events concern a signal or multiple actions of potential intrusions.
- Response: when the detector raises an alarm, active response component reports complete information on the detected attack, deployed countermeasures and assists SSO in choosing convenient controls or implements appropriate ones. It may be autonomous in that it is capable to select the most appropriate set of actions, to counter detected attack, and implement them. Passive response component instead, simply, alerts SSO on the detected malicious actions.

The three step process of collect, analyze and respond, of an idrs can be implemented by components of the architecture given in figure 1.9 [217]. In this generic architecture, ids detector interacts with three information sources. The sensors provide detector with information on system activities. Configuration information allows the detector to evaluate the actual state of the system. Knowledge base saves possible patterns, of known intrusive or normal activities, needed for performing the detector process. The response component receives information on system configuration and detector outputs. This information is required not only to implement appropriate corrective actions, but also to generate a complete report on the detected attack [93], [217].

Based on their data collection, analysis and response components, intrusion detection systems have been classified into different categories. The main ids taxonomies proposed by Debar et al. and Axelsson are presented in the next section.

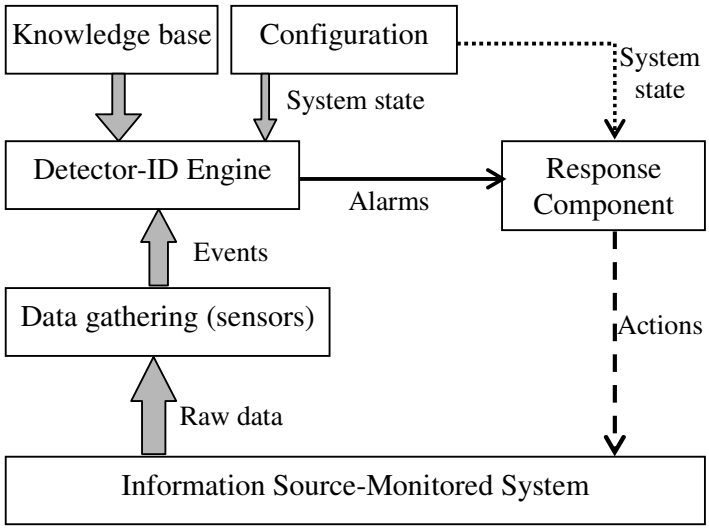


Figure 1.9 : Generic architecture of an idrs [93]

1.7.3 Intrusion detection systems classification

The most common and widely used ids taxonomies were proposed by Debar et al. and Axelsson [30], [94]. The first ids classification proposed by Debar et al. was based on ids characteristics. It focuses on detection methods, behavior on detection, audit source locations and detection paradigm of an ids. It categorizes research prototypes and commercial intrusion detection systems into four classes. Identified classes are further refined using different additional criteria in the revised and enhanced taxonomy [94]. Considered properties in the improved taxonomy were extended by detection paradigm of an ids to separate between state and transition based monitoring capabilities of these systems. Moreover, Debar et al. have refined data sources of ids to include applications log data and ids alerts (figure 1.10). Main ids classes of both taxonomies will be presented below.

Debar et al. revised taxonomy defines following classes of ids systems as presented in figure 1.10 [94]:

- Detection method: this property is concerned with the central component of an ids, its analysis engine. The data analyze engine of an ids can be either behavior or signature based.
  - Behavior based: Behavior or anomaly based ids extracts normal or valid behavior of the system using on non intrusive log data. It assumes any deviation, to the expected behavior of the monitored system, exceeding a prespecified threshold as an intrusion.

- Knowledge based: Knowledge, misuse, signature or also policy based ids disposes of a knowledge base of known attacks and vulnerabilities of the monitored system. These ids raise an alarm only if logged activities match one of saved attack signatures or attempt to exploit known vulnerabilities.

– Behavior on detection: This characteristic focuses on response component of an idrs. According this property, an idrs allows a passive or active reaction :

  - Passive: Passive idrs only alerts SSO on that an attack has taken place. It implements no countermeasures to defend against detected malicious activities.

  - Active: Active idrs triggers corrective actions when an attack is detected (changes file permission, generate script for system patching, restore system, …)

– Audit source location: this property concerns data collection components of ids. Sensors of an ids collect audit data of a single host, network, application or service or other resources of the monitored system.

  - Host-based: Host-based ids processes log data generated by a single machine. Host-based sensors collect sequences of system calls, user commands or syslog information, as discussed in chapter 6.

  - Network-based: Network-based ids analyses network traffic log data. Network-based sensors extract log data from management information base (MIB associated to the Simple Network Management Protocol, SNMP) or sniffed network traffic, using different sniffers.

  - Application-based: Application-based ids processes log data generated by a single application. Sensors of these ids filter network traffic and extract log records of a specific service (HTTP-sensor, FTP-sensor, …).

  - Intrusion detection alerts: This type of ids focuses on correlation of different alerts generated by other ids.

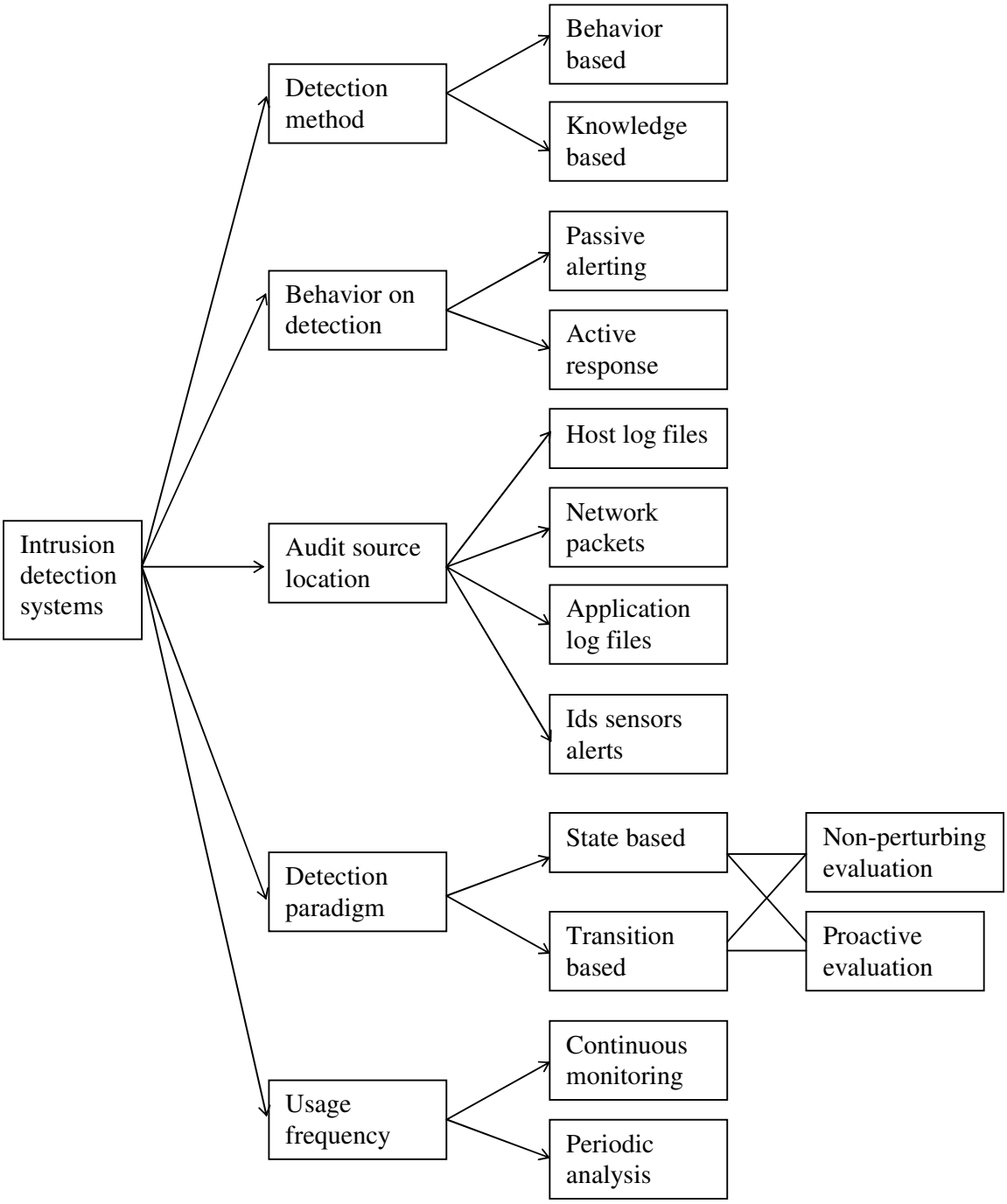


Figure 1.10: Debar et al. revised ids taxonomy

– Detection paradigm: This characteristic focuses on detection mechanisms of an ids. An ids evaluates state or transition between system states as secure or insecure. Moreover, its evaluation can be based on either system observations or complete simulation of system states and transitions.

  - State based ids concentrates on system state either secure or insecure. It evaluates system states based on vulnerability knowledge base and reference configuration

of the system. It identifies insecure states when the configuration is changed or one of the vulnerabilities is matched.

- Transition based ids evaluates the transitions between states either secure or insecure. It focuses on specific events that trigger system transition from secure and insecure states.

  State and transition analysis for these ids can be performed in non-perturbing or proactive way. Non-perturbing analysis of observations is similar to the vulnerability assessment process where each state or transition is checked against known vulnerabilities. The state or transition is insecure if single vulnerability is matched otherwise it is secure. Proactive analysis instead actively triggers events to exploit vulnerabilities. Generated state or transition of simulated system is compared to real one to check whether it is secure or not.

– Usage frequency: This property focuses on the detector of an ids. It describes how an analysis engine processes log data.

  - Continuous ids performs a real time analysis of any activity in the system immediately after it takes place.

  - Periodic ids performs a batch analysis of log data. The state of monitored system is evaluated using all logged data within a prespecified time window.

Axelsson work was inspired by Debar et al. taxonomy. However, Axelsson thought of the detection method property as the most discriminating factor between ids. This property was distinguished to all other ids characteristics. It was taken as a root node in the tree structure of Axelsson's taxonomy. Based on the detection principle, Axelsson's taxonomy classifies ids as anomaly, signature or signature inspired systems. Signature inspired systems rely on compound detectors which use both attack signatures on normal behavior patterns. All three classes of ids are then divided either into programmed or self learning subclasses.

Axelsson has proposed a second taxonomy that deals with other ids characteristics. Included characteristics are deduced from ids desired properties, presented above in section §1.7.1. Axelsson has used different terms than those integrated in Debar et al. taxonomy. He has treated an extended number of ids desired properties in this taxonomy. Ids characteristics of Axelsson's taxonomy are the following [30]:

– Time of detection: real-time and postponing detection.

- Granularity of data processing: continuously of batch processing.
- Source of audit data: network data or host based data.
- Response to detected intrusion: passive or active.
- Locus of data processing: centralized or distributed.
- Locus of data collection: centralized or distributed.
- Security: the ability to defend against attack on ids.
- Degree of interoperability: the degree of operation with other ids.

Intrusion detection systems classifications are required both to identify capabilities of ids and reinforce organization's security. They are required in choosing appropriate ids, according to the security posture that assesses the current security position of the monitored system. Moreover, they are useful to provide insight into possible improvements and extensions of available detection systems. Different research surveys have discussed additional requirements of intrusion detection systems including [72] and [217].

1.7.4 Intrusion detection normalizing activities

Motivated by the idea of systems interoperability, DARPA and IETF (Internet Engineering Task Force) have assigned two workgroups to normalize respectively design and communication between idrs.

1.7.4.1 Common Intrusion Detection Framework workgroup

The workgroup of DARPA has developed the Common Intrusion Detection Framework (CIDF). CIDF was proposed to standardize ids architectures. Moreover, it uses common protocols and application programming interfaces (API) to interoperate different ids.

An ids architecture based on CIDF consists of four main components:

- Event generator boxes (E-boxes): collect and format event from the target environment and send them to A-boxes.
- Analysis engines (A-boxes): analyze the messages of E-boxes and forward their conclusion either to R-boxes or other A-boxes.
- Response engines (R-boxes): consume A-boxes messages to carry out required corrective actions.

– Event database (D-boxes): store messages exchanged between all CIFD components for further use.

All the four black boxes exchange data in a standard format called GIDO (Generalized Intrusion Detection Object). GIDO are represented using the standard common format defined for CIDF. E-boxes events, A-boxes conclusions and R-boxes corrective actions are encoded and formatted into GIDO before they are forwarded to other components.

CIDF components interact and cooperate for detecting and responding to attacks. E-boxes generate GIDO using collected events from the environment they are specialized in. E-boxes formatted events are forwarded to A-boxes components. A-boxes analyze received objects and produce new ones that support its analytical conclusions. R-boxes consume A-boxes GIDO to carry out different corrective actions. D-boxes save all components outputs for further use.

GIDO are defined using CISL API (Common Intrusion detection Specification Language). CISL proposed by CIDF workgroup is flexible and extensible language. It is based upon expression types capable to represent event data, analysis results and response directives from different ids. Thus, it allows required flexibilities to interoperate multiple idrs in different stages of the intrusion detection process [58].

1.7.4.2 Intrusion Detection work group

Intrusion Detection Work Group (IDWG) was created within IETF to define a common data format and procedures for improving information sharing and cooperation between ids and other systems interacting with them. It has found that CIDF specification is not suitable to be an internet standard for exchanging information between ids. In addition, proposed protocols and exchange format are not in concordance with XML standards.

IDWG has defined a specification language based on XML that describes Intrusion Detection Message Exchange Format (IDMEF). IDMEF is a standard encoding format for information and alerts exchanges between ids components. Furthermore, it is useful for interoperating different commercial systems and ids research prototypes and normalizing exchanged alerts between them (alerts correlation and aggregation).

IDWG has proposed also IDXP protocol (Intrusion Detection eXchange Protocol) to transport IDMEF objects. IDXP is an application level protocol to exchange intrusion detection

messages. It is based on other protocols that allow authenticity and confidentiality of exchanged messages [58], [94], [308].

1.8 Conclusion

In this chapter, we have introduced multiple information security concepts. Based on theses concepts, we have presented different attack types that aim at compromising the three main security services namely confidentiality, integrity and availability discussed before. We have analyzed attack future trends based on different evolution factors including attacker skill and tools sophistication. To defend against these attacks, we have presented different prevention and detection security countermeasures. Various cryptosystems, authentication techniques and access control models and policies have been discussed to avoid attacks against organization resources. The attack detection solution presented in this chapter focuses mainly on idrs. We have presented the generic architecture of an idrs and its components. Based on these components, we have discussed many idrs categories of Debar et al. taxonomies. Different normalization efforts which concentrate on idrs components have been also presented in this chapter. The next chapter will focus on analysis and response mechanisms of an idrs. Several detection models and response strategies used respectively by analysis and reaction components of an idrs will be studied and summarized in this part.

CHAPTER 2

RELATED WORK: INTRUSION DETECTION ANALYSIS AND REACTION MECHANISMS

2.1 Introduction

Analysis mechanisms of existing intrusion detection systems have been based on a wide variety of statistical, artificial intelligence, data mining, signal processing and other techniques [11], [79], [89] [400], [403], [415]. However, machine learning and data mining techniques are the most extensively applied in misuse and anomaly detection. They may be categorized into three main classes depending on their learning approaches. Techniques based on supervised learning approach require complete training datasets in which data instances are structured into groups relying on their output labels or values. Semi-supervised analysis techniques use partially labeled training datasets where instances of the target output only are predefined. The last category concerns unsupervised techniques that require no prior knowledge on output values. They are given with unlabeled datasets from which representative models of discovered useful patterns are derived.

Response mechanisms of ids are broadly categorized into passive or active. Passive response mechanisms simply alert the SSO on mounted and detected attack and eventually provide them with a detailed detection report. Active response mechanisms instead have defensive or preventive reactions against intrusive activities. They are capable to select corrective actions and implement or assist the SSO in implementing them. Multiple classes of active reactions are identified by developed taxonomies such as in [67], [368]. They include static, dynamic and cost sensitive classes identified based on the criterion of response selection methods. Proactive and delayed are also two classes of active response determined depending upon the deployment time criterion.

For these two components of an irds, multiple analysis techniques and response strategies have been experimented. Techniques of supervised and unsupervised learning categories are commonly adopted by analysis mechanisms of existing ids. Furthermore, cost sensitive response mechanisms either proactive or reactive have gained great attention, the last decade. In this chapter, many of existing analysis mechanisms and their adopted techniques are reviewed. Moreover, several designed response mechanisms and their response processes and cost metrics are presented in this chapter.

2.2 Main techniques of intrusion detection analysis mechanisms

Machine learning techniques are widely adopted in designed analysis and detection mechanisms of ids. Other data mining techniques also are extensively applied in analysis and detection steps of the ids process. This section presents a structured review of several previous ids experiments, their analysis processes and experimental results, relying on their adopted techniques. Subsequent sections will respectively summarize commonly applied supervised and unsupervised machine learning and data mining techniques in intrusion analysis.

2.2.1 Supervised machine learning techniques

Supervised techniques focus on hidden relationships between input and output variables. They aim at extracting and describing them by mappings between used variable. Learned knowledge of these techniques will serve to build concise models and predict right output values of the input instances. Depending on involved techniques, derived predictive models are explicitly or implicitly expressed by different forms including decision trees or neural networks or probability vectors.

Supervised machine learning algorithms form two main categories, of classification and regression, based on their tasks. Classification methods focus on categorical output features. Their constructed learners, called also classifiers, serve to predict output classes of given data instances. Classification techniques are the most applied by analysis mechanisms of existing intrusion detection systems. Regression methods instead concern continuous outputs. Their built regressors or regression models are involved in predicting real valued outputs of the input data observations.

This section specifically focuses on supervised classification methods that have gained increased attention in the intrusion detection field. Furthermore, several analysis engines involving these methods are also surveyed in the current section.

2.2.1.1 Decision trees

Decision trees are one of the most commonly adopted supervised learning techniques in applied fields as well as in research. A decision tree is a hierarchical model that includes decisions and their consequences. It consists of decision and leaf nodes and branches or edges. A decision node corresponds to a test attribute. An edge specifies a possible outcome of the corresponding test attribute. A leaf or terminal node represents a class label. In the tree structure, the root is the top decision node that has no incoming branch. However, internal decision nodes have both incoming and outgoing edges.

The decision tree construction process consists of two main phases. Attribute selection is the first phase of the tree building process. It is required to reduce the complexity of the learning task and optimize the decision process. Such phase is involved by different other learning algorithms specifically when dealing with highly dimensional feature space. In decision trees, attribute selection phase identifies relevant features that will serve as decision nodes. It can be based on various selection criteria including information gain and gain ratio [310], [311]. The gain ratio criterion was proposed by Quinlan using the Shannon Entropy. Given that a data set D and an attribute A, the amount of information required to identify class label of an instance in D corresponds to the D entropy which is estimated as follows:

$$Entropy(D) = -\sum_{i=1}^{n} p_i \log_2(p_i) \quad (2.1)$$

where p_i corresponds to the probability that a given instance of D belongs to the class C_i; $p_i = |C_{i,D}| / |D|, i = 1..n$, $|D|$ and $|C_{i,D}|$ are respectively cardinalities of the set D and instance subset of class C_i in D.

The information amount regarding data partitions of D, D_j, $j=1,\ldots, m$, defined based on the values of A, a_j, $j=1,..., m$, is determined by:

$$Entropy_A(D) = -\sum_{j=1}^{m} \frac{|D_j|}{|D|} Entropy(D_j) \quad (2.2)$$

The information gain due to the use of data partitions of A is estimated by the difference between information requirements before and after involving these partitions. It is given by:

$$Gain(D,A) = Enterpoy(D) - Entropy_A(D) \quad (2.3)$$

The gain ratio of *A* corresponds to its normalized information gain using split information values. The latter represents potential information generated by splitting *D* with respect to *m* outcomes of the test attribute *A*. Split information and gain ratio are evaluated as follows:

$$SplitInfo_A(D) = -\sum_{j=1}^{m} \frac{|D_j|}{|D|} log_2\left(\frac{|D_j|}{|D|}\right) \quad (2.4)$$

$$GainRatio(D, A) = \frac{Gain(D, A)}{SplitInfo_A(D)} \quad (2.5)$$

In the feature selection phase of the decision tree construction process, the most relevant attribute has the highest gain ratio.

In the second phase of tree building, the most relevant feature is taken as root node of the tree structure. The branches of this node are determined using the training set. Each branch defines a new sub-tree. The root node of the sub-tree corresponds to the most appropriate attribute of those remaining with respect to the adopted feature selection criterion. Its outcomes are evaluated based on a given data partition. This process relies on the top-down approach in building decision trees. It starts from the root node and then recursively performs to create descendent nodes until satisfying the stopping criteria such as tree size.

Classification of a new example using generated decision tree is initiated at the root node. The value of the root feature in the given instance is tested and the convenient edge leading to the appropriate sub-tree is selected. By moving down to the next root node of the new sub-tree, the same decision process is recursively executed until branching on a leaf node. The latter is considered as the most appropriate class associated to the given example.

Various algorithms have been developed for decision tree induction. ID3 and C4.5 are among the most known top-down based decision tree algorithms. They were proposed by Quinlan [311]. They use respectively information gain and gain ratio as feature selection criteria. ID3 is the predecessor of C4.5. Both algorithms have been widely applied in the intrusion detection field for implementing misuse as well as anomaly detection systems.

In their signature based intrusion detection system, Ye et al. have adopted machine learning techniques specifically decision trees. Layered classifier of Ye et al. consists of two levels of induced decision trees. Decision trees of both levels provide intrusion warning values to determine whether a processed system call sequence is normal or intrusive. Low level decision trees correspond to single event classifiers. They are built using only system call logs

and a single predictor feature which is the event type attribute. The induction algorithm for low level classifiers processes a single event each time to grow the tree structure. Decision nodes for building low level decision tree classifiers are chosen from 284 possible system calls of the Solaris operating system. Leaf nodes correspond to normal and different attack classes. They determine intrusion warning values of processed system call sequences.

Upper level decision trees or state-ID classifiers are generated using preprocessed system call training set. For these classifiers, original data sets are transformed into state sets using data preprocessing techniques including moving window technique. The latter identifies different states of a system call data set by considering the timestamp of each call and an observation window of fixed size. Determined states represent distinct fixed size sequences of system calls. They will serve as test features of the upper level decision tree classifiers. In the proposed two level classifier, Ye et al. use single event classifiers to determine states for upper level decision trees. However, in this case identified states may correspond to sequences of system calls of variable lengths.

Ye et al. have tested different variants of single event, state-ID and layered classifiers using the DARPA 98 and simulated system call data sets. In these variants, test attributes outcomes are either binary or multi-valued respectively when testing on existence or occurrence count of a given system call or state within processed sequences. Separately conducted tests for single event and state-ID classifiers show that the tree structure of the latter is more appropriate than the former. State-ID classifier allows also simple classification rules. Moreover, it outperforms single event decision trees in terms of classification and false alarm rates. For less than 20% of false alarm rate, it ensures near 90% as a classification rate which exceeds the double in the case of low level classifiers. However, its performance is less good than tested variants of layered classifiers. The count variant of layered classifier has better performance than the existence variant, single event and state ID classifiers. It nearly achieves 95% of the classification rate for less than 5% of false alarms [419].

Ben Amor et al. have conducted different experiments that aim at comparing the appropriateness of the decision tree and naïve Bayesian network techniques for intrusion detection field. The main findings of these experiments state that on one hand decision tree classifiers have better performances than naïve Bayesian networks in identifying normal and DOS attack instances. On the other hand, naïve Bayesian networks outperform decision trees in terms of probe attack detection rate. However, both types of detection models fail in identifying U2R and R2L attack instances [41]. Tabia has extended these experiments to

improve decision tree and naïve Bayesian detection models and then their capabilities to detect unknown attacks. He has proposed two types of improvements to decision tree generation algorithms. The first enhancement focuses on relaxing feature selection measure in order to construct decision trees with extended numbers of test nodes and leaves. The Tabia's extension uses second ranked feature as a root node of built tree or sub-tree instead of the attribute associated with highest gain ratio as in C4.5 algorithm of Quinlan. The second improvement concentrates on stopping criteria and pruning options. The relaxations of stopping criteria and pruning options aim at generating large decision trees that fulfill the main requirement of separation between normal and attack instances. Tabia's extension with its two improvements generates large decision trees called compatible decision trees. The latter are capable to reduce classification error and increase testing probabilities of new unseen events previously confused with normal events.

Tabia has experimented compatible decision trees and compared them to those standard, generated using the original standard C4.5 algorithm. Using a training set extracted form the simulated web traffic, built decision trees by standard and relaxed versions of C4.5 algorithm include respectively 144 and 5105 test nodes. Testing results confirm better accuracy of compatible decision trees compared to that standard. Furthermore, compatible decision trees remarkably outperform standard trees in terms of new attack detection rate. They are capable to detect 88.51% of new attack instances instead of 6.9% for standard trees [381].

Wu et al. have also adopted decision tree techniques in designing their detection and traceback mechanism of DDOS, distributed DOS. In this class of attacks, intruders are based on two entity types namely handlers and agents to mount their coordinated intrusive actions. They manage multiple intermediary agents or handlers that in turn control an extended number of agents or zombies. The latter serve as supports of DDOS attacking tools. DDOS attacks process consists of two main stages. The control stage focuses on identifying vulnerable hosts on the Internet and determining handler and agent hosts. The attack stage exploits communication links initialized in the latter stage and implements the coordinated attack using zombies' tools indirectly launched by attackers through their handlers.

Detection and traceback mechanism of Wu et al. performs in two steps. On one hand, the detection module identifies malicious traffic generated by DDOS attacking entities. On the other hand, the traceback module reconstructs attacking path based on the spoofed IP address and closest router to the victim. The DDOS detection module uses decision tree classifiers built using features and traffic signature data sets, the preprocessed data packets within a

single minute time interval. Possible leaves of decision tree classifiers include normal and four types of DDOS attacks namely, TCP, SYN, UDP and ICMP flooding. When analyzing preprocessed traffic, DDOS detection module alerts on each intrusive instance the traceback module. The latter triggers data collection of different routers linked to the victim and then traces back the path of mounted DDOS attack.

Carried experiments on detection and traceback mechanism of Wu et al. show promising results of detection as well as traceback module. Using simulated network traffic, the detection module achieves high performance with false positive and negative rates ranging respectively between 1.2% to 2.4% and 2% to 10%. Furthermore, the traceback module misidentifies attack and normal edges of reconstructed paths respectively by about 8% to12% and 12% to 14% [411].

Several other intrusion detection experiments have involved decision tree techniques in their analysis engines. Sinclair et al. have implemented an intrusion detection system, NEDAA (Network Exploitation Detection Analyst Assistant), which uses different rule learners. Decision tree techniques were one of the involved learners. Such learners generate multiple decision trees using sniffed network traffic for normal and anomalous connections. Built decision trees are then transformed into classification rules to be included in NEDAA's expert system. The latter uses learned rules either for connection filtering or anomalous event detection [352]. Lee et al. also have adopted decision tree techniques to built one attack class detectors. Constructed detectors depend on different classes of the DARPA taxonomy. Sangkatsanee et al. as well have based the classification part their real time detection system on decision tree techniques. Conducted tests on one class detectors and real time detection system respectively in [218] and [335] using KDD 99 dataset have shown appropriateness of detection trees to intrusion detection. Additionally, the have illustrated high detection rate and low resource consumption of the decision tree detection model. Bouzida et al. have compared decision tree to neural network detectors. They have confirmed that each detector complements the other and thus their integration within the same analysis engine is possible in order to improve the overall performance of an ids. Makkithaya et al. have experimented C-fuzzy decision tree technique for intrusion detection. C-fuzzy decision tree detectors consist of a decision tree structure in which different nodes are identified using fuzzy C-mean clustering algorithm. Additional details on these decision tree detectors are discussed in [53], [177]. Besides, different experiments have tested and compared decision trees and Bayesian

classifiers in intrusion detection, such as [40], [41], [272]. The next section reviews two main Bayesian classifiers and presents many analysis components based on these.

2.2.1.2 Bayesian classification

Bayesian classification methods are based on the Bayes theorem to predict class membership probabilities of a given data example. Let X a p-dimensional data example and H_j a hypothesis stating that X belongs to an output class C_j, j=1,…,N, the membership probability of X is estimated by the conditional probability $P(H_j/X)$. The latter is the posterior probability that assesses the probability that the hypothesis H holds given the observed data example X. It is estimated using the Bayes theorem as follows:

$$P(H_j/X) = \frac{P(X/H_j)P(H_j)}{P(X)} \quad (2.6)$$

where $P(H_j)$ and $P(X)$ are prior probabilities respectively of the output class C_j and the observed data example X. $P(X/H_j)$ is the posterior probability of the observation X conditioned on the hypothesis H_j. These probabilities are estimated relying on frequencies of observed values in the training set.

Naïve classifier and Belief networks are two Bayesian classification techniques widely applied in intrusion detection. Classification processes and several intrusion analysis mechanisms based on these techniques are presented in the next sections.

2.1.2.2.1 Naïve Bayes classifier

Naïve Bayes classifier is based on the class conditional independence assumption. The latter states that feature values are conditionally independent given the target output class. This assumption simplifies posterior probability computation as follows:

$$P(C_j/X) = \frac{P((x_1,...,x_p)/C_j)P(C_j)}{P(x_1,...,x_p)} = \frac{P(C_j)\prod_{i=1}^{p} P(x_i/C_j)}{\prod_{i=1}^{p} P(x_i)} \quad (2.7)$$

Where X=<x_1 ,…, x_p> a p-valued vector of features respectively f_1,…, f_p and $C_j \in C = \{C_1,...,C_N\}$ the set of possible output classes.

For each processed data example *X*, naïve Bayes classifier estimates posterior probabilities as given by (2.7). Then, it assigns *X* to the most likely output class, C_X, associated with the maximum posterior probability as follows:

$$C_X = \underset{C_j \in C}{arg\,max} \frac{P(C_j)\prod_{i=1}^{p} P(x_i/C_j)}{\prod_{i=1}^{p} P(x_i)} \qquad (2.8)$$

or also, $C_X = \underset{C_j \in C}{arg\,max}\, P(C_j)\prod_{i=1}^{p} P(x_i/C_j)$, because $P(X)$ is constant for all output classes. Prior probabilities of classes, $P(C_j)$, and posterior probabilities of feature values conditioned on classes, $P(x_i/C_j)$, involved in this decision rule are estimated using the training instances.

Naïve Bayes classifier has been adopted by several intrusion detection experiments in implementing corresponding log analysis mechanisms. Panda et al. [289] have designed a network anomaly detection using the naïve Bayes classifier. The proposed Bayesian anomaly detector has been trained and tested using preprocessed network traffic datasets provided by the DARPA benchmark. Testing results show that naïve Bayesian classifier ensures higher detection rates of considered attack classes than the experimented back propagation neural network. Moreover, it is less costly than this in terms of computation time both in the training and testing phases. However, the neural network detector has lower false positive rates, comparatively to Bayesian detector, except for probe attacks.

Farid et al. also have designed two learning algorithms for mining network traffic and detecting traced attacks. These algorithms are inspired by the naïve Bayes classifier process. Furthermore, they use additional machine learning techniques namely decision tree and clustering to build improved detection models that appropriately represent and detect normal and intrusive patterns [113], [114].

The first algorithm proposed by Farid et al [113] extends the learning process of the naïve Bayes classifier by a data splitting phase as performed in decision tree learning. The second learning algorithm instead includes a distance based clustering step [114]. Several experiments have been performed to evaluate the two learning algorithms using preprocessed DARPA data sets. Different subsets of features have been involved in these experiments. Using a set of 19 features, testing results of these two learning algorithms illustrate their high accuracies that exceed 99%. Moreover, they show that both algorithms outperform naïve

Bayes classifier in terms of detection rates. However, specifically the algorithm including a splitting step achieves lower false positive rate than the naïve Bayes.

Muda et al. [264] have also adopted the k-means clustering and naïve Bayes classifier in their two stage intrusion detection process, KMNB. The first stage of the process focuses on clustering training data to determine groups of similar instances. However, the number of discovered clusters in this stage is limited to three representing respectively normal, DOS and other attacks instances. For each of these clusters, the second stage of KMNB estimates prior and posterior probabilities relying on the learning process of the naïve Bayes classifier. It aims at and deriving more specific groups regarding included output classes in a cluster.

Multiple experiments have been carried out to test and compare naïve Bayes classifier and KMNB process. They have been based upon training and testing data sets extracted from the preprocessed DARPA traffic log data. Experimental results show that KMNB ensures 99.89% and 0.41% respectively as detection and false alarm rates. K-means clustering in KMNB increases its detection rate by nearly 2% compared to the naïve Bayes classifier. Moreover, it reduces the false alarm rate of KMNB nearly by 7% comparatively to this. Reached results validate those of Farid et al. experiments in that the naïve Bayes classifier is more effective when trained using a data partition than the whole training set.

More recent experiments have been conducted by Sharma et al. to evaluate and compare their naïve Bayes based multilayer detection approach to decision tree based ids. In this approach Sharma et al. have proposed three layers of naïve Bayes classifiers. In each layer, the corresponding classifier uses a reduced feature set to recognize fixed attack types. Every log data instance is sequentially processed by classifiers of the three layers to identify its true type including DOS, probe, R2L or U2R attack. Results of carried experiments on layered approach and decision trees based ids show slight dominance of the latter in terms of detection rates of DOS, probe and R2L attacks on the former. However, the former outperforms decision trees based ids both in terms of detection rate of U2R attacks and time taken to build detection models [346].

2.1.2.2.2 Belief networks

Bayesian belief networks encode causal relationships between variables and represent their joint probability distribution [178], [293]. They assume that class conditional independence hypothesis applies to subsets of variables but not to all of them, as supposed by the naïve Bayes classifier. A belief network consists of two components namely graphical and

numerical. The graphical component is a directed acyclic graph. Each of its vertices corresponds to a discrete or continuous random variable. The latter may be observed or hidden. Observed variables like protocol type or service attributes of network traffic logs are evaluated for each processed data example. Hidden variables may correspond to features to be predicted such as the degree of intrusiveness or the output class of the given log data example. Each edge in the graph encodes a probabilistic dependence between a node and its descendant. A node is conditionally independent of all of its non-descendants, given its parents. The numerical component concerns conditional probability tables, each of which determines the conditional probability distribution of a variable given its parents.

The belief network learning process is typically performed in two steps. In the case where the network structure is unknown, the first step identifies relationships between variables and builds the graphical structure. The last step estimates conditional probability tables of involved variables. Generated belief network will serve then to classify given data examples relying on Bayes conditional probability rule given by (2.6).

Ben Amor et al. have experimented a simple type of belief networks called naïve Bayesian network in detecting network attacks. The naïve Bayesian network consists of two layers of nodes. It includes a single parent node in one layer and all its descendants in the other layer. The parent node represents the only hidden variable to be predicted and thus it usually corresponds to the output class of the processed data instance. Its child nodes in the other layer represent all observed variables of the network and correspond to considered features of given data examples. Complete independence between child nodes in the context of their parent is also assumed for the naïve Bayesian network.

A series of experiments have been conducted on the naïve Bayesian network using the DARPA preprocessed datasets. They aim at validating several hypotheses that mainly concern densities estimation, discretization of continuous feature and considered attacks granularities. The main findings of these experiments state that normality hypothesis does not apply to all continuous log features thus more generalized density estimation method increases accuracy of belief network based detector by more than 12%. Moreover, discretization of a subset of continuous features associated with finite values has slightly improved the accuracy of the detector. These two results are validated by experiments involving high granularity, 38 attack types or 5 attack classes of DARPA taxonomy, but not those using low granularity of attacks, specifically, two classes based experiments. However, five classes based belief network detector adopting generalized density estimation and discretization methods for continuous

attributes ensures the highest accuracy that exceeds 92%. Detailed results and complete findings of carried experiments are given in [41].

Additional experiments have been performed by Ben Amor et al. in [40], [41] to compare detection models based respectively on naïve Bayesian networks and decision trees. Experimental results for different attacks granularities show that both types of detection models have high accuracies that exceed 91%. For all these experiments, decision tree slightly outperforms naïve Bayesian network based model even after discretization of continuous attributes of the selected subset. When compared to the winner of KDD99, these detection models seem complementary rather than competitive. On one hand, the decision tree model is better than the KDD99 winner in detecting DOS and U2R attacks. On the other hand, naïve Bayesian network outperforms the KDD99 winner in identifying R2L and probe attacks.

Additionally, in his thesis, Tabia has tested several variants of belief networks. The main finding of the carried experiments states that these classifiers fail to detect novel attack instances specifically those of U2R and R2L classes. The reason behind this common failure is two-folded as discussed by Tabia. On one hand, classifiers capabilities to handle new unseen behaviors and adapt to the particularities of intrusion detection are insufficient. This is usually due to low probabilities assigned to unseen events of new attack instances and their confusion with those normal. On the other hand, training and testing data sets induce confusion to these classifiers. As an example, in the preprocessed DARPA data sets, R2L and U2R are represented by low proportions in the training set and only new unseen examples of these two attack classes are included in the testing set. Moreover, these datasets include some incoherencies in that the same data instance is duplicated with different labels such as for normal and R2L classes.

To improve belief network detection capabilities of new attacks, Tabia has proposed different enhancing rules for the Bayesian classification process as detailed in [381]. Improved Bayesian network classifiers have been tested using the DARPA and real internet traffic data sets. Results of conducted experiments using DARPA data sets show that detection rates of unseen attacks have increased by more than 50% for novel DOS and R2L attacks and 25% for new U2R attacks. This has remarkable improved new attack detection rate of enhanced Bayesian networks compared to the standard ones to reach 63.14% instead of 5.12%. These improvements are also validated by experiments performed on web traffic. These experiments report more than 90% of increasing in the detection rate of new attacks after applying enhancing rules of belief networks.

Enhancing rules have been also adopted in the tree augmented Bayesian network, TAN, of Tabia [381]. TAN is a variant of the naïve Bayesian network. It explicitly represents dependencies between child nodes or features such that they form a tree structure. For this class of belief networks, Benfarhat et al. have proposed another improvement mechanism based upon expert knowledge. The latter may concern different aspects linked to the attacks, detection models and data sets. In [43], authors particularly focus on attacks or normal instances proportions in testing data sets. This knowledge is encoded into rules and then applied to the classification results of TAN classifier.

Different experiments of extended TAN classifier have been conducted on preprocessed DARPA data sets. The extended version of TAN classifier ensures better performance than the standard one, as reported by results of these experiments. Its accuracy is increased by more than 3% compared to the standard TAN classifier that correctly classifies nearly 93% of normal and anomalous data examples of the testing set. Moreover, this extension of TAN outperforms the Tabia enhanced version even if the reduced DARPA taxonomy is considered.

Multiple other intrusion detection works have been focused on belief networks. Kruegel et al. have adopted Bayesian networks to aggregate outputs of different detection models [202]. Tuba et al have proposed a detection model based on large probabilistic networks. They have discussed a design methodology for these networks. This idiom based methodology thought of network structure as a set of linked fragments each of which can be built using predefined templates or idioms. Thus, it speeds up and simplifies the construction of large belief networks. Feng et al., as well, have adopted dynamic Bayesian networks, DBN, in predicting goals of system call sequences. State variables included in these networks concern system calls and their goals either normal or anomalous. Markovian property is assumed between system calls such that current system call depends on the previous one and the current goal of the sequence. Moreover, the goal of the current system call sequence is supposed dependent to the initial goal. Probability tables for variables of both classes namely system calls and goals are estimated using the training set. DBN have been also applied in detecting privacy attacks targeting sensitive data. Discussed Bayesian network based detection models and others are detailed in [21], [119], [392].

In a recent work [272], Natesan et al. have experimented ensemble based approach using decision trees and Bayesian classification techniques in intrusion detection. They have shown that decision tree outperforms Bayesian based ensemble, in terms of detection rate, for DOS and Probe attack classes and inversely for U2R and R2L classes. This complies with the main

finding in [40] about the complementarity of decision trees and Bayesian classification techniques in intrusion detection.

Multiple other techniques including Markov models and neural networks have been used to deploy several analysis and detection mechanisms. Following two sections respectively focus on these techniques and their built mechanisms.

2.2.1.4 Hidden Markov model

Markov process is a mathematical model that describes the system dynamic behavior over time. It is a simple model that exhibits dependencies in system behavior in terms of probabilities. It is a stochastic or a random process for which Markov property holds. The latter states that the future behavior of the system is conditionally independent to the past given its present behavior. Thus, Markov process is memoryless and it does not save any previous activities or states because next state of the system is influenced only by the actual one. Moreover, the next state is independent to the manner of how current state was reached.

Markov model can be discrete or continuous time based. Discrete Markov model deals with the system state changes at discrete time points. However, continuous Markov models focus on systems for which state changes occur anywhere in the time. The Markov model can have either a finite or infinite state space. Models with finite state space are called Markov chains. Multiple intrusion detection works have been focused on this detection model [364], [415].

The strict assumption of Markov that the next state depends only on the actual state is not sufficient for modeling different real world processes. It allows a simple model that focuses on system observable states or outputs. To deal with such complex processes, an improved model is required. Moreover, this extension of observable Markov model should concentrate on other inherited aspects in the target system [126], [166].

A two hierarchy level model namely Hidden Markov model (HMM) was proposed to deal with complex stochastic processes. HMM distinguishes between states and their outcomes. In fact, when observations are given to the model, only outputs are observable and states behind these remain hidden. Therefore, it is called hidden Markov model. The theoretical foundation of HMM was developed by Baum and colleagues [36].

HMM is a generative model. It consists of hidden Markov chains of states and sequences of observations generated by each one. HMM is based on a hierarchical structure composed of two levels of states. The upper level is a Markov process with hidden states. The lower level

consists of outputs or observable system signals. The first level process fulfills Markov's assumption. However, the second one supposes that the actual output signal depends upon the current hidden state [192], [194].

Classification process based on HMM models involves two main steps. The training step focuses on generating HMM models. In this step, hidden states are determined and transition probabilities between these are estimated using the training set. Prior probabilities of the identified states are also computed at this step. They determine starting probabilities of the HMM at different states. Furthermore, emission probabilities of observed symbols are estimated at the training step. Each of these corresponds to the probability of generating a given output symbol when HMM is in a given state. The classification step using the HMM involves the learned probabilities. For each processed data sequence, posterior probabilities of an observed sequence conditioned on the computed HMM models are estimated. They correspond to observation probabilities of the processed sequence under given HMM models. Based on the Bayes' rule, this sequence is assigned to the output class of HMM associated with the maximum of observation probability.

HMM have been used in modeling temporal and spatial dependencies in sequential data. Various applications in information security, Bioinformatics and speech processing have employed different types of HMM. In these domains, ergodic and left-to-right HMM types are commonly applied. In fully connected or ergodic HMM, each state should be reached from any other state in a finite number of steps. Left-right or Bakis HMM has different constraints about states and transition between them. It imposes transition to the same state or others with increased indexes when time increases, but not to low index states. Additional constraints can be associated to left-to-right HMM such as the jumping step to avoid large changes in state indexes and reduce possible transitions of each state [73], [194], [312].

In intrusion detection field, initial experimentations of HMM have been conducted by Forrest and his team, after their seminal paper, on self and non self system calls, that has introduced a new type of log data and participated in founding the immunological approach for information security [124]. HMM and other techniques have been also tested by Warrender et al. in discriminating between normal and intrusive sequences of system calls. Fully connected HMM type has been adopted in [405] for modeling normal behaviors of selected programs. Generated program profiles using HMM use fixed numbers of states. Furthermore, they are capable to process system call sequences of variable lengths.

Warrender et al. have trained HMM using normal system call traces of selected programs. Each of these models uses a number of hidden states equivalent to the numbers of unique system calls invocated by the corresponding program. The transition probabilities are evaluated for each of the identified states. Moreover, conditional probabilities of observing an output symbol given a state are also estimated using training sets. Generated HMM are independent to program trace length in training and testing sets. Each of the HMM processes a single trace system call a time. In the testing phase, it is considered as a nondeterministic finite automaton. For any processed system call of the trace, it explores the possible paths including required state transitions and output symbols to generate this call. A mismatching system call is identified when it is generated using transition probabilities less than the prespecified thresholds for normal behavior. It is usually inserted by an intruder in normal traces to implement his attacking objective. Thus, any processed trace supporting a mismatching system call is considered as anomalous by the built HMM.

Testing experiments performed for normal behavior HMM and other models use sets of unseen normal and intrusive traces. Normality thresholds in these experiments range between 0 and .001 for generated HMM. For selected programs, HMM have better performance in terms of true and false positive than other frequency and rule based detection models. Their average true positive rate reaches 99% for less the 0.05% as an average false positive. Furthermore, they are capable to achieve better results when extended numbers of states are included in building HMM, as stated in [405]. However, training such HMM is very expensive in terms of computation cost compared to frequency and rule based models.

Left-to-right HMM with single or two jumping steps and fully connected HMM have been also experimented by Yeung et al. HMM was applied in modeling programs and users' profiles using respectively system calls and shell commands in these experiments. The dynamic profiles built for programs and users have the ability to capture temporal dependencies respectively in invocated system calls and shell commands. Captured dependencies in these profiles ensue the discrimination between intrusive and normal system call or command sequences.

System calls and shell command data sets involved in Yeung et al' experimentations were divided into three subsets. Training and threshold determination data sets concern specifically normal behaviors of programs and system users. Testing data sets instead include normal and intrusive system calls or shell commands. Training sets are preprocessed before building HMM models that concern a single state of normal behavior. For each program, system call

traces of its invocated processes are determined. Typed command within each user session are also structured into tokens and included in the session trace. Then, different sequences of each trace in system call and shell command training sets are extracted using a fixed width moving window with a step size of 1. Resulting training data sequences will serve in building respectively programs and users normal profiles based on ergodic and left-to-right HMM types.Generated temporal profiles of programs and users' normal behaviors are then involved in classifying new traces. A given trace is considered as intrusive if it includes an anomalous sequence. A sequence is abnormal when its observation probability under given HMM is above the computed normality threshold. The latter is evaluated for each threshold determination data set. It is estimated by the mean or minimum value of sequences observation probabilities.

Conducted tests on generated HMM models use testing sets including both normal and intrusive traces. Various experiments of normal behavior HMM of selected Unix programs have tested different lengths of system call sequences. The results of these experiments show that ergodic HMM outperforms left-to-right model in terms of true positive when using sequences of reduced length. Left-to-right HMM require sequences of increased length to achieve high true positive rate. In the experiments focusing on shell commands, both types of HMM models do not appropriately discriminate between normal and intrusive traces specifically for reduced length sequences. Such failure, as stated by Yeung et al., depends upon weak temporal relations between user commands. Thus the static technique such as event frequency allows better detection rate than HMM models in these experiments. When using increased length sequences, performances of both types of HMM models are improved. Ergodic HMM allows a slightly better detection rate than left-to-right HMM, but the latter is less expensive in terms of computation time [421].

Standard HMM has been also experimented in network intrusion detection. Ariu et al have proposed a multiple classifier system based on HMM called HMMPayl in order to detect web attacks specifically those manifesting through HTTP service. HMMPayl processes logged HTTP payload, which summarizes invocated HTTP requests, in three steps. Feature selection step focuses on preprocessing HTTP payload and formatting them into fixed length sequences. A moving window with a fixed width was adopted in [24] to extract payload sequences. Preprocessed normal HTTP payloads only are involved in training HMM for the multiple classifier system. Built HMM include the same number of hidden states that corresponds to the length of processed sequences. Observed symbols for these HMM are

identified from produced sequences. Moreover, transition and emission probability matrices are randomly initialized when training different HMM of the multiple classifier system. The latter includes five HMM according to Ariu et al. settings. All the HMM are provided with the same input sequences. However, they use different transition and emission matrices.

The pattern analysis step next aims at evaluating emission probabilities of each processed payload trace using HMM. In this step, each HMM processes extracted sequences of a given trace. The emission probability of the trace for that HMM is estimated by the mean value of output probabilities of its different sequences. Multiple classifier system of Ariu et al. determines five emission probabilities regarding its HMM for each processed payload trace. In the classification step, emission probabilities of a single trace are fused using the minimum or maximum method. Then, the resulting probability decides whether the processed payload is normal or intrusive depending on the predefined normality threshold. True and false positive rates are involved in the threshold determination [24].

Testing experiments aiming at evaluating HMM based multiple classifier system of Ariu et al. use simulated and real traffic for normal and anomalous HTTP requests. Their normality thresholds are fixed such that the false positive rate is initialized to either 0.1% or 1%. Experimental results for DARPA data set show that HMM based multi-classifier system is capable to detect more than 98% of intrusive HTTP requests. For the same false positive rate, HMM ensure also high true positive rate for real traffic data sets that exceeds 86%. However, for a false alarm rate of 0.1%, the multiple HMM system achieves detection rates exceeding 94% and 77% respectively for simulated and real test sets [25].

Several other intrusion detection experiments have been focused on testing detection capabilities of HMM. Huang has proposed in his thesis the profile HMM, PHMM, for modeling users normal behaviors. PHMM combines HMM and sequence alignment techniques to construct a normal user profile based on sequences of typed commands. Hu et al. have proposed an improved training scheme for standard HMM. This scheme is based on partitioning long sequences of system calls in order to speed up the training phase of HMM. Sultana et al. instead have based their intrusion detection experiments on large frequent sequences. Similarly to Huang, they have illustrated remarkable enhancement of this on building time of their improved HMM, I-HMM, based ids. The latter ensures also near same detection performance to ids based on the standard HMM. All these experiments and the designed HMM based detection mechanisms are detailed in [163], [166], [168], [377].

2.2.1.5 Artificial neural networks

Inspired by the human brain, scientists such as Minisky and Papert, Rumelhart and McClelland, Hopfield and others have contributed to the theoretical foundation of Artificial neural networks, ANN [47], [261]. ANN is a powerful technique for modeling complex relationships between input and output data. It consists of a network of computational units that implement a mapping function to approximate the desired output relying on a training data set. The network units or neurons are highly interconnected. Each unit receives inputs to compute its activation and feeds a single output to other neurons that perform the same task. The connections between all processing units are weighted. These weights are updated from iteration to another to adapt the network to desired outputs [278], [298], [342].

In neural network, processing units are organized into layers. The input layer is the first layer of the network structure. Neurons in this layer don't perform any task rather than feeding input data to neurons of another layer. Generally, the number of neurons in this layer depends on the dimensionality of the processed data set. The ANN structure disposes of a single input layer which is connected to the first hidden layer of neurons and may be to others, in specific architectures such as Recurrent Neural Network, RNN. A neural network may support single or more hidden layers. Neurons of these layers process input data and then forward their activations to processing units of the next hidden or the output layer. The last is the final neuron layer in the network structure. It returns the decision of the network to the given problem. Its neurons may be connected to input neurons or those of the last hidden layer. Their number is fixed depending upon the treated problem. A single or multiple neurons form the output layer when dealing respectively with function prediction or classification problems.

ANN are extensively experimented in multiple real problems including information security. They have been involved in building useful and highly adaptive models of user or system behavior relying on incomplete or even noisy data. Thus, they are widely applied in intrusion detection systems where experimented attackers can sometimes alter log files to hide their traces, [298], [333], [344]. Neural networks have additional qualities that offer the potential to resolve different problems encountered by other approaches in the intrusion detection field [5], [63], [298]. They have the ability to generalize from incomplete data. Moreover, their input data are free from any statistical assumption. Furthermore, they have other advantages such as improved analysis capabilities and the possibility to update constructed models, when retrained. In spite of these qualities, ANN are appealing black boxes by their results. They allow no explanation on how results are reached. Moreover, they have another drawback

namely parameters setting that increases their application cost in intrusion detection and other fields. Some critical parameters for the neural network model should be empirically estimated and optimized. This requires additional costs in terms of computation as well as data availability [5], [133], [203], [298], [342].

Neural networks are categorized into feed-forward and recurrent networks depending on connections and units arrangement within the network. MLP (Multi Layer Perceptron) and Elman are among the most known respectively feed-forward and recurrent networks. Feed-forward neural networks allow single direction signal or activation flow. In RNN, one or many neurons in a hidden layer wait for inputs from other neurons in the same or other layers. The internal connection feedback in the RNN implements the effect of previous decisions on the current outputs of neurons [15]. In this section, several ids supporting supervised ANN based analysis engines are summarized. Subsequent sections in the current chapter will focus on unsupervised networks.

Multiple neural network topologies have been also experimented either in modeling attacks signatures or normal behavior patterns or both. In their neural network intrusion detector, NNID, Ryan et al. have applied a two layer MLP network to identify legitimate users based on the distribution of their executed commands. NNID uses user vectors that correspond to collected statistics about user commands over a period of time. Then, it tries to recognize the distribution of commands as normal or intrusive. It doesn't take into account neither command orders nor command arguments. Only the set of commands and their frequencies are considered in computing user vectors. The two layer back propagation MLP experimented by Ryan et al. includes 100 input units. Its single hidden layer supports 30 neurons. However, the number of units in the output layer depends on the number of users in the training set.

Testing results of the NNID on reported sessions of 10 users show its high accuracy. NNID achieves 96% and 7% respectively as detection and false alarm rates. Furthermore, by focusing on command distribution, NNID allows more flexibility to intrusion detection system and preserves user privacy. However, it may generate increased false alarms, specifically, when dealing with an extended number of users. In this case, the discrimination between user sessions becomes difficult and requires numerous relevant features [329].

Ghosh et al. as well have adopted standard feed-forward networks in their misuse and anomaly intrusion detection system. The hybrid ids designed by Ghosh et al. relies on MLP networks in modeling processes behaviors both in normal and under attack situations. In

addition to the detection of known attacks, it aims at classifying online data and taking advantages of the generalization capabilities of ANN to recognize future unseen attacks.

Built neural nets for each process use its preprocessed sequences of system calls. Ghosh et al. have adopted distances instead of simple enumeration of extracted sequences as inputs to different neural nets [133]. For each process, its normal or intrusive traces are formatted into sequences of six consecutive system calls. Afterwards, the distance between each extracted and one of the reference sequences is evaluated using the devised distance metric. The latter concerns common system calls and their positions within the two sequences. Computed distances will serve as input to different neural nets that include input units similar in number to available reference sequences. Output layers in these networks support a single neuron. Moreover, a single hidden layer is included in each ANN. The number of hidden units in this layer is variable and determined for each ANN depending upon its testing performance.

Conducted tests on hybrid ids use different normal and intrusive data sets extracted from the DARPA 98 testbed. The misuse detection system is capable to achieve a detection rate exceeding 90% but with high false positive rate of 18.7%. The anomaly detection system instead ensures more than 80% as a detection rate for less than 8% of false positive. The main findings of these experiments confirm high sensibility of the misuse detection system to slight changes in attack signatures. Additionally, the anomaly detection system can generalize from learned patterns and hence it is capable to recognize new unseen attacks. However, its generated false alarm rate remains unacceptable specifically for commercial systems [133].

Ghosh et al. have also tested different machine learning techniques including feed-forward backpropagation and Elman recurrent networks [134]. In performed tests, MLP and Elman based program normal profiles are generated using data sets extracted from the DARPA evaluation program. The results of these tests show higher performance of the Elman nets comparatively to the MLP. MLP based anomaly detection recognizes 77.3% of intrusions with 2.2% of false positive rate. Elman networks reach the same accuracy level with no false alarm. Furthermore, they are capable to achieve total detection with near 9% of false positive rate. This illustrates promising capabilities of the Elman networks in detecting unseen attacks as well as reducing false positive rates of anomaly detection systems [134], [263].

Similarly to Ghosh et al., Alarcon-Aquino et al. have conducted a study on standard feed-forward and two types of recurrent networks, namely Elman and fully connected RNN. This study aims at evaluating performances of ANN based detection models in detecting HTTP attacks. The designed and implemented tests of feed-forward, Elman and fully RNN based

detection models use training and testing datasets of normal and anomalous web application queries. They are preprocessed and the resulting fixed length binary sequences are taken as inputs to different neural nets. All experimented neural nets have input layers of 64 units. Multilayer feed-forward detection model includes two hidden layers of 15 neurons and an output layer of 5 units. Elman network has the same architecture as the feed-forward network. It includes a similar number of units in each layer except hidden layers in which neurons are doubled compared to those in the chosen feed-forward net. Fully connected RNN has a similar structure to Elman network with a single hidden layer.

At the training phase, fully RNN was the least costly in terms of epochs or training iterations to satisfy the stopping criterion. It reaches the predefined training error of .015 for about 75 epochs. Three layer feed-forward and Elman networks require respectively more than 200 and 100 iterations to fulfill the same goal. Furthermore, fully RNN outperforms the two other neural nets. It ensures high accuracy that exceeds 94%. Moreover, it preserves low false negative and positive rate of respectively about 0.87% and 4.37% [15], [344].

Previously presented intrusion detection experiments solely focus on supervised neural nets based analysis components. Multiple others such as those of Bivens et al., Canady and Mahaffey and Sheikhan and Jadidi, have proposed the combination of neural nets and supervised or unsupervised machine learning techniques. Bivens et al. have designed a modular network intrusion detection system based on neural networks, NIDNN, which analyzes network traffic (tcpdump data) to develop windowed traffic intensity trends. They have selected both supervised and unsupervised neural networks respectively for preprocessing and analysis components of their system. NIDNN system monitors traffic of selected set of ports for different network sources or hosts. It preprocesses network traffic by sources with respect to the set of ports and processing time intervals. Self organizing maps, SOM, an unsupervised network, are then used to cluster sources in order to group hosts with similar traffic intensity trends together, section §2.2.2.4 details SOM based clustering. The number of clusters or source groups is constant, initially established at the training phase of SOM. After this step of behavior-based clustering, traffic intensities of each group are normalized and then given to the analysis component of NIDNN system. An MLP analyses pre-processed data within each time window and decides which group has attacked the victim, if an attack was detected [50].

In their preliminary experimentations of the hybrid misuse detection system [62], Cannady and Mahaffey have adopted the same neural net combination as in [50]. In these experiments,

network log data is structured into sequences with respect to a predefined time window. The events of each sequence are clustered based on SOM derived groups. Afterwards, the outputs of the clustering step are fed to the input layer of the MLP based analysis component. In this component, clustering decisions are fused to label the processed sequence as normal or attack.

Conducted tests by Cannady and Mahaffey on the hybrid misuse system use simulated FTP attacks. The results of these preliminary tests show that hybrid neural nets are capable to detect complex known attacks as stated in [62]. Bivens et al. experiments of the NIDNN system have been based on the DARPA data sets. Two tcpdump test sets have been designed for these experiments. In addition to normal instances, they respectively include examples of all attacks and single DOS attack. NIDNN is able to perfectly discriminate normal and DOS instances of the second test set with no false alarm. However, it achieves an unacceptable false positive rate that reaches 76% for the first test set [50].

Sheikhan and Jadidi have also proposed a hybrid detection model that combines association rules and supervised neural nets. Association rules are one of widely adopted data mining techniques to capture relationships between attribute values as explained further in section §2.2.3.1. The integration of these two techniques within the same analysis component aims at taking advantage of their capabilities in order to improve the detection rate of the hybrid misuse detection system, specifically for complex attacks such as U2R and R2L. This system is based on a three step process to generate signatures of different attacks and normal classes. The first step of feature selection identifies a reduced set of relevant log attributes to be involved in the association rules generation. Feature selection step is performed on a subset of training data that concerns U2R and R2L attacks. The second step focuses on rules generation. In this step, the training subset of U2R and R2L is reduced, according the selected features, and then preprocessed. Afterwards, association rules that concern both classes, U2R and R2L, are generated using the resulting dataset. In the last step, an MLP network is built using preprocessed training set that concerns classes of the DARPA reduced taxonomy. The trained MLP includes 41, 35 and 5 units respectively in its input, hidden and output layers.

Built MLP network and generated rules are saved for classifying further traffic data examples. The hybrid detector of Sheikhan and Jadidi initially classifies each data instance using the trained MLP. Then, the rule based classifier is triggered depending upon the outputs of the MLP network. It is involved in making precise decisions about processed data examples that were previously labeled as U2R or R2L by the MLP network. Conducted tests of the hybrid use training and testing datasets extracted from the DARPA testbed. They show that the MLP

and hybrid detection models have comparable performances in terms detection and false alarm rates. However, the hybrid detection model is definitely better than MLP network in detecting U2R and R2L attack instances. Additionally, these two attack classes are the least represented in the testing set, therefore their detection rates have almost no effect on the overall performance of the hybrid detection model [347].

Several works have surveyed previous intrusion detection experiments involving different types of neural networks. Mousa [263] has presented a survey of research efforts in intrusion detection based on neural nets. Different network or host or even application based misuse and anomaly detection systems using supervised or unsupervised neural nets are discussed in this survey. More recent reviews have been presented by Ahmed et al. and Shah et al. on ANN approaches to intrusion detection [6], [344]. Additionally, the authors in [6], [7] have developed a thorough empirical study on different supervised and unsupervised neural networks aiming at raking them according to numerous criteria including cost and detection rate. Ravi Kiran et al. have adopted unsupervised learning technique in selecting log features to supervised neural net based ids [320]. Gu et al. have discussed different neural nets architectures and how integrating them with the genetic algorithms to achieve intelligent intrusion recognition [144]. Mukkamala et al. have also conducted a detailed comparison between multi-layer feed-forward networks and support vector machine, SVM, in the intrusion detection field [266]. The SVM technique was adopted in implementing several analysis mechanisms. The following section discusses some of these.

2.2.1.6 Support Vector Machine

Support Vector Machine (SVM) is a binary classification technique proposed by Vapnik and colleagues [52]. It is based on geometrical interpretation of classification problem where the objective is the definition of the hyperplane with the maximum margin that separates instances of two classes. SVM classifiers are based on mapping functions in order to transform original data space into high dimensional space. Within the resulting space, they search for the optimal linear hyperplane separating the two classes. The optimal hyperplane is determined by a small fraction of the training instances, associated with both classes and referred to as support vectors. Furthermore, it leaves the maximum margin between support vectors. SVM is known as the optimal margin classifier which has improved classic linear classifier based on the idea of margin maximization. It transforms the classification problem into an optimization problem which aims at finding a useful tradeoff between margin

maximization and classifier performance [61]. A comprehensive and detailed presentation of the theoretical foundation as well application fields of the SVM can be found in [61], [399].

The last decade, SVM has gained increased attention owing to its speed, scalability and generalization capacity. It is becoming one of extensively experimented machine learning techniques in multiple domains for many reasons. SVM speed makes it suitable for real time system. An SVM classifier requires a reduced training time, compared to other error minimization techniques, and allows an online classification of new incoming examples. Additionally, SVM scales better than other machine learning techniques such as neural networks when dealing with highly multidimensional space where weight estimations become difficult and distribution of an extend number of training instances is unavailable. SVM is also able to cope with symbolic features and noisy data. Moreover, it has shown promising capabilities including high generalization capacity in multiple fields [61], [265], [266], [331].

Mukkamala et al. have implemented the analysis engine of their intrusion detection system, SVMID, using an SVM classifier. In SVMID, Gaussian radial basis kernel function was adopted for mapping eight feature data vectors to a higher multidimensional space. Mukkamala and colleagues have also appropriately initialized parameters of the SVM classifier in order to severely penalize noisy data vectors and avoid overfitting. SVMID was prototyped using nonlinear binary SVM algorithm of SVM light package [107], [409].

In conducted experiment, Mukkamala et al. have trained SVM classifier using both intrusive and normal traffic data. Anomalous data include different types of attacks. Over carried experiments, SVM classifier has achieved a high detection rate on testing data that reaches 94% [265]. In other experiments [266], Mukkamala et al. have performed different tests aiming at comparing ANN and SVM techniques using the DARPA datasets. They have illustrated that SVM and ANN based detection models ensure comparable detection rates. They have also shown that these learning techniques have compatible performance levels with a slight dominance of SVM technique. On one hand, SVM requires shorter training and running time than ANN. On the other hand, ANN is more suitable than SVM when dealing with multi-class problems.

Additionally, Tran et al. have proposed OTAD, a network anomaly detection based on one class SVM. In OTAD, an SVM classifier models network normal behavior patterns. It uses a Gaussian radial basis function in mapping included attributes to high dimensional feature space. Furthermore, it is built using Tcpstat computed statistics from collected network traffic data. Tcpsat is a network monitor tool. It reports in online or offline manner different statistics

on network interfaces activities using sniffed traffic. In OTAD, the traffic data set is structured into subsets regarding a predefined time window. Then, Tcpstat preprocesses resulting subsets and forwards evaluated traffic statistics to the SVM based analysis engine. A subset of Tcpsat reported statistics, including percentages of TCP, UDP and ICMP packets, is considered in building normal behavior detection model of OTAD. Conducted tests of OTAD have considered different time windows in structuring network traffic log of the DARPA datasets. Best performance of OTAD is reached using a time window of 300 seconds. For such time window, OTAD ensures an accuracy of 71% with 10% of false alarms [389].

The binary SVM classifier was also adopted by Heller et al. of Colombia University to implement their Windows registry anomaly detection. The SVM based intrusion detection system was developed to detect abnormal access to Windows registry database. Training and testing data sets for this ids correspond to preprocessed registry queries collected using Windows NT system. Each query concerns a single registry in the database. It is a valued vector that includes the invocating process, the result and the success status of the query and other components.

SVM based Windows registry detection system was tested using different mapping or kernel functions such as linear and Gaussian kernel functions. An SVM classifier with linear kernel has shown the best accuracy level comparatively to other mapping functions. However, it allows a low detection rate when compared to the probabilistic anomaly detection algorithm of Eskin [110]. The latter algorithm accurately detects abnormal access to Windows registry with less than 10% of false alarms. It uses required discriminative information and consistency checks as discussed in [110], [159]. As stated by Heller et al., such information could be integrated into a strong kernel function to improve accuracies of SVM detection models [159]. Different kernel functions have been tested in intrusion detection field including linear [294], polynomial [382] and Markov kernels [422].

Previously presented detection models are based on binary SVM classifiers, which are not applicable in multiclass classification problem unless one of the proposed extension approaches is adopted. Two main approaches have been widely discussed in the literature to extend the SVM algorithm to the multiclass version. The first approach is based on breaking the multiclass problem into several binary problems. The second approach instead considers all classes at once. It proposes a generalization of two-class SVM to the multiclass case where all classes are simultaneously considered [72], [226], [253], [331]. The latter approach is simple to implement. However, the former is commonly adopted in different domains.

Multiple researches have been conducted in order to evaluate capabilities of multiclass SVM in intrusion detection. Xu in [413] has adopted the multiclass SVM technique to implement the classifier construction component of his proposed intrusion detection framework. In this component, multiclass SVM is based on the decomposition approach. Built multiclass SVM classifier uses the radial basis mapping function. Furthermore, it concerns normal and intrusive network traffic.

Initial experimentations of the multiclass SVM detection model of Xu have been conducted on small fractions, about 2% and 10%, respectively of the KDD 99 training and testing sets. The experimental results illustrate promising capabilities of the multiclass SVM classifiers in intrusion detection. The tested classifier ensures comparable detection rates to the KDD winner in identifying normal traffic and probe attack instances. Moreover, it outperforms the KDD winner. It increased detection rates of U2R and R2L attacks respectively by more than 8% and 2%, comparatively to the KDD winner. However, it requires an increased training set to achieve a better detection rate of DOS attacks than this [299]. Additional improvements including feature selection are also needed in order to boost the performances of multiclass SVM classifiers, as discussed in [413].

In a recent work, Mewada et al. have experimented several multiclass SVM algorithms based on the decomposition approach. In their network based ids, generalized multiclass SVM detection models are capable to recognize normal and anomalous traffic instances, according to the reduced DARPA taxonomy. They were trained and tested using network traffic logs of the DARPA 99 dataset. Several kernel functions have been tried in building and testing multiclass SVM detection models. Overall performed tests, radial basis function based multiclass SVM classifiers ensure high detection rate exceeding 90% compared to other kernels. Additionally, with the same kernel, the multiclass SVM outperforms one class SVM classifier adopted in OTAD of Tran et al. [258], [389].

Several other works have integrated different machine learning techniques with multiclass or single class SVM within the same detection mechanism. Peddabachigari et al. [294] have designed a hybrid ids by integrating one class SVM and decision tree techniques. The same idea was recently explored by Mulay et al. in designing their misuse and anomaly detection systems [267]. Yu et al. have also proposed a hierarchical detection model that combines single and multiclass SVM classifiers [423]. This model is capable to detect instances of flooding attacks, a subclass of DOS attacks, in two steps. In the first step, one class SVM classifier discriminates between normal and intrusive traffic. In the last step, multiclass SVM

classifier labels flooding instances based on their communication protocol into TCP, UDP or ICMP flooding. All these integrated analysis engines and others are detailed in [127], [267], [294], [420], [423]. Additional other works such as [193], [339], [382] have integrated SVM with clustering techniques in designing their detection engines. More recent ones have proposed a new technique of multilevel SVM that dynamically determine support vectors of the separating hyperplane and compared this to the SVM technique using KDD 99 data [2].

Similarly to risk minimization, decision trees and Bayesian techniques, instance based classification has been experimented in the intrusion detection field. The following section reviews multiple intrusion detection experiments involving nearest neighbor classifiers in their analysis components.

2.2.1.7 Nearest neighbor

K-Nearest-Neighbors (k-NN) is an instance based classification method. It is widely applied in the non parametric classification. In k-NN classification, each output class is represented by instances of its training data set, which correspond to the memory of the designed classifier. A given data instance is assigned by the k-NN classifier to a class associated with the k closest neighbors to this. Closest neighbors are identified among those of the classifier memory using a similarity measure. The latter is commonly defined in terms of distance metric including the Euclidean distance [155]. K-NN classifiers using Euclidian distances or other similarity functions have been extensively adopted in different domains such as information security.

Inspired by previous experiments of text categorization methods, Liao et al [230] have designed their k-NN based anomaly detection model. The latter aims at classifying session processes as normal or anomalous using their invocated system calls. At the training phase, data sets that consist of normal sequences of system call are preprocessed to derive characteristic vectors of executed processes. A process vector summarizes given sequence of system calls. It consists of several weights each of which concerns one of selected system calls for the whole training set. A system call weight is estimated using different parameters including system call frequencies within the sequence and the overall training set. Vectors of weights are then involved in classifying further system call sequences relying on an improved k-NN classification process. This process includes three steps. In the first step, each processed sequence including system call not among those selected is considered as anomalous. The second step labels the given sequence as normal if its similarity to the closest training process vector is equal to 1.0. The last step determines the average similarity between the processed

instance and its *k* nearest neighbors. Then it assigns given data instance to normal or attack classes depending on whether its average similarity exceeds a prespecified threshold or not.. These classification steps are performed for any processed data example. Different tests of the k-NN detection model use datasets of the DARPA 98. For a threshold of .87 and 5 neighbors, this model is capable to recognize more than 90% of normal and intrusive sequences. Moreover, it ensures .082% of false positive.

Optimizing k-NN classifier has been also investigated by Toro-Negro et al. in designing their misuse detection system. The authors have proposed a genetic learning process to train k-NN detector. In this process, the training phase of the k-NN classifier was formulated as an optimization problem. A candidate solution to this problem is a weight vector that assesses the relevance of features to k-NN classification task. The best solution corresponds to a weight vector associated with the highest classification accuracy of k-NN classifier at the training phase. It is saved for classifying further processed data instances.

Preliminary tests conducted on designed misuse detection system use data sets of DOS attack extracted from the DARPA database. The evolutionary learning process was initialized to 400 iterations with 50 individuals and .5 as mutation probability. Classification accuracy of the optimized k-NN detector in these experiments ranges between 95% and 99%. Moreover, candidate weight vectors state that five features out of 38 included in specifying data examples are the most relevant in classifying DOS instances. Complete details on genetic algorithm settings and conducted experiments are given in [387].

Additionally, Ghasemzadeh et al. have designed a hybrid detection model that combines rule induction and k-NN. Two main implementation alternatives of the hybrid detection model have been discussed in [132]. In the first alternative, the selected rule learner is applied to the given training set and induced rules are then encoded into valued vectors. After that, the original training set is preprocessed and the resulting set includes only features of rules vectors. The latter alternative merely extends the original training set by valued vectors of the induced rules. Data sets generated in both cases will serve as training sets to the k-NN classifiers. Both implementations of the hybrid detection model aim at decreasing the sensitivity of k-NN classifiers to noise based on induced rules. In the first case noisy instances are filtered based on feature vectors of high confidence rules. Whereas the influence of noisy examples is reduced in the second case relying on high prediction capabilities of rules features involved in the extended training set.

Hybrid detection model of Ghasemzadeh et al. has been tested with its two prototypes RBR and ABR respectively for the above discussed implementation alternatives. Training and testing data sets for these prototypes and included classifiers are extracted from the preprocessed network traffic of DARPA testbed. In these experiments, ABR, RBR and other classifiers ensure comparable true positive rates exceeding 99%. However, rule based classifier outperforms all prototyped detection models in terms of detection rate.

Nguyen et al. have adopted k-NN method for the designed detection component of their anti-DDOS framework. This component aims at achieving earlier detection of DDOS attacks by classifying the network status. Based on the main two stages of DDOS attacks namely control and attack as stated in section §2.2.1.1, Nguyen et al. have identified three classes of network status involved in the earlier detection of DDOS attacks. Therefore, each processed connection is labeled as a pre-attack, attack or normal.

The designed k-NN based model to classify network status includes feature selection and normalization steps. In the first steps, most relevant features to predict DDOS attacks are selected and then the training set is reduced according to these. Afterwards, attributes' values in the resulting data set are normalized to reduce differences between feature scales as stated in [275]. The reduced and normalized training set includes network sessions associated with the three distinguished classes. Any given network session is assigned by the k-NN model to one of the output classes based upon the majority of its k closest neighbors. Experiments carried on the earlier DDOS detection model use datasets of the DARPA 2000 that focuses on coordinated attacks. They have shown that this model is capable to identify more than 91 % of normal and intrusive sessions. However, the misclassification rate of this model exceeds 10% specifically for pre-attack sessions.

Additional experiments integrating k-NN with other machine learning techniques within the same detection engine are discussed in [270]. Some others involving k-NN classifiers in reducing training sets and eliminating outliers are presented in [132], [374]. Moreover, k-NN classifiers have been applied by Ho [161] in filtering ids alarms and identifying normal and suspicious ones. Other experiments aim at exploring the usefulness of feature selection, max and min decision rules and the integration of clustering to k-NN detection models are also detailed in [350], [352].

In addition to the reviewed supervised learning techniques, several clustering algorithms have been widely applied in intrusion detection. They are mainly involved in preprocessing log

data and implementing the analysis mechanisms of ids. The following section presents most adopted clustering techniques and reviews their experimentations in intrusion detection.

2.2.2 Unsupervised machine learning techniques

Supervised learning techniques have been commonly adopted in misuse and anomaly detection. However, when structured training datasets are not available, unsupervised techniques are required for intrusion detection analysis mechanisms. Clustering techniques are provided with unlabeled training sets. They aim at identifying regularities according to an inner criterion, generally based on a distance or proximity metric, in input datasets. Relying on discovered regularities, different groups of data objects are worked out. A given data object is assigned to one of derived groups only when it shares some common properties with its objects.

Clustering techniques perform into two main steps. The initial step focuses on cluster identification. It consists of grouping similar training patterns into different subsets. At this step, exhaustive and mutually exclusive clusters are determined. Their number may be predefined or induced from the training data set. The last step of the clustering process concerns the classification based upon discovered clusters. At this step, labeled clusters from those initially identified are involved in classifying new data observations.

When applying clustering techniques, their associated methods address to two main issues of similarity among patterns and evaluation of discovered clusters. Similarity or dissimilarity between patterns is obviously based on distance metrics. Although multiple distance metrics have been adopted, the Euclidean distance is the most applied in estimating similarity between two patterns for clustering algorithms. Derived groups based on selected similarity metric are evaluated using clustering or partitioning criterion. The latter may correspond to the sum squared error between cluster objects and its mean, maximum distance between cluster objects or any other dependent criterion to clusters and their objects. It aims at optimizing cluster compactness and overlapping.

Clustering methods can be classified into several categories. However, in the intrusion detection field, mainly methods of four categories, namely partitioning, density based, hierarchical and model based clustering, are the most applied. Partitional algorithms start with an initial number of partitions or clusters either predetermined or automatically derived from a given data set. Then, they iteratively refine determined clusters to meet objectives associated with the selected optimization criterion. Density based clustering algorithms determine dense

regions of the data space that correspond to its clusters separated by low density components or outliers. Hierarchical clustering algorithms derive hierarchies of clusters represented as a tree structure called dendrogram. A dendrogram describes a hierarchical decomposition of given data patterns based on their similarities. At different levels of the hierarchy, branches of each cluster identify its sub clusters, except the last one. At the lowest level of the hierarchy, each cluster is represented by similar data objects. The last category of model based clustering supposes that each cluster is generated based on a mathematical model. It attempts to optimize the fit between data objects and this model. Statistical and neural network based clustering that use respectively Expectation maximization and self organizing maps are examples of model based methods [105], [155], [410].

2.2.2.1 Partitional clustering

Partitional clustering methods have been adopted in various real life applications, including intrusion detection. K-means is the most widely applied distance based partitional clustering algorithm in misuse and anomaly detection. It is mainly based on three steps to determine clusters of a given dataset. Initially, it randomly chooses *k* cluster centers from the dataset. Then, it assigns a given data instance to the closest centroid depending upon evaluated distance measures. Finally, it replaces each cluster's centroid by the mean of its members. The last two steps are repeated by the k-means algorithm until a stopping criterion, such as no change for each cluster, is met.

K-means and other clustering algorithms have been tested in the intrusion detection field. Sabhnani et al. have conducted different experiments to evaluate the effectiveness of several machine learning algorithms in network attack detection including two distance based partitional algorithms namely k-means and nearest cluster. The latter is based on k-means to determine initial cluster centers. Furthermore, it adopts the Euclidean distance measure, similarly to k-means, to identify and assign the processed data example to the nearest cluster. Nearest cluster, k-means and other machine learning algorithms have been tested in [330] using KDD datasets. Experimental results presented by Sabhnani et al. show that k-means algorithm outperforms nearest neighbor and other supervised and unsupervised machine learning algorithms specifically in detecting denial of service and privilege elevation attacks.

In their defensive and offensive mechanism against distributed denial of service attack, Yu et al. have adopted k-means clustering to implement the defensive part. Based on collected application log data, k-means clustering will build client normal profile that will serve in

classifying future logged session. Suspicious sessions detected based on generated profiles are then dropped depending upon their trust values computed at the classification step [424].

Munz et al. have proposed the Network Data Mining (NDM) approach for flow based anomaly detection. This approach is based on the k-means clustering to process and classify collected network flow records. The latter are first structured based upon prespecified network services into several datasets. However, those associated with non included services form three other data sets of TCP, UDP and ICMP protocols. Constructed services logs are next preprocessed with respect to equally spaced time intervals to derive different data sets. K-means algorithm is then applied to each of these data sets. Computed cluster centroids will classify logged flow records based on their (protocol, port) pairs and using a distance measure. Initial experiments of NDM approach using locally generated and real traffic data prove its feasibility. Moreover, they have shown promising detection results of the k-means clustering and thus it may be useful for real time intrusion detection as stated in [268].

Nieves has applied the k-means algorithm to detect different attack classes. In the designed experiments, he has preprocessed training sets extracted from the DARPA experimental sets in which categorical and continuous features were respectively encoded and normalized. K-means algorithm was tested for different values of the parameter k using preprocessed data sets. Illustrated results confirm the appropriateness of this clustering algorithm in identifying different classes of network attacks. However, determining the right number of clusters to the k-means algorithm is also critical to reach good detection and false alarm rates [277].

Multiple extensions have been proposed to improve the k-means algorithm [108], [145], [179], [239]. They specifically focus on two main shortcomings of the k-means namely the number of clusters and the empty clusters. These two weaknesses are known respectively as the number of clusters dependency and degeneracy problems [145]. The first problem concerns how determining the appropriate number of clusters, the parameter k. The second problem is about how eliminating empty clusters among those discovered by the k-means. These two problems have been resolved by Y-means clustering algorithm, an extension of the k-means proposed by Guan et al.. Y-means clustering partitions the given data set into a predefined number of clusters that ranges between 1 and n, the size of the data set. Then, it replaces any empty cluster, of those identified, by a new one and reassigns normalized data instances to existing clusters. The last two steps are iteratively performed until removing all empty clusters. Afterwards, the appropriate number of clusters is automatically determined by splitting and merging steps of the Y-means algorithm. The latter was tested using DARPA

experimental data. On average, it has automatically identified 20 clusters and reached 86.63% and 1.53% respectively as detection and false alarm rates. Detailed experiments conducted using Y-means are presented in [145]. Additionally, a thorough comparison of Y-means clustering to variants of k-means was recently conducted in [89]. It takes account different criteria including performance and efficiency.

In their invasion detection system, Jia et al. have proposed an improved version of the k-means algorithm [179]. Two additional parameters have been included in this extension namely the cluster radius and nearest neighbor threshold. These parameters are involved in deciding when new clusters are created and existing ones are merged for the extended version of k-means. Derived clusters of normal or anomalous behavior based on this clustering algorithm will serve next in generating rule for the invasion system. Preliminary tests of the improved version of the k-means show that the number of discovered clusters of DOS attacks depends upon the fixed nearest neighbor threshold and cluster radius. Moreover, the number of discovered clusters directly affects the performance of the invasion system. As illustrated by conducted experiments on DOS attacks, a reduced number of large clusters ensure low detection and false alarm rates and inversely for an extended number of small clusters. This stresses the main shortcomings of the proposed extension of the k-means that concern the determination of appropriate nearest neighbor threshold and cluster radius.

K-means and other clustering algorithms have been adopted also in preprocessing log data before applying supervised classification techniques. In their experiments, Chairunnisa et al have used k-means and genetic based clustering to discover network traffic clusters. The derived clusters will serve in generating k-nearest neighbor based detection model to classify collected data instances [71]. Bharti et al. have also adopted the k-means algorithm to filter training and testing traffic log data before building decision tree based detection models [45].

Portony in [307] has proposed a partitional clustering algorithm for his anomaly and misuse detection system. The designed algorithm performs in one pass, differently to k-means that requires multiple passes to determine final cluster centroids. It relies on a single linkage method that assigns each processed data example to its closest cluster. Moreover, it discovers clusters with a fixed width. Initially, Portony's algorithm uses a set of empty clusters. Each normalized data example is assigned to a cluster only if it falls within a predefined radius from the center of this cluster. In the other case, the distances between the given data example and cluster centers exceed the fixed radius, a new cluster is created with the processed instance as a center. Different variants of Portony's algorithm have been tested using various

datasets of the DARPA testbed. Initial results of these tests show the designed ids using different variants is capable to reach near 80% of detection for less than 6% of false positive. Detailed experiments of the Portony's algorithm and their results are given in [307].

2.2.2.2 Density based clustering

Density based clustering overcomes the drawbacks of most partitioning methods by allowing arbitrary shaped clusters instead of spherical shaped ones. Its discovered clusters correspond to dense regions in the data space. Each dense region is a group of data points within a predefined radius and contains at least a minimum number of points. It is extended each time if its data points form new dense regions [155].

DBSCAN is the most widely applied density based clustering algorithm in the intrusion detection field. It requires two input parameters namely the radius and the minimum number of points to determine different classes of data points including core, neighbor and outlier. Based on core points that represent centers of dense regions, it identifies data clusters as maximal sets of density connected points in the data space.

DBSCAN algorithm has been tested in anomaly intrusion detection using DARPA and honypot[1] data sets [286], [387], [363]. Although, it ensures higher detection rate than k-means for DARPA experimental data, its false positive rate is unacceptable comparatively to this [286], [287]. Conducted experiments on honypot data show that DBSCAN and single linkage based algorithm of Portony ensure comparable detection rates. Moreover, for a fixed false positive rate of 10%, both clustering algorithms outperform the k-means clustering [363].

Wang et al. [402] have proposed a framework for adaptive and online detection of web attacks. They have adopted in this framework an extended version of the DBSCAN clustering algorithm. The improved DBSCAN is capable to online learn. It updates the built detection model each time changes in system normal behavior are detected. Moreover, it rebuilds this model if outliers marked as suspicious audit data exceed a certain threshold. The generated model based on the improved version of DBSCAN clustering also ensures online detection of web attacks. It was tested using a large data set of web logs specifically for HTTP service. Reached results show that this adaptive detection method using the online version of DBSCAN is more effective than the static method based on k-nearest-neighbors in terms of

[1] Honypots are net hosts supporting multiple vulnerabilities to attract attackers

the detection and false alarm rates as well the CPU processing time. Complete results of conducted experiments and their details are presented in [402]

2.2.2.3 Hierarchical clustering

As discussed before, hierarchical clustering algorithms derive tree structures called dendrograms of discovered clusters. Two main approaches can be adopted to build dendrograms. They determine the two main classes of hierarchical clustering algorithms. The first class includes agglomerative hierarchical clustering. Algorithms of this class are based on a bottom up approach. Whereas, the second class groups divisive clustering algorithms that adopt top down approach to derive tree structure of generated clusters.

An agglomerative or bottom up clustering algorithm starts with the lowest level of the hierarchy. It considers each singleton set of given data as a cluster. Then, it successively merges clusters, to derive larger ones, based upon the predefined dissimilarity criterion until reaching the root cluster of all data points or the termination condition. Candidate dissimilarity criteria in merging two clusters are based on distance between data points in these. Minimum and maximum distance criteria are among those applied by agglomerative clustering. They respectively identify nearest and farthest neighbors to clustering algorithms of this class. Two nearest clusters are merged by these algorithms only if they include respectively the closest or farthest data point pair where each point belongs to one cluster.

Divisive clustering algorithms start with all data points of the given data set in one root cluster. Afterwards, they iteratively split each cluster into smaller ones until a stopping condition is satisfied or each data point forms a cluster. The splitting step of top down clustering algorithms can be based on partitional clustering including k-means.

Although, both hierarchical clustering approaches reach the same result, divisive clustering is always considered more computationally expensive than agglomerative. Thus, bottom up clustering algorithms are commonly adopted in several application fields. BIRCH is one of the most applied agglomerative clustering algorithms in intrusion detection. It characterizes each cluster by its centroid and radius. Moreover, BIRCH algorithm introduces two main concepts of clustering feature and clustering feature tree. A clustering feature summarizes each discovered cluster. It corresponds to a three dimensional vector that includes the number and the linear and square sum of cluster data points. Clustering features are additive in that when two disjoint clusters are merged, the clustering feature of the resulting cluster

corresponds to the sum the two initial clustering features. This is useful in building the tree structure of clustering features.

Clustering feature tree instead determines the cluster hierarchy of a given data set using computed clustering features. It identifies non-leaf nodes represented by clustering features of their child clusters. BIRCH algorithm uses a branching factor parameter that specifies the maximum number of child clusters by non-leaf node. Furthermore, it has a threshold parameter that determines the maximum diameter of leaf node clusters. BIRCH algorithm performs in multiphase where the first scan generates an initial clustering feature tree which is improved in one or more additional scans of the data set [155], [429].

Mahmood et al. [245] have designed Echidna, a hierarchical clustering algorithm to classify network traffic. Echidna is inspired by BIRCH. Moreover, it is able to deal with different types of categorical, numerical and hierarchical attributes, such as IP address. Echidna uses required preprocessing mechanisms to transform attributes of different types and then to evaluate dissimilarity measure between data points involving them. Similarly to BIRCH, the tree structure built by Echidna corresponds to several levels of clusters each of which is represented by a feature vector. The latter includes sufficient statistics to determine the cluster centroid and radius. In leaf nodes, this is involved in estimating the maximum and average intra-cluster distances required to generate traffic reports based upon Echidna clusters.

Echidna has been tested using the DARPA datasets. Comparatively to AutoFocus, a clustering based network traffic analysis tool is more efficient in terms of the detection rate and computation time. Additionally, Echidna is capable to detect more attacks than AutoFocus.

In the frame of the European project Safeguard that aims at enhancing survivability of critical infrastructure, Burbeck et al. [60] have proposed a hierarchical clustering based agent detector. The latter uses an extended version of the BIRCH algorithm, ADWICE (Anomaly Detection With Incremental Clustering), to model the normal behavior of the monitored network and detect its anomalous traffic. ADWICE is an incremental agglomerative clustering. It includes the first phase of clustering feature tree generation of BIRCH. In this phase, ADWICE uses the parameter of maximum number of clusters produced by the algorithm, additionally to the branching factor and threshold parameters of BIRCH. This parameter was introduced to ensure the generation of sufficient number of clusters that appropriately represent normal behavior of the monitored system. Moreover, ADWICE supports the threshold parameter in the detection phase where the belief threshold of each processed data example is evaluated to decide its normality or intrusiveness. The belief

threshold is distance based estimated by considering a tolerance interval of the anomaly detection system.

ADWICE algorithm has been experimented using the DARPA preprocessed data set. Reported results show that ADWICE is capable to detect 95% of normal and anomalous traffic records. Furthermore, it reaches a false positive rate of 2.8%. When tested using different data sets of DARPA attack classes, the anomaly detector based on ADWICE ensures high accuracies except for R2L class that reaches 31% only. Another extension of the BIRCH and ADWICE algorithms has been also proposed and experimented in [167]. It ensures better detection results than these, but it is slightly less effective in terms of the false positive rate.

Zang et al [428] have also experimented the capabilities of hierarchical agglomerative clustering in detecting botnets, networks of compromised end hosts using remote commands. For their experiments, they have simulated anomalous flows of a botnet. Collected botnet flows are then time based merged with real normal internet traffic. The resulting data set is preprocessed to derive a new one including sixteen time based features. Afterwards, the adopted hierarchical clustering algorithm that relies on the basic process of agglomerative methods [155] is applied to the preprocessed data set. Its discovered clusters distinguish near perfectly botnet flows from those normal. Although, its detection capabilities are comparable to those of k-means algorithm, it is less efficient than this in terms of computation time.

Agglomerative hierarchical clustering has been also applied by Maggi et al. in clustering system calls of the monitored host. The clustering based detection approach proposed by Maggi et al. consists of two main steps. The first step focuses on clustering arguments of an invocated system call. Discovered clusters aim at characterizing each system call through capturing relationships between values of its invocation arguments. A representative model is generated for each of these clusters to classify further inputs. It will assign a probability to the processed system call that assesses to which degree it belongs to the considered cluster. The second step of the proposed approach concentrates on modeling normal behavior of system programs. It represents program flows by first order Markov chain models for which each state corresponds to invocated system call clusters.

Different experiments have been conducted to evaluate the accuracy and computation cost of the designed host based ids. Selected system call sets for generating clusters, Markov models and testing them have been extracted from the DARPA datasets. The designed ids ensures high detection rate and low false alarm rate for tested system calls, as shown by experiment

results [244]. Moreover, its computation overhead does not overload the monitored system specifically at the testing phase.

2.2.2.4 Model based clustering

Clustering using neural network is one of the most widely adopted model based clustering methods in intrusion detection. It is typically based upon self organizing or Kohonen maps, an unsupervised neural network type proposed by Kohonen [196]. Self organizing Map, SOM, has also gained great attention in several applications and research fields including feature selection, visualizing and exploring multidimensional data. A SOM usually consists of two or three dimensional grid of units. The latter grid maps patterns of high dimensional source space into points of two or three dimensional target space. Pattern mapping preserves topological order such that the proximity relationships between patterns in the input space are also respected by those of the output space.

In the grid, a model vector is attached to each neuron. It represents a group of patterns of the input space. Neurons model vectors are involved in the two main steps of the SOM learning algorithm namely competition step and cooperation step. In the first step, the selected winner of the competition is the neuron associated with the most similar model vector to the given input example. Similarities between model and input vectors are usually distance based determined using the Euclidean or other distance measures. In the second step, not only model vector of the winner or the best matching unit is updated but also those of its neighbors. Thus, the winner cooperates with its neighbors in the map to best represent input patterns and relationships between them, such as the proximity relation [105], [155].

Self organizing maps have been widely applied in the intrusion detection field [64], [210], [401] [426]. Labib et al. have designed NSOM a real time network anomaly detection using SOM. SOM based detector of Labib et al uses time based features of collected and preprocessed network packets. Involved features summarize source and destination addresses and protocol type for each subset of logged traffic packets. Normalized values of these features form input vectors to SOM network. Two variants of Kohonen maps have been tested in NSOM. Simulated and real data sets that concern normal and DOS instances are included in these experiments. Tested two dimensional maps are respectively based on rectangular and hexagonal neighborhood shaped in updating model vectors of winner's neighbors. Results of conducted tests show that the latter outperforms the former SOM variant in terms of classification results. For both data sets, winners of the map have similar behavior in that their

distances to the processed DOS instances are much higher than those to the normal examples. Additionally, Labib et al have graphically illustrated testing results of SOM. They have also stressed promising capacities of SOM specifically for visual discrimination between normal and anomalous traffic [210].

Other intrusion detection systems use enhanced detectors involving Kohonen maps. A two level hierarchy of SOM has been adopted in the dynamic intrusion detection system of Lichodzjewski et al. [231]. The dynamic ids takes account only of the basic features of network connections such as duration, protocol type, service and other attributes. For each of these features, a SOM is built. Before that, logged values for each basic feature are preprocessed in order to provide the corresponding SOM with required inputs that support time representation. Basic feature SOM are then considered as potential inputs to the single map of neurons of the second level. This SOM aims at presenting a unified view of network connections. It uses input data of reduced dimension provided by the selected subsets of neurons of the first level SOM. All selected subsets have the same size. Furthermore, the outputs of involved neurons in these subsets are normalized before forwarding them to the second level SOM.

Different experiments have been performed on the proposed hierarchical SOM detector using the preprocessed DARPA datasets. They are two folded. On one hand, they aim at deducing labeling rules and validating them based upon frequently selected winners or clusters of the second level SOM. On the other hand, they seek to evaluate detection performance of the designed dynamic ids. The results of these experiments show that two best matching units of the second level SOM are capable to distinguish anomalous connections. Generated classification rules involving these units ensure 0.02% and 33% respectively as false positive and negative rates [231]. Reached detection performance lets us deduce that the hierarchical SOM detector confounds a little fraction, 0.02%, of normal connections as anomalous ones. However, 33% of the classified intrusive connections by this detector are confused with normal connections. This may be due to the involvement of same best matching units of second level SOM in classifying normal and anomalous connections as stated in [188].

Kayacik has extended two levels of hierarchical SOM detector of Lichodzjewski et al. by a third one. SOM of the added third level concern specific neurons of the second level SOM, eventually those inducing confusion between normal and intrusive connections. These neurons are identified based on their selection frequencies as winners for classifying both normal and abnormal connections. For each of these neurons, a third level SOM is created. It

is trained on a reduced data set for which the corresponding neuron is considered as the winner to classify included connections. Third level SOM are involved each time one of uncertain neurons of the second level wins the competition (is selected as the best matching unit). Testing results on the whole KDD 99 set show that the extended SOM hierarchy remarkably outperforms the two level version in terms of false negative rate. Near 0.3% of intrusive connections are confused with normal ones by this extension instead of 33% for the two level hierarchy. However, this extension has an increased false positive rate, which reaches 1.7%, comparatively to 0.02% for the initial SOM hierarchy in [231].

Ibrahim [172] has designed a network misuse detection system that uses growing hierarchical SOM, GHSOM, based detector. GHSOM is an improved version of the SOM neural network. It was proposed by Rauber et al. [319] to overcome some shortcomings of the SOM network including static structure and number of units in the map. GHSOM consists of hierarchically structured layers of independent SOM. At the top of the hierarchy, the root SOM is a single unit map. For every level in the hierarchy, each neuron in a given map may be branched on a SOM in the next level only if the dissimilarity between its model vector and represented patterns reaches a certain threshold. GHSOM algorithm also imposes other criteria on maps growing as well as on hierarchy extension.

The GHSOM based detector has been tested and compared to other detectors. Testing and training data sets for different detectors have been extracted form preprocessed DARPA data. They include normal instances and intrusive data examples of several attack types. Reached results show that recurrent network based detector slightly outperforms GHSOM detector in terms of true positive and negative rates. Furthermore, both neural networks based detectors ensure higher detection rates than KDD99 winner for R2L, U2R and probe attacks.

Different other intrusion detection experiments have adopted SOM based detectors. In [401] Vokorokos et al. have applied SOM in detecting normal and anomalous users' behaviors using features characterizing their activities. Mitrokotsa et al. have proposed a detection approach of DOS attacks relying on a variant of SOM that uses a two dimensional grid of large neuron number. Their approach also aims at evaluating SOM detection and visualization capabilities of anomalous network traffic. In the same paper, they have summarized several SOM based detection approaches [262].

Analysis engines of different ids presented in this section are based upon supervised and unsupervised machine learning techniques. Multiple other intrusion detection works have experimented additional data mining techniques in analyzing and identifying attack traces in

the log data. The next section will focus on commonly applied data mining techniques, namely association rules and frequent episodes, and analysis engines.

2.2.3 Other data mining techniques

The foundation of data mining approach for intrusion detection was developed by Lee et al. [221] of Colombia state university. Lee et al.' approach was specifically based on association rules and frequent episodes techniques. Both data mining techniques have been applied in analyzing sequential pattern and discovering relationships between feature values or data records within log data. They are capable to automatically produce appropriate and concise detection models. These rule-based models capture characteristics of normal or intrusive activities contained in processed log data. The next two sections review main intrusion detection experiments involving association rules and frequent episodes techniques.

2.2.3.1 Association rules

The problem of discovering association rules was initially introduced by Agrawal et al [3]. The Apriori algorithm was the first algorithm for mining association rules. It was initially applied in discovering association between sales in large databases. Each transaction in databases is composed by a set of values or items, regarding the considered attributes. The length of an itemset is determined by the number its items. The itemset support corresponds to the percentage of the database records that contain this itemset. Association rules focus on frequent itemsets. They specifically aim at discovering associations among items and deriving correlations between multiple attributes in large databases.

An association rule is an implication of the form:

$$X \rightarrow Y \quad [c,s]$$

Where X and Y are two disjoint itemsets ($X \cap Y = \varnothing$), c and s are two measures of rule interestingness that represent respectively confidence and support of an association rule.

The support of an association rule is the rate of records that include both itemsets ($X \cup Y$). It measures the statistical significance of the rule. The confidence is the ratio of both itemsets support to antecedent itemset support ($c = \frac{support(X \cup Y)}{support(X)}$). It assesses strength or trustworthy associated with the discovered rule. An association rule is considered significant only if its support and confidence exceed respectively pre-specified minimum support and

minimum confidence values. Different steps of discovering frequent itemsets and association rules of the Apriori algorithm are detailed in [3], [220].

In intrusion detection field, first experiments of association rules have been conducted by Lee et al. in building users' profiles based on typed commands. In their intrusion detection project, Lee et al. have adopted a modified version of the Apriori algorithm. In this version, preprocessing step, where data are converted to binary format, of the initial algorithm was dropped. Such modification helps in discovering multiple relations between large values of different attributes. Lee and co-workers have implemented their extension using a row vector structure for each frequent itemset. The row vector of bits records transactions that include the frequent itemset.

Improved algorithm of Lee et al. incrementally determines frequent itemsets with respect to derived feature classes. The final set of frequent itemsets is then used in generating association rules. Confidences and supports of discovered rules are also computed based on evaluated row vectors. Afterwards, these rules are merged depending upon fixed constraints that concern their left and right sides. The resulting rules, with significant confidence values, specify user normal profile. Rule based normal profiles will be involved in classifying further logged sessions. Mined patterns for each processed session are compared to the saved normal profiles. The similarity score between extracted and saved patterns is then evaluated to decide whether a given user session is normal or anomalous [222], [223]. In conducted experiments [222], Lee et al. have adopted five weeks session logs extracted from DARPA data to test detection capabilities of rule based profiles. The results of these experiments show that four weeks generated profiles are capable to recognize all anomalous sessions of the fifth week logs. Furthermore, when tested using intrusive traffic data, rule based profiles are able to capture specific patterns that distinguish anomalous from normal activities, as stated in [221].

Several studies have been conducted on association rules based detection models. In her network intrusion detection system, Tsai has proposed an improved version of the Apriori algorithm capable to transform association rules into classification rules [391]. Similarly to Lee et al.' extension, the proposed algorithm performs in two steps. Katkar et al. have also proposed one pass incremental association rules mining algorithm. In this algorithm, hash tables saving discovered patterns and their frequencies are created using log data of each attack type. They are then involved in generating association rules and evaluating their supports and confidences. For each attack type, its profile includes only association rules satisfying support and confidence thresholds. Detection models built using association rules

mining algorithm of Katkar et al. save attack profiles and hash tables. Saved profiles are involved in analyzing given data instances. Hash tables will serve in incremental update of attack signatures when new data examples are appended to train sets.

Katkar et al.' algorithm has been tested using traces of different DOS attacks in traffic data sets extracted from DARPA preprocessed data. In the training step, rule based signatures of 6 selected DOS attacks are constructed using reduced data sets that include 10 features. In the testing step, built rule based detection models fail to appropriately recognize DOS instances except for two attack types namely teardrop and pod attacks. They achieve detection rates below 1% of other DOS attack types, including smurf, neptune, back and land attacks, the complete list is given in section §6.4. Furthermore, their false alarm rates exceed 4% for certain DOS attack types such as smurf and pod [187]. Such failure of rule based detection models in recognizing DOS instances may depend on many reasons. In fact, many DOS instances are confused with normal data examples specifically in DARPA test set [40], [381], [385] and therefore rule based detection models have increased false positive rates for certain DOS attacks. Additionally, selected features are relevant for the discovered patterns of the given DOS attacks but not others and thus their detection rates are below 1%. Also, rule selection in building attacks signatures may require in addition to support and confidence other criteria such as testing accuracy adopted in [391].

In [187] multiple other experiments focusing on association rules based detection models have been summarized. Additional works have integrated association rules with other machine learning techniques including neural nets [365] and clustering [240], [252], [430]. Furthermore, fuzzy association rules approach initially proposed in [242] has gained increased attention in the intrusion detection field. Several ids have been based on this approach [252], [348], [373] to overcome increasingly large log data sets and adaptability shortcomings.

2.2.3.2 Frequent episodes

The goal of mining frequent episodes is to discover inter-audit records patterns. A frequent episode is a set of events that describes an entity (system, user, program,…) behavior. Such a set of events occurs frequently within a time window with respect to a given minimum frequency and partial order. In serial episodes, events occur in a sequence with the same order each time. Whereas, parallel episodes are free from such constraint.

Given a database with timestamped records, a sequence of events or itemsets $E=\{e_1,e_2,\ldots, e_n\}$ occurring in the interval $[t_1,t_2]$ if it starts at t_1 and ends at t_2. An interval is a minimal

occurrence of *E* if it contains *E* and no one of its subinterval includes another occurrence of *E*. The support of an itemset e_1 is the ratio between the number of minimal occurrences including e_1 and the total number of events [224], [243], [250].

A frequent episode rule is an expression of the form

$$X,Y \rightarrow Z \quad [c,s,w]$$

Where *X*, *Y* and *Z* are itemsets that define an episode, *c* and *s* represent respectively the confidence and support of the frequent episode rule within a time interval *w*.

Manilla et al. have proposed an algorithm for mining frequent episodes [250]. The frequent episodes algorithm consists of three phases. The recognition phase in which serial or parallel episodes are discovered incrementally. In this phase only episodes that satisfy the minimum frequency and window size are saved. In the building phase, candidate episodes with an increased size are discovered. They use candidate collection recognized in the last building phase. These two phases are iteratively and alternately performed in generating all frequent episodes with different sequence lengths. In the last phase, discovered frequent episodes serve for rule generation. The final set of frequent episode rules consists of those with confidence levels exceeding a prespecified minimum confidence.

The frequent episodes technique was experimented in intrusion detection by Lee and his team of the Colombia state university. An improved version of Manilla et al.' algorithm was proposed by Lee et al. for discovering frequent sequential patterns using logged network traffic. The enhanced algorithm discovers serial episodes in two main steps. The first step finds out frequent associations among items of the most relevant features, axis attributes. The second step generates frequent episodes using extracted associations. This two step algorithm builds episodes that combine both relationships among attributes and sequential patterns between log data records [223], [224].

Lee et al. have widely experimented the proposed frequent episode algorithm. These tests aim at determining different parameters and resolving same shortcomings associated with discovered frequent episode rules. For instance, in [221] Lee et al. have preprocessed the training set according various time windows. The frequent episode algorithm was then tested using resulting traffic data sets in order to determine an appropriate window size to evaluate time based features and generate frequent episode rule set. The results of these tests illustrate stable rule sets after reaching a time window of 30 seconds. Furthermore, the best detection rate of frequent episode rule based model is achieved when evaluating traffic attributes, time

based features, each 30 seconds, as stated in [221]. Other tests [223] focus on numerous issues closely linked to rule based detection model including merging and usefulness of count attributes in mined rules.

An improved version of Lee et al.' frequent episodes mining algorithm has been adopted also in hybrid ids of Hwang et al. The misuse system was based on SNORT engine. The anomaly detection system, ADS, uses an episode mining engine that discovers and saves normal behavior rules from internet connections. In ADS, rare episodes are considered as indicators of malicious activities. Moreover, Hwang et al. have also adopted a rule pruning step in their algorithm in order to reduce the large set of episode rules that can be generated [171]. Kokorina has enhanced Hwang et al.' work. The designed hybrid ids uses intrusive and normal frequent episode rules respectively for signature based and anomaly detection systems. It includes four main components namely preprocessing, pattern discovery, update and decision components. Preprocessing component reduces given data sets and formats them as required for the pattern discovery component. The latter generates frequent episode rules that specify known attack signatures and normal behavior patterns. In the hybrid ids, built attack and normal patterns are saved in their corresponding databases. Additionally, the attack pattern database of the hybrid ids includes SNORT signatures after transforming them into frequent episode rules. Saved patterns of both databases are periodically revised by the update component of the hybrid ids. This component inserts new patterns of unknown attacks or normal behavior to the respective databases depending upon the decisions of system administrator and the decision component. The latter determines whether processed data is normal or anomalous regarding saved frequent episode rules. Furthermore, the decision component fulfills other tasks such as selecting another feature set for preprocessing component and informing pattern discovery component on required pattern revisions [197].

The hybrid ids of Kokorina was tested using collected honypots log data. Different experiments have been conducted on hybrid ids with complete and reduced test sets. They have been focused on the detection capabilities of rule based detection model regarding selected attacks of several classes including U2R and probe. In these experiments, only frequent episode rules associated with 1, 2 and 3 items have been considered. Furthermore, time windows size applied in preprocessing data set and rules generation varies from 2 to 20 seconds. Relying on generated rule based normal and attack detection models, the hybrid ids is capable to recognize most of tested traces. These results illustrate the appropriateness of

frequent episode technique to intrusion detection. Detailed results of the performed experiments and main findings of Kokorina's work are thoroughly discussed in [197].

The frequent episode technique has been also applied in detecting masquerade attacks. Gigstad has adopted frequent episode technique to reduce false alarm rate of an ids through generated filtering rules. Luo et al. have proposed a network ids based on fuzzy frequent episodes. Additional details of these intrusion detection experiments and others involving classic and fuzzy frequent episodes are respectively discussed in [137], [243]

Main findings of studied analysis and detection engines:

The majority of previous intrusion detection works, discussed in the current chapter, propose simple detection models, each of which is merely based on a single technique. Several machine learning, data mining and other artificial intelligence techniques are adopted in generating known attack signatures or discovering normal behavior patterns of systems, users or programs. Other works are based on the integration of more than a single detection model, building then integrated or hybrid detection models. Base detection models may be generated using single or multiple artificial intelligence techniques. In these works, two main strategies are adopted to integrate base detection models, namely sequential and parallel strategies.

The sequential integration strategy assumes that build detection models are applied regarding a prespecified order, such that outputs of one are processed or refined by the next detection model. Sequentially integrated detection models are usually built using various artificial intelligence techniques such as in [204], [247], [264], [348]. It is typically applied when dealing with a hybrid detection model integrating unsupervised and supervised machine learning techniques as described in [139], [247], [264].

The parallel integration strategy instead takes simultaneously account all base detection models. For any processed data example, outputs of base detection models are all combined together to derive the final decision of the integrated model. Combined detection models are commonly constructed using single technique such as SVM or ANN [135], [254], [267], [296]. They may be also based on different learning techniques as proposed in [276], [330].

All proposed hybrid detection models using either sequential or parallel integration strategies are based on a static set of base models. They always apply the same combination of base detection models independently to processed data examples. This may involve additional costs, when processed example is correctly labeled by all base detection models whereas a subset of them is sufficient. Furthermore, it may induce erroneous decisions specifically if

processed data is correctly labeled by a subset but not by another of detection models. Therefore, selective integration of detection models, which was not investigated before, may be a potential and promising solution to overcome mentioned shortcomings and others as discussed in the introduction of this report.

2.3 Response mechanisms of intrusion detection systems

The analysis component of idrs has extensively gained attention of researchers in the intrusion detection field. Minority of previous research works has focused on the idrs response component and proposed potential improvements of its reactions. A response component concerns post attempt or attack detection behavior of an idrs. Its implemented actions after detection may range from a simple alert to a complex reaction involving multiple environment and attack dependent factors and others. Several taxonomies have been proposed to identify different categories of idrs responses. Depending on the automation level, three main categories namely notification, manual and automated responses are identified in [67], [368]. This taxonomy was also studied and its response classes are detailed in the recent work of Shameli-Sendil et al. [345]. Notification components alert the SSO on detected or potential threats targeting the monitored computing environment. However, manual and automated response components provide the SSO respectively with possible and recommended corrective actions against identified threat. The later automatically implements recommended actions which are manually selected and deployed by the former. Another commonly adopted taxonomy categorizes intrusion responses into passive and active such as in [23], [30], [94]. Passive responses correspond to simple notifications to the SSO whereas; active responses include various corrective actions against detected threats. In [368], Stakhanova et al. have proposed an extended taxonomy that includes above discussed categories. Furthermore, other response characteristics have been considered by Stakhanova et al. to devise a detailed taxonomy. However, Stakhanova et al.' taxonomy [368] does not meet mutually exclusiveness principle because a response component may be assigned to more than single category such as for notification and passive responses. In this section, the response taxonomy identifying passive and active categories will be adopted in presenting previous works.

2.3.1 Passive response components

Passive responses have no mitigation effect on mounted and detected attacks. Passive response components merely inform the SSO about detected threats. They always raise an

alert. They may also provide the SSO with additional information including logged intrusive activities and a report on mounted threats depending upon the capabilities of involved detection models. Corrective actions against detected threats are further decided and implemented by the SSO. Most of existing ids are passive [369]. SNORT is one the most known signature based passive ids. Its response component alerts the SSO and reports detected attacks. Additional other passive idrs are presented in [67].

2.3.2 Active response components

Active response components aim at mitigating incurred or potential damage of mounted attacks. They automatically select or assist the SSO in choosing appropriate security controls and then deploy them on the monitored computing environment. Active responses have been categorized based on response time into proactive and reactive in [23]. Proactive response components preempt attackers' activity sequences and react before attacks take place relying on potentially intrusive activities predicted by analysis engines. Reactive response components instead implement their corrective actions only after detection of attacks. Furthermore, automated responses have been structured into static, dynamic and cost sensitive categories depending on how response's actions are selected [369]. Additional other response categories are deduced by involving characteristics such as adjustment and cooperation abilities of automated response components [369], [372]. This section focuses on automatic response strategy design, without human intervention. Response selection and response time criteria are the only considered in presenting previously designed response components.

2.3.2.1 Static response selection

Static intrusion response components have prespecified action lists. Depending on detected attack types, corresponding action lists are then implemented by static response components. Static responses are designed by the SSO or with the collaboration of a security expert for theses components. Furthermore, the deployment of selected responses is always delayed until identifying true types of detected intrusive activities and therefore static response components are mainly reactive [8], [123].

2.3.2.2 Dynamic response selection

Dynamic intrusion response components dispose of possible defense actions against potential attacks. Sets of response actions to be implemented are selected by these components

depending upon criteria related to detected attacks including confidence and severity. Selection criteria are first evaluated for each attack and then appropriate response actions are chosen generally in real-time. Dynamic components offer uch more flexibility than static ones in designing response strategies. Furthermore, they are capable to take account of additional environment dependent information rather than attack type. Dynamic response components may save states on their previous reactions and hence they are able to learn from them to improve future idrs reactions [10].

Most of dynamic response components ensure delayed reactions. EMERALD and AAIRS of respectively Porras et al. and Carver et al. are examples of idrs ensuring reactive dynamic reactions. EMERALD is a hybrid distributed detection system. It relies on a hierarchy of monitors deployed at three main levels namely service, domain and enterprise. Monitors of each level focus on threats targeting the resources of that level. Furthermore, over different levels, monitors hierarchically cooperate to detect malicious activities, aggregate raised alerts and respond against recognized attacks. Each monitor in EMERALD architecture consists of profile and signature engines, resource object and resolver. The two analysis engines perform respectively anomaly and misuse detection. The former uses computed statistical profiles to identify unacceptable deviation to normal behavior of the monitored service, domain or network. The later verifies logged event sequences against saved signatures to determine those similar to encountered malicious activities. A resource object component provides all other components with required data such that the independence of deployed monitor to its target service, domain or network is preserved. A resolver component receives analysis reports forwarded by profile and signature based engines, either internal or external to its monitor. It is an expert system capable to coordinate analysis reports and implement responses. A resolver is able to correlate intrusion results issued from analysis engines. Furthermore, it has the possibility to request analysis reports from other resolvers at lower levels in order to derive global decisions about security state as well as required responses to monitored target. Resolver of each monitor is also capable to trigger appropriate responses against reported intrusive activities. Response methods available to resolver are saved by the resource object of its monitor. Each method is associated with evaluation metrics namely threshold and severity that determine its deployment conditions. The threshold metric assesses the confidence assigned by the analysis engine to detected intrusive activities, whereas the severity metric specifies to which degree a response method is appropriate to anomalous

activity sequences. A resolver combines both metrics in designing monitor's response [67], [123], [306].

The EMERALD response component is stateless in that its previous experiences are not considered in designing the current response. An adaptive response component based on agent technologies was designed by Carver et al. of Texas A&M University [67]. It has the capacity to learn from its environment and past experiences in order to improve further deployed responses [8]. Adaptive agent based intrusion response system, AAIRS, of carver et al. fulfills post detection phase for passive ids. It includes multiple agent types. Interface agents of AAIRS interact with passive ids monitoring target system. They keep track of ids activity history including their false and true positive rates. Built model of each ids will serve in generating analysis report and evaluating attack confidence metric by interface agent. Analysis report and confidence metric are forwarded to a master analysis agent. The latter identifies whether the detected attack is a new or continuation of an existing one. In the former case, it creates a new analysis agent to generate a response plan against the new attack. Whereas, in the latter case, a master agent forwards detection report and confidence metric to an analysis agent specialized in countering the detected attack. An analysis agent processes and mitigates detected attack with the cooperation of several other agents. It generates a sequence of actions, to counter detected attack, in which response taxonomy, policy specification and tactics agents and response toolkit are involved. The detected attack is initially classified by the response taxonomy agent. Afterwards, the goal and limit of response actions are determined by a policy specification agent regarding multiple aspects including legal and ethical aspects. Then, selected response actions are forwarded to tactics agent that decomposes them into specific actions. Tactics agent also triggers the appropriate component of response toolkit to implement specific actions of the generated response. Decisions of analysis and tactics agents are adaptively taken depending upon the success of their previous responses, which are saved by a logger agent of AAIRS. In Carver et al. response system, the SSO feedback is always required in updating information [183] and guiding further responses [8]. However, Carver et al. have not presented in which step of the response process the SSO interacts with AARIS. Moreover, as stated in [123], they have given no additional information on main algorithms performed by AAIRS agents including response action selection, actions decompositions and response success evaluation.

2.3.2.3 Cost sensitive response selection

Above discussed dynamic response components of EMERALD and AAIRS are reactive. AAIRS only is capable to adaptively design action sequence depending upon its previous responses. However, neither EMERALD nor AAIRS takes account of response cost criterion in selecting response actions or implementing them. Another category of response components focusing on cost sensitive design of ids reaction has gained the attention of many researchers, last decade. Initially, Lee et al. have integrated cost factors in detection component of an ids. They have designed cost sensitive detection model in which three main cost factors, specifically damage, response and operational costs, are considered. Damage cost assesses potentially incurred damage by an attack on monitored resources. Response cost determines required defense cost against current or potential intrusion. Operational cost estimates the cost of performing intrusion detection including all processing steps from log data collection until the analysis report generation. First two cost factors will determine consequence costs depending upon possible output decisions of the analysis engine of an ids. Cumulative consequence and operational costs will then gauge the total expected cost of intrusion detection. The latter is involved in generating cost sensitive detection models for Lee et al.' ids.

Lee et al. have adopted DARPA attack taxonomy in their cost model. Base damage and response costs are prespecified for each attack subclass of those identified. These cost factors associated with the criticality of the target asset will respectively estimate costs of incurred damage and countering the mounted attack. The response decision, in this model, is based on trade-offs between evaluated damage and response costs regarding the mounted attack and target asset, such that an intrusion is mitigated only if its damage cost exceeds selected response cost. This is also reflected in estimating the consequence cost of Lee et al.' model [225], [368], [372].

In the cost model of Lee et al, damage and response costs are statistically initialized. This is unrealistic because neither base costs nor criticality of the victim reflect its security requirements and severity of its exploitable weaknesses. Moreover, Lee et al. have not detailed multiple critical algorithms to their cost model including how evaluating base damage and response costs and selecting appropriate controls to mitigate reach damage.

Balepin et al. have also proposed an adaptive response component based on derived hierarchy and directed graph of system resources. In Balepin et al.' response component, system

resources are structured based on their types into hierarchies. Such hierarchies are useful in selecting response actions because the resources of the same type are often protected with same security controls. However, they are not sufficient in selecting appropriate controls since thorough information on monitored system and its resources is required. Therefore, system map expressed in terms of resources directed graph is included in Balepin et al.' response component. In the system graph, resources represent different nodes and edges express dependencies between them. Each node in the graph is associated with cost value and basic response actions. The cost value is commonly assigned by the system administrator or information owner depending upon the importance of the represented resource. The response actions are capable to re-establish normal function of the protected resource if attacked. Each of these actions has an activation condition and a list of affected nodes when implemented. Furthermore, each node in the graph is assigned response actions. Response actions of each target node are selected based on a cost benefit analysis step. For this step, intrusion cost and response cost and benefit are estimated. Intrusion action cost is determined by the sum of cost values of damaged map nodes due to the mounted intrusive action. Cost and benefit of a response action are approximated relying on costs of map nodes respectively damaged or restored to normal function due to implemented corrective action. A gain matrix is then constructed using evaluated costs. This will serve in determining the optimal response actions, achieving maximum benefit and minimum response cost, to be deployed against detected intrusive actions [35], [372].

In Lee et al. and Balepin et al. response components reactively defend against detected attack. They are based on analysis of tradeoffs respectively between damage and response costs and cost and benefit of potentially deployed response. Foo et al. have also proposed an improved trade-off based response component that has the capacity to implement proactive reactions. ADEPTS of Foo et al. is an adaptive graph based response system to rule out attacks in a distributed system of interacting services [122]. It relies on intrusion graphs, I-Graph, that model paths potentially followed by attackers to achieve their goals and spread intrusion behavior from a service to another. An I-Graph consists of nodes expressing intrusion sub-goals and edges representing pre and post-conditions between them. It models dependencies between sub-goals and possible alternatives of arranging them to fulfill the final goal of an intrusion including information leakage or denial of service. Each alternative or sub-goals path explicitly states how attackers sequentially proceed to meet the objective of the implemented deliberate actions.

In ADEPTS, reported alerts are mapped to I-Graph to determine followed paths and estimate the likelihoods and locations of the achieved sub-goals. Responses against detected intrusions are then selected and deployed to prevent and limit potential propagation of intrusive behavior to other services in the distributed system. ADEPTS responses are selected depending on their past efficacies against considered attack classes and disrupts to system normal activities. Furthermore, confidence levels of detection engines decisions are also included in designing ADEPTS responses. The response control center of ADEPTS chooses and filters candidate responses based on evaluated parameters. Most appropriate response actions, according these, are then deployed by response execution agents. A feedback system supported in ADEPTS tracks implemented responses and revises their effectiveness indexes [122], [123], [372].

Although, ADEPTS prevents against spreading intrusive actions to other services of the distributed system, the effectiveness of its implemented response remains always dependent to parameters and designed intrusion graph and granularity of its sub-goals. In fact, in ADEPTS response process, disruption index evaluation for deployed response strategies is not detailed. Furthermore, intrusion graph, which is semi automatically generated, may include elementary sub-goals of mounted intrusive actions. In this case, identifying graph node using generated alert and its confidence factor is extremely complex unless additional information on implemented attack actions is available.

Several other limits of ADEPTS have been enumerated by Stkhanova et al. in designing their cost sensitive preemptive response component [368]. The latter overcomes some of discussed shortcomings and includes multiple other improvements, compared to ADEPTS. The response component of Stkhanova et al. is based on tradeoffs between the damage cost incurred by unexpected actions and response cost of potentially deployed corrective actions. Its response process includes three main steps namely pre-emption decision making, candidate responses identification and optimal response selection. The first step focuses on determining the time point when a response should be fired. Therefore, a probability threshold expressing an acceptable confidence level from which an attack in progress should be ruled out is predefined. For each detected intrusive event sequence, its probability of occurrence is compared to the threshold. The candidate response identification step is initiated only if the occurrence probability of a malicious sequence exceeds the tolerance threshold. This second step of the response process aims at maintaining required balance between damages and benefits of early response. It is based on estimated damage and response costs of potential response actions against detected intrusive event sequence. Included actions in the candidate

set are those associated with weighted damage costs, using computed occurrence probability, exceeding their response costs. The final step of the response process determines the optimal action to be deployed among the candidate set. The optimal action ensures the highest expected value expressed in terms of its success and risk factors and probabilities of success and risk of the detected event sequence. Success and risk factors of a response action assess respectively its appropriateness in the past and negative effect on the target of an attack. Selected optimal response action based on these parameters ensures the highest benefit and least risk to the monitored computing environment [368].

Lee et al. and Balepin et al. response components ensure cost sensitive delayed reactions. Likewise, ADEPTS and Stakhanova et al. response components automatically select cost sensitive response actions which are proactively implemented. Toth et al. and Strasburg also have designed adaptive response components able to deploy delayed or proactive reactions depending on capabilities of analysis engines. Toth et al.' response approach has inspired multiple reaction components of successors such as Balepin et al. and Stakhanova et al. Network based response component of Toth et al. relies on built network model to evaluate the effects of deployed responses on operational services. A network model consists of multiple elements of different types and dependencies between them. An element included in Toth et al. model may be a network service, resource, user, topology or other that can potentially be affected by the deployed response actions. Furthermore, dependencies between these elements reproduce their direct or indirect relationships in the network system.

The response component of Toth et al. evaluates response action impacts based on a network model using an impact evaluation function. The latter uses dependency trees of entities, network resources or users, to evaluate their capabilities. A dependency tree of each entity is extracted from the network model. It illustrates the relationships and their types, AND and OR relations, between the target and other entities in the network model. Additionally, it represents dependency degrees within the tree that range between 0 and 1. A dependence degree expresses to which degree the operation of the target entity relies on the availability of another within the network model. A dependency tree of an entity will serve in evaluating its capability. The capability of an entity is a value within the interval [0,1]. It expresses to which degree the target entity normally operates either when response actions are implemented. The capability value of an entity is determined using a recursive algorithm when its dependency tree includes other entities. Estimated capabilities associated with penalty constants are then involved in gauging penalty costs that assess unavailability costs of network model entities. In

Toth et al.' response component, penalty costs due to candidate response actions are always evaluated. The most appropriate response action to be preemptively or reactively deployed is that yielding the least penalty cost among those candidates as stated in [388].

The response component of Toth et al. solely focuses on availability cost in the automatic response selection. Whereas, integrity and confidentiality impacts are also extremely important is designing response strategies. This component requires a detailed graph that captures, direct and indirect, dependencies between elements and potential extensions to represent a further expansion of the network system. Additionally, penalty costs of elements are not constant and need to be dynamically determined depending on mounted intrusive actions, their objective and dependency trees of target entities [372], [388].

In a recent work, Strasburg has designed a host based intrusion response framework that ensures cost based selection of preventive or defensive actions respectively against predicted or detected intrusive activities. The developed selection methodology within this framework determines most appropriate response strategies depending on their expected values and the security policy of the monitored system. It estimates the expected value of a response by the difference between its approximated benefit and cost, both have the same scale and range within [0,1] interval. A response cost is determined by its damage and operational costs. The latter expresses cost requirements to deploy a response, whereas, the former evaluates its potential damage on the target in the post deployment phase. The damage cost of a response is estimated based on its impacts in terms of confidentiality, integrity and availability, (CIA) on the protected system resources. The operational cost of a response is determined relying on organization associated cost and monitored system estimated value. The latter is determined in terms of CIA based on, assumed well established and available, security policy of the organization. The intrusion cost is also assessed by Strasburg methodology in order to approximate a response benefit. This methodology expresses the intrusion cost similarly to the response cost without normalizing operational cost because, as stated in [372], the implementation cost of intrusive actions may exceed their incurred damage. The overall benefit of a response is then determined by its estimated reduction effects on intrusion impact and mounting costs. Among those candidates, Strasburg's response component selects the response strategy associated with the highest benefit.

Although, Strasburg's response components has remarkably improved cost sensitive response selection through focusing on environment dependent parameters including security policy and security requirements, its response process supports several shortcomings. In fact, this

process does not initially illustrate how candidate response actions are selected. Afterwards, it evaluates intrusion cost similarly to response cost using in both cases damage and operational costs. But, the operational cost of an intrusion has a different interpretation than that of a response. Furthermore, its estimation process, which is not expressed by Strasburg methodology, is extremely complex and requires detailed information obviously difficult to reach or unavailable such as attacker skill and attacking tools. Moreover, in the response process of Strasburg component, it is also useful to take account of the target vulnerability state in estimating damage cost of a candidate response as well potential intrusion.

In addition to Strasburg and Toth et al., Anuar et al. have recently designed a response model that combines proactive and reactive reactions. This model proposes two response zones namely active and passive relying on intrusion time frame. Within the active zone, a response component should be capable, according this model, to implement its preventive or defensive reaction depending upon the mounted attack progress [23]. Strasburg et al. [372] have proposed a useful cost model to response component potentially deployed within this active zone. This cost model relies on three factors, namely response operational cost, goodness and impact on the target system, in order to evaluate candidate responses. Evaluation, selection and deployment of response actions according the cost model of Strasburg et al. are performed with respect to the security policy of the monitored system. Cuppens et al. [86] have also extended the security policy to include response requirements called also reaction policy. Formally specified reaction policy rules [86] are involved in identifying detected intrusion and assessing its impact regarding violated security policy rules. Furthermore, they are useful to dynamically revise the requirements of security policy, such as access policy, and fire appropriate reactions against detected intrusion.

In recent works, Baayer and Regragui [31] and Zaghdoud and El Kahtanai [426], have improved existing cost sensitive response components. Enhanced cost model of the response component in [31] takes account of false positive cases. It aims at optimizing response costs when dealing with such cases. Layered response component proposed in [426] was based on agent technologies. After processing detection decisions by high level layers, the last layer of the response component designs and implements appropriate responses respectively using specialized and mobile agents. Multiple other response approaches have been discussed also in [121]. Designed response components based on these approaches are also compared regarding their efficiencies, complexities and cost effectiveness in the same work.

Main findings of studied response engines:

Previously reviewed cost sensitive response components have been based on different factors including response cost, damage cost, operational cost, response benefit and others. Most of them lack detailed processes to selected candidate response actions depending on detected attack or attempt of attack and the target system. Furthermore, they commonly identify appropriate responses based on cost benefit or cost damage tradeoffs with consideration of neither the security policy nor the current security state of the monitored system. Strasburg's response component specifically is based on the security policy of the target to select and deploy response actions. However, no one of these response components has taken the current security state, vulnerabilities and implemented controls, of the target system in evaluating cost factors. Additionally, some of these components use static parameters such as penalty cost in [388] and probability of intrusion occurrence in [372]. In recent works such as [345], [371] response cost evaluation is discussed but neither detailed processes to estimate involved parameters is presented nor thorough risk model to this aim is designed. Further improvements of these response components are potentially possible if security standards such as those of NIST and ISO are adopted and common vulnerability databases including NVDB and OSVDB are involved in corresponding response processes.

2.4 Conclusion

In this chapter, different detection models based on supervised and unsupervised machine learning and data mining techniques are presented. Supervised techniques such as decision trees, naïve Bayes, neural nets and support vector machines have been commonly adopted in generating appropriate detection models for known attacks. Whereas, unsupervised detection models based solely on clustering techniques have promising capabilities in detecting unknown attacks. Other data mining techniques including association rules and frequent episodes have been also successfully applied in misuse and anomaly intrusion detection.

Additionally, different response components are reviewed in this chapter. Response processes of several dynamic and cost sensitive response components are also detailed in the current chapter. A specific focus is given to cost models of tradeoffs based response components. Cost factors and their relationships within these response models are identified. Moreover, evaluation processes of involved factors are discussed as well response action selection and associated decision criteria. In the next chapter, our idrs framework will be introduced with its

two main components of multimodel analysis and risk driven response. Subsequent chapters of this work will be dedicated to thoroughly present these components.

CHAPTER 3

THE PROPOSED INTRUSION DETECTION AND RESPONSE SYSTEM FRAMEWORK

3.1 Introduction

Existing intrusion detection systems use different log analysis techniques. A minority of these take account of post detection reaction mechanisms. Both analysis and response components of theses idrs systems have many drawbacks as previously discussed in the introduction. They require more improvements to appropriately defend against increasingly complex attacks and meet security requirements of the monitored computing environments. In this thesis, proposed enhancements address structural aspects as well as operation steps of existing and future idrs systems. They are associated to designed idrs framework herein introduced.

3.2 Proposed framework

Designed framework proposes an improved structure for existing and future idrs systems. Besides components of the CIDF, it includes other ones deemed extremely useful according to identified problems of existing idrs. Moreover, it supports an idrs life cycle that states different operation steps of involved components. Both, components of the proposed architecture and steps of the idrs life cycle are presented in subsequent sections.

3.2.1 Problem formulation

The main goal of intrusion detection systems is to generate more certain, precise and accurate results as much for detection as for response. Several methods have been suggested and adopted by ids to deal with these concerns. However, ids methods support several weaknesses that induce failure at different levels of the intrusion detection process. Previously, detection methods fail to capture different aspects associated with mounted attacks or monitored system normal behavior. The majority of these methods rely on single detection model approach. They solely address to a single relation type including nonlinearity, temporal or association,

within logged intrusive or normal events. Therefore, their generated detection models focus on a single aspect or facet of system normal behavior or attacker intrusive actions. Additionally, these detection methods are facing to the problem of data sets availability. Indeed, reduced data sets that appropriately report different attack classes are available for these methods. This remarkably limits capabilities of their generated detection models to recognize slight changes in an entity's behavior and thus affects the performance of the ids.

Additional other detection methods have been based on multiple model approach. However, they share same weaknesses with the single model approach unless other reinforcing techniques, including boosting or bagging or various learning algorithms, are adopted. Although these improvements partially overcome weaknesses of the former approach, they induce additional drawbacks for the latter. The resulting detection methods always use static combinations of generated models. They involve the same sets of detection models in analyzing any collected log data set and recognizing instances of normal and different attack classes.

The majority of the proposed intrusion detection methods take account of a single log data type including network, host or application logs. In addition, they neglect model generation and updating steps within the ids operation process. A global description is always given for both tasks that briefly concerns compositions of training data sets and settings of the algorithms involved in building detection models. As such, no structured process is presented to deal with how preprocessing data sets, selecting features subsets and generating and updating detection models. Moreover, ids systems are devoid of a knowledge base component. Such component saves the domain dependent and intrusion detection knowledge of extreme usefulness for idrs. It maintains knowledge on detection models, their use conditions and performances, and previous experiences of idrs systems. It also gives idrs with required knowledge on computing environment assets, vulnerabilities and security controls.

Additionally, some of previously designed ids integrate response actions to the detection process. However, response capabilities of an idrs are determined by its response component, as stated by the CIDF framework. For active ids, they range from a simple alert associated with a list of applicable controls, given to the SSO, to an automatic selection and implementation of a combination of countermeasures, depending on detected attacks. Moreover, it is common for existing idrs to defend against detected attacks without involving priorities. Their response components similarly treat attacks of the same class regardless of their respective effects on target assets of the computing environment. The major drawback of

these idrs is their failure to provide any information on the extent of inflicted damages to target assets. Indeed, their designed response components exclusively focus on the principle of reducing likelihoods of threats instead of the global cost of underlying risks.

Known above enumerated and other drawbacks, the proposed idrs framework in this thesis addresses many of them. It is based on multimodel and risk driven approaches respectively for detecting attacks and designing responses against them. Our multimodel approach allows several evaluation facets to the idrs systems in order to conduct a thorough analysis of the current security state of the monitored computing environment. It ensures multiple log data sets processing using appropriate combinations of detection models. Processed data sets by the proposed component may be issued from single or several sources. They differently trace intrusive or expected activities within the monitored system by involving multiple features. Their features subsets depend on sources or specificities of single source log. Network, router and firewall log types are determined by considering their sources. Whereas, intrinsic and content logs are examples of log types identified for single source. They focus respectively on basic and content attributes of collected network connections. Log data sets of different types and their feature subsets are determined by preprocessing and formatting collected log data regarding a prespecified time window. They serve in generating and updating detection models. They are also involved in selecting best combinations of detection models for the analysis component of the proposed idrs framework. Formally, J different log types are considered by the idrs framework. Each log data set of given type j, $j=1..J$, is expressed using the whole feature set of log type j, F^j, such that $F^j = \{f_l, l = 1..L_j\}$ and f_l is the l^{th} feature of log type j. Feature set available to the idrs framework over all considered log types is represented by the set $F=\{ F^j, j=1..J\}$.

In the proposed idrs framework, each log type is associated with a set of detectors or detection models. A detection model is implemented using machine learning, data mining, and other artificial intelligence techniques, as illustrated in the previous chapter. Classification, prediction, pattern matching, genetic and other techniques have been widely applied in constructing detectors for ids [130], [273], [285], [337], [342]. They are also useful in building detection models to the analysis component of our idrs framework. Additionally, this component merely imposes that any participating detection model in assessing current logged activities, independently to its generation technique, should provide an abstract output among those prespecified. Such condition requires additional processing steps when dealing with

other techniques, but it is obviously satisfied by classification techniques. Thus, only supervised classification techniques are considered in constructing detection models in order to simplify the design and illustration of associated processes to the analysis component of the proposed idrs framework.

Supervised classification techniques are widely applied in different domains including information security. Classification models built using these techniques use data sets whose examples are entirely labeled with their true output class. Labeled training data for these models concern multiple output classes. Formally, for any given training data set *TR* of *N* examples, $TR=\{(x_r,y_r),\ r=1..N\}$, where $x_r \in \Re^P$ is a *P*-dimensional data instance, x_l^r is the value of feature *l* of the instance *r*, $l \in \{1,\ldots,P\}$, and $y_r \in C=\{c_1,\ldots,c_{Q+1}\}$ the set of possible output classes. The classification model derived based on the training set *TR* is represented the mapping M_c such that $M_c : \Re^P \rightarrow C$. When given with an unlabeled data example, $x \in \Re^P$, the built classification model predicts its output class, $y \in C$, such that $M_c(x) = y$.

Classification models are mainly categorized into binary or multiclass depending on their output sets. Multiclass classification models are above summarized, whereas binary models have an output set *C* such that $C=\{c_q, c_{\bar{q}}\}$ where c_q is the only known class and $c_{\bar{q}}$ groups other possible output classes than c_q. Binary classification models are also known as supervised outlier detection models. A given data example $x \in \Re^P$ is an outlier if $M_c(x) \neq c_q$ and then it is dissimilar to subsets of *TR* associated to the target output class c_q [349].

Binary or multiclass classification models may be also categorized based upon details of their output decisions into abstract, rank and measurement as detailed in section §4.6. Additionally, several forms are useful in representing classification models. Most widely adopted representations include rule set, decision tree, graphical network and automata [34], [101], [155].

In our idrs framework, detection models generated using classification techniques are based on training, validation and testing datasets set of the corresponding log type, respectively TR_j, VS_j, TS_j, $j=1..J$. For each log type *j*, its training set TR_j consists of N_{tr} labeled data instances, $TR_j = \left\{(x_r, y_r) / x_r =< x_1^r,...,x_{L_j}^r >, y_r \in C, r = 1..N_{tr}\right\}$ where x_r is a data instance of L_j dimensions, x_l^r is the value of feature *l* of the instance *r*, $l \in \{1,\ldots,L_j\}$, and y_r the output class

associated with x_r, $y_r \in C$, the set of possible output classes, $C=\{c_1,\ldots,c_{Q+1}\}$. In our idrs framework, the output set C includes Q attack classes of reduced DARP taxonomy, c_q, $q=1..Q$, and the normal class, c_{Q+1}. Additionally, the training set TR_j is reduced to $TR_{i,j}$ using the projection function, $\pi()$, and selected feature subset, $F_i \subset F^j = \{f_l,\ l=1..L_j\}$. For instance, the reduced training set $TR_{i,j} = \pi_{F_i}(TR_j)$, such that the feature subset $F_i = \{f_l, l = 1..L_{j-1}\}$, consists of N_{tr} unlabeled data examples. Testing and validation data sets of each log type j including respectively N_{ts} and N_{vs} unlabeled data instances are given by: $TS_j = \{x_u / x_u =< x_1^u,...,x_{l_j}^u >, u = 1..N_{ts}\}$ and $VS_j = \{x_v / x_v =< x_1^v,...,x_{l_j}^v >, v = 1..N_{vs}\}$. Similarly to the training set, they are reduced if required respectively for generated detection models evaluation and validation.

In the idrs framework, detection models construction, using machine learning or other techniques, is preceded by a feature selection step. A wrapper approach is adopted in this step where the learning techniques involved in generating detection models are also included in selecting relevant features for them. For each log type j, its detection model subset DM_j includes N_j models, $DM_j=\{M_i / i=1..N_j\}$. Every detection model, $M_i \in DM_j$, is generated using a reduced feature set $F_i \subset F^j$, $F_i = \{f_l,\ l=1..P\}$, as further discussed in section §4.4. In addition, its reduced training, validation and testing data sets are derived using the projection function, $\pi()$, its selected feature subset, F_i, and datasets of log type j respectively, $TR_{i,j} = \{(x_r, y_r) / x_r =< x_1^r,...,x_p^r >, y_r \in C_i, r = 1..N_{tr}\}$, $VS_{i,j} = \{x_v / x_v =< x_1^v,...,x_p^v >, v = 1..N_{vs}\}$ and $TS_{i,j} = \{x_u / x_u =< x_1^u,...,x_p^u >, u = 1..N_{ts}\}$. A generated detection model using supervised machine learning technique has an output set, $C_i=\{c_q,\ q=1..N_{i,j}\} \subseteq C$. The latter enumerates classes of patterns recognizable by the detection model, M_i.

Additionally, in the proposed idrs framework, detection profiles are defined for constructed detection models. They are saved by the idrs knowledge base. Detection profiles subsets are required in fulfilling critical processes of the multimodel analysis component in the proposed idrs framework. For every log type j, the corresponding detection profile set, DP_j, is determined depending on its detection models in DM_j, $DP_j=\{P_i,\ i=1..N_j\}$. Each profile, P_i, is a data structure supporting generation, historical and performance information on the detection model M_i.

Generation information includes the representation of computed detection model such as a rule set, tree structure or others, as illustrated in chapter 6. It additionally concerns the output

set, relevant feature subsets and learning set of the generated detection model. Relevant feature subsets of a detection model, M_i, are determined depending on its output classes of C_i and the training set of considered log type, TR_j. For every output class, $c_q \in C_i$, its relevant feature subset, $F_{i,q}$, is derived from attributes of log type j, F^j, based on a wrapper approach. Selected feature subsets of different output classes in C_i, $\{F_{i,q},\ c_q \in C_i,\ q=1..N_{i,j}\}$, are then involved in specifying the overall set, F_i. The latter will serve in generating detection model M_i as subsequently detailed in section §4.4. The learning set of a detection model, $\zeta_i \subset TR_{i,j}$, includes all recognized data instances of this model when tested using its training set, $\zeta_i = \{(x_r, y_r) / (x_r, y_r) \in TR_{i,j}, M_i(x_r) = y_r, r = 1..N_{tr}\}$. Learning sets of detection models are required by different processes of the proposed idrs framework, specifically detection models fusion within the multimodel analysis component, as discussed in section §4.6.5. They are also revised using historical data sets of generated detection models and involved in updating them.

Historical information focuses on previous experiences of detection models. The historical data set, H_i, of each detection model, $M_i \in DM_j$, summarizes its earlier involvements in analyzing log data. It saves processed log data examples at different time points and associated assessments, both of the participating and combined detection models. The historical data set of each detection model serves in dynamically revising its learning set and then in reinforcing its detection capabilities. This is detailed in a subsequent section of chapter 4. Additionally, historical data sets are also included in updating testing performances of detection models.

Performance information of a detection model summarizes its detection effectiveness using corresponding validation and testing data sets. Relative scores of detection models, $sr_{i,j}$, considered in our idrs framework, express their performances. They are assessed using validation and testing confusion matrices of detection models for included log types as further discussed in section §4.4. Moreover, the relative score of a detection model is revised each time it is involved in an analysis task. In such case, the testing confusion matrix of the detection model is updated using its historical data set. For every detection model, M_i, validation and testing confusion matrices respectively VM_i and TM_i are saved by its profile. They are involved in different computation steps of the multimodel analysis component including selection and fusion of detection models.

A detection profile saves all above discussed information. For each log type j, computed profile set, $DP_j = \{P_i, i=1..N_j\}$, depends solely on generated detection models for that log type, $DM_j = \{ M_i, i=1..N_j\}$. Each profile, $P_i \in DP_j$, concerns a single detection model M_i, $M_i \in DM_j$. It is represented by a multidimensional data structure, $P_i = <M_i, C_i, \{F_{i,q}, q=1..N_{i,j}\}, F_i, \zeta_i, H_i, VM_i, TM_i, sr_{i,j}>$, that saves required information on the corresponding detection model. Detection profiles are built and updated by the generation component of the idrs framework, discussed later in this chapter. Furthermore, they are required by processes of the multimodel analysis component of our framework in evaluating the security state of the monitored system, selecting detection models and processing log data as discussed in chapter 4.

In the proposed idrs framework, analysis and detection process performs in near real time. At each time point t, log data of different types are collected and preprocessed by the corresponding component of the idrs framework. The resulting data set, $D_t = \{x_{t,1}, \ldots, x_{t,j}, \ldots, x_{t,J}\}$, is analyzed using dynamically selected detection model subsets, each of which concerns a single log type. Initially, a subset of candidate detection models, $\{CS_j / CS_j \subseteq DM_j, j=1..J\}$, is determined for each log type. Its candidates are already built using selected feature sets of the considered log type. Then, the most effective combination of these models, $S_{t,j} = \{M_i / M_i \in CS_j\}$ and $S_t = \{ S_{t,j}, j=1..J\}$, to process currently available log data of type j, are identified relying on an integrated criterion. The latter includes environment and model dependent factors that focus respectively on attack signs and performances of detection models, as respectively explained in sections §4.4.4 and §4.4.

In the designed multimodel analysis component of the idrs framework, every data example of given type, $x_{t,j} \in D_t$, $j=1..J$, is processed by detection models of the corresponding combination, $S_{t,j} \subset S_t$, $j=1..J$. Detection models are assumed independent and providing outputs at the abstract level $\{ c_q / c_q \in C_i, i=1..|S_{t,j}|, j=1..J\}$. They are fused to derive the final decision of the combined detection model on currently logged and analyzed activities of the monitored computing environment. Different fusion levels are discussed in section § 4.6, among these, the decision level is only being considered by our combined detection model. Furthermore, many fusion methods operating at the decision level have been experimented in previous works [184], [185], [412]. However, evidential combination methods are specifically considered in our idrs framework. This is due to two main reasons. On one hand, evidential fusion rules are useful in dealing with uncertainty in detection models decisions and including context dependent information within the combination process. On the other hand, most of

them are appropriate in coping with conflict situations between selected detection models. The proposed combination method relies on the Transferable Belief Model, TBM, of Smets. It fulfills two fusion stages within and between selected combinations of detection models, $S_{t,j} \subset S_t$, $j=1..J$. The main steps of the fusion process are presented in detail in section §4.6.5 of the next chapter. For each processed log data set, D_t, the combined detection model provides a detailed decision. The latter corresponds to the assigned label to analyzed system activities and its confidence value. Multimodel analysis component decision on currently reported and processed security state of the monitored computing environment is then forwarded to the response component.

The combined detection model decision is critical to risk driven response component of the proposed idrs framework. It is required to evaluate risk position of the monitored system and design cost effective response programs against attacks facing it. As such, a risk model is developed for the response component of our idrs framework. It focuses on assessment and treatment of computing environment risks. Furthermore, it uses several parameters in addition to the aggregated decision of the multimodel analysis component, interpreted in this context as the probability of threat occurrence. The designed risk model is an extension of generic models such as Annual Loss Expectancy (ALE) of DoD and risk methodologies of National Institute of Standard and Technology (NIST), to meet requirements of the monitored computing environments [149], [325]. It includes two additional components, implicitly considered by generic models, in assessing risks of the computing environment, as summarized in section §5.2. These components concern respectively the severity of supported vulnerabilities and effectiveness of operational security controls of the monitored computing environment and its assets. The former focuses on potential extent of harm caused by mounted attacks known that the gravity level of supported weaknesses by assets of the target system. The latter component instead takes account of reduction effect of deployed security countermeasures on computing environment inflicted damage by mounted attacks.

In this work, the designed response component is based on normalized risk management model. The latter is mainly inspired by the risk management standard of ISO-27005 and other risk standards and security guidelines [149], [173], [175], [325]. It includes two major and interdependent parts of risk assessment and risk treatment, as recommended by ISO-27005. The first part relies on two steps respectively of risk parameters identification and evaluation in order to assess the basic risk of the target system when attacked. In this report, we interchangeably use risk parameters or risk elements. The second part complements the first

one. It aims at treating assessed risks through choosing a treatment option and implementing its underlying actions.

In the proposed model, the first step of the assessment part focuses on identification and determination of different risk parameters relying on included risk components. In this step, six risk elements are considered in determining risk of the monitored system and assessing their monetary values. They are structured into three main groups of risk exposure, vulnerabilities severity and controls effectiveness. The exposure group concerns indirect risk elements, namely, asset value, impact and threat likelihood. These parameters are involved in determining risk exposure, the potential damage due to mounted attacks, of the monitored system or its target assets. Vulnerability and control groups instead include direct risk elements to estimate respectively the gravity of potential exploit of supported flaws and efficacy of deployed security safeguards. For each identified direct or indirect risk element, dependent variables involved in its evaluation are also determined in this step. All these risk elements are discussed in detail in sections §5.2 and §5.3.

In the proposed risk model, we assume that the monitored computing environment is composed of N assets determined by the corresponding set $A=\{a_i,\ i=1..N\}$. Assets support multiple flaws identifying then the vulnerability set V, $V=\{v_j, j=1..M\}$, of the computing environment. Additionally, the monitored computing environment has different security controls that form its initial protection strategy, $SS_0=\{s_l, s_l \in SC\}$, SC the set of L possible security controls, $SC=\{s_l, l=1..L\}$, given by the Annex A of the ISO-27001. Each asset, a_i, of the computing environment is in turn determined by its set of vulnerabilities and an initial control subset, respectively, $V_i \subset V$ and $SS_{i,0} \subset SS_0$. The latter includes dedicated or common controls considered in protecting against mounted attacks of the threat set $T=\{c_q,\ q=1..Q\}$ and $T \subset C$, the set of possible output classes.

The second step of the risk assessment part focuses on basic risk evaluation for each target asset and then the computing environment. It takes account of risk parameters identified and determined in the latter step focusing on risk elements of exposure, vulnerability and control groups. In the evaluation step, basic risk, R_i^B, of each target asset, a_i, is estimated based on its exposure due to detected attack, c_q, vulnerabilities severity and controls effectiveness that respectively define parameters $X_{i,q}$, Y_i and $Z_{i,0}$. Therefore, it is expressed as a function of these parameters, $R_i^B=\Phi(X_{i,q}, Y_i, Z_{i,0})$ as detailed in section §5.4. The exposure determines the

potential loss inflected by the detect attack, c_q, on the target asset, a_i. It is also estimated relying on the impact of the victim, I_i, and the likelihood of detected threat, $L_{i,q}$, as stressed by defined function to this aim, in section §5.4. The impact element assesses the potential loss of an asset due to its supported vulnerabilities, V_i. It takes account of the asset value and impact factors of determined categories of vulnerabilities depending on their exploit goals. The likelihood of a detected threat expresses to which degree it is successful such that the normal function of the target asset is affected. It is estimated based on the combined decision of the multimodel analysis and detection engine and the failure rate of operational security controls. The latter rate expresses to which degree functional controls are incapable to defend against the exploit of supported vulnerabilities by the detected threat. Vulnerabilities severity and controls effectiveness are both determined based on supported flaws of an asset. The former risk element solely considers flaw groups, determined using exploit gravity criterion, and severity scores of associated vulnerabilities. The latter element takes account of different vulnerability categories, identified relying on exploit goal criterion, and estimated efficacies of operational security controls in defending against supported flaws. Expressed functions and designed processes to evaluate basic risk and associated parameters are presented in detail in section §5.4.

Appraised basic risk of the monitored system, R^B, due to mounted and detected attacks is then treated in the second part of the proposed risk management model. The mitigation option is chosen in this part. It aims at reducing the assessed basic risk such that the resulting residual risk, R_k^R, of the computing environment doesn't exceed a prespecified tolerance level, $\bar{\tau}$, with respect to allocated security budget, β. The residual risk of the computing environment, R_k^R, corresponds to unmitigated risk after applying selected control combination or a security strategy, $SS_k=\{s_l, l=1..L_k\}$and $R_k^R = R^R(R^B, SS_k)$. In our response component, designed risk mitigation process is capable to determine the optimal security strategy, SS_k^*, that minimizes both the residual risk and security investment cost. Therefore, a minimization program was developed with this aim. The risk cost function considered by this program takes account of both damage cost and remediation cost. Two main constraints are associated to the risk cost minimization program. They ensure for each designed security strategy that reached residual risk and cost of selected controls do not exceed respectively the tolerance level and the security budget.

The developed minimization program considers effects of both installed and added security controls on risk cost. Optimal security strategies determined by this program to defend against detected threats are cost effective. Furthermore, they include appropriate controls from those recommended by security experts for detected attacks. Different controls of designed response strategies and expert recommended subsets are extracted from the Annex A of the ISO-27001, the set of possible security countermeasures applicable to the information security domain. These two parts of the proposed risk management model and their associated steps are thoroughly discussed in the respective sections §5.3, §5.4 and §5.5, of the chapter 5. Following section presents the framework structure that supports proposed improvements and copes with the challenges of future idrs systems.

3.2.2 Architecture

New requirements of future idrs systems, previously introduced, are supported by components of the following architecture.

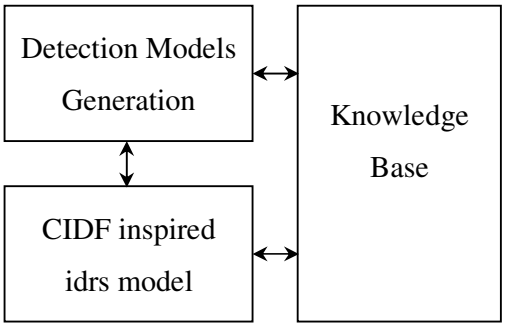


Figure 3.1: Idrs framework structure

Basic components included in this structure deal with critical tasks of preparation, action and reaction of idrs process. Detection models generation component fulfills preparation task for the idrs system. It generates detection models relying on reduced training sets, with respect to selected feature subsets, and chosen learning techniques. Knowledge base component assists in the achievement of various tasks within different steps of the idrs life cycle. Moreover, it performs additional tasks to maintain up-to-date and shareable idrs knowledge. In our framework, knowledge base component has a distinctive role to other components in that it ensures integration of domain knowledge in the idrs and its core process. It revises, evaluates and maintains an environment dependent and global knowledge. Idrs knowledge is critical for assessing security state, approximating inflicted damage and designing response strategies.

Detection models generation and knowledge base components represent two extensions of CIDF framework.

CIDF inspired component of our architecture focuses on different processing steps of idrs system. In the proposed idrs model, structural recommendations of CIDF are fulfilled and its components namely Event, Analysis, Response and Database-boxes are considered. However, in this work a special attention is given to critical components of CIDF including analysis and response boxes, regarding above defined problem. A multimodel approach is adopted in designing A-boxes. Moreover, a risk driven approach is involved in modeling R-boxes. Both components and their associated processes are respectively detailed in chapters 4 and 5.

Known the CIDF framework, the above presented architecture enhances this generic idrs structure. It extends basic components of CIDF by adding P-boxes and K-boxes respectively for preparation and knowledge base components. The former component concerns general conditions and basic steps towards computing appropriate detection models. The latter instead focuses on the integration domain dependent knowledge in the idrs process. Six instead of four components of CIDF is well adapted and extremely useful to satisfy requirements of future generation of idrs. Furthermore, it ensures appropriate background to support further idrs improvements required for evolving computing environments including high speed network and parallel systems.

3.2.3 Intrusion detection and response life cycle

The proposed idrs framework, through its different components, involves following steps' activities:

i. Preliminary traffic analysis and idrs model generation
ii. Security analysis towards feature based models selection
iii. Fusion of selected models
iv. Real time risk driven security or response program design
v. Knowledge creation and policy based evaluation and validation of idrs knowledge

Previously discussed idrs components are concerned with these activities that cover preparation, detection, response and revision steps during the idrs process. They participate in idrs life cycle by fulfilling single or several of its activities. Initial step's activities concern detection model development component. The latter preprocesses training datasets in order to

identify most appropriate predictor features. Then, training datasets are reduced regarding selected feature subsets. The resulting data sets serve next in constructing detection models, which are saved for subsequent use.

Next three steps of the life cycle are about activities achieved by two main components, analysis and response, of the idrs framework. The multimodel analysis component track signs of malicious events and control changes of relevant features for any tested data set. It identifies subsets of feature exhibiting remarkable changes to saved references. These feature subsets are then involved in selecting different detection model combinations to analyze collected datasets and explain the origin of supported changes. Afterwards, the analysis component combines subsets of selected detection models at the decision level. Decisions made by the combined detection model and other parameters are required to the risk driven response component. The latter assesses damage incurred as a result of detected attacks. Depending upon the reached damage level, it designs most appropriate response strategies to counter detected attacks and reduce subsequent risks of the computing environment to an acceptable level. Additionally, it is capable to selectively revise the security program of the monitored system by considering its current security state and the decision of SSO.

Knowledge base component mainly focuses on the last step of the idrs life cycle. It copes with the creation and revision of idrs knowledge, which is required by components of the idrs framework and possibly by other security systems. Derived knowledge concerns, for instance, detected attacks, their inflicted damages or implemented corrective actions to rule them out. Idrs knowledge can be revised if new pieces of information become available. Revised knowledge includes updated detection models, if appropriate datasets are collected or their learning sets are updated, and control combinations, when new applicable countermeasures are available or others are developed. Additionally, knowledge that concerns computing environment assets, supported vulnerabilities or other entities of the framework can be also acquired, saved by the idrs knowledge base and revised. However, before involving created or revised knowledge in different tasks of the idrs process, it should be evaluated and validated. In our idrs framework, knowledge revision, evaluation and validation are policy based activated. Knowledge revision, evaluation and validation policies specify requirements, conditions and rules that ensure effective implementation of any of these activities if required.

Different activities within each step of the idrs life cycle will be detailed in this report. The first and last steps of the life cycle are briefly discussed respectively in the next two sections.

Remaining steps that represent the main focus of this work are introduced in subsequent sections of the current chapter and deeply addressed in the next two chapters.

3.3 Detection model generation

Detection model generation component of the proposed framework fulfills two main activities. On one hand, it preprocesses and formats collected log data with respect to the prespecified time window. On the other hand, the generation component computes detection models after performing additional actions including feature subsets selection and preprocessed log data reduction, as illustrated by figure 3.2. Both activities are critical in achieving the initial step of the idrs process. They are selectively performed either to construct new or update existing detection models. In these cases, the SSO and Knowledge base component are involved.

Idrs preparation tasks, associated with model generation component, are identically performed for available data sets of different log types. Processed data sets include various features. They can be issued from several sources or derived according to specificities of collected logs. They are grouped into multiple subsets, as mentioned before in section §3.2.1. These datasets can be labeled or not depending on their utilization, training, testing or validation, and considered learning techniques. Moreover, they may include normal or anomalous log data examples, as categorized by DARPA [190], or a combination of these.

Within model generation component, each data set of given log type includes three subsets of training, testing and validation respectively TRj, TS_j, and VS_j, $j=1..J$. As their names already suggest, these datasets serve respectively to the construction, evaluation and validation of detection models. They support both normal and intrusive log examples. These datasets are unlabeled except training dataset that should satisfy the requirements of selected learning techniques and prespecified output sets for detection models, $C_i \subseteq C$, $i=1..N_j$. Furthermore, all three data sets are reduced before the generation of detection models using their selected feature sets.

Diverse artificial intelligence and data mining techniques may be adopted by the model generation component, as discussed above. But in this work, commonly applied classification techniques, in previous ids researches, such as decision trees, support vector machine, nearest neighbors, and others are considered. They are useful in building heterogeneous detection models for our idrs framework. Constructed models are also complementary and focus on different aspects that concern normal and intrusive behaviors. Therefore, their fusion

improves the discriminative power of the resulting combined detection model unless conflicts between them are resolved. Furthermore, it enhances reporting capabilities (expressiveness, explanation) and boosts global accuracy and precision of the idrs.

For every log type, $j=1..J$, the model generation component takes account of the corresponding training set and selected learning algorithms and feature subsets in building its detection model subset, $DM_j=\{M_i,\ i=1..N_{i,j}\}$, $j=1..J$. Relevant feature subsets for detection models are extracted from the whole set of the given log type. They are identified using either selection criterion or adopted learning techniques. They are also restructured into different subsets depending on considered output classes and involved in other activities of the idrs process, as discussed in section §4.4. The common process of the detection model generation for a single log type is illustrated by the diagram of figure 3.2.

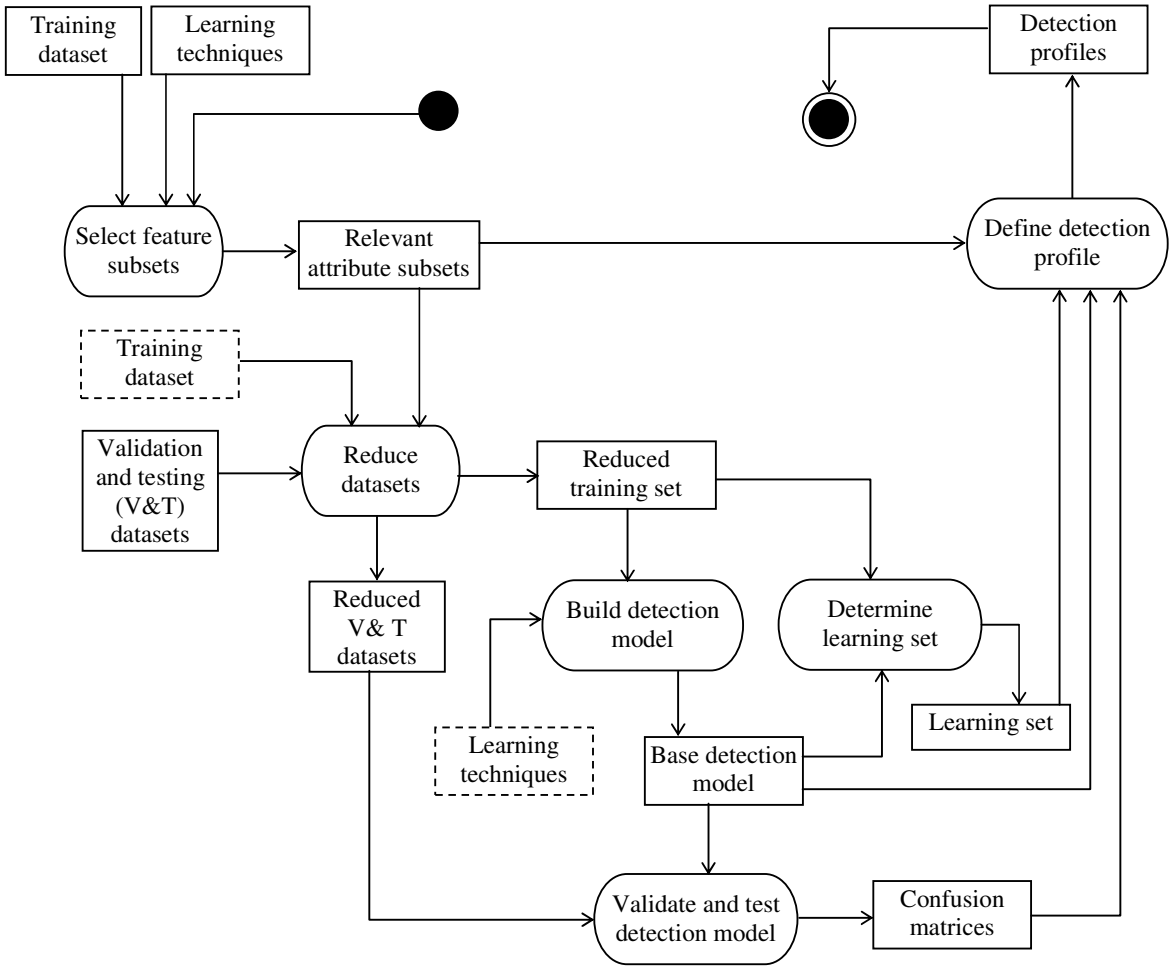


Figure 3.2: Detection model generation process

Preliminary log analysis step, as stated in the idrs life cycle, covers two elementary and sequentially performed activities of feature selection and data reduction. Feature selection is the initial activity of model generation process. It aims at reducing dimensionality of log data sets before processing them. It can be implemented based on two main approaches of filter or wrapper selection. The former uses different criteria including information gain, gain ratio or others, as discussed in section §2.2.1.1. Whereas, the latter relies on adopted learning techniques to determine relevant feature subsets involved in building detection models. In our framework, both approaches are useful to detection model generation component. Furthermore, the two approaches can be adopted in implementing feature selection activity of this component's process with respect to used artificial intelligence techniques to generate detection models. However, this work takes account only of classification techniques. Moreover, as stated in the literature, the wrapper approach always outperforms filter, in terms of accuracy, in different domains, including pattern classification [32], [78], [155], [383]. Therefore, this approach is adopted for detection model generation component. Additionally, selected feature subsets using wrapper approach are involved in the second step of the idrs life cycle, feature based detection model selection, as detailed in section §4.4.

In the depicted diagram, feature subsets are selected using included learning techniques. Based on selected feature subsets, training, testing and validation datasets are reduced as required by next activities within the generation process. A detection model is then built by applying the selected learning technique on reduced training set. Afterwards, it is validated and evaluated using unlabeled examples of respectively validation and testing reduced datasets. At this step, confusion matrices of generated model are computed to be further included in assessing its performance, as discussed in section §4.4. Additionally, the learning set of each detection model is computed by a separate activity in the process of the generation component. Built detection model, its relevant feature subsets, confusion matrices, learning set and other information are then included in its profile. The latter is saved in the idrs knowledge base.

Presented process of the detection model generation component can be enhanced by including additional activities. For instance, data filtering activity performed before feature selection may be required for this process. It eliminates outliers from training datasets and provides more appropriate feature selection results than before. Another activity that consists of structuring given data sets depending on specificities of learning algorithms seems extremely useful for this process. It aims at providing selected algorithms with appropriate datasets

before reaching the feature selection activity [169]. These extensions and others may improve capabilities of detection models and enhance their profiles in the knowledge base. The latter component and its management process are succinctly discussed in the next section.

3.4 Knowledge base management

Knowledge is defined as actionable information or information in a context. It corresponds to validated information which is required in performing actions or fulfilling tasks in a given context. Different kinds of knowledge are distinguished including shallow or deep and tacit or explicit depending respectively on a memory term to acquire knowledge (short term memory) and ease of codification (or verbalization) [29], [281]. Knowledge is created, verified and validated before transferring them into the knowledge base. These steps form the knowledge management process that allows useful knowledge to resolve domain specific problems.

Although, main processes of information security systems are based on expert and domain knowledge, a minority of these systems support knowledge management facility as one of their basic components. As an example in ids systems, the analysis and detection processes use knowledge discovered from collected log data. However, existing ids systems include databases instead of knowledge bases, except those based on expert systems. The core difference between a knowledge base and database is that the former supports an inference engine which is lacking the latter.

Knowledge base consists of rule and fact bases and an inference mechanism. The inference engine relies on a rule base to revise existing or acquire new knowledge. Such rule based approach in designing knowledge base is the oldest and the most widely adopted in capturing expert knowledge. Whereas, the last decades, ontology based approach has gained great interest in knowledge representation. Ontologies specify formal information models of concepts of given domain and the relationships between them [316], [407]. They are easily understandable by computer users and security personnel. Moreover, they offer required flexibility and upgradeability in representing knowledge and acquiring new concepts within the target domain [234], [235], [318].

Recently, benefits and flexibility of ontologies have attracted considerable attention in the information security domain. Multiple experiments have been carried out in this domain to derive useful ontologies to classify attacks [302] and [316] or design knowledge bases [77], [234], [235], [370]. Ontology based knowledge bases are designed in order to reinforce the management of information security specifically after the publication of the ISO-27001 that

focuses on ISMS (Information Security Management System). Following this standard and knowing high intricacy of currently developed security systems, knowledge base become a critical component of these systems not only for reinforcing security management but also for enhancing technical security, including intrusion detection, vulnerability and threat assessment.

Ontology based knowledge base is effective and well adapted to our framework. Based on domain semantic captured by ontologies, it is capable to provide idrs components and their associated activities, in the idrs life cycle, with required knowledge. Additionally, idrs knowledge can be easily shared with other security systems and revised by the SSO or security experts.

Ontology proposal for our idrs knowledge base component should be based on previous works. It enhances existing ontologies in information security domain such that new concepts, dependent to intrusion detection and response, and relationships between them are included. It is derived through a thorough revision of idrs components and their entities. The scope this ontology is confined to solely concerns activities of the idrs life cycle, above discussed. Main concepts and relationships of the intrusion detection and response ontology are depicted in figure 3.3 below.

Elements of the idrs ontology cover different idrs processes including detection model generation, analysis and detection and reaction. In this ontology, an *asset* represents any kind of worthy resources of the monitored computing environment. It supports several vulnerabilities. *Vulnerability* represents any weakness that can serve as a source of compromise or exploit. Obviously, an *exploit* concerns single or multiple vulnerabilities of the target asset. It is used by a malevolent entity to mount an attack. An *attack* consists of a combination of an agent (entity), malicious actions and a fixed goal. It manifests through leveraging these sources of compromise. Both, malicious actions of attacking agent and expected behavior of an asset are represented by *detection profiles*. Basically, a detection profile includes designed model that captures specificities of given intrusive or normal actions. Additionally, to reduce damage or decrease the realization chance of an exploit, a *security strategy* is implemented to protect computing environment assets. It consists of a combination of security controls, which are effective against potential realization of an exploit. A *control* is a security countermeasure that has corrective or preventive effect on exploits realization. *Risk* element concerns incurred damage by an asset regarding an exploit

realization and implemented security strategies. Historical or *residual risks* save traces on previously inflicted damages and their treatment strategies.

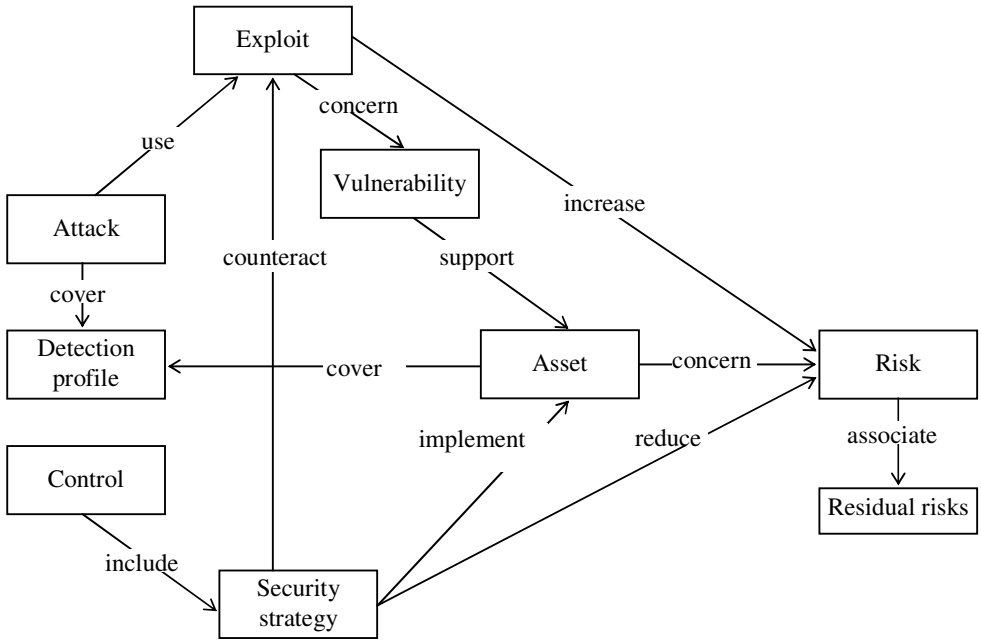


Figure 3.3: Idrs concepts and their relationships

The presented concepts can be extended. Additional entities of intrusion detection and response field or common with other information security fields can be considered in the idrs ontology. Moreover, other concepts associated to information security standards, guidelines and best practices are possibly included in the ontology.

Ontology based idrs knowledge base design and implementation process includes several steps. Automated tools associated with the ontology integrated environment such as Protégé supports these steps [131], [309]. Protégé environment is an open source platform associated with useful tools for editing ontologies and building ontology based applications. This platform integrates OWL (Ontology Web Language) a formal language for encoding ontologies to knowledge bases. OWL ontologies are expressed as a hierarchy of classes, instances and proprieties, the main OWL concepts. OWL classes focus on ontology concepts and group their associated domain instances or objects into different sets. OWL proprieties or relations are binary and link instances between them or to data types. Protégé platform ensures also data introduction for the knowledge base. The data can be collected from monitored systems, public databases or also security standards. Protégé can be extended by various inference engines, including Jess and CLIPS engine, that rely on rules deduced from encoded ontologies or introduced manually [84], [131], [305]. Security experts and SSO can

define their own rules and encode them to the inference mechanism through tools such as DLRule and SWRL (Semantic Web Rule Language) that focus on OWL hierarchies [76], [106], [120], [129], [407].

The idrs knowledge base supports knowledge created by processes of the inference mechanism. Before applying acquired knowledge, it should be evaluated and validated. Policy based evaluation and validation strategy of idrs knowledge relies on two main steps. The evaluation step tests idrs function with consideration of created knowledge. It aims at assessing the consistency of newly derived knowledge. The validation step instead focuses on idrs performance after integrating acquired knowledge. The selection, use and interpretation of decision criteria associated with both steps should be also policy based fixed. Moreover, the evaluation and validation policies explicitly determine performing requirements of each step's process.

Knowledge base design and implementation for idrs include many complex tasks. Discussed concepts and relationships are the core elements of the idrs ontology. The latter requires further extensions and deeper enhancements to effectively meet key requirements of idrs systems. Furthermore, idrs knowledge base development requires more effort to come up with appropriate creation, evaluation, and validation processes. Additionally, the knowledge creation process should cope with automatic (inference and idrs processes) and manual knowledge acquisition. Whereas, evaluation and validation processes should deal with knowledge appropriateness and system efficiency. Further investigations of all these processes may be extremely useful for idrs and other security systems supporting knowledge base component.

3.5 CIDF inspired idrs model

The proposed idrs framework complies with CIDF standard. The common structures of different CIDF components are included in our model, as presented before, to ensure required modularity and interoperability with other systems. However, core components of CIDF namely analysis and response, respectively A-boxes and R-boxes, only have received considerable attention in this thesis. Multimodel based A-boxes are designed in the proposed idrs framework. They rely on dynamic selective combinations of detection models to fulfill analysis and detection tasks. Their selection processes dynamically identify most effective detection models to be included in the current analysis task. Selected detection model combinations are then hierarchically fused within an evidential environment using the

proposed combination process. After that, the combined decision of A-boxes is forwarded to risk-driven R-boxes in order to select the most appropriate corrective actions. R-boxes are based on devised risk management model. The latter is compliant with risk standards of FIPS 65 and ISO-27005 and meets objectives of NIST guidelines, NIST SP800-30. It supports two main processes of risk assessment and risk treatment. The former identifies and appraises damages incurred by target assets of the computing environment due to detected attacks. While, the latter performs an optimization program to retrieve cost-effective response strategies that rule out detected threats and reduce inflicted risks to an acceptable level. Processes of A-boxes and R-boxes are detailed respectively in chapter 4 and 5 of this report.

3.6 Conclusion

In this chapter, we have introduced the proposed idrs framework and detailed its structure and the associated idrs life cycle. Furthermore, we have briefly discussed added components, namely detection model generation and knowledge base components, to CIDF framework, their usefulness and potential tools to deal with them. These two components enhance idrs systems at functional and interoperability levels. The first component extends the idrs process by including preparation tasks that emphasize the new requirement of existing and future idrs. While the second component focuses on the integration of dynamically changing instead of static knowledge in idrs and ensures their interaction with other security systems. In addition to structural improvements and their benefits for idrs, other functional enhancements that concern analysis and reaction components are introduced in current chapter and detailed in the next two ones. The multimodel analysis component of the proposed framework is presented in the following chapter. Only supervised classification techniques are considered in designing the multimodel analysis and detection engine and studying and modeling its processes.

CHAPTER 4

ADAPTIVE ANALYSIS AND DETECTION

4.1 Introduction

According to CIDF framework, intrusion detection systems should include single or multiple A-boxes. Analysis and detection components or A-boxes play a central role in idrs process. They process log data initially collected, preprocessed and formatted by event generators, E-boxes. Their decisions about the current security state of the target system are sent back to the response component or R-boxes of the idrs.

E-boxes provide A-boxes with required log data to perform analysis and detection tasks. They are implemented by sensors that collect log data at different levels including application, host and network. Moreover, E-boxes allow various types of log data depending on their granularity. Based on these two criteria, namely operation and granularity levels, several event generation mechanisms are distinguished. Main taxonomies and widely adopted classes of logging mechanisms are discussed in this chapter.

A-boxes are differently designed from an ids to another, but their main role should be preserved whatever their design and detection principles. They can be single model based A-boxes as commonly encountered in the majority of existing ids. In this case, analysis and detection components are called simple or basic A-boxes because no additional elements are required to implement them rather than detection models. However, complex A-boxes include multiple detection models and thus other elements to manage them are becoming necessary. Furthermore, A-boxes can be designed to track signs of given patterns or assess deviation from them or both. In this work, we propose complex A-boxes that dynamically and adaptively select detection models based on a preliminary analysis of collected and preprocessed log data. Following sections of this chapter deal with the architecture and different components of designed A-boxes.

Depending on included detection models, A-boxes output decisions take different forms and allow various levels of details. Abstract outputs or labels, either single or ranked labels,

provided by A-boxes summarize current security state and specify types of attacks targeting monitored system. Whereas numerical outputs of A-boxes assess the current security state of the monitored system (normality or intrusiveness of currently analyzed behaviors) and determine confidence or certainty degrees assigned to this assessment. Decisions of A-boxes, labels or confidence degrees, are required to R-boxes either to inform the SSO or to design and implement corrective actions against detected threat. In our idrs framework, A-boxes may be based on multiple machine learning, data mining or other techniques, as discussed in previous two chapters. However, in the current chapter, two types of machine learning techniques, namely the binary and multiclass classification, are solely considered in designing A-boxes of the proposed idrs framework. The outputs of binary or multiclass detection models are supposed at the abstract level. Multimodel analysis engine based on these detection models allows outputs at measurement level. Its valued outputs are forwarded to risk driven R-boxes in order to design appropriate response strategies depending on determined risk levels. Proposed risk model to R-boxes of our idrs framework is subsequently detailed in a separate chapter. The current chapter focuses on second and third steps of the idrs lifecycle that specifically concern multimodel analysis of log data and detection of intruder activities.

4.2 Proposed analysis component, structure and main processes

The proposed analysis component is based on a multimodel approach. It takes account of various subsets of detection models depending on log types included by the model generation component. Our multimodel approach is different to those in [102], [135], [139] [202] [204], [254], [288], [296], [330]. It considers several subsets of heterogeneous detection models instead of single or multiple subsets of homogenous detectors based on neural networks or other techniques. Furthermore, subsets of detection models in our approach are neither static nor duplicated to process considered datasets. They are dynamically and adaptively determined, as presented in section §4.4. Depending on processed datasets and specificities of generated detection models, these subsets are selected. Moreover, decisions of selected detection models are hierarchically combined using a bi-level fusion structure.

Additionally, our analysis component ensures many enhancements comparatively to existing in [135], [156], [222], [254]. It takes account of context dependent information, scores and reliability factors of detection models, throughout its components and their implemented processes. It is not designed to specifically operate at network or host levels [135], [156], [288] but it is capable to analyze datasets concerning same sequences of activities even if

issued from different sources. A typical example of the last alternative is host activities that may be reported by system calls of the Basic Security Module, BSM, of the Solaris system, or also by host based features extracted from network sensor logs, as discussed in section §4.3.

The multimodel analysis engine takes account of different log types. Indeed, log datasets of different types ensure several descriptions that concern same system activities to be analyzed. They can be issued from different sources or collected at various levels but they differently report performed activities within the same time interval. As such, varieties of log types may serve as inputs to the proposed analysis engine regarding the monitored system. For instance, in a network environment, activities can be traced in traffic logs and other log types generated by various sources including switchers, firewalls, routers and proxies. Additionally, traffic logs can be structured into different log types depending on their specificities, as discussed in the next section, by considering network services or layers of the OSI stack. Similarly, host activities can be also traced in several log datasets including those generated by operating system functionalities, such as process management, file system management, security management components [156], or other network based logs.

For every log type, its collected dataset is processed using a combination of appropriate detection models. Combinations for considered log types are feature based constructed. They include detection models that concern norm and anomalous behavior and thus the proposed idrs framework is useful both for anomaly and misuse detection. Anomaly detection models focus on the expected behavior of the monitored system. They extract different patterns from logged normal activities through considering various projection facets, including nonlinearity, association and temporal relations. Similarly, misuse detection models identify abnormal conditions associated to logged activities of intrusive behavior affecting to the normal function of the monitored system. They are constructed based on available datasets of known attacks.

Several techniques are useful to build misuse and anomaly detection models for the idrs framework, as presented next in the section §4.3. However, in this chapter and for the sake of simplicity, one and multiclass supervised classification techniques are specifically considered in generating detection models. One class detection models determine whether current security state of the monitored environment is expected or of known type, depending on the target output class. While multiclass detection models assess to which degree processed event sequences are similar to computed signatures of concerned output classes.

Anomaly and misuse detection models are involved in different steps of the main process of our analysis engine. This process is performed for each collected and preprocessed log

dataset. It consists of three main steps namely selection, analysis and fusion, as depicted in figure 4.1. Each of these steps corresponds to a basic process of the analysis and detection engine. The selection process identifies combinations of appropriate detection models to be involved in current log analysis step, with respect to considered log types. It uses feature and model based criteria to identify best detection models. The analysis process performs à thorough checking of collected log datasets using detection model combinations of the previous selection step. It ensures that every selected combination is provided with the dataset of the corresponding log type. Moreover, within each combination, it formats the considered dataset as required by included detection models. After processing log datasets, output labels of participating detection models forwarded to the fusion process. The latter evaluates beliefs of involved detection models on selected output labels using a distance based approach. It then performs two combination levels on detection model output decisions. In the first level, decisions of detection models are fused within each combination, hence an aggregated decision is reached for every log type, as though a single detection model has processed the associated dataset. In the second level, fused decisions of considered log types are aggregated another time to derive an overall assessment of the current security state of the monitored system. This is required in designing defensive security strategies of R-boxes.

The main process of our analysis and detection engine enhances those existing by adding two processing steps. Commonly designed processes include two basic steps of log analysis and decision making. However, our analysis and detection process additionally supports the selection and fusion steps to meet requirements of the multimodel adaptive engine. Its elementary processes that respectively correspond to above enumerated steps are illustrated by figure 4.1. Subsequent sections of this chapter will focus on these processes. They present in detail respectively selection, analysis and fusion components of the multimodel engine. Before that, widely adopted log data types by ids are presented in the next section. Furthermore, their applicability and usefulness to the proposed multimodel analysis engine are also discussed in the same section.

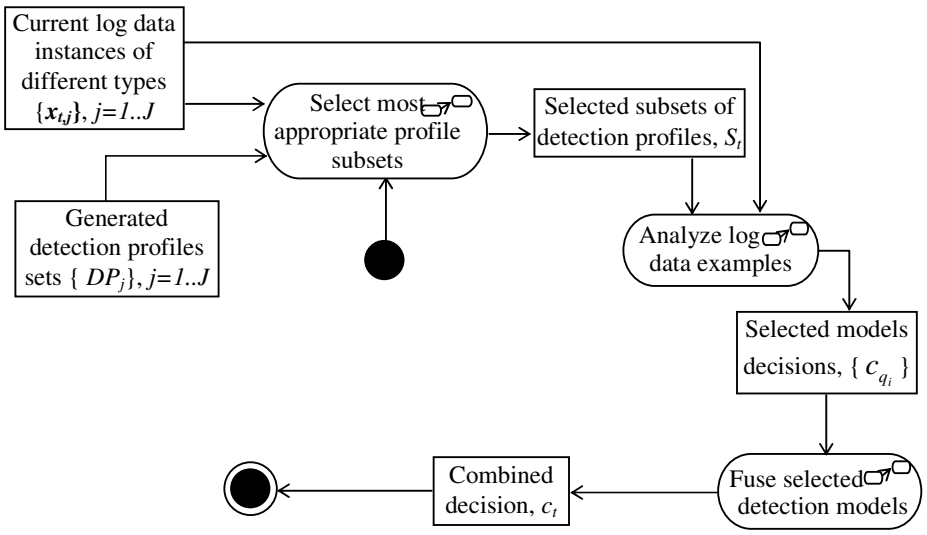


Figure 4.1: Analysis and detection process of the multimodel engine

4.3 Log data

Event generator or E-boxes of CIDF is the initiator of the idrs process. According the CIDF framework, E-boxes accomplish event generation within the idrs process by preprocessing and formatting log data produced by the computing environment. Their produced events are specifically sent back to the security state evaluator or A-boxes of the idrs. They concern use conditions and performance indicators of the monitored computing environment. These events are extracted from raw log data collected and recorded by different data collection mechanisms or sensors.

Depending on sensor attributes, several taxonomies of data collection mechanisms for idrs have been proposed. Moreover, collected data by these mechanisms can be structured into different formats, either normalized or proper, to appropriately satisfy requirements of A-boxes.

In the current section, we will present three of the most widely adopted classes of data collection mechanisms in idrs systems. Included features, normalized format and applicability of these sensors in our idrs framework are also discussed in the same section.

4.3.1 Sensor taxonomies

Data collection facilities for idrs are widely studied and surveyed such as in [30], [93], [191], [213], [217], [297], [427]. They are classified into different groups relying on their intrinsic features and the environment where they operate. In [427], Zamboni has proposed different taxonomies, each of which uses single decision criterion and identifies two classes of sensors. These taxonomies were structured based on the abstraction level into conceptual and practical classifications. At the conceptual level, idrs sensors can be considered as centralized or

decentralized, according to their locations and components of the computing environment they monitor. Additionally, they can be categorized into direct or indirect sensors based on how log data are obtained. At the implementation level, data collection can be fulfilled by an integrated or supported part of the monitored component. Then, it identifies respectively internal and external sensors. Moreover, data collection facility can be implemented to acquire logs of single or interconnected hosts. Through discussed taxonomies, Zamboni's main finding states that host based data collection mechanisms are the most useful and suitable for idrs. Furthermore, he has argued, specifically in this group, that all mechanisms are based on direct logging. Compared to the indirect group that includes network based mechanisms, host based internal sensors fulfill required completeness, reliability and scalability for idrs. But, this is not always reachable and depends on the environment where idrs sensors are deployed and the nature of potential attacks to be detected. For instance, in an open source environment host based and internal sensors can be implemented and deployed but they remain inefficient for detecting network denial of service attacks.

Larson [212] has proposed a detailed taxonomy of log data collection mechanisms. Four groups of sensors are identified in this taxonomy. They form the main branches of the taxonomy tree structure. The first group focuses on implementation techniques of sensors. It determines different sensor classes based on considered implementation aspects namely time points of triggering data collection mechanisms and levels of granularity of logs they provide. The second group characterizes data collection as triggered or performing on action mechanisms. Several triggers, actions and responses that concern sensor's behavior are considered by subclasses of this branch. The third group is based on location or where logging mechanisms can be inserted. User and system location levels are considered to discriminate between sensors of this group. Determined subclasses include application, system and object manipulation considered respectively as entry, exit and network data collection mechanisms. The last group in Larson's taxonomy presents most widely discussed and adopted subclasses of sensors even in classifying idrs. Sensors of determined subclasses are merely categorized as application, host and network based collection mechanisms [33], [94], [146], [400]. All three types of sensors will be discussed in the following section with special attention given to their collected data, generated profiles or patterns and their usefulness to our idrs framework.

4.3.2 Host based log data

Host based sensors collect different types of data that concern monitored host and its use conditions, invoked by users or programs. They are closely dependent to auditing mechanisms

of the operating system running on the target host. Audit logs collected by these mechanisms are not specifically dedicated to idrs but also useful to other security systems, such as access control and reinforcing accountability. They report user or program behaviors on the monitored host.

Host based audit logs have different abstraction levels as illustrated in figure 4.2. High level audit logs correspond to various events that describe currently performed activities using resources of the target host. These events concern, for instance, logging and disk management mechanisms of the host. They are generated by running programs. Furthermore, these events are captured and processed by operating systems and reported by auditing mechanisms. Auditing facilities of Windows and syslog of UNIX systems allow such high level audit logs.

Low level audit logs consist of sequences of system calls. They allow information about the implementation level of currently performed activities. Kernel and application calls that implement triggered activities on the target host are collected by low level mechanisms. BSM module of Solaris operating system is an example of the low level auditing facility.

In both levels, different features are appended to raw audit data including timestamp and identity of the agent, user or program. Audit logs provide useful information to idrs systems to track user or program activities on a single host. Moreover, they allow idrs systems to scale well with the increased load of the monitored system and react in a timely fashion as pointed by Zamboni [427]. Host based idrs uses different forms of audit logs, such as event logs, shell commands and system calls, to generate profiles of system, user or program normal behavior or given attack types [162], [191], [395], [400], [404]. Many of these systems are based on subsets of features included in raw data such as arguments of the system calls [143], [400]. Additional other systems use statistical and many derived features, computed using basic attributes of logged audit data. Designed profiles by these systems obviously involve frequencies of system call types or argument within given time intervals [111], [484].

In our idrs framework, the proposed analysis engine can operate at the host level and detect attacks that target single machine. Furthermore, it is well adapted to a thorough assessment of the security state of monitored host when more than single audit log type is considered. However, to detect network attacks, this engine requires an aggregated log dataset over all hosts of the network. Such alternative is seldom explored such as in [251].

Type	Date	Time	Source	Category	Event	User
Information	27/01/2010	22:43:48	Browser	None	8033	N/A
Error	27/01/2010	22:43:46	Dhcp	None	1002	N/A
Warning	27/01/2010	22:37:17	Dhcp	None	1003	N/A
Information	27/01/2010	22:36:48	Tcpip	None	4201	N/A
Information	27/01/2010	22:26:07	Service Control Manager	None	7036	N/A
Information	27/01/2010	22:26:04	Service Control Manager	None	7036	N/A
Information	27/01/2010	22:26:04	Service Control Manager	None	7035	System
…	…	…	…	…	…	…

(a) Example of system event log for Windows platform

close, close, open, close, close, close, close, close,
close, execve, open, mmap, open, mmap, mmap, munmap,
mmap, close, open, mmap, mmap, munmap, mmap, mmap,
close, open, mmap, mmap, munmap, mmap, close, open
mmap, close, open, mmap, mmap, munmap, mmap, close
close, munmap, open, close, ioctl, ioctl, close, close,
close, close, exit

(b) Sample of system calls of single process logged by the BSM module in DARPA experiments

Figure 4.2: Examples of high (a) and low (b) level audit data

4.3.3 Application based data collection

Application logs form two main classes of system and user application logs. System applications correspond to different functionalities of the operating system, specifically communication functionality. These functionalities called also services including HTTP, FTP and Telnet are associated with several communication ports. Activities of running services on different ports can be logged by high level data collection mechanisms associated with operating systems such as Syslog. User applications including web and database applications are also managed by operating systems and based on their services. Their behaviors are logged by auditing mechanisms of the operating systems. Additionally, user applications may support proper auditing facilities. Their sensors are capable to directly record and report different events involving application resources

User application sensors are considered by taxonomies of Zamboni as direct logging mechanisms. They are assigned to the group of dedicated auditing facilities that log high level events about user activities in target applications. Collected events by application sensors concern access types, required resources and additional information. They are appropriate in designing application or user normal behavior profiles to ids. Furthermore, they are useful in generating attacks signatures. However, direct application sensors have two main drawbacks.

On one hand, collected data are specific to given applications and hence they don't allow same description neither in terms of format nor in terms of details due to the particularity of considered features. On the other hand, application sensors are disconnected from any auditing mechanisms of operating systems and specifically report application events. Thus, application ids are vulnerable to various attacks that target environments on which these applications are running.

By revising the last weakness, we deduced that user level log data, collected by application sensors, are not sufficient to protect against low level attacks. In fact, running applications may be indirectly targeted by low level attacks. This type of attack aims at disrupting normal function of critical infrastructure, including network protocols and other basic functionalities of operating systems, for these applications. A possible solution to this drawback consists of the integration of application ids with host or network based ids to ensure collection and analysis of log data at both user and system levels [103], [153].

For such problem, our idrs framework offers the typical solution. It is capable to resolve above discussed drawback of application ids by including multiple log datasets. Analysis and detection process of the proposed idrs framework takes account of different traces, of performed activities, issued from direct and indirect application sensors and thus it is able to include logs at user and system levels. This ensures required security and low risks specifically for highly critical applications.

4.3.4 Network based data collection

Network sensors globally offer two types of log data to idrs that concern respectively network devices and traffic. They indirectly collect useful information about network devices relying on simple network management protocol, SNMP. Such Internet standard protocol ensures information exchanging between different network devices. It also assists administrators in managing their network infrastructure. MIB (Management Information Base) associated with this protocol is the main data collection source of sensors at this level. Saved information by MIB includes routing tables and various traffic counters that concern respectively configuration and performance of network devices. This information is indirectly collected by network sensors, formatted and then presented to A-boxes of idrs [93], [94], [217], [423].

Network traffic sensors intercept and collect local traffic within the network. Packet sniffers are the most widely adopted traffic sensors. Associated to the network interface, they passively listen and log exchanged packets across the network. They provide administrators and automated security tools with useful information about network activities after decoding

captured packets. Derived attributes from packet headers by sniffers form the basic or intrinsic feature set. The latter includes IP address, port number, transport protocol and other flags. Additional features can be determined using logged network traffic. For instance, content feature set can be generated relying on payloads of collected packets and their semantic interpretation. It groups features such as number of failed logins and number of accessed files. Moreover, time based or traffic features can be derived from the basic set by considering time intervals within which logged data are processed according to single or multiple time windows. Such set includes features like number of connections to the same host or service within the last two seconds. Another feature set that concerns host activities can also be deduced from intrinsic attributes. Its attributes are determined based on history or archive window that focuses on a fixed number of past connections involving the considered host.

Previously discussed feature categories have been proposed in the frame of DARPA intrusion detection evaluation program. Simulated network traffic by DARPA has been preprocessed in order to ensure training and testing datasets that cover all features of these categories [90], [189]. Additional attribute categories can be defined based on the intrinsic features either by focusing on a particular layer of the OSI stack such as in [283] or by considering new requirements of existing computing environments [96], [125], [152], [198].

Network traffic logs are more appropriate than SNMP logs for intrusion detection. In fact, the latter allows untreatable logs by computerized systems unless encoded to required format. However, the former provides a flexible log format that can be customized according to system requirements. Moreover, it ensures an extended description of network activities that partially covers different aspects discussed at host and application levels. Therefore, network sensors are almost the most suitable source of logs for idrs systems. They are also appropriate to the proposed idrs framework. Collected network traffic and identified feature subsets offer required redundancy to multisided and stringently track of attack signs. Furthermore, they are useful to idrs detection models to appropriately analyze and precisely evaluate the current security state of the monitored computing environment. However, wide availability of these sources of data from anywhere on the network increases their risks to be tampered. In fact, the Internet provides attackers with various and sophisticated tools to hide their deliberate actions and alter log files unless appropriate security controls are correctly deployed.

Above discussed sensors provide corresponding idrs with required log data. Furthermore, aggregation of their collected logs ensures more complete datasets as well as for designing patterns, for attack or normal behavior, as for identifying them. Aggregated log data could

enhance idrs capabilities. Additionally, they are useful not only for systems with which idrs cooperate, such as network forensic systems, but also for system administrators. In intrusion detection field, log data aggregation is facing various problems. Lacking of standards and platform independent formats for collected data at different levels are serious problems that stay behind difficulties of log data aggregation in intrusion detection.

Proposed formats such as of Bishop [48] and Chen et al. [75] focus specifically on log data at host level. Both formats are flexible, extensible and platform independent. Additionally, Chen et al.' proposed format is based on the XML (eXtensible Markup Language). Its structure and logged events are expressed using XML.

The Intrusion Detection Work Group (IDWG) of the Internet Engineering Task Force (IETF) body has proposed a common format for idrs alerts. The Intrusion Detection Message Exchange Format (IDMEF) was recently considered to express logs for correlation based idrs. It relies on the XML language. Moreover, it is useful to represent outputs of different sensor types. But, the IDMEF format addresses to alerts formatting and exchange procedures to enforce information sharing between different idrs. This has inspired several works in the field of alert aggregation and correlation, which has gained a lot of attention in the last decade [200], [251], [396].

In our idrs framework, intrusion detection alerts can be considered as metadata. They can't be processed by idrs analysis engine, as for collected log data. However, adapting the basic approach of this engine to alerts aggregation is one of promising research perspectives of the proposed idrs framework. Additionally, considering aggregated log datasets from different sensors at the analysis step of idrs is also a potential direction to enhance performance and co-integrate the proposed model with those of others security fields such as forensic analysis.

Commonly used log data types discussed in this section are useful to our idrs framework. Furthermore, their associated log datasets can be easily processed by E-boxes, P-boxes and multimodel A-boxes. The analysis process of the last component includes different steps that are thoroughly studied in the next three sections.

4.4 Detection model selection

Detection model selection is one of the key components of the proposed multimodel analysis engine in the idrs framework. It ensures dynamic and context dependent selection of detection models, as detailed in the next three sections. The model selection process, on which relies this component, aims at decreasing the computation load of the analysis engine. It fulfills preliminary tasks in order to achieve selective processing of collected log datasets. Thus, for

each log type, a combination of most appropriate detection models, rather than the whole set, is included in the current analysis task.

Model selection processes identify single or subset of most effective detection models with respect to prespecified, simple or integrated, criteria. They may be static or dynamic. Static selection determines a fixed subset of detection models to be involved in analyzing all unknown patterns. Dynamic selection instead is online performed. Furthermore, for each unknown pattern, it identifies most appropriate detection models, to label it [327]. Static and dynamic processes include two main steps of evaluation and selection. The evaluation step assesses capabilities of detection models depending upon selected criteria. The selection step finds out then a single or subset of best models with respect to these criteria.

Several metrics and search methods have been widely experimented in classifier selection. Performance and diversity metrics are involved in classifier subset selection [12], [65], [136], [326]. However, single classifier selection methods are solely based on performance metrics, such as in [205], [361], [410]. These metrics are commonly adopted also by ranking methods, which are useful, both for single and subset selection. Furthermore, ranking methods are appropriate specifically for subset selection when dealing with an increased number of detection models. Therefore, the proposed selection process ranks detection models based on their assessed scores before identifying the most effective subset to be involved in the current analysis task.

The proposed selection process for the multimodel analysis consists of different steps. Its initial processing steps focus on a thorough checking of changes in observed values of key features derived by the model generation component. Then, other steps fulfill multiple assessments by involving detection profiles. They aim at feature based identifying most effective detection model combinations, each of which concerns a single log type.

In the selection process, involved detection profile subsets of different log types are stored in the idrs knowledge base. Each profile saves detailed information about the target detection model including its selected relevant feature subset and confusion matrices, as discussed in section §3.3. Information about detection models is included in different steps of the selection process. Relevant feature subsets serve in conducting preliminary checking of collected datasets of different log types. Determined features as abnormal by these processing steps signal first signs of intrusive events in processed log datasets. Additionally, they are considered in computing global scores of detection models. Global scores involve both data dependent and performance factors. The former focuses on relevant features of a detection model and those flagged abnormal. It concerns the flagged feature factor determined for every

detection model, as discussed later in this section. This factor expresses to which degree a detection model is appropriate to process currently checked log data.

The latter factor takes account of performances of detection models. It can be expressed using different metrics including ROC curve and others detailed in [65], [66], [182], [361], [362]. But in our work, specifically global performance metrics are applicable because multiple heterogeneous detection models are considered and their outputs are assumed at the abstract level. Furthermore, the commonly available information about detection models' performances is extracted from their confusion matrices. Global performance metrics involved in assessing scores of these models may be gauged using validation and evaluation confusion matrices. Multiple, pairwise and non-pairwise, performance metrics are next discussed in the frame of detection model score assessment.

Global scores of detection models expressed in terms of their performances and flagged feature factors are required to the selection process. At any time point, the latter involves computed scores in ranking detection models of given log types. Then, it identifies subsets of appropriate detection models to be included in the current log processing task. The selection process and its main steps are depicted in figure 4.3.

The selection process continually analyzes security states of the monitored system relying on three checking steps. These steps fulfill a preliminary analysis of the security state of the monitored system. The first two processing steps are simultaneously performed for all considered log types. They aim respectively at conducting a preliminary assessment of current security state and identifying any abnormal changes to the expected behavior of the system. Thus, they respectively involve most efficient norm detection models and norm relevant attributes, as detailed later in this section. Identified signs of intrusion in these steps are thoroughly tracked in the next one. The last step conducts a stringent control of signaled changes through checking observed values of relevant features of different attack classes. It aims at collecting more evidences and providing a preliminary explanation about the origin of previously identified intrusion signs. Feature checking steps of the selection process involve relevant feature subsets and reference vectors respectively of normal and attack classes. Collected evidences, on security state and abnormal changes in relevant feature values, over the three previous steps are compiled before performing the selection step. They serve to evaluate and rank candidate detection models. The last step of the selection process is then conducted on ranked sets of detection models. For every log type, it decides which are appropriate and well adapted detection models to tell us the truth about the current security

state of the monitored computing environment and decipher hidden aims behind unexpected changes in log datasets.

Inputs to the selection process consist of different subsets of detection profiles, reference vectors of normal and attack classes and preprocessed dataset of currently collected logs of different types. Depending on considered log type, the corresponding detection profile subset specifies its detection models and their confusion matrices and relevant feature subsets, as detailed in chapter 3. Additionally, norm and attack relevant feature subsets for every log type are deduced from those selected by its different detection models. They determine reference vectors that characterize normal or intrusive behaviors. Reference vectors of normal and attack classes are computed based respectively on their training instances of given type. Currently collected and preprocessed log dataset includes data examples of all considered log types. Detection profiles, reference vectors, discriminative attributes and current preprocessed log dataset are required to the five step process of the proposed model selection component. These steps and their associated processes are subsequently detailed.

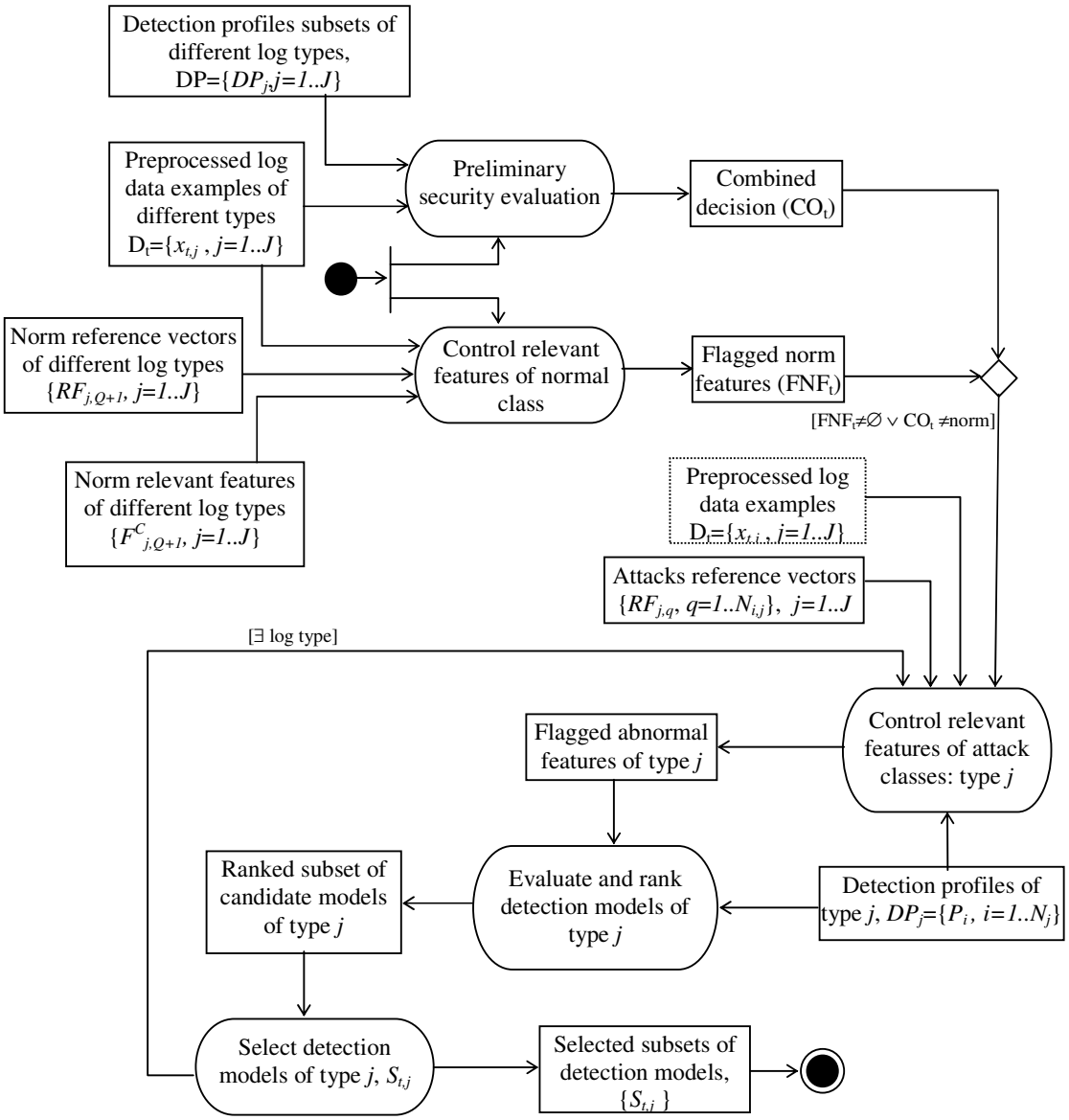


Figure 4.3: Detection model selection process

For these elementary processes, we assume that J log types are considered by our idrs framework. Each log type j is associated with different attributes that form the set $F^j = \{f_{l,j}, l = 1..L_j\}$ and therefore the set of available features for the idrs framework is F, $F = \{F^1,...,F^j,...,F^J\}$. Additionally, the set of possible output classes in the idrs framework corresponds to C. The set $C = \{c_1,...,c_q,...,c_{Q+1}\}$, includes possible attack classes, c_q, $q=1..Q$, and the normal class, c_{Q+1}. Considered attack classes are determined based on the reduced DARPA attack taxonomy. Each output class in C is associated with several reference vectors, each of which is computed using corresponding training instances of given log type, as discussed later in this section.

We suppose also that the set of generated detection profiles over all log types is DP, $DP=\{DP_j,\ j=1..J\}$. Additionally, the subset of detection profiles of every log type j $DP_j=\{P_i, i=1..N_j\}$, depends on its constructed detection models, $DM_j=\{M_i,\ i=1..N_j\}$. A detection profile, P_i, as discussed in the previous chapter, is a data structure built for every detection model, M_i. It saves information about generation, history and performance of the concerned detection model. Detection models of M_j are generated using different techniques and datasets and relevant feature subsets, $F_i \subseteq F^j$, $i=1..N_j$, selected from those of log type j. Furthermore, they are capable to recognize patterns of different classes determined by $C_i \subseteq C$, $i=1..N_j$ and $C_i=\{c_q,\ q=1..N_{i,j}\}$, the set of output classes of the detection model M_i. Relevant features, training sets, historical data and more details about detection models are included in generating their profiles of DP_j. Within each set, DM_j, detection models are evaluated and their relative and global scores are estimated, as discussed next in this section. Moreover, these models are structured into two main subsets depending on whether they are capable to recognize normal system activities or not. This is required to first steps of the detection model selection process as detailed respectively in sections §4.4.1 and §4.4.2. In this work, we interchangeably use DP_j and DM_j to denote detection models of log type j, when additional information about detection models is required, DP_j set is used. The following sections discuss relevant feature identification and score assessment for detection models and reference vector computation for included output classes.

a) Relevant feature subsets

In the proposed framework, feature space is partitioned into different categories depending on included log types, as discussed above. Construction of detection models of considered log types is preceded by a feature selection step based on the wrapper approach. Although, such approach is computationally expensive, compared to the filter approach, it is more appropriate when dealing with heterogeneous classification techniques, such as in our idrs framework. Furthermore, it ensures accurate detection results due to the validated effectiveness of selected feature subsets.

For every log type j, the detection model $M_i \in DM_j$ disposes of different subsets of most discriminative attributes derived depending on its output classes in C_i, $\{F_{i,q} \ / \ F_{i,q} \subseteq F^j,\ q=1..N_{i,j}\}$. Its relevant feature set over all classes is determined by F_i, such that $F_i = \bigcup_{q}^{N_{i,j}} F_{i,q}$ and $F_i \subseteq F^j$. The learning algorithm involved in selecting feature subsets for different output classes, $\{F_{i,q},\ q=1..N_{i,j},\ c_q \in C_i\}$, is also adopted to generate the detection

model M_i using the overall set F_i. Relevant attribute subsets of detection models, $\{F_i, i = 1..N_j, M_i \in DM_j\}$ are included in their profiles and saved in the idrs knowledge base. Additionally, selected feature subsets for detection models of given type are structured depending output classes in C. For every log type j, the relevant feature subset $F_{j,q}^C$ of the output classes, $c_q \in C$, $q=1..Q+1$, is deduced, such that $F_{j,q}^C = \bigcup_i^{N_j} F_{i,q} / c_q \in C_i$. It basically characterizes the output class c_q when dealing with log data of type j. It is involved in performing one of the feature control steps of the selection process. Moreover, the attribute set $F_{j,q}^C$ serves to compute the reference vector, $RF_{j,q}$, of the output class c_q in C relying on its training instances of log type j. Existing methods for determining reference vectors of output classes are discussed below.

b) Reference vectors

Reference vectors of output classes are required by different steps of the model selection process. They mainly focus on relevant feature subsets of considered output classes in order to globally summarize their training sets. In our framework, each output class is represented by multiple reference vectors depending on included log types.

Various techniques are useful to determine reference vectors of attack classes or system normal behavior. The sample mean is the simplest technique to derive reference vectors for the model selection process. In computing reference vectors, numeric features, continuous or discrete, are represented by their mean values over the training sample. Symbolic attributes instead are expressed by their median values in determined vectors.

$$RF_{j,q} = < \bar{x}_1, ..., \bar{x}_l, ..., \bar{x}_{L_q} > ,$$

$\bar{x}_l$: mean or median value of the selected attribute f_l, $f_l \in F_{j,q}^C, l = 1..L_q$, and

$F_{j,q}^C$: Set of relevant features of the output class $c_q \in C$, $q=1..Q+1$, of log type j, over all its generated detection models in DM_j.

Clustering techniques are also applicable to this problem. Reference vector of each output class corresponds to centroid or medoid of the corresponding cluster. K-means and other center based clustering algorithms, as discussed in section §2.2.2, can be adopted to determine reference vectors of the output classes.

Control intervals

Computed reference vectors are critical for two feature control steps of the selection process. They serve in identifying and tracking signs of anomalous activities. Initial signs are

determined by norm relevant features exhibiting unacceptable changes regarding associated reference vectors to the normal class. These signs are thoroughly tracked by the second checking step. The latter focuses on relevant attribute subsets of different attack classes. Furthermore, it marks any of these features as a source of intrusion only if its current observed value satisfies imposed constraints with respect to the involved attack class and its reference vector. For both checking steps, variations between relevant attributes observed and saved values by reference vectors are compared to determined thresholds. Thresholds of different selected features decide whether reached changes, in features observed values, are expected or intrusive regarding respectively norm and attack classes' reference vectors.

In the proposed selection process, thresholds of different variables can be empirically determined either by assuming normality or approximating their probability distributions. The first alternative supposes that involved features are independently and identically distributed. For a given confidence level α, observations of a relevant variable, f_l, that takes values falling within the control interval, $I = [\bar{x}_l \pm Z_{\alpha/2}\sigma_l]$, represent (1-$\alpha$) percent of training data examples of normal behavior or different attack classes, where $\bar{x}_l$, σ_l^2 and $Z_{\alpha/2}$ are respectively sample mean and variance of feature f_l observations and tabulated value of the normal distribution. In previous works, different experiments have been conducted using thresholds of 3-sigma to the sample mean, to discriminate between normal and anomalous behaviors [292], [414], [416]. The second alternative is based on testing different known distributions, regarding the characteristics of the considered variable. The distribution function that well fits observations of the target variable is included in thresholds determination.

Thresholds for specifying control intervals to normal or intrusive behaviors can be also estimated using Techbychev inequality, hypothesis testing or other tests. These techniques are based on derived distribution to assess to which degree current changes in relevant features' values are normal or intrusive. For instance, Tchebychev inequality states that $\left(1-\left(1/\lambda^2\right)\right)$percent of observed values of the feature f_l are within λ standard deviation to the mean $p\left(\left|x_l-\bar{x}_l\right| \geq \lambda\sigma_l\right) \leq 1/\lambda^2$, if λ=3 then 89% of observations of output class c_q lay within the control interval $I = [\bar{x}_l - 3\sigma_l, \bar{x}_l + 3\sigma_l]$ [410], [414].

In our idrs framework, reference vectors of considered output classes are determined based on sample mean or median of training sets. Thresholds of control intervals may be determined using identified distributions of considered relevant variables. But, this alternative is quite

costly, specifically when dealing with an increased number of relevant variables, such as for log datasets. Therefore, the first alternative of threshold determination, above discussed, seems more appropriate to the proposed selection process. Additionally, the normality assumption should be maintained for treated variables.

Previously discussed techniques allow a single reference vector for any pair of training set and output class. However, it is possible, in different steps of the model selection process, to represent each output class by several reference vectors or a codebook. Output class codebook consists of a set of code vectors which basically represent the associated training data set. Each code vector encodes a region of given output class partition. A given log data example is assigned to a region only if it ensures the smallest distance to the corresponding code vector. Codebooks of different classes can be worked out based on unsupervised techniques including clustering as discussed above [390].

c) Detection models relative scores

Relative or performance scores are determined for detection models of different log types. They are involved in determining global scores as discussed in section 4.4. They are also critical in performing the first checking step of the selection process. Most appropriate detection models are selected based mainly on their performance to conduct the preliminary security assessment step. Multiple performance metrics [65], [182], [215], [361], [362] are potentially applicable in assessing relative scores of detection models. These metrics have been subdivided into three main categories according to Caruana et al. [66]. Threshold metrics are applicable to different detection models whatever outputs they allow at the abstract, measurement or also rank level. They require the definition of a fixed threshold that will serve in interpreting assessed performance results according these metrics. Ranking or ordering metrics summarize performances of detection models over multiple thresholds. Thus they impose an ordering on models' outputs according predicted values. These metrics apply to detection models at the measurement level, where predicted values serve in ranking output labels [116]. Additionally, they may be evaluated using training results of detection models, of different output types, when built using cross validation technique [410]. The last category of probability metrics commonly focuses on detection models at the measurement level. Its metrics allow much more detailed performance evaluation than the two other categories. They involve the predicted value for each processed pattern in assessing performance of the concerned detection model.

Classification accuracy is commonly accepted and widely adopted performance metric of the first category [361]. The accuracy of a detection model determines the proportion of its

correctly recognized normal or intrusive patterns. It is estimated based on the confusion matrix of each detection model. A confusion matrix is usually a square matrix that depends on output classes included in generating each detection model. For each output class, it counts numbers of correctly and incorrectly recognized instances by the corresponding detection model. For instance, a detection model capable to discriminate between normal and intrusive activities has the following confusion matrix:

Table 4.1: Confusion matrix of a binary detection model

Predicted class / True class	*Attack*	*Normal*
Attack	*tp*	*fn*
Normal	*fp*	*tn*

Elements of the confusion matrix determine true positive and true negative counts, *tp* and *tn*, for correctly classified patterns of respectively attack and normal classes. Additionally, false negative and false positive, *fn* and *fp*, counts correspond respectively to intrusive instances confused with normal ones and the inverse. The accuracy of the detection model is computed based on these counts as follows:

$$Acc = \frac{tp + tn}{tp + fn + tn + fp} \quad (4.1)$$

Similarly, the detection error of a detection model is estimated based on its confusion matrix, using type I and II errors respectively *fp* and *fn*, by the following equation:

$$Err = \frac{fp + fn}{tp + fn + tn + fp} = 1 - Acc \quad (4.2)$$

Multiple other performance metrics such as sensitivity, specificity and precision can be deduced based upon the confusion matrix. Sensitivity, or also recall or hit rate of a detection model, assesses its effectiveness in detecting intrusive activities. Specificity or true negative rate instead estimates the effectiveness of the detection model in recognizing normal examples [361]. The inverse specificity corresponds to the false alarm or false positive rate, FPR, of a detection model. Precision metric determines the proportion of correctly detect intrusive instances among all classified as attack. True positive and true negative rates, TPR and TNR, and precision of a detection model are respectively determined by:

$$Recall = \frac{tp}{tp + fn}, \quad (4.3)$$

$$Specificity = \frac{tn}{tn + fp} = 1 - FPR \quad (4.4),$$

$$Precision = \frac{tp}{tp + fp} \quad (4.5)$$

Receiver Operating Characteristic, ROC, and Recall/precision curves allow different performance metrics falling within the ordering category. These graphical tools and others appropriately visualize performances of detection models. Among these tools, ROC curve has gained much more attention and wide acceptance in multiple domains including signal processing, medical domain and machine learning. A ROC curve is a plot of detection model recall versus (1-specifity), TPR versus FPR as depicted in figure 4.4. It illustrates relative tradeoffs between TPR and FPR of a detection model across varied thresholds. The performance of a detection model based on its ROC curve is estimated using the area under the ROC curve, AUC. A detection model performs better than another only if it has the highest AUC. Performance evaluation based on AUC of machine learning and data mining algorithms is illustrated and detailed in different Fawcett's works including [116], [117].

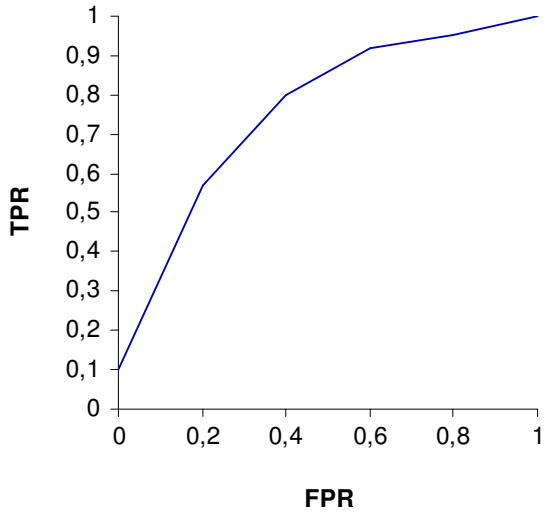


Figure 4.4: ROC curve

The last category of probability metrics includes root mean squared error, *RMSE*, and other metrics. *RMSE* assesses the deviations between predicted and true output values. For a given binary detection model M_i, its numerical predictions, $p_{i,j}$, that assess intrusiveness degrees of processed test examples, x_j, $j=1..J$, fall within the interval [0,1], $p_{i,j} \in [0,1]$. The *RMSE* of the detection model is estimated by : $RMSE = \sqrt{\frac{1}{J}\sum_j \left(p_{i,j} - a_j\right)^2}$ (4.6)

Where the binary variable a_j is equal to 1 if the true class of processed example x_j is "attack" and 0 otherwise. *RMSE* and other error and information based metrics falling within the probability category have been applied in comparing classification models [155], [165]. Furthermore, these metrics have been combined with others of threshold and ranking categories in order to propose more powerful performance evaluation tools. Caruana et al. [65] have proposed *SAR* metric based on Squared error, Accuracy and Roc curve. *SAR* includes most correlated metrics according conducted experiments in [66] but its application is restricted to classification models generating outputs at the measurement or confidence level. Lavesson et al have also proposed a generic performance function that allows classification model assessment using multiple metrics depending on the target domain [214], [215].

In our idrs framework, probability metrics are not applicable in assessing relative scores of detection models allowing abstract outputs. Additionally, ordering metrics are not useful for such context unless training results using cross-validation are adopted in evaluating relative scores of detection models using the selected ranking metric. However, in model selection component of our framework, detection models relative scores are estimated based both on their validation and testing results, $\{(VM_i\ ,\ TM_i),\ i=1..N_j\}$ as stated in section §3.2.1. Therefore, threshold metrics are the most appropriate to evaluate performances of detection models. Several, pairwise and non-pairwise, threshold metrics in addition to those previously discussed apply to detection model relative score assessment. Balanced accuracy and error rates, *BAR* and *BER*, are two non-pairwise performance metrics useful in assessing efficacy of detection models. They are determined based on confusion matrices of detection models as follows:

$$BER = \frac{1}{2}\left(\frac{fn}{tp+fn} + \frac{fp}{tn+fp}\right) = 1 - BAR \quad (4.7)$$

Additionally, success rate ratio, *SRR*, is a candidate pairwise performance metric to evaluate relative scores of detection models [54]. Success rate ratio, $SRR^j_{u,v}$, of a pair of detection models (u,v) using data set D_j is determined by :

$$SRR^j_{u,v} = \left(1 - Err^j_u\right) / \left(1 - Err^j_v\right) \quad (4.8)$$

where Err^j_u and Err^j_v are the error rates of detection models respectively u and v when tested using data set D_j. It compares a pair of detection models in terms of their accuracy on the same test set. The two detection models are comparable in terms of ensured accuracies only if

SRR ratio is equal to 1, otherwise one detection model dominates the other. The overall mean *SRR* of detection model *u*, is determined by:

$$SRR_u = \frac{1}{J(N-1)} \sum_{v \neq u}^{N} \sum_{j}^{J} SRR_{u,v}^{j} \quad (4.9)$$

It is estimated overall tested *N-1* detection models and *J* test data sets, $\{D_j, \text{j=1..}\ J\}$.

Relevant features, reference vectors and relative scores determination is required for the five steps of the selection process. Elementary processes of these steps are respectively presented by following sections.

4.4.1 Preliminary security evaluation

Preliminary security evaluation is the initial step in the detection model selection process. It aims at evaluating current preprocessed data examples of different types, $D_t=\{x_{t,j},\ j=1..J\}$, as normal or anomalous. The security evaluation process of this step is useful to identify preliminary signs of potentially intrusive events. Furthermore, it guides next steps of the main process towards the selection of most appropriate detection models capable to sift and assess the current security state of the monitored computing environment.

Preliminary security evaluation is achieved by performing different actions of the corresponding process as presented in figure 4.5 and the appendix A. This process is solely based on anomaly detection models, capable to recognize normal activities of the monitored system. Thus for each log type *j*, the profile subset, $DP_{j,Q+1} \subseteq DP_j, DP_{j,Q+1} = \{P_i, \text{i} = 1..\text{N}_{\text{j,Q+1}}, \text{c}_{\text{Q+1}} \in \text{C}_\text{i}\}$ that specifically concerns norm detection models, is considered by the preliminary security evaluation process. For every log type, the latter selects the best norm detection model associated with the highest relative score, $sr_{i,j}$. to analyze currently considered log data example, $x_{t,j}$. Outputs of selected detection models are next fused over all considered log types. A voting method such as the minimum vote is appropriate for deriving the combined decision of the preliminary security evaluation step on current security state of the monitored system [317].

Another alternative eventually applies to the preliminary security evaluation step. It consists of the selection of a subset instead the best norm detection model for every log type. A subset of top ranked norm detection models based on their relative scores is selected to fulfill the preliminary evaluation of collected log data. In this case, the combined decision of the evaluation step is derived using two aggregation stages. In the first stage, output labels of norm detection models are combined within the selected subset, for each log type. In the latter stage, first level decisions are fused over all considered log types to decide whether the

current state is normal or anomalous. Additionally, the preliminary security evaluation step may be based on fixed or variable size subsets, depending upon a prespecified lower bound of detection model relative scores. Voting methods are also appropriate to fuse norm detection model outputs in such alternative.

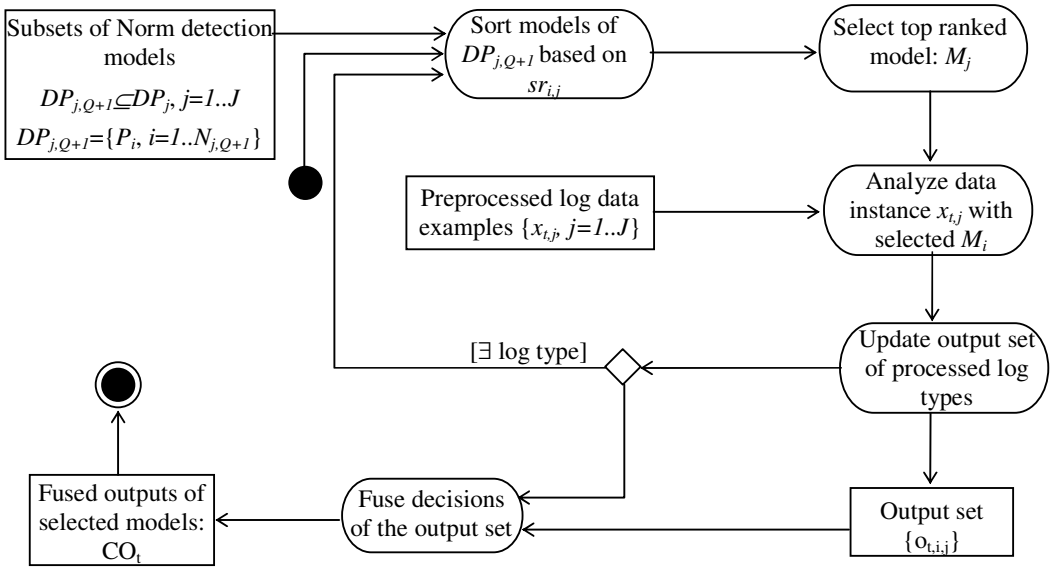


Figure 4.5: Preliminary security evaluation process

4.4.2 Control relevant features of the normal class

This second step of the selection process overlaps with the first one of preliminary security evaluation. For every log type, it takes account of the determined relevant feature subset, $F^C_{j,Q+1}$, and reference vector, $RF_{j,Q+1}$, that characterize normal behavior of the monitored system. Computed tolerance intervals for different features in $F^C_{j,Q+1}$ using the reference vector $RF_{j,Q+1}$, as previously discussed, are also required by this checking step. For any processed data example of given type, $x_{t,j}$, the latter detects changes in observed values of relevant features with respect the reference vector of the normal output class. Then, it flags a checked feature as abnormal, only if it is associated with unacceptable deviation with respect to the corresponding tolerance interval. All these activities of norm relevant features control process are depicted by the diagram of figure 4.6 and detailed in the appendix A.

Norm relevant features control process is simultaneously performed with the preliminary security evaluation step. On one hand, it reinforces reached decision of the initial step by an advanced monitoring of changes in norm relevant attributes values. On the other hand, it reports potential attack signs through flagged features as abnormal.

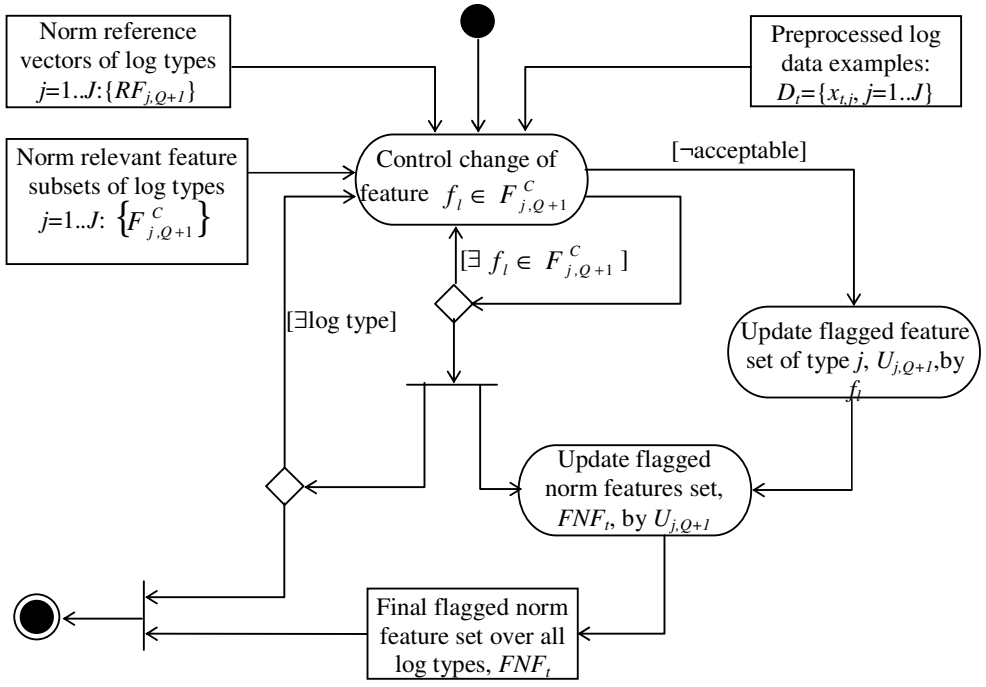


Figure 4.6: Norm relevant feature control process (FCP-1st step)

Norm features control and preliminary security evaluation steps induce redundant verifications of each logged data example. Such redundancy is intentionally included in the selection process. It increases the chance of identifying intrusion signs by inserted double control. Moreover, it ensures that all detection models capable to recognize normal system activities are considered in the two first steps of the selection process, even those excluded by the security evaluation step are involved in the norm feature control step through revising changes in values of their relevant features. Additionally, it reinforces the decision to interrupt the main process of the multimodel analysis engine at this level when treated data instances of different log types are evaluated as normal.

Outputs of security evaluation and norm features control steps are tested. In the case of abnormal behavior in the first or unacceptable changes in the second, the next step of attack relevant features control of the selection process is started. Performed actions of this third step accomplish a thorough tracking of previously signaled intrusion signs.

4.4.3 Control relevant features of attack classes

The third step of the model selection process is carried depending on decisions of the previous steps. The control step of relevant features of different attack classes is initiated only if the

current security state is evaluated abnormal or at least a single norm relevant feature is flagged, for any processed log type. It separately focuses on feature subsets of considered log types. The third checking step is iteratively performed for each log data type, as illustrated in figure 4.7. An iteration of the control process concerns a single attack class. It takes account of selected features of that class, $F^{C}_{j,q}$, their observed and saved values respectively in the current log data instance, $x_{t,j}$, and stored reference vector, $RF_{j,q}$. It checks changes between observed and saved values with respect to prespecified control intervals, each of which corresponds to a single relevant feature. Any of relevant features is marked as abnormal only if its observed value falls within the control interval. Flagged features subsets, U^{T}_{j} and $U_{j,Q+1}$, respectively in the current and precedent control steps will determine the set of abnormal attributes, U_j, of considered log type j. The resulting set of abnormal features is forwarded to the evaluation and ranking step of the model selection process. It will mainly serve in evaluating global scores of candidate detection models and sorting them according to these scores. Different actions of the control process focusing on attack relevant features are illustrated for a single log type in figure 4.7 and presented in detail in the appendix A.

As stated in the idrs life cycle, an initial analysis of the monitored system security is conducted relying on designed three checking processes. The preliminary security evaluation process determines whether currently report security state is normal or anomalous. Its decisions are reinforced by the next process. The latter controls deviations of observed values of norm relevant features to the expected behavior of the system. The last process checks if taken values of attack relevant features correspond to a known intrusive behavior.

In our framework, preliminary security evaluation and feature control processes are complementary. They achieve a twofold objective. On one hand, they fulfill an initial assessment of the current security state of the monitored system. On the other hand, they identify key features that will serve in selecting most effective detection model combinations. Preliminary security evaluation and norm feature control processes continually supervise security state of the target environment over all considered log types. Attack feature control process is triggered when at least one of previous processes reports unexpected behavior within analyzed log data. Outputs of feature control processes are critical for identifying candidate detection models, evaluating their scores and ranking them, as detailed in the next section.

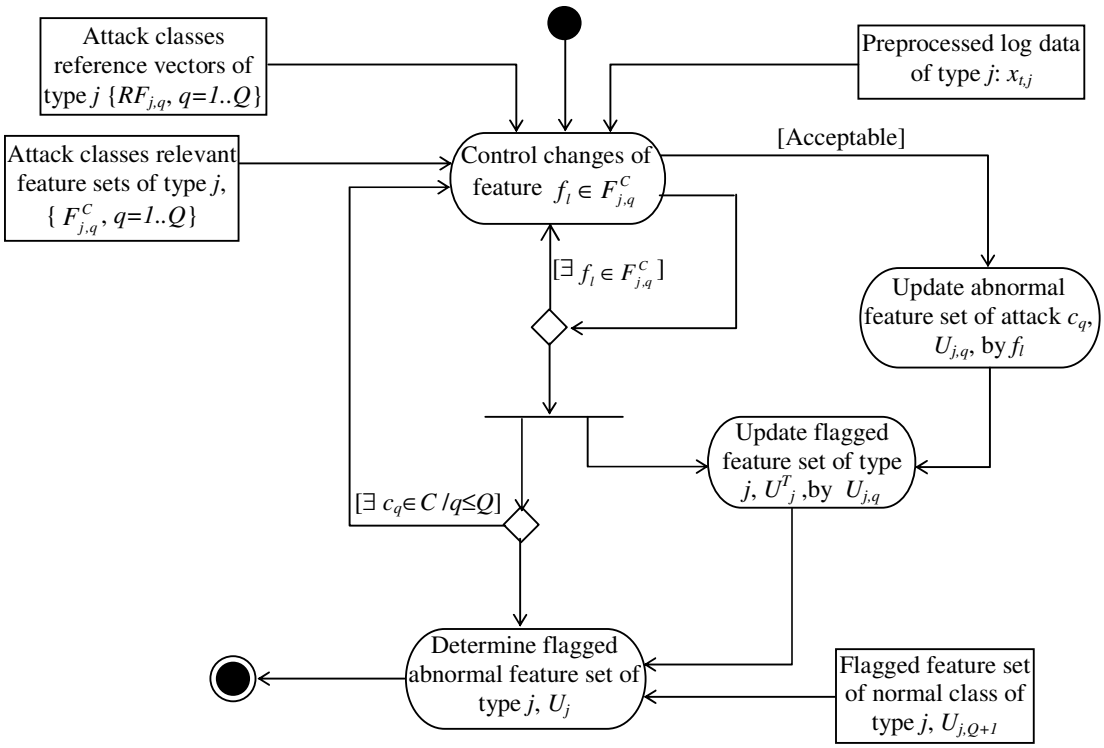


Figure 4.7: Control of attack classes relevant features for single log type (FCP-2^{ed} Step)

4.4.4 Detection model evaluation and ranking step

This step of the selection process aims at estimating scores and sorting detection models of each log type relying on these scores. Detection model scores serve as the main criterion in the selection process. They combine two types of factors namely performance and data dependent factors. Performance factors are included in estimating detection model relative scores as discussed before. The single data dependent factor included in this step will assess appropriateness of detection models to analyze current security state. It focuses on relevant feature subsets of generated detection models and those flagged in the two previous steps of the selection process.

The global score of a detection model is determined based on its relative score and flagged feature factor. The latter expresses to which degree the concerned detection model is adequate to explain marked changes in the monitored system behavior. Appropriateness of a detection models, according this factor, depends on the fraction of common features, $\delta_{i,j}$, between its relevant attribute set and those reported as abnormal. Thus, the flagged feature factor is determined after performing feature control steps of the selection process. It involves both, the output of control steps, U_j, and relevant feature subsets of generated detection models, $\{F_i, i = 1..N_j, M_i \in DM_j\}$, for every log type. The common feature subsets, $\{\delta_{i,j}, i=1..N_j\}$, of

detection models in DM_j are initially determined, $\delta_{i,j} = F_i \cap U_j$. Then, the flagged feature factor, $ff_{i,j}$, of every detection model, $M_i \in DM_j$, is computed as follows:

$$ff_{i,j} = \frac{|\delta_{i,j}|}{|F_i|} \quad (4.10)$$

where, |X| is the cardinality of the set X.

Most appropriate detection models to process logged data have higher values of the flagged feature factor. For every log type, evaluated factors are also involved in determining the candidate detection model subset, to be considered in raking and selection steps. Detection models associated with not null flagged feature factor values only are included in this set. The global scores of detection models are then estimated using a weighted sum of relative scores and flagged feature factors based on the following formula:

$$sg_{i,j} = \frac{1}{2}\left(sr_{i,j} + ff_{i,j}\right) \quad (4.11)$$

Equal weights in global scores computation indicate that both performance and appropriateness are essential factors to evaluate detection models and select the most effective combination of them to assess current security state of the system. Weight coefficients may be revised depending on whether performance or appropriateness is more important to evaluate detection models. For each log type, evaluated global scores serve also in ranking candidate detection models them.

Main activities performed in evaluating and ranking and selection steps for each log type are depicted in figure 4.8. Inputs for these activities include flagged feature subset reported by the control processes and generated detection profiles for considered log type. The initial activity of the evaluation and ranking step determines the common feature set, $\delta_{i,j}$, for each detection model of log type j, M_i. The common feature set of a detection model corresponds to its relevant attributes reported as abnormal depending upon the output of control steps and processed data example. It serves in fulfilling the next two activities of the evaluation and ranking step. On one hand, the common feature set of a detection model determines whether it is included or not in the set of candidates, CS_j. In the other hand, it is involved in estimating flagged feature factor for the corresponding detection model and then its global score, with the consideration of its evaluated relative score, $sr_{i,j}$, and respectively equations (4.10) and (4.11). Candidate detection models are ranked based on their global scores. Afterwards, the resulting subset is processed in the last step of the selection process to identify the most effective detection model combination to analyze logged data example of type j.

4.4.5 Detection model selection step

This last step of the selection process iteratively treats subsets of ranked candidate detection models, CS_j, $j=1..J$, of different log types as presented by figures 4.3 and 4.8. Its main goal consists of identifying most effective combinations, each of which concerns a single log type, to be involved in conducting a thorough assessment of current security state of the system. Multiple selection methods may be useful to implement this step. Three of them are discussed in this section.

The first selection method identifies fixed size combinations of highly ranked detection models. For ever log type, top ranked detection models with respect to their global scores and the prespecified combination size are selected to analyze current log data example. Although this method is simple and intuitive, its selected detection model combinations may include redundancies. Redundant cases within a selected combination are identified when two or more detection models have nearly identical relevant feature subsets. In such cases, only a reduced subset of common attributes between those flagged abnormal and those relevant for selected detection model is considered in analyzing logged data. This may lead to a partial assessment of the current security state of the monitored system and therefore the final decision of the combined detection model may be affected.

The second selection method reduces redundancies within selected detection model combinations by imposing a coverage condition. For each selected combination, the latter condition is satisfied only if relevant feature subsets of involved detection models ensure the maximum coverage of flagged attribute set of considered log type, U_j. This second method is slightly different to the first one. Furthermore, it determines combinations including variable number of detection models, with respect to an upper bound. However, similarly to the first method, its selected detection models are among highly ranked.

The third selection method aims at identifying diversified combinations of detection models for processing collected log data. Various diversity measures may be useful for this method. The main objective behind using these measures is boosting global performances of selected combinations that include complementary and independent detection models. The diversity of a pair or subset of detection models may be expressed using similarity, agreement, correlation [100], [205], [209], [269] or other measures. Different research works have tested diversity measures in designing multiple classifier systems [12], [206], [326]. Disagreement measure and Q statistics are among widely adopted diversity measures in these experimental studies. The disagreement measure assesses the difference between two classifiers based on

proportions of inversely classified examples. Yule' Q statistic was adopted in [205] and [208] to evaluate the dependency between a pair of classifiers. Disagreement and Q statistic are respectively evaluated for a pair of classifiers, (u,v), as follows:

$$dis_{u,v} = \frac{N^{10} + N^{01}}{N^{11} + N^{00} + N^{10} + N^{01}} \quad (4.12) \text{ and,}$$

$$Q_{u,v} = \frac{N^{11}N^{00} - N^{10}N^{01}}{N^{11}N^{00} + N^{10}N^{01}} \quad (4.13)$$

Where, N^{11} and N^{00} are numbers of times both classifiers, u and v, are respectively correct and incorrect. N^{10} and N^{01} are numbers of times only single classifier, respectively, u or v, is correct.

New diversity measures have been proposed in other works such as [12], [13], [37], [170]. Aksela et al. [12] and [13] have proposed new diversity measures that focus on the significance of classifiers prediction errors. Weighted count of errors and correct results, WCEC, and exponential error count, EEC, are two measures of the error diversity category introduced by Aksela et al. WCEC takes account of both positive and negative classification results of a pair of classifiers. It gives emphasis to cases when both classifiers are correct or they made the same errors. Using arbitrary weights, WCEC is estimated by:

$$WCEC_{u,v} = N^{11} + 1/2\left(N^{10} + N^{01}\right) - N^{00}_{diff} - 5N^{00}_{same} \quad (4.14)$$

N^{00}_{diff} and N^{00}_{same} are counts of the number of times both classifiers made respectively different and same errors using given test sample. EEC measure takes account of exponentially weighted error count by the number of classifiers making it. It is evaluated for each combination, CB of k candidate classifiers, as follows:

$$EEC_{CB} = \frac{\sum_{i=1}^{k}\left(N^{0}_{i,same}\right)^{i}}{N^{1}_{all}} \quad (4.15)$$

Where, $N^{0}_{i,same}$ is the count of same errors made by i classifiers and N^{1}_{all} is the number of correctly classified examples by all classifiers of the combination CB.

Although, researches in the field of multiple classifier systems theoretically stress the assumption of strong correlation between diversity and accuracy of classifiers, no practical diversity measure is yet proposed to support such hypothesis. Therefore, the majority of these works have identified a weak correlation between diversity measures and accuracy of multi-classifier systems [1], [56], [170], [206], [207], [237], [269]. Other works instead have illustrated the effectiveness of diversity measures in improving performance of classifier combinations [37], [46], [139].

Multiple other selection methods are also candidates and may be appropriate to the detection model selection component of the proposed idrs framework. They include methods based on the integration of two diversity measures or more such as in [37]. Other selection methods combine performance and diversity measures. Furthermore, they use different search methods such as genetic algorithms to identify most effective combinations of detection models that ensure the best tradeoff between performance and diversity. The method based on GMES (Genetic Ensemble Member Selection) algorithm of Löfström [236] is an example that combines diversity and performance measures. Furthermore, it uses genetic search to select best combinations of classifiers. Additional other selection methods have been discussed and experimented in [135], [326].

Variants of top ranked and diversity based selection methods are useful for the last step of the selection process. Multiple combinations of appropriate detection models according one of discussed criteria are identified, each of which concern a single log type. They will fulfill the step of log analysis. The latter is presented in detail in the next section.

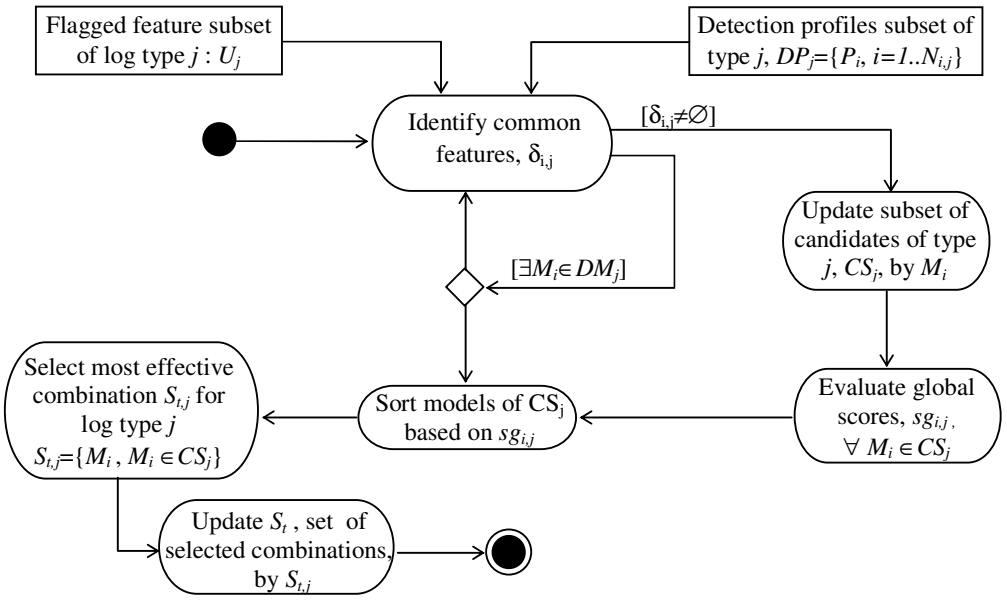


Figure 4.8: Evaluation, ranking and selection of single log type detection models

4.5 Log data analysis

Feature based selected detection models combinations of $S_t=\{S_{t,j}, j=1..J\}$ are involved in the next activity of the multimodel analysis process namely log analysis step. Each subset will analyze preprocessed and formatted log data examples of the corresponding type. Before this

step, for each type *j*, collected log data within a prespecified time interval is preprocessed as imposed at the training phase. Afterwards, the resulting preprocessed log data instance is reduced according to relevant feature subsets selected by detection models in $S_{t,j}$. Reduced data instances are then analyzed by corresponding detection models, $M_i \in S_{t,j}$. Different actions of the analysis process are performed for log data instances of considered types. The activity diagram in figure 4.9 illustrates main actions of the log analysis process for single log type.

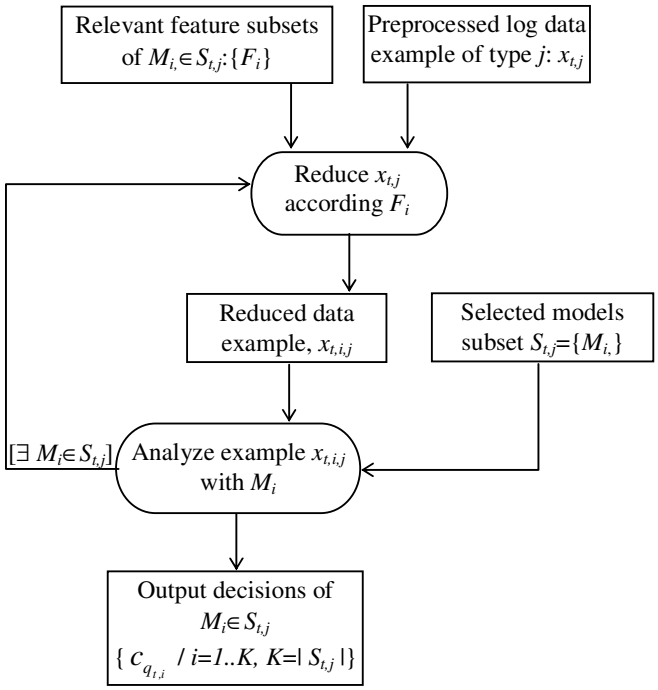


Figure 4.9: Log analysis process for single log type

The output set of the analysis process for each log type *j* consists of output labels, $c_{q_{t,i}} \in C_i$, each of which is assigned by a selected model, $M_i \in S_{t,j}$, to currently assessed security state of the monitored computing environment. More stringent assessment of the current security state of the system may be conducted when decisions of detection models within and overall output sets are considered, even those conflicting. Therefore, two level fusion process is proposed, as stated in the third step of our idrs framework, in order to derive a combined decision on the activities of the monitored system over all included trace types. The next section details different steps of the detection model fusion process.

4.6 Detection model fusion

Subsets of selected detection models will process collected log dataset. To derive the final idrs evaluation of the current state of the monitored system, these detection models should be

combined. Several combination levels are discussed in this section, but only fusion at the decision level is retained for our idrs framework. Moreover, various methods are candidates to fuse selected detection models at the decision level. However, uncertainty based aggregation methods only are introduced in this section. Furthermore, a specific attention is given to evidential combination methods for which two fusion rules are detailed namely Dempster rule and Smets's conjunctive rule. The latter rule specifically is included in modeling the idrs fusion problem. In this problem, only one and multiclass detection models are considered.

4.6.1 Fusion methods

Information fusion concerns combination of often imperfect and heterogeneous information in order to have global, complete and high quality information required to take right decision and implement appropriate actions [51]. Different fusion methods have been proposed to combine information from several sources. They form three main groups based on fusion levels namely, data, feature and classifier combination methods. All three groups may be considered at different levels of the classification process [208], [326]. Methods of the two first fusion levels are less explored comparatively the third one. They focus respectively on preprocessing and integration of acquired raw data and fusion of feature subsets extracted from several sources. Methods of these groups are adopted in a wide range of applications including image and signal processing.

Methods of the third group, also called decision fusion methods, are the most widely studied and investigated in literature [4], [51], [326], [412]. They propose different strategies to ease the multi-classifier problem. They are subdivided into two subgroups depending on whether they focus on classifiers (classifier structure) or their outputs. Methods of the first subgroup are concerned with the design of multi-classifier systems and how are generated base classifiers using available training data. Bagging and boosting are common methods of this subgroup. They focus on a combination of multiple classifiers generated by sampling from the original training data set. But, they use different techniques to generate the training sample of each base classifier.

In the second subgroup, combination methods solely operate on classifiers outputs. Output information of base classifiers can be assigned to one of the three levels: abstract, ranked and measurement. Type I classifiers output abstract labels that indicate the most probable classes to processed data examples. Output information of type II classifiers corresponds to partial or completely ranked lists of class labels. Most likely output classes for these classifiers are

given by top ranked labels of the lists. Type III classifiers instead allow soft outputs interpreted as confidence or certainty factors assigned to considered class labels. Processed data instances by these classifiers are assigned to output classes associated with highest confidence values.

Different variants of voting method, such as the majority and weighted plurality voting methods, and other probabilistic and evidential methods are useful to fuse type I classifiers outputs [206], [271]. Combination methods of type II classifiers at decision level are based either on reduction or reordering approaches. They aim at improving the rank of the true class of given input either by reducing or resorting class labels over all lists. The largest class of combination methods focuses on type III classifiers, also referred to as probabilistic classifiers. These fusion methods thought of returned certainty or confidence values by each classifier as probability, fuzzy or belief measures [51], [326], [393]. Same of probabilistic and evidential fusion methods of measurement and abstract levels have been experimented in the intrusion detection field such as in [184]

In our idrs framework, detection models are supposed providing outputs at the abstract level. Combination methods operating at the abstract level, specifically evidential methods, are the only considered in this work. In such context, evidential fusion methods are appropriate to deal with conflict in detection models decisions. Furthermore, they allow much more flexibility to integrate context dependent knowledge in designing the combined detection model. Additional other benefits supporting the adoption of evidential methods in designed detection model fusion process are subsequently discussed.

4.6.2 Dempster's rule based combination

Evidential fusion is based on Dempster's orthogonal combination rule. The latter is widely known and commonly applied in several fusion problems. Various other fusion rules have been devised relying on Dempster's rule to cope with weaknesses of this including Smets's conjunctive rule, Dubois and Prade' rule, Yager's rule and others reviewed in [228], [301]. Dempster's combination rule and Smets's conjunctive rule only are studied in this work.

4.6.2.1 Dempster Shafer Theory

The mathematical theory of evidence is a generalization of probability theory to simply and directly represent ignorance. The Dempster-Shafer theory (DST) of evidence is a powerful tool for representing knowledge, updating beliefs and combining evidences relying on Dempster's combination rule [343]. Thus, it becomes attractive for modeling complex

systems and practical for multiple applications treating uncertainty in different domains such as classification, information fusion, and medical diagnosis [227].

DST is based on a universe of discourse known as the frame of discernment and denoted by Ω. The frame Ω is a set of mutually exclusive and exhaustive hypotheses, $\Omega = \{w_1,...,w_M\}$. All possible subsets of Ω, A⊆Ω, are also hypotheses and they form the superset of 2^M elements. The impact of evidence on a subset of the power set can be measured by mass functions or the basic probability assignment (bpa). The latter is a mapping function of the powerset to the interval [0,1]. Formally, its properties are the following:

$$\begin{aligned} &m: 2^M \rightarrow [0,1] \\ &m(\varnothing) = 0 \ \ and \ \ \sum_{\forall A \subseteq \Omega} m(A) = 1 \end{aligned} \quad (4.16)$$

The hypotheses associated with not null mass are called focal elements. They represent the only elements in Ω dealt with in computing belief values. The belief function is based on the mass function to evaluate the total belief committed to a given hypothesis *A* via all its subsets as given by the following formula:

$$Bel(A) = \sum_{B \subseteq A} m(B) \quad (4.17)$$

The plausibility also relies on bpa. It is the sum of all masses associated with any subset *B* that intersect with A

$$Pl(A) = \sum_{B \cap A \neq \varnothing} m(B) \quad (4.18)$$

Bel and *Pl* represent respectively the lower and upper bound that locate the probable impact of evidence on the hypothesis *A*. They specify respectively the minimum and the maximum extents to which current evidence allows to believe in *A* [14], [28], [328]

Beliefs assigned to focal elements are discounted when their source is not totally reliable. Discounting consists of redistributing support degrees between focal elements based on source reliability degree, α. Discounting operation was initially proposed by Shafer [343]. It assigns reduced parts of belief masses of different hypotheses to the uncertain set Ω. Discounted belief function ${}^{\alpha}m$ is defined as follows:

$$\begin{cases} {}^{\alpha}m(A) = \alpha\, m(A), \ \ \forall A \subset \Omega \\ {}^{\alpha}m(\Omega) = 1 - \alpha + \alpha\, m(\Omega) \end{cases} \quad (4.19)$$

The discount coefficient, (1-α), is null when the considered source is fully reliable. In the other case, the discount coefficient is equal to 1, the information source is unreliable and its associated belief function represents the total ignorance.

4.6.2.2 Dempster's rule of combination

Dempster's rule aggregates two or more independent evidences within the same frame of discernment and from different sources into a single belief function. The combined belief function expresses support of the given proposition in both evidences bodies.

Consider Bel_1 and Bel_2 two belief functions and m_1 and m_2 their respective bpa associated to independent evidences defined in the same frame Ω. The combined bpa that represents the aggregated impact of different pieces of evidences on the hypothesis A is defined as follows:

$$\begin{aligned} \forall A \subseteq \Omega,\ \ m(A) &= m_1 \oplus m_2(A) \\ &= K \sum_{C,B \subseteq \Omega; C \cap B = A} m_1(B) m_2(C) \end{aligned} \quad (4.20)$$

where the normalization coefficient K is expressed by:

$$K = 1 \Big/ \left(1 - \sum_{C \cap B = \varnothing} m_1(B)\ m_2(C) \right) \quad (4.21)$$

It expresses the degree of agreement between sources; if this coefficient is null, it means the complete conflict between sources and the combination of their beliefs is impossible. The combined beliefs of two distinct sources ($Bel(A) = Bel_1 \oplus Bel_2$) can be also computed using their fused masses and *Bel* equation of respectively (4.20) and (4.17).

DST is useful when dealing with incomplete and possibly contradictory information. It does not require a prior knowledge on hypotheses probability distribution for performing evidences combination as in Bayesian scheme. However, the DS combination scheme is similar to the Bayesian scheme in that evidences are assumed to be statistically independent [343]

4.6.2.3 Existing fusion methods

In the proposed idrs framework, different subsets of heterogeneous detection models are selected to process currently logged datasets. The outputs of involved detection models are supposed to be at the abstract level. They correspond to different subsets of class labels each of which is associated with selected detection models of given log type. Each class label represents the output decision of a selected detection model on the processed log data instance. Several methods have been proposed to fuse heterogeneous detection models at the abstract level [14],[271], [327]. Vote methods are widely known fusion methods that operate at the abstract level. They solely focus on output labels of detection models. They use various rules including majority vote to derive the combined decision. Xu et al [412] have proposed different variants of the majority vote that also deal with the combination of abstract or type I

classifiers. Additionally, two other evidential fusion methods of Xu et al. and Parikh et al. are also useful in combining abstract classifiers. These methods focus on output labels and prior knowledge of participating classifiers to derive their combined decision [291], [412].

Xu et al. evidential combination method is based on detection model global information. Recognition, Substitution and Rejection rates (RSR) of attack classes and normal behavior are used in this method [412]. These performance metrics are evaluated for each detection model, $M_i \in \{M_1,...,M_I\}$, with respect to output classes considered at the training phase, $C = \{c_1,...,c_{Q+1}\}$. Assuming that all detection models have the same output set, for each output class c_q, these metrics respectively correspond to proportions of patterns correctly classified, confused with other classes and not recognized by detection model M_i. They are computed using the confusion matrix of each detection model for testing set as follows:

$$r_q^i = \frac{tp_q}{np_q} \quad (4.22) \text{ and}$$

$$s_q^i = \frac{fn_q}{np_q} \quad (4.23)$$

Where r_q^i and s_q^i are respectively recognition and substitution rates of detection model *i* when processing a sample of np_q data examples of class c_q. Recognized and substituted examples of class c_q by detection model M_i are determined respectively by true positive, tp_q, and false negative, fn_q, patterns in the confusion matrix. Rejected patterns of each output class by detection models are also included in confusion matrices considered by Xu et al. They are involved in determining rejection rate of each detection model which corresponds also to $(1-r_q^i-s_q^i)$. Assessed performance metrics will serve in computing belief mass (m_i) of each hypothesis in the frame of discernment, $\Omega=\{A_1, ..., A_q,..., A_Q\}$, where the hypothesis A_q states that processed data example *x* belongs to the output class $c_q \in C$.

In SRS method [291], the output decision of a detection model, M_i, on processed data example x is $c_{q_i} \in C \cup \{c_{Q+2}\}$ such that, $M_i(x) = c_{q_i}$ and c_{Q+2} is the rejection class. Depending on outputs of each detection model, two main cases are treated by the SRS method in computing bpa of processed data example *x*. Belief masses on hypotheses of Ω are estimated according SRS as follows:

- Rejected sample: $c_{q_i} = c_{Q+2}$ and it is the complete ignorance case and $m_i(\Omega) = 1$,

– Recognized example: $c_{q_i} \in C$, two focal elements are identified (A_{q_i} and $\neg A_{q_i} = \Omega - \{A_{q_i}\}$) and belief masses are estimated by:

$$\begin{cases} m_i\left(\{A_{q_i}\}\right) = r_q^i \\ m_i\left(\neg A_{q_i}\right) = s_q^i \\ m_i\left(\Omega\right) = 1 - r_q^i - s_q^i \end{cases} \quad (4.24)$$

The bpa of detection models given by (4.24) will be fused using the orthogonal combination rule of Dempster, (4.20), as discussed in the above section, to assign processed instance *x* to the most appropriate class.

As in Xu et al.' method, Parikh et al.' predictive rate combination scheme, PRM, is based on classifier level information. In PRM, predictive rates instead of recognition, substitution and rejection rates are involved in estimating belief masses of hypotheses. They are estimated using the confusion matrix of each detection model. The predictive rate of each class takes into account misclassified instances of other classes. It assesses the capacity of a detection model to recognize patterns of a given output class. For each column of the confusion matrix, the predictive rate of detection model M_i for class c_q corresponds to the ratio between correctly recognized patterns, tp_q, and total patterns identified as belonging to the class c_q, ($tp_q + fp_q$).

$$p_i^q = \frac{tp_q}{tp_q + fp_q} \quad (4.25)$$

In PRM, bpa of hypotheses are estimated using predictive rates when processed data examples are not rejected by detection models. For a given detection model, M_i, if its output for processed data instance *x* is $c_{q_i} \in C$, then identified focal elements in this case are A_{q_i} and $\neg A_{q_i} = \Omega - \{A_{q_i}\}$, their bpa are determined as follows:

$$\begin{cases} m_i\left(\{A_{q_i}\}\right) = p_i^{q_i} \\ m_i\left(\neg A_{q_i}\right) = 1 - p_i^{q_i} \end{cases} \quad (4.26)$$

Illustrative examples given in [184] detail how performing detection models fusion using RSR, RPM and other methods.

Several other combination methods have been based on DST. They propose different improvements to overcome weaknesses of Dempster's rule as discussed in [228], [301], [359], [425]. Smets has also worked out another version of Dempster's rule, Smets's conjunctive

rule, in the frame of his interpretation of DST, called the Transferable Belief Model (TBM) [353]. The following section introduces the TBM model of Smets and his fusion rule.

4.6.3 Smets's conjunctive rule based fusion

TBM is an interpretation of Dempster-Shafer Theory of belief function proposed by Smets. It provides a model, for representing quantified beliefs, totally unlinked to the probability model. This model is solely based on credibility or belief functions as appropriate measures of beliefs. Moreover, belief functions are free from any assumption that link them to probability functions such as randomness concept, additivity rule and prior probability distribution on the frame of discernment [353], [354], [356].

The TBM was based on two-level structure: credal and pignistic. The credal level is concerned with entertaining beliefs. It relies on two components including static and dynamic. The static component quantifies beliefs of given user using belief function. The dynamic component instead deals with belief revision when new pieces of information become available for user. The pignistic level focuses on belief based decision making. Beliefs at this level are quantified by probability functions. The pignistic transformation proposed by Smets ensures construction of these probability functions from belief functions at the credal level.

In the TBM framework, the credal level precedes the pignistic level. At any time, the former quantifies, updates and combines beliefs. But, when decisions have to be made, the latter constructs pignistic probabilities from belief functions.

The credal level:

This level includes two components. The first component focuses on beliefs representation and evaluation. But, the second component deals with belief revision.

The static component:

The initial and crucial step in formulating any problem with TBM is the definition of the frame of discernment, Ω, that contains all states of the nature, w_i: $i = 1, \ldots, M$. Then, beliefs that support any subset $A \subseteq \Omega$ are quantified using belief function from the power set 2^M to the interval [0,1].

$$\begin{aligned} &m : 2^M \rightarrow [0,1] \\ &\sum_{A \subseteq \Omega} m(A) = 1 \end{aligned} \quad (4.26)$$

Basic belief assignment (bba) or belief mass function, *m()*, is one possible form to express belief function. Belief mass, *m(A)*, quantifies that part of user belief allocated to the

hypothesis A , i.e., that the actual world belongs to A and due to lack of information, user does not support any strict subset of A. A mass function is normalized only if $m(\varnothing)=0$ as in DST. The belief degree on A, $bel(A)$, is obtained by summing all bba given to $X\neq\varnothing$ and $X\subseteq A\subseteq\Omega$

$$bel(A)=\sum_{\varnothing\neq X\subseteq A} m(X) \quad (4.28)$$

In the TBM framework, Smets does not assume the normality condition, $m(\varnothing)=0$, except under the hypothesis of closed world where Ω is exhaustive. However, under the assumption of open world, the bba of the empty set, $m(\varnothing)$, is interpreted as the belief mass that supports the actual world does not belong to Ω. In this case, the bba assigned to $\varnothing$ is not specific and does not support any subset, $A\subseteq\Omega$, because $\varnothing$ supports at the same time A and $\neg A$. Additionally, the positive mass allocated to the empty set in the TBM has a different interpretation in the dynamic component.

Coarsening and refinement: given that Θ and Ω two frames of discernment. A mapping ρ from the powerset 2^{Θ} to the powerset 2^{Ω}, (ρ: $2^{\Theta}\rightarrow 2^{\Omega}$), is called a refining if it verifies following properties:

$$\begin{aligned} &1-\ \{\rho(\{\theta\}),\theta\in\Theta\}\subseteq 2^{\Omega} \text{ is a partition of } \Omega \\ &2-\ \forall A\subseteq\Theta,\ \rho(A)=\bigcup_{\theta\in A}\rho(\{\theta\}) \end{aligned} \quad (4.29)$$

Θ is called a coarsening of Ω and Ω is a refinement of Θ. The bba m^{Θ} on Θ can be transformed into bba on the refinement Ω relying on vacuous extension as follows [95], [97]:

$$m^{\Theta\uparrow\Omega}(\rho(A))=m^{\Theta}(A) \quad \forall A\subseteq\Theta \quad (4.30)$$

Dynamic component:

Beliefs issued from distinct sources and quantified by bba can be aggregated, at the credal level, using different operators. In the TBM, these operators are called combination rules. The Smets's conjunctive rule aggregates two bba, m_1 and m_2, as follows:

$$(m_1\textcircled{\cap}m_2)(A)=\sum_{\substack{X\cap Y=A\\ X,Y\subseteq\Omega}} m_1(X)\ m_2(Y), \quad \forall A\subseteq\Omega \quad (4.31)$$

In such combination, sources are supposed to be fully reliable (assumed to tell the truth). This rule is also called unnormalized Dempster's rule of combination where the normalization factor K is omitted and the conflict between sources is expressed by the combined mass allocated to the empty set, $m_{12}(\varnothing)$. Normalized belief masses, in this case, are determined as follows [257], [355], [356]:

$$(m_1 \oplus m_2)(A) = \frac{(m_1 \textcircled{\cap} m_2)(A)}{1-(m_1 \textcircled{\cap} m_2)(\varnothing)} \qquad \forall A \neq \varnothing \qquad (4.32)$$

The pignistic level:

When the decision must be made, belief masses determined by the combination rule induce probability measures at the pignistic level [354], [355], [359]. The probability function that allows such measures using bba is called betting probability function and denoted *BetP*. It is constructed from bba using the following pignistic transformation:

$$BetP(w) = (1-m(\varnothing))^{-1} \sum_{B \subseteq \Omega, w \in B} \frac{m(B)}{|B|}, \quad \forall w \in \Omega \quad (4.33)$$

Where B is subsets of Ω and |B| its cardinality.

Such transformation ensures uniform distribution of belief mass of B to its elements. Determined probabilities of different elements will be adopted to complete the decision process and select the best decision [353], [358].

The TBM framework proposes powerful tools to deal with different forms of uncertainty. In our idrs framework, TBM tools are appropriate to represent and treat uncertainty associated with decisions of detection models and conflict between them. Moreover, this evidential model offers required flexibility to incorporate domain specific knowledge within the developed model of the target problem. In this work, the proposed evidential model for the fusion component of our multimodel idrs problem is based on the TBM framework. The Smets's conjunctive combination rule will be adopted to aggregate decisions of detection models within and between selected subsets of considered log types.

4.6.4 Detection model fusion problem

In our idrs framework, different subsets of detection models identified by the selection process will analyze collected log datasets. Each of selected detection models exchanges no information with those in the same subset or in other subsets. Furthermore, detection models outputs on the current security state of the monitored system are supposed independent. They will be combined within an uncertain environment relying on an evidential method. As such, the TBM framework is adopted in modeling the idrs fusion problem. Moreover, the designed fusion method is solely based on Smets's conjunctive combination rule. Many reasons support such choices and approve the appropriateness of TBM and Smets's conjunctive rule in this context.

The TBM model is appropriate to our idrs framework. It is highly structured and its two levels meet with the requirements of the proposed multimodel analysis engine. At the credal level, beliefs of selected detection models are evaluated based on their outputs and learning sets. Then computed beliefs are fused using Smets's conjunctive rule that is one of integrated TBM tools. At this level, context dependent knowledge is quite easily integrated in the proposed fusion process. Uncertainty in detection models decisions can be dealt with when evaluating beliefs. Moreover, the conflict between selected detection models can be treated by required mechanism when fusing them.

At the pignistic level, combined beliefs are transformed into probabilities to make a decision. This is required to multimodel analysis engine to label the current security state of the monitored computing environment. Furthermore, it is critical to the response component of the proposed idrs framework in order to assess inflicted damage and design appropriate defense strategies against detected attacks.

TBM model offers required tools to appropriately support further extensions of the proposed idrs framework. It is useful when dealing with imperfect data either in training sets or historical datasets as discussed later in the proposed process. Moreover, it is well suited to cope with unknown attacks in idrs when adopting the open instead of the closed world hypothesis, as further discussed in this work. Additional other qualities of the TBM model, such as multilevel fusion, remarkably improve capabilities of idrs by allow precise detection decisions and detailed intrusion reports.

In this work, another class of fusion methods is proposed based on TBM model. In this class, fusion methods are capable to deal with outputs at different levels including abstract, ranked and confidence levels. Furthermore, they are independent to natures of outputs, specifically at the measurement level where output values have different interpretations.

Instead of solely focusing on outputs of detection models, our evidential fusion method takes account of prior knowledge of detection models, the learning sets. This knowledge is not static as in PRM and RSR but dynamically revised depending upon previous experiences of detection models. A learning set of a detection model supports learned and recognized patterns of considered output classes in training this model. The proposed method based on learning sets is capable to fuse heterogeneous detection models at the abstract level. It is inspired by previous works of Denœux and Zouhal on evidential nearest neighbor classifier [98], [431]. Furthermore, this fusion method uses the distance based approach to evaluate the beliefs of selected detection models based upon their derived learning sets. It additionally applies Smets's conjunctive rule in combining evaluated beliefs of detection models within

and between selected subsets for considered log types. Two main steps of the fusion method namely beliefs evaluation and combination are summarized in the proposed TBM based real time multimodel detection process. The latter is thoroughly discussed in the next section.

4.6.5 Real time multimodel detection process

In the proposed idrs framework, different subsets of heterogeneous detection models are selected to process currently logged datasets. The outputs of involved detection models are supposed to be at the abstract level. They correspond to different subsets of class labels each of which is assigned by a selected detection model combination to the processed log data instance. Our TBM based detection process computes the final decision of the combined model relying on those of selected detection models and their learning sets.

In the proposed process, the initial step of updating learning sets of detection models based on their historical data sets is performed depending upon fixed criterion, such as historical dataset size or time constraints. The next steps of the process are fulfilled for any processed data instance. Beliefs evaluation step uses the distance based approach of Denœux to assess detection models beliefs on processed log data instances. Afterwards, beliefs fusion step performs two aggregation levels, within and between selected subsets, to evaluate the beliefs of combined detection model on currently processed logs. Smets's conjunctive rule is applied in both fusion levels. Betting probabilities estimation step is finally conducted using the pignistic transformation and combined belief masses to take the right decision about reported and analyzed security state of the monitored system.

In our idrs framework, multiple log types are considered in reporting and analyzing monitored system activities. Let us assume that we have J intrusion log types associated with different feature subsets constituting the set $F = \{F^1, \ldots, F^j, \ldots, F^J\}$. Also let us denote $C = \{c_1, \ldots, c_q, \ldots, c_{Q+1}\}$ as finite set of output classes, c_q, $q=1..Q$, represent attack classes of DARPA reduced taxonomy and c_{Q+1} corresponds to the normal class. We interchangeably use labels of the set, $L = \{1, \ldots, q, \ldots, Q+1\}$ $i=1..Q+1$ to ease notation problems as needed.

For every log type j, we denote $DM_j = \{M_i \ / \ i=1..N_j\}$ the set of detection models generated using relevant feature subsets $\{ F_i, F_i \subset F^j\}$ and dataset of type j. The dataset of each log type is structured into TR_j, TS_j and VS_j that denote respectively training, testing and validation subsets. These composite datasets include both normal and intrusive data examples. Every detection model M_i in DM_j of log type j is trained, validated and tested using respectively datasets $TR_{i,j} \subseteq TR_j$, $VS_{i,j} \subseteq VS_j$ and $TS_{i,j} \subseteq TS_j$. $TR_{i,j}$, $VS_{i,j}$ and $TS_{i,j}$ are respectively training, validation and testing sets of M_i based on datasets of type j. They are determined using the

projection function, π(), selected feature subset, $F_i=\{f_l, l=1..P\}$, and datasets respectively TR_j, VS_j and TS_j, as stated in section §3.2.1. The projection function defined by $\pi_{X_i}(R)=R_i$ is based on the set of attributes, X_i, to reduce the relation R into R_i. The resulting sets of the projection of TR_j, VS_j and TS_j based on F_i are respectively reduced training, $TR_{i,j}$, validation, $VS_{i,j}$, and testing, $TS_{i,j}$, sets determined as follows:

$$TR_{i,j}=\left\{(x_r,y_r)/x_r=<x_1^r,...,x_p^r>,y_r\in C_i,r=1..N_{tr}\right\},$$

$$VS_{i,j}=\left\{x_k/x_k=<x_1^k,...,x_p^k>,k=1..N_{vs}\right\} \text{ and}$$

$$TS_{i,j}=\left\{x_u/x_u=<x_1^u,...,x_p^u>,u=1..N_{ts}\right\}$$

All datasets involved in generating detection model M_i concern only its selected feature subset, $F_i\subseteq F^j$. The latter is iteratively determined depending on output classes of $C_i=\left\{c_q,q=1..N_{i,j}\right\}\subseteq C$, as discussed in section §4.4. Throughout the idrs life cycle, a historical data set, H_i, $i=1..N_j$, is built for each detection model. It stores previous experiences of that detection model. Of course, some of the processed data instances may not concern all models M_i, $i=1..N_j$, and therefore they are not stored in the history of certain detection models. Historical data sets are also involved in updating prior knowledge on detection models including their confusion matrices, TM_i, $i=1..N_j$, initially computed using testing sets $TS_{i,j}$.

The historical data set, H_i, of detection model M_i summarizes its previous experiences at different time points. At given time τ, the data raw appended to H_i corresponds to the reduced data example x_i^τ of the logged instance, $x_{\tau,j}$, using the projection function, π(), and selected feature subset, F_i of the corresponding detection model. Output decisions respectively of the model M_i, $c_{q_{\tau,i}}$ or simply $q_{\tau,i}\in L_i$, the set of output labels of M_i, and the combined detection model, $q_{\tau,c}\in L$, are obviously included in labeling each data instance of the historical set H_i, $H_i=\left\{h_{\tau,i}\,\middle|\,\exists\tau\in\{0,...,t-1\},h_{\tau,i}=<x_{\tau,i},y_{\tau,i}>,y_\tau=\left\{q_{\tau,i}\cup q_{\tau,c}\right\}\right\}$. Therefore, the historical data sets may include imprecise data examples for which the associated output labels correspond to subsets of C, $y_\tau\subset C$, but not a single label. Such case is encountered when the involved and the combined detection models in current analysis task output different labels.

The learning set ζ_i of the detection model M_i is determined based upon its training set, $TR_{i,j}$. It is gradually updated using the historical data set, H_i, of the involved detection model. The learning set of each detection model includes precise examples, each of which is associated with a single output label.

Initially, the learning set ζ_i of each detection model corresponds to its revised training set. After the validation phase, generated detection models are tested using their training sets in order to identify substitution cases. The latter include training examples incorrectly recognized by the learned detection model. The learning set of each detection model is then derived based on its revised training set.

The learning sets of detection models are gradually updated based on historical data sets. In this context, real time log analysis induces very large historical data sets for detection models. Although, increasingly large historical datasets improve detection models decisions, they may produce extended learning sets that affect performance of the analysis engine in terms of computation and storage loads. In order to remedy to these problems, we proceed by prototyping the historical data sets.

Prototyping historical data sets

Many prototyping methods are discussed in literature [16], [128], [303]. They aim at representing large datasets by prototypes without affecting the decision making process based on this. Prototyping methods form two main categories namely generation and selection. Generation methods determine artificial prototypes by merging data examples of the processed set. Selection methods instead identify representative prototypes among instances of the dataset relying on data reduction, artificial intelligence and other approaches [303], [351]. Methods of both categories are confused [238], [321] as stated in [128] and usually considered as prototype selection methods [16], [303].

Prototype selection methods are commonly based on nearest neighbor technique. Garcia et al. [128] have categorized and experimented near fifty of these methods. Other methods use clustering techniques for the identification of prototypes [44], [241]. In [99], clustering based method is adopted to select prototypes for training evidential neural networks. Ravindra also in [321] has tested different methods including distance based for selecting prototypes from large datasets.

In the proposed multimodel detection process, clustering, distance based and other methods discussed in [16], [44], [128], [241], [303] can be adopted in selecting prototypes of learning sets. For instance, distance based methods can be applied in evaluating dissimilarity between randomly selected prototypes and data examples in historical datasets. When the dissimilarity exceeds a given threshold, the tested prototype in the historical set is then included in the considered learning set. Distance based and other prototype selection methods are performed each time historical datasets of intrusion detection models exceed a certain size.

The learning set, ζ_i, summarizes the knowledge state of the detection model M_i about processed data examples of each output class in C_i. Each element in ζ_i will participate in expressing its own knowledge on processed data examples as belief degrees. For any given data example x, each selected individual $x_r \in \zeta_i$ states that x belongs to c_r with a certain belief. The reliability of x_r depends on its measured distance to x, $d(x_r,x)$. A reliability coefficient is then determined for x_r and involved in discounting its beliefs.

Prototyping effort

Learning sets of detection models, ζ_i, $i=1..N_j$, include data examples of different output classes considered at the training phase. Data instances of each learning set have precise labels and this is also the same for prototypes appended to them. However, prototypes derived from historical data set, H_i, of the detection model M_i may have imprecise labels. Therefore, the proposed prototyping process aims at selecting prototypes with precise labels from historical data sets in order to update learning sets of detection models, $M_i \in DM_j$. It uses clustering and nearest neighbor techniques to respectively find out prototypes and their precise labels. The prototyping process is performed in two steps. In the first step, the standard clustering process is fulfilled in order to determine clusters and their centroids. In the second step, precise labels are specified for cluster centroids based on their computed nearest neighbor sets.

In the initial step of the prototyping process, the number of clusters may be statically determined depending upon the number of possible output classes for the considered detection model. Likewise, the number of distinct single labels within the historical dataset may be also useful in initializing the number of clusters. Based on the prespecified number of clusters, k, the clustering process partitions the historical data set into groups. Non empty clusters are then returned and their centroid, $P^*_{i,x} = \{x^*_l, l = 1..k\}$, are computed. In the final step of the prototyping process, centroids are assigned to precise labels, $P^*_{i,y} = \{y^*_l, l = 1..k\}$. Afterwards derived prototypes are inserted in the learning set of the considered detection model.

In the last step of the process, each cluster centroid is associated with a nearest neighbor set, φ_l, from the historical data set. Computed nearest neighbor sets are involved in labeling generated prototypes. Two main cases may be encountered depending on whether all elements in the nearest neighbor set belong to the same output class or not. In the former, the generated prototype is similarly labeled to its neighbors in the computed set. In the latter case, labels of elements in the neighbor set are aggregated to derive the most probable label of the

identified prototype. The labeling process of centroids involves similar computation steps as in processing new cases, detailed below. It takes account only of centroid neighbors, determined by set φ_l, in performing different computations and a single fusion step. Then, it chooses precise labels of centroids based on estimated pignistic probabilities. Identified labels and computed centroids determine prototypes of historical datasets, $P_i^* = \{s_l^* =< x_l^*, y_l^* >, l = 1..k\}$, in order to update learning sets of detection models.

When selected, nearest neighbor set of a given cluster center, $x_l^* \in P_{i,x}^*$, has distinct, either precise or imprecise, labels. The labeling process of the corresponding centroid has to determine precise label to ease the problem of updating learning sets. It is inspired by evidential kNN process proposed by Denœux [98]. In this context, the frame of discernment is represented by C. It can be coarsened into C_i depending on the specificities of considered detection model M_i. Bba of each selected neighbor $x_r \in \varphi_l$ is determined based on $d(x_r, x_l^*)$, the distance between x_r and the centroid x_l^*, as follows :

$$\begin{cases} m_{i,l}^r(y_r) = \alpha_l^r, y_r \subseteq C_i \\ m_{i,l}^r(C_i) = 1 - \alpha_l^r \\ m_{i,l}^r(A) = 0, A \in 2^{C_i} \setminus \{C_i, y_r\} \end{cases} \quad (4.34)$$

and $\alpha_l^r = \alpha_0 \phi(d_{r,l})$ (4.35)

where α_0 is a constant dependent to examples in φ_l. It represents the reliability coefficient of x_r in determining instances of $c_r \in C_i$. $\phi(d_{r,l})$ is a decreasing monotone function verifying following conditions:

$$\begin{aligned} &\phi_r(0) = 1 \\ &\lim_{d \to \infty} \phi_r(d) = 0 \end{aligned} \quad (4.36)$$

Combined beliefs of k_r neighbors in φ_l of the centroid x_l^* associated with the same label y_r are given by

$$\begin{cases} m_{i,l}^{(r)}(y_r) = 1 - \prod_e^{k_r} (1 - \alpha_{i,l}^e) \\ m_{i,l}^{(r)}(C_i) = \prod_e^{k_r} (1 - \alpha_{i,l}^e) \\ m_{i,l}^{(r)}(A) = 0, A \in 2^{C_i} \setminus \{C_i, y_r\} \end{cases} \quad (4.37)$$

Combined beliefs over distinct labels in φ_l using Smets's conjunctive rule are determined as follows by:

$$m_{i,l}^{C_i}(w) = \sum_{\substack{A_r \cap B_u = w \\ A_r, B_u \subseteq C_i}} m_{i,l}^{(r)}(A_r)\, m_{i,l}^{(u)}(B_u),\ w \subseteq C_i \quad (4.38)$$

Pignistic probabilities of atoms in C_i are then determined based on combined beliefs over all neighbors in φ_l by:

$$betP_{i,l}(w) = \frac{1}{1 - m_{i,l}^{c_i}(\varnothing)} \sum_{B \subseteq C_i, w \in B} m_{i,l}^{C_i}(B) / |B|,\ \forall w \in C_i \quad (4.39)$$

Precise labels of centroids, $x_l^* \in P_{i,x}^*, l = 1..k$, are determined by $y_l^* \in P_{i,y}^*, l = 1..k$, such that

$$y_l^* = \arg\max_{\forall w \in C_i} \left(BetP_{i,l}(w) \right) \quad (4.40)$$

Generated and labeled prototypes, $P_i^* = \{s_1^*, ..., s_k^*\}$, of the historical dataset H_i are then appended to the learning set of the detection model M_i. Additional other cases, not considered at this level and may be encountered by the labeling process, need further studies.

Updating learning sets of detection models is required by the fusion step of the detection process. The latter step focuses on the evaluation and combination of beliefs with respect to selected detection model decisions on currently processed log data examples. This step is presented in detail below.

Processing new cases

In our idrs framework, the knowledge base saves different detection profiles for considered log types, $j=1..J$, as discussed in section §3.2.1. Each detection profile, $P_i \in DP_j$ concerns a generated detection model, M_i, its training set, $TR_{i,j}$, historical data set, H_i, and other information, $P_i = <M_i, H_i, \ldots>$. At any time point t, different subsets of detection profiles, $\{S_{t,j}, j=1..J\}$, are selected and then the associated detection models are involved in processing logged data examples of J different log types determined by $D_t = \{x_{t,1}, \ldots, x_{t,j}, \ldots, x_{t,J}\}$. $S_{t,j}$ is always used to designate selected detection models of type j rather than their profiles in current and subsequent sections. Furthermore, each detection model is assumed to generate independent outputs to other candidates or selected models in $S_{t,j}$.

The log analysis process of the proposed multimodel engine is similarly performed for all considered log types, $j=1..J$. To process the data example $x_{t,j}$, a subset $S_{t,j} \subset DM_j$ of detection models, generated using feature subsets of log type j, is determined by the selection process,

$S_{t,j}=\{M_i / M_i \in DM_j\}$, as detailed above in section §4.2. For any detection model $M_i \in S_{t,j}$, bba of a given data example $x_{t,j}$ are constructed using individuals in its learning set, ζ_i. By considering all selected subsets of detection models $S_{t,j}$, $j=1..J$, our intrusion detection problem consists to assign current security state of the system differently reported by several log types, $x_{t,j} \in D_t$, to one class of C and therefore to decide which is the true among $Q+1$ possible hypotheses stating that D_t is assigned to the class c_q, $q=1..Q+1$. In this problem, the frame of discernment is represented by C. Moreover, bba on different hypotheses of C, for any $x_{t,j} \in D_t$, are expressed using distance based approach. Additionally, abstract outputs of selected detection models, $M_i \in S_{t,j}$, and their learning sets, $\{\zeta_i, i=1..N_{i,j}\}$ are considered in estimating belief masses of any processed log data example, $x_{t,j} \in D_t$, as detailed below

Techniques proposed to select cases in ζ_i to participate in computing bba

Two main techniques have been proposed to select data examples of ζ_i to be involved in computing bba. The first technique selects a fixed number of nearest neighbors to the data example $x_{t,j}$ depending on the chosen distance measure. The second technique summarizes the learning set by several prototypes using different methods as discussed earlier. For both techniques, a reduced set of examples of ζ_i is involved in evaluating beliefs. In this work, the first technique is adopted. For any processed data example $x_{t,j}$, the set $\varphi_{t,i,j}$ of k nearest neighbors to $x_{t,j}$ in ζ_i regarding the output label of selected detection model M_i, $q_{t,i}$, is derived based on evaluated distance measures. For any selected example $x_r \in \varphi_{t,i,j}$, bba of $x_{t,j}$ is expressed based on (4.34) as follows:

$$\begin{cases} m^r_{t,i,j}(\{c_r\}) = \alpha^r_{t,i} \\ m^r_{t,i,j}(C_i) = 1-\alpha^r_{t,i} \\ m^r_{t,i,j}(A) = 0, A \in 2^{C_i} \setminus \{C_i, \{c_r\}\} \end{cases} \quad (4.41)$$

where $0 < \alpha^r_{t,i} < 1$ and it is expressed using (4.35) by

$$\alpha^r_{t,i} = \alpha_0 \phi_r(d_{t,r})$$

α_0 is a dependent parameter to examples in $\varphi_{t,i,j}$ and belongs to [0,1]. It expresses reliability of selected data example x_r in recognizing data instances of class c_r. $\phi_r(d_{t,r})$ is a decreasing function depending on the distance $d_{t,r}$ between x_r and $x_{t,j}$. It should satisfy conditions given by (4.36). As detailed in [98] and [398], it can be defined by the following exponential function:

$$\phi_r(d_{t,r}) = exp\left[-\gamma_r(d_{t,r})^2\right] \qquad (4.42)$$

where γ_r is a normalization coefficient estimated based on the mean distance, d_r^m, between data examples in ζ_i labeled as $c_r \in C_i$, $\gamma_r = 1/\left(d_r^m\right)^2$

Combined beliefs of two examples x_r and x_l of $\varphi_{t,i,j}$ associated with the same output class c_r are determined by:

$$\begin{cases} m_{t,i,j}^{(r,l)}(\{c_r\}) = 1-(1-\alpha_{t,i}^r)(1-\alpha_{t,i}^l) \\ m_{t,i,j}^{(r,l)}(C_i) = (1-\alpha_{t,i}^r)(1-\alpha_{t,i}^l) \\ m_{t,i,j}^{(r,l)}(A) = 0,\ A \in 2^{C_i} \setminus \{C_i, \{c_r\}\} \end{cases} \qquad (4.43)$$

Over the k elements of $\varphi_{t,i,j}$, belief masses of the processed data example, $x_{t,j}$, are given by :

$$\begin{cases} m_{t,i,j}(\{c_r\}) = 1-\prod_l^{k_r}(1-\alpha_{t,i}^l) \\ m_{t,i,j}(C_i) = \prod_l^{k_r}(1-\alpha_{t,t}^l) \\ m_{t,i,j}(A) = 0,\ A \in 2^{C_i} \setminus \{C_i, \{c_r\}\} \end{cases} \qquad (4.44)$$

At this step, belief masses of the selected detection model M_i are determined by fusing bba over all k elements of $\varphi_{t,i,j}$ using (4.44). They are discounted with respect to reliability coefficient, α_i, of the involved detection model. This coefficient is estimated using the testing confusion matrix of the detection model M_i. Moreover, it is updated each time the model M_i is selected to process logged data. Discounted beliefs of selected detection model M_i are evaluated as follows:

$$\begin{cases} m_{t,i,j}^{C_i}(\{c_r\}) = \alpha_i\ m_{t,i,j}(\{c_r\}),\ c_r \in C_i \\ m_{t,i,j}^{C_i}(C_i) = (1-\alpha_i) + \alpha_i\ m_{t,i,j}(C_i) \\ m_{t,i,j}^{C_i}(A) = 0,\ \ \varnothing \neq A \in 2^{C_i} \setminus \{C_i \cup \{c_r\}\} \end{cases} \qquad (4.45)$$

Fusing on all

To be combined with other belief masses on different hypotheses of C_i of the detection model M_i, bba should be expressed on the frame of discernment C. Depending on generated detection models and their output classes, C_i, different transformations are applicable to represent their bba on C. In our idrs framework and for the sake of simplicity, one class and multiclass supervised detection models specifically are considered. In the first category of

multiple class detection models, output classes of each model represented within the defined frame of discernment, C, and thus their bbas are evaluated, as discussed above using (4.45) on C. However, one class detection models are associated with a reduced output set and then the reduced frame of discernment for these models is determined by $C_i = \{c_q, c_{\bar{q}}\}$. The latter is considered as a coarsening of the frame of discernment C. Bbas initially determined using (4.45) for $c_r \in C_i$ are now expressed on C based upon the following transformation:

$$\begin{cases} m^{C}_{t,i,j}(\{c_r\}) = m^{C_i}_{t,i,j}(\{c_r\}), \forall c_r \in C \\ m^{C}_{t,i,j}(C \setminus \{c_r\}) = m^{C_i}_{t,i,j}(c_{\bar{r}}) \\ m^{C}_{t,i,j}(C) = m^{C_i}_{t,i,j}(C_i) \end{cases} \quad (4.46)$$

Evaluated and represented bba on C using (4.45) and (4.46) are then fused twice relying on Smets's conjunctive rule. The initial fusion level evaluates combined beliefs for each considered log type j, $j=1..J$. Beliefs of selected detection models of $S_{t,j}$ to process log data examples of type j, are fused based on the following:

$$m^{C}_{t,j}(w) = \sum_{\substack{A \cap B = w \\ A,B \subseteq C}} m^{C}_{t,u,j}(A)\, m^{C}_{t,v,j}(B), \; w \subseteq C \quad (4.47)$$

where M_u and M_v are two selected detection models of $S_{t,j}$ and their beliefs on the security state reported at time t by $x_{t,j} \in D_t$ are respectively evaluated by belief masses $m^{C}_{t,u,j}$ and $m^{C}_{t,v,j}$ expressed by (4.45) or (4.46).

The second aggregation level determines fused beliefs of D_t over all considered log types, with respect to selected subsets of detection models, $S_{t,j}$, $j=1..J$. The beliefs of the combined detection model on currently analyzed security state of the monitored system are expressed as follows:

$$m^{C}_{t}(w) = \sum_{\substack{A \cap B = w \\ A,B \subseteq C}} m^{C}_{t,j}(A)\, m^{C}_{t,h}(B), \; w \subseteq C \quad (4.48)$$

where $m^{C}_{t,j}$ and $m^{C}_{t,h}$ are combined bbas respectively for log types j and h evaluated using (4.47).

Deciding the output class

Fused beliefs of the second level by (4.48) will determine pignistic probabilities, $BetP_t$, of different atoms of C to decide to which class is assigned processed log data in D_t. Betting

probabilities of atoms are expressed using the pignistic transformation (4.33) and combined beliefs of (4.48) as follows:

$$BetP_t(w) = \frac{1}{\left(1 - m_t^C(\varnothing)\right)} \sum_{B \subseteq C, w \in B} \frac{m_t^C(B)}{|B|}, \forall w \in C \quad (4.49)$$

Currently analyzed security state of the system is assigned to the output class associated with the highest $BetP_t$ evaluated by (4.49) as follows:

$$q_{c,t} = \arg\max_{\forall w \in C} \left(BetP_t(w)\right) \quad (4.50)$$

Afterwards, the final decision, selected label and betting probability $BetP_t$, of the combined detection model on analyzed log, D_t, is sent back to the risk driven response component of the proposed idrs framework. It is initially considered by this component to decide whether a reaction is required or not. When a response should be implemented, it is also involved in designing appropriate security strategy by the risk driven response component. Additional parameters are considered with the probability of detected attack, $BetP_t$, in assessing risk cost for this component.

4.7 Conclusion

Adaptive analysis engine proposed in our idrs framework is based on the multimodel approach. It takes account of multiple log data types. Moreover, it extends those designed in previous works by including real time detection model selection and fusion. Designed multimodel analysis engine is based on a three step process. The latter focuses on performed activities of considered components respectively of selection, analysis and fusion. At a given time point *t*, log datasets of different types are collected and preprocessed. The resulting set includes single instance for each considered log type. The selection component is then involved in identifying appropriate combinations of detection models with respect to considered log types and assessed security indicators. Afterwards, the analysis component treats available log data instances of different types using corresponding detection model combinations of the previous step. Finally, the fusion component derives the combined decision of the multimodel analysis engine. It is based on two combination levels that are preceded by beliefs evaluation for all selected detection models based on their abstract outputs and learning sets. The combined decision of the multimodel detection engine on current security state of the monitored computing environment is forwarded next to the response engine

In this chapter, proposed improvements of the analysis engine fulfill two main activities of the idrs life cycle namely selection and fusion. The selection component was initially based on three checking steps to analysis system security of the system. The first two steps determine whether current state is normal or anomalous. In the case of an abnormal behavior, the last checking step thoroughly tracks signs of potential attack. The checking steps determine also several feature subsets that report identified intrusiveness signs. These subsets are involved in guiding the selection of well adapted detection model combinations to assess current security state of the system. The fusion component was based on Smets's conjunctive rule to combine selected detection models at the abstract decision level. The distance based approach of Denœux was adopted to express beliefs of detection models on processed log instances. Evaluated beliefs were fused twice to derive the final decision of the multimodel analysis engine. The probabilistic output of the analysis engine is then considered by the risk driven response component of our idrs framework to design appropriate security strategies against detected threats.

The next chapter of this thesis focuses on the risk driven response component of our idrs framework. Furthermore, it presents in detail the proposed idrs risk management model and how security strategies are designed relying on this.

CHAPTER 5

RISK DRIVEN RESPONSE, THE IDRS RISK MANAGEMENT MODEL

5.1 Introduction

In our idrs framework, combined decision forwarded by the multimodel analysis engine is one of the basic parameters of the response component, R-boxes. The latter is based on a risk driven approach. As such, a risk model is proposed to assess inflicted damage by mounted attacks on the assets of the computing environment. Relying on this model, a risk cost minimization program is developed in order to design the most appropriate security strategy to be deployed.

In this chapter, the proposed risk management model is presented, as well as its identification and estimation parts are thoroughly discussed. Furthermore, the risk treatment part of this model and the associated optimization program are also stringently detailed in subsequent sections of the current chapter.

5.2 Proposed risk model

In our multimodel intrusion detection and response framework, detected attacks or attempts of attacks are actively treated by included risk driven response engine. The latter assesses incurred damages due to these attacks. Then, it identifies the most appropriate combination of countermeasures, from those recommended by security experts, to rule out detected threats and mitigate their negative effects on target assets. Both steps followed by the response engine in processing detected attacks are imposed by the risk management model proposed to these aims.

Risk management model of the response engine consists of two interdependent parts. On one hand, the assessment part ensures the identification, determination and evaluation of inflicted damage by detected threats. On the other hand, the treatment part focuses on the selection of

cost effective control combination to reduce assessed risks. The risk assessment part of the proposed model is concerned with the analysis and evaluation of risks. The risk treatment part concentrates on the post assessment step, specifically, the selection and implementation of a treatment option.

The proposed risk management model relies on four main components namely assets, threats, vulnerabilities and security controls. Threats and assets are critical components and commonly considered by risk management methodologies. Threats component groups deliberate actions of insider or outsider entities who attempt to inflict damage to target assets. Assets component includes any worthy computing resource to the organization. The assets of the computing environment support several vulnerabilities or weaknesses that represent the main source of harm to them. To defend against potential exploit of supported flaws by malicious actions of an attacker, appropriate security safeguards or controls are selected and implemented with respect to the security policy of the target environment. Moreover, they can reduce or neutralize negative effect arising from weaknesses exploit. Dependencies between considered risk components are presented in figure 5.1.

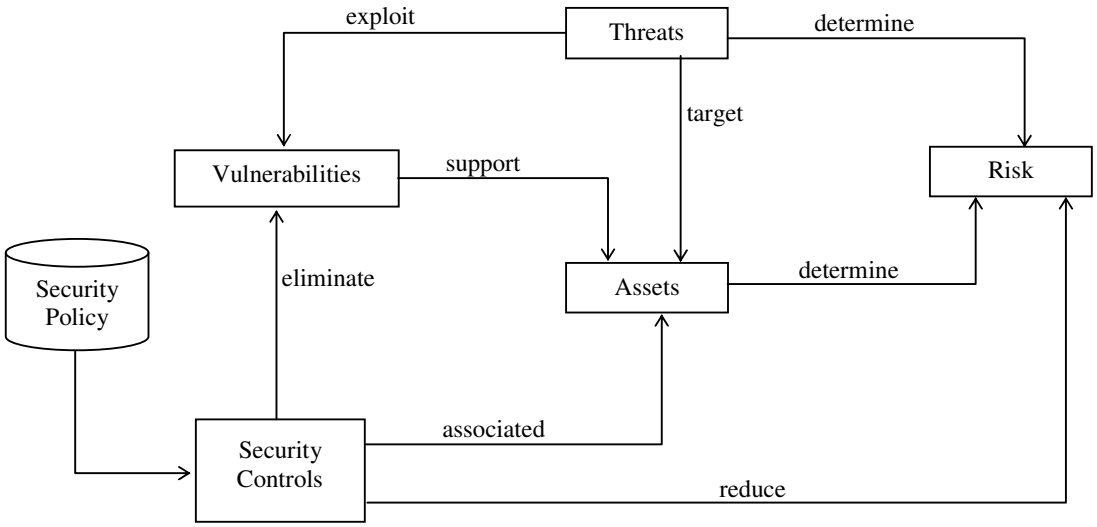


Figure 5.1: Dependence diagram of risk components

The proposed risk management model is depicted by figure 5.2. The risk assessment part focuses on analysis and evaluation of the current risk level, the basic risk of the computing environment. It uses collected data on attackers' means, exploited vulnerabilities, targeted assets and safeguards. The risk treatment part specifically concerns mitigation option where cost-effective security controls are selected to reduce basic risk and ensure acceptance of residual or remaining risk. Each identified part of the risk management model is supported by processes that implement its associated activities.

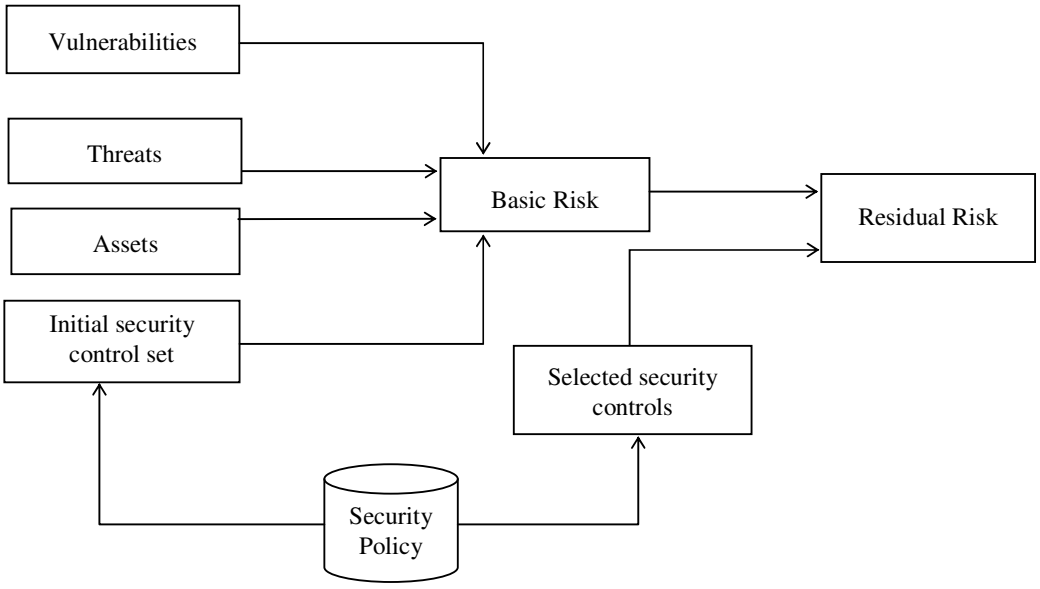


Figure 5.2: The proposed risk model

The risk assessment in our model is performed in two steps. Initially, risks are analyzed where different risk elements or parameters are identified and associated variables are determined. Then, evaluation processes of identified risk elements are designed by considering determined variables of the previous step. The risk analysis process is initiated by identifying risk elements closely related to above discussed components. Key elements identified in the proposed risk model are required by both steps of risk assessment. They include commonly considered elements by normalized, public or proprietary risk assessment methodologies namely impact, likelihood and exposure elements. Moreover, other risk elements are explicitly considered by our risk assessment process, specifically the severity of supported vulnerabilities and the effectiveness of countermeasures. Such improvement proposed by our risk model emphasizes the roles of environment dependent elements that were neglected or implicitly included, through key elements, in managing risks. Furthermore, it allows more precise and objective estimations of risk levels and thorough risk management than the existing ALE (Annual Loss Expectancy) or NIST (National Institute of Standards and Technology) inspired methodologies [149], [325].

In the analysis process, risk determination focuses on variables involved in estimating risk elements. For each identified parameter, its determinant variables are found out regarding specificities of considered risk components. They reflect single or multiple aspects that the target risk element is concerned with. Determined risk elements can be estimated using quantitative or qualitative methodologies. Depending on selected methodology, involved activities in the determination step may focus on scale definition. Moreover, they may

concern procedures design to process collected data and combine different features of considered risk components in order to evaluate determined parameters. In the proposed management process, different variables that concern identified risk elements, presented in figure 5.3 are determined. Their estimation is based on a quantitative methodology. Risk identification and determination steps of the analysis process are discussed in section §5.3.

Risk evaluation follows identification and determination steps of risk analysis. This final step of the risk assessment process is based on identified parameters and their determinant variables. It defines estimation processes of the identified elements and basic risk of each target asset of the computing environment. These processes involve selected combinations of determined variables with respect to considered risk elements and their dependencies. Basic risk evaluation process focuses on identified risk elements. It expresses combined effect of involved parameters that reflect the current security state of the asset or the computing environment. In the proposed risk model, basic risk is evaluated by combining elements including exposure, severity of vulnerabilities and effectiveness of controls as depicted in figure 5.3. The exposure element determines the expected damage incurred by the target asset due to exploit of its vulnerabilities by detected threats. Asset impact and threats likelihoods are main parameters to determine the exposure. The former relies on the asset value to express its potential loss due to supported flaws. The latter assesses the probability that an asset is affected by an attack when occurring. The severity of vulnerabilities and effectiveness of deployed security controls depends on current security state of the target asset. In one hand, the severity parameter focuses on the extent of damage inflected by an attack if a sequence of vulnerabilities is exploited by this. On the other hand, the effectiveness parameter concerns the reduction effect of existing security controls on assessed risk. The basic risk assessment process is thoroughly studied in sections §5.3 and §5.4.

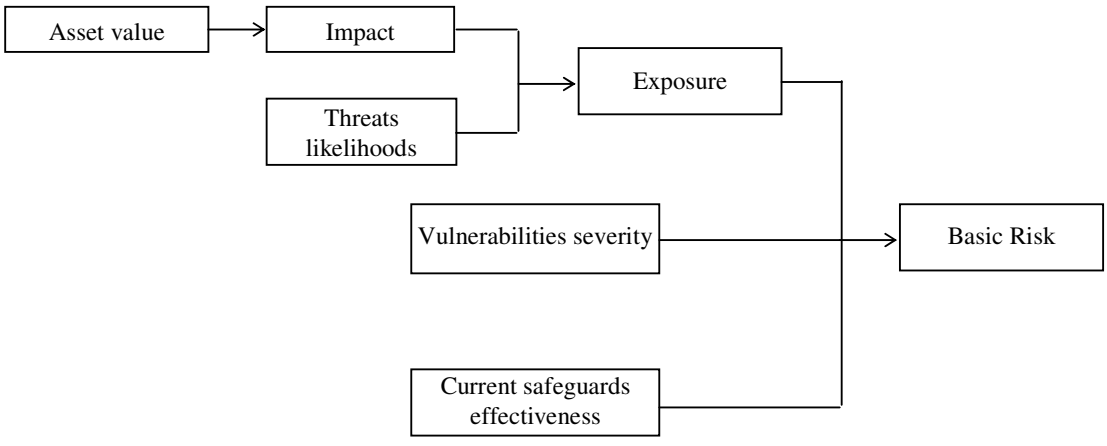


Figure 5.3: Basic risk assessment

Risk management consists of risk assessment and treatment. After assessing risk, normalized risk management methodologies impose the selection and implementation of a treatment option [148], [175], [325]. Risk reduction option is the most significant for our idrs framework, thus it is the only considered in this study. Such option reinforces the current security state of the computing environment. It relies on the selected combination of security controls to reduce current risk of the computing environment to an acceptable level with respect to available security budget, as globally illustrated by the process of figure 5.4. The proposed risk reduction process aims at decreasing the global risk cost, damage and security investment costs. It prioritizes risks to identify and urgently mitigate the most serious. Moreover, this process ensures selection of the best control combination that reduces residual risk to meet tolerance criterion and saves maximum security budget. Residual risk determination and control combination selection in this process are iteratively performed as summarized by figure 5.4. The whole risk mitigation process and its optimization problem are detailed in section §5.5.

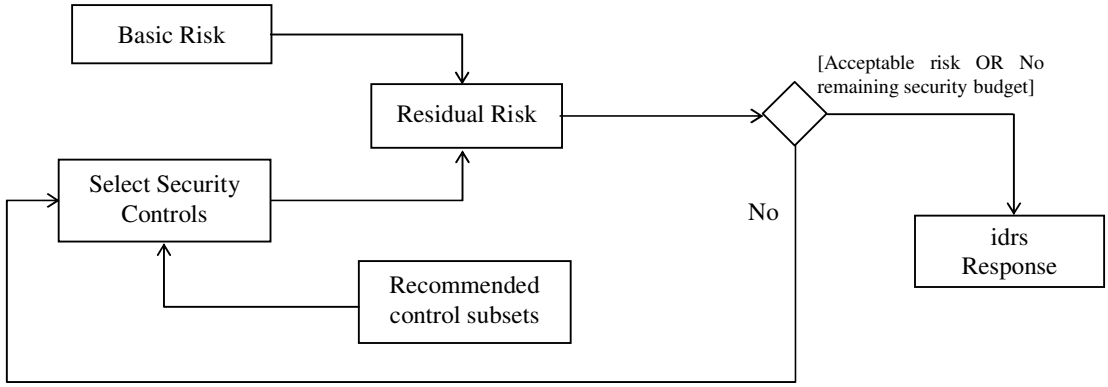


Figure 5.4: Risk treatment

In the proposed risk management model, two main vulnerability classifications are considered. The first classification discriminates vulnerabilities of the computing environment based on their exploit objectives as given by publicly available databases such as open source vulnerability database, OSVDB. Determined classes by this include privilege elevation, service privation and other classes. They are involved in assessing threats likelihoods and controls effectiveness parameters.

The second classification identifies different flaw groups depending on the target environment. Availability and implementation states of remediation against these vulnerabilities are basic criteria considered in determining groups of this classification. The latter mainly consists of

patched, unpatched and unresolved vulnerability groups. These three groups are specifically considered by our risk model in determining severity element of supported vulnerabilities for the assets of the computing environment.

In our risk model, we assume:

A: a set of N assets of the monitored computing environment, $A=\{a_i, i=1..N\}$.

V: a set of M vulnerabilities supported by the computing environment; $V=\{v_j, j=1..M\}$.

V^C: a set of P vulnerability categories, $V^C = \{V^r, r = 1..P\}$.

V_i: a set of vulnerabilities supported by the asset a_i, $V_i = \{v_j / \mathrm{v_j} \in V\}$.

T: a set of Q possible threats, $T \subset C$, the set of possible output classes, $T=\{c_q, q=1..Q\}$.

SC: a set of L security controls, $SC=\{s_l, l=1..L\}$.

SS_0 : the initial control combination or security strategy of the computing environment and the subset $SS_{i,0} \subseteq SS_0$ protects against exploit of vulnerabilities, V_i, supported by the asset a_i .

S: the set of designed candidate combinations of security controls to mitigate assessed risk of the computing environment to an acceptable level.

$SS_{i,k}$: the control combination of the asset a_i associated to the security strategy $SS_k \subset S$ of the computing environment, $SS_k=\{ s_l, s_l \in SC, l=1..L_k\}$.

Risk assessment and treatment parts of the proposed model are presented in following three sections. Risk analysis, evaluation and mitigation steps of the model are respectively discussed in these sections.

5.3 Risk identification and determination

Previously identified risk elements, namely asset value, impact, threats likelihoods, exposure, vulnerabilities severity and controls effectiveness, based on included components are detailed in this section. Moreover, variables involved in their determination processes are globally discussed in following subsections.

5.3.1 Asset value

Assets identification and valuation is a critical step in quantitative risk analysis methodologies. It is required to enumerate and determine how worthy are potentially target assets of the computing environment. However, with the increased size of organizations and complex and dynamic nature of their networks, this step becomes more complicated and time consuming.

Assets can be automatically identified using asset management systems. These systems track changes of the computing environment when new assets are added or other are removed. They periodically update information about assets including their functional and security statuses and locations. Moreover, asset management systems are able to discover supported flaws when associated with vulnerability scanners [39].

Identification step in risk management provides global information about computing environment assets in order to understand their criticalities and assess their degrees of implication in achieving the organization's mission. Assets of computing environment can be grouped into different classes. The most known classification identifies tangible and intangible assets classes. Tangible class concerns hardware and other physical assets. Intangible assets correspond to software, information, intellectual property and other logical assets. Pfleeger has defined six class asset classification by discriminating between assets of computing environment and required resources for their function. The first three classes, namely hardware, software and data, identify computing environment intrinsic assets. The last three classes of documentation, supplies and people skills group assets not considered as a part of the computing system but they are required for its proper operation [300]. Farahmand uses also five class assets taxonomy. Information assets class of this taxonomy concerns data and documentation resources of Pfleeger. Whereas, system class includes any combination of people, software, hardware and information assets [112].

In this work, the main classes of Pfleeger's taxonomy are considered. Any combination of software, hardware and data assets is included in system class, similarly to Farahmand's taxonomy. The last class is considered in our risk model because attacks generally target combined rather than elementary assets. For instance, if attackers aim at disrupting printers or corrupting client database, these are reachable only by jeopardizing management systems of target assets respectively printing server and database management system in this example.

Assets of different classes can be valuated using several methodologies discussed in literature. Quantitative methodologies determine assets worth in terms of monetary values. Different costs are considered by valuation processes of these methodologies such as purchase, design, implementation, function and maintenance costs. Qualitative methodologies rely on different labels or scoring systems to estimate values of assets. Cost factors considered by these methodologies are included in defining scoring tables or selecting qualitative terms. These tables or chosen labels guide the determination of the right valuation scores or labels.

Assigned value, score or label, reflects the importance of a given asset within the computing environment [112], [340].

The methodologies of the first class are subject to unreasonableness due to their over or under-valuations. Likewise, second class methodologies are considered more subjective when using inappropriately structured scoring tables or scales. But, whatever the selected approach, the main objective of this step is the definition of an asset hierarchy in terms of their importance or criticalities within the computing environment [151].

In this work, the valuation process relies on a quantitative approach because this is critical in assessing inflicted damage by detected threats as well as designing appropriate defense strategies against them. Additionally, it mainly focuses on above discussed asset classes, explicitly, hardware, software, data and systems that group support assets of the computing environment [300]. Different quantitative valuation methods are candidates for implementing this process, as stated in [104]. However, the proposed quantitative process will specifically focus on an economical approach to determine how worthy are assets of the computing environment. Such methods thought of the business value of an asset as its expected value as explained later. Moreover, they suppose that a fraction or the total amount is lost if the corresponding asset is targeted by a threat. In the proposed process, the value of an asset, $a_{i,}$ is expressed relying on its business income and operating cost.

The business income or the return of an asset corresponds to the fraction of the organization revenue insured by this asset through implementing single or multiple activities within several business functions. Different approaches were proposed to gauge the business income of given assets. In [180], revenues of assets are broadly determined based on downtime and liability costs. Downtime cost quantifies income loss in terms of productivity and revenue when an asset is targeted by an attack within a given time interval. Liability cost evaluates the amount to be paid by the organization to its partners due to service disruption. Other approaches approximate the business income of single asset using revenues of business functions in which it is involved [376].

The operating cost of an asset depends on its nature. It corresponds to required cost to recover normal function of the target asset. It includes one or a combination of acquisition, installation, function, maintenance, design, implementation and configuration costs depending on the target asset. These costs are approximated based on previous experiences of the organization and the actual market prices.

Assets values are appropriately estimated if business processes of the organization are well structured and documented with regard to recommendations of the ISO-27001 which concerns information security management system (ISMS) [173]. ISMS implementation requirements are mainly centered on business functions of the target organization and their security needs. Each business function is organized into several activities. These activities are represented by different processes that transform inputs into required outputs by the corresponding business function. By identifying different business processes of the organization, their importance and required assets, the proposed valuation procedure of the risk model may produce more objective and precise estimations of asset values [173], [376]. The process proposed in section §5.4.1.1 assumes similar conditions to those discussed above in determining asset value.

5.3.2 Impact

Impact determination is a critical step of any quantitative risk analysis methodology. It focuses on estimation of potential losses of computing environment assets due to supported weaknesses. Impact element in risk assessment methodologies expresses the magnitude of harm that could result from potential exploit of target asset vulnerabilities.

Potential loss due to supported flaws can't be exactly evaluated neither for assets nor the computing environment. However, it is estimated based on an intricate process that combines historical data, knowledge of target systems and judgments of risk assessors. In quantitative risk analysis methodologies, impacts are commonly measured using numerical scales. Different numbers within the selected scale are assigned to expected magnitudes of potential damages. ALE methodology, for instance, recommends the expression of assets impacts directly in terms of monetary values because it is legally more acceptable and useful in evaluating risks [149].

Impact estimation processes are based on several factors. A class of processes approximates potential losses of exploit relying on multiple cost factors, while another class takes account of different security concerns in their estimations. Cumulative costs, either direct or indirect, of possible damages are considered by processes of the first class as the most appropriate estimate of supported weaknesses' impact. Direct costs such as loss of sales, materials, labor costs and productivity are included in the processes of this class because they reflect tangible part of this impact. Whereas, intangible impact costs are difficult to approximate due to their variable nature and dependence on out of control attributes. These costs are closely related to external agents to the organization like competitor (loss of competitivity), and costumer (loss

of confidence), and operation standards (breaches to operation standards). Moreover, they represent the major security cost to preserve business continuity [375]

The second class of processes is based on NIST risk assessment standard. Processes of this class take account different security concerns, such as confidentiality and integrity, in their estimates of assets impact. They determine the overall impact of the target asset by considering its single potential loss in terms of each selected security principle. These processes are well suited to information security domain as well as to our idrs framework. On one hand, their assessments are based on security concerns of the target organization. On the other hand, their considered list of security requirements is not restrictive and can be extended depending on the environment and operation sector of the organization [181].

Standards and guidelines such as [149], [150], [260] and [325] try to identify the magnitude of harm resulting from compromises independently to other assessments that aim at evaluating compromise probability. Processes based on NIST risk management framework define adverse impact as any potential damage associated to unauthorized disclosure, corruption or disruption of computing environment assets. They express impacts in terms of loss or degradation of any or a combination of three main security concerns namely confidentiality, integrity and availability.

Confidentiality impacts focus on potential damage of unauthorized discloses of saved or exchanged data assets. They reflect an important part of the intangible impact of the organization vis-à-vis its partners and customers and the regulation. For example, successful divulgation of client credit card number would result in customer confidence and business losses. Impacts on integrity instead resume additional costs supported by the organization due to improper modifications of its systems or information assets. The cost of corrupted information or contaminated systems could stem from increased fraud tentative, wrong decisions and conflicts with organization's partners. Losses of availability affect mission critical assets of the organization. They lead to disruption of organization functions and decrease operational effectiveness due to lost productivity time and inability of legitimate users to access and use their systems.

NIST generic risk assessment process [325] uses qualitative labels to describe corruption, disclosure and disruption impacts. In our quantitative risk model, relative impacts of supported weaknesses to considered security concerns will be determined using numerical scales used by the normalized CVSS system, as detailed in subsequent sections. The potential damage due to supported vulnerabilities of a given asset will be function of its value and

relative impact factors. Moreover, different weight coefficients will be associated to selected security principles and considered in determining relative impact factors of an asset. This was neglected by all previous impact assessment processes, which assume the same importance for all security requirements over all assets. Such improvement of the impact assessment process is inspired by reality where the same asset requires various levels of protection depending upon its environment, usage, location within the computing environment and stored information.

In the proposed risk model, the impact of a given asset a_i is determined using its value, required security services and supported weaknesses as expressed by the following function:

$$I_i = f_1(U_i, W_i, V_i) \quad (5.1)$$

where:

I_i : the impact of the asset a_i due to its supported weaknesses

U_i : the expected value of the asset a_i

W_i : vector of weights that expresses the importance of considered security principles, including integrity, availability, confidentiality and other security requirements, to the asset a_i. It can be evaluated for each asset by security expert and organization' managers regarding the security policy and the mission of the organization

V_i : set of supported vulnerabilities of a_i

The impact of the asset a_i is partially determined based on its relative impact factors expressed in terms of confidentiality, integrity, availability and other security principles included in W_i. A vector of relative impact factors, $\Gamma_{i=<\gamma_{i,1}, \ldots, \gamma_{i,n}>}$, should be evaluated for each asset a_i. Its appraisal is basically achieved by considering supported weaknesses, V_i, of the target asset a_i and their impact scores assessed by publicly available vulnerability scoring systems as discussed in §5.3.5. Relative impact factors, Γ_i, and weights of required security services, W_i, are involved in determining the overall impact factor, τ_i, of the asset a_i. The potential loss due supported flaws of the asset a_i is then estimated using its value and the overall impact factor. Involved variables and the associated function to determine impacts are discussed in section §5.4.1.1.

5.3.3 Threats likelihoods

One of challenging steps in quantitative security risk assessment methodologies is the determination of likelihoods of malicious events experienced by different assets of the computing environment. Threats likelihoods element is essential in our basic risk determination process. It expresses how likely target asset is being compromised if an

unwanted event occurs. In ALE methodology, likelihood is interpreted as the annualized frequency of malicious event occurrence. For other risk assessment methodologies including GISAM and NIST, it is thought of as the probability of potential exploit of an identified security breach by a malicious event [216], [325].

Objective estimation of likelihoods requires an extended historical dataset on mounted attacks, supported weaknesses and target assets associated with perfect knowledge of the monitored system. These are seldom available for actual organizations and even if they dispose of little historical dataset, available risk assessment methodologies fail to objectively predict likelihood of occurrence of a malicious event that impacts computing environment assets. In fact, the majority of these methodologies rely on expert judgment and evaluate likelihoods of threats occurrences using terms such as low, medium or high or using numerical ratings. Qualitative methodologies use these labels to identify risk levels of target assets based on impact-likelihood matrix as in [17], [216], [325]. However, quantitative methodologies determine risks of a given asset using selected functions. The latter functions combine different risk parameters including likelihoods of threats occurrences and expected consequences of intrusive actions.

Other risk assessment methodologies are based on likelihood estimation processes. Different dependent factors to entities mounting attacks or threat sources are considered by these processes. Schechter has identified two main categories of factors related to threats frequencies. The first category concerns positively correlated factors to threats occurrences such as the number of potential attackers and potential benefits. However, the second category focuses on disincentives to attacker resumed by negatively correlated factors like intruder risks, to be caught, and system resistance. Both factor categories are considered by the process proposed to estimate threats frequencies [337], [338]. Farahmand has founded his likelihood determination process on NIST recommended factors, specifically; attackers' motivations, means and opportunities of exploit [112]. Bellefeuille has proposed a probability estimation process that involves several dependent factors to security incidents. Incorporated factors concern categories such as attacker motivation and capability and vulnerability nature [39], [390]. Motivation and capability of attackers are represented in this process by two factors namely target attractiveness and ease of exploit. Vulnerability related factors are expressed by characteristics such as availability of exploit and Internet wide frequency of attacks. Considered factors are scored using different scales. The probability of an attack is then linearly determined based on cumulative scores over all included factors [39].

In previous risk analysis methodologies, threat likelihood is interpreted as the probability or frequency of unwanted event occurrence on computing environment assets. This event can be originated from nature or accidentally or deliberately by different entities or threat agents. It may cause variable damages to computing environment assets. Furthermore, deliberate actions inducing these events look for several objectives ranging from simple information leakage to complete disruption of the target asset.

In this work, mal-intentioned events mounted by internal or external entities are the only considered threats by the proposed risk management process. These deliberate events aim at compromising computing environment assets and inflicting harm to them. They negatively affect target assets only if they successfully exercise supported weaknesses or bypassing deployed security controls or both. Otherwise, any attack or attempt of an adversary, when occurring, has no impact on computing environment assets. In the proposed risk model, the likelihood of a malicious event corresponds to its probability of affecting the target asset. This depends jointly on its occurrence and effect on the victim. Moreover, at this step, we consider an attack, c_q when occurring, negatively affects the target asset only if it exploits at least one of its supported weaknesses.

In our framework, detected threat, c_q, is considered as a random variable with possible values in the set of threats, T. Its values can be determined by an embedded sub-system such as the analysis and detection engine of our idrs, as discussed in chapter 4. In this work, a probability distribution of c_q, $p(c_q)$, is defined. For each attack class, $p(c_q)$ is interpreted as the probability that the class c_q takes place. Probabilities of different values of c_q in T are periodically estimated by our combined detection model. They are included in estimating threats likelihoods element of our risk model.

In the proposed risk model, threat likelihood is approximated by the probability that mounted threat c_q affects target asset a_i. Based on Bayes theorem of conditional probability, this probability is determined by the product of the probability of the occurrence of c_q and the probability that c_q affects a_i when occurring. It can be expressed as follows:

$$p(c_q \text{ affects } a_i) = p(c_q)\,p(c_q \text{ affects } a_i / c_q \text{has taken place}) \quad (5.2)$$

The conditional probability included in (5.2) of likelihood estimation can be determined by considering the current security state of the target asset a_i, specifically, its security controls and supported flaws as discussed in sections § 5.3.6 and § 5.3.2.

5.3.4 Exposure

Exposure element in risk analysis determines the expected compromise associated with an attack exploiting a vulnerability in the target asset. It takes account critical components of risk model, specifically, assets supported vulnerabilities and potential threats. Impact element of risk analysis focuses on the first two components where expected potential loss experienced by an asset is determined by considering its intrinsic sources of harm, the supported weaknesses. However, the exposure element adds another risk dimension to those of the impact parameter. This dimension concerns asset extrinsic origins of damage, either internal or external malevolent entities to the organization and their capabilities. The exposure element of the proposed risk model expresses expected damage inflicted by a given threat agent with consideration of its potential success in exploiting single or multiple weaknesses of the target asset.

Information security risk exposure can be quantitatively or qualitatively estimated. It is expressed using numerical values either monetary or scaling values for quantitative methodologies. Qualitative risk analysis methodologies instead use ordinal scales that globally rank expected magnitude of risk exposure, usually, as high, medium or low. Both classes of methodologies thought of expected loss due to the unwanted event as the product of its occurrence frequency and expected consequence.

In quantitative methodologies, particularly ALE based, risk exposure or also Annualized Loss Expectancy is assessed in terms of monetary values by multiplying the Single Loss Expectancy (SLE) and the Annual Rate of Occurrence (ARO) of an intrusive event. This was largely criticized in other risk analysis methodologies [313], [365]. In fact, exclusive reliance on such assessment of risk exposure provides a global insight and hides details of the actual security strategy of the computing environment. However, hidden information is extremely valuable to decision maker, security managers and other steps in the risk management process. For instance, if the monitored computing environment is faced to the following situations:

Case 1: ARO = 9 /year and SLE= 2000$

Case 2: ARO = 0.5/year and SLE = 36000 $

According to the ALE global assessment of the exposure, both situations have the same extent because they ensure identical expected loss. Moreover, they should be similarly treated. Nevertheless, decision maker and security managers consider the first case as the most catastrophic and needs to be urgently addressed because it is associated with the highest frequency. Furthermore, it may concern client database in e-banking environment or an access

server of an Internet service provider. However, the second case corresponds to an ordinary and manageable situation since it is associated with low frequency, even if it concerns highly critical assets. Additionally, if the second case occurs for low criticality assets of the organization, its treatment should be done after appropriate mitigation of high priority risk associated to the first situation.

Other quantitative and qualitative methodologies determine the risk exposure level based on predefined matrix. The exposure assessment matrix is formed by different lines and columns associated respectively with scores or labels of threats likelihoods and assets impacts. Expected risk exposure is identified for given labels or scaling values by the intersection between the corresponding likelihood line and impact column as illustrated by table 5.1 [295], [325].

In our risk model, the exposure of the computing environment is considered as a function of the impacts of its assets and likelihoods of malevolent activities targeting them. Potential damages of the target asset, due to supported vulnerabilities, are estimated based on its expected value and required security services. Likelihoods of attackers' actions are expressed in terms of probability of a successful exploit of target asset weaknesses. Expected exposure due to exercise of supported flaws by a given threat can then be determined, identically to ALE based methodologies, a combination of above discussed risk elements or any other as discussed in [55], [332]. In the proposed risk model, the exposure, $X_{i,k}$, of the target asset a_i is expressed using its impact, I_i, and the likelihood of the mounted attack, c_q, $L_{i,q}$.

$$X_{i,q} = f(I_i, L_{i,q}) \quad (5.3)$$

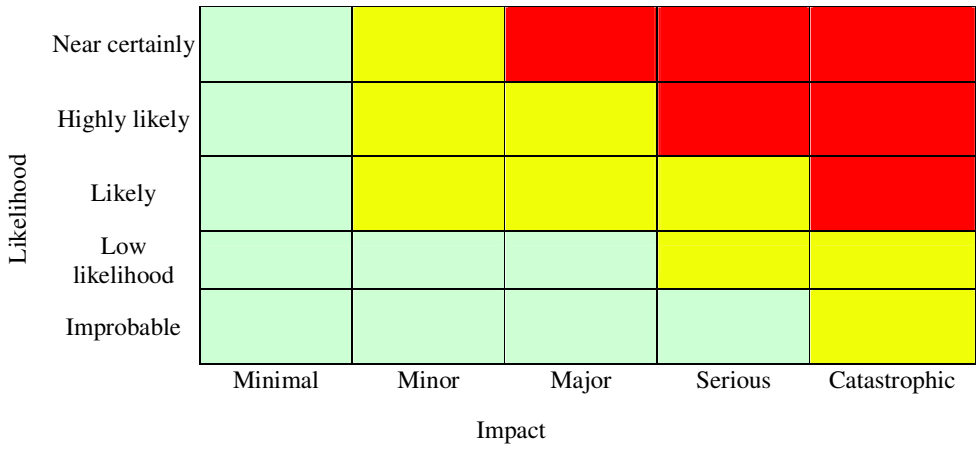


Table 5.1: Example of exposure matrix[2]

[2] An example of risk matrix for information system security available at: www.faa.gov

5.3.5 Vulnerabilities severity

In information security context, the same attack causes variable losses even if it targets comparable assets of two organizations in the same exploitation domain. This is closely dependent to the target environment and attacker related factors. In fact, the consequence of mounted attack is manageable if technical, operational and managerial controls of the target computing environment are periodically revised. Nevertheless, if increasingly discovered vulnerabilities are left without any remediation; malicious activities of an attacker could probably cause catastrophic damages that can't be supported by the target organization

Losses due to the malicious actions depend also on attacker capability, skill and experience, and means. Expert attackers are able to carry out their intrusive actions even if only difficult to exploit vulnerabilities are supported by the victim. Always, they use their proper attacking toolkit either totally developed by them or partially modified Internet tools according to their needs. Actually, wide variety and highly sophisticated attack tools available on the Internet motivate even beginners and thoroughly assist them in mounting their attacks.

In addition, both classes of beginner and experimented attackers have an unlimited access to detailed information, from trusted sources, on recently discovered vulnerabilities and their potential exploit. They are capable to design the most appropriate strategy that supports their motives and satisfies their objectives. Thus, their deliberated actions are becoming well structured, precise and more devastating.

Attackers can identify multiple flaws of the victim relying on simple to use vulnerability scanner tools. Based on their knowledge and discovered weaknesses, they try to determine different attacking scenarios that fulfill their objectives. Generally, they look for scenarios associated with a single vulnerability to minimize their logged traces. But, when such scenarios are not available, they include sequentially exploitable vulnerabilities in designed attacking scenarios to increase their success chance. Such aspect should be considered in estimating potential damage of malicious events experienced by the target computing environment.

Unfortunately, it is difficult to determine exact vulnerabilities exploited by an attacker. But, in our risk model, vulnerabilities severity element tries to express above cited risk factor by the mean of flaws scores. Such risk element gives an insight on the potential exploit of a collection of supported flaws. It focuses on the extent of attacker opportunity to implement deeper attack and inflict more damage to the target asset knowing its vulnerabilities and their

severity scores. In fact, severity scores are available for disclosed vulnerabilities through different normalized and public databases including National Vulnerabilities Database (NVDB) and Open Source Vulnerabilities Database (OSVDB). They are calculated by standardized vulnerability scoring systems such as the Common Vulnerability Scoring System (CVSS) [88], [280], [284].

Vulnerabilities scoring systems are proposed and implemented by trusted organizations such CC-CERT, NIST, SANS, Microsoft and others. Their main objective is to provide information technology managers with required information to prioritize and remediate most damaging technical flaws [74], [271]. These systems process nearly real-time disclosed vulnerabilities. Moreover, they are associated with comprehensive historical databases that save traces of discovered vulnerabilities. These databases are publicly available. Besides, they are structured according Common Vulnerability and Exposure (CVE) naming standard that recommends an extensive set of features to describe technical vulnerabilities [87].

Several public and proprietary scoring or rating systems have been implemented. US-CERT system ranks discovered vulnerabilities using severity scores ranging from 0 to 10. For this system, scores are determined based upon different factors which are dependent to vulnerabilities, their environment of exploit and effects on Internet infrastructure [256], [394]. CERT system was considered as threat oriented rather than vulnerability oriented because it focuses on general factors that fail to ensure standard scoring of different vulnerabilities. Moreover, it provides security managers with general threat level to given vulnerability which is not effectively useful for prioritization and remediation ends [140]. The SANS vulnerability analysis system concentrates on flaws origins and their criticalities. The Symantec proprietary system determines vulnerability severity scores based on access levels required to exploit discovered flaw and its impact on the target asset [380]. The security response group of Microsoft manages its own rating system that prioritizes user reported vulnerabilities based on their exploitation and impact levels [82], [259]

Above discussed scoring systems are not normalized. They support different weaknesses either in considered factors about vulnerability or scoring process itself. The majority of these systems include few vulnerability features in their scoring processes which are not appropriate in rating flaws of different categories. The single standardized scoring system that overcomes these shortcomings is CVSS (Common Vulnerability Scoring System) of the Forum and Incident Response and Security Team (FIRST) [82].

CVSS is the most widely accepted scoring system managed by FIRST. It reinforces the vulnerability management policy of the organization by adopting harmonized scoring process for all of its software vulnerabilities. CVSS is based on standardized scoring process. It offers an open framework that allows a detailed description on how determining scores of different vulnerabilities. Several organizations use CVSS for publishing their vulnerability bulletins, communicating severities of vulnerabilities in their commercial products or managing flaws of their operational software. CVSS scores and processed vulnerabilities are publicly available in different databases such as NVDB, OSVDB and bugtrack [59], [280], [284].

CVSS severity scores are determined using a numerical scale of 0 to 10. They are based on three relative scores associated with considered metric groups. Basic metric group concerns constant vulnerability characteristics over time and across computing environments. These metrics include access vector, access complexity and impact. The temporal metric group instead focuses on vulnerability features that change over time. Exploitability and remediation factors of this group emphasize changes respectively in the available exploitation technique and remediation status of discovered vulnerability. Metrics of the first two groups are determined when vulnerability bulletin is released by software vendors or security analysts. The last group of environmental metrics concerns features related to vulnerability context within the organization. Metrics of this group are evaluated by users of the computing environment. They aim at integrating organization specific features in scoring process. Last two groups of metrics are optional for all vulnerabilities. But, metrics of the first group are critical in computing CVSS scores.

In the proposed risk model, potential effect due to multiple exploits was included. Its detailed estimation is not feasible since complete data on attack stages and exploited vulnerabilities are not available. We propose a partial approximation of such effect by considering a global severity factor of supported vulnerabilities. Such risk element broadly resumes offered opportunities to threat agent to make deeper attack. Moreover, it informs on the extent of possible damage when an attacker has the potential to exploit more than single flaw of the target asset. Global vulnerabilities severity factor of the proposed model will be determined based on supported flaws of the target assets, their CVSS scores and common weaknesses of NVDB and OSVDB databases.

In this work, severity factor is computed for each asset of the computing environment. Initially, supported vulnerabilities of the target asset are discovered using widely available automated tools. Then, severity scores of identified flaws are evaluated. Before this,

discovered vulnerabilities are dispatched between different groups of the second flaw classification, considered by our risk model. This categorization is based on NVDB. It identifies patched, unpatched and unresolved vulnerabilities of those discovered in the computing environment. It solely focuses on the gravity of potential damage associated with different vulnerability groups. Several criteria can be considered in this step including the efficacy of available countermeasures [9], [26], the level of exploitation potential [397] and general popularity, with consideration of top ranked vulnerabilities of SANS [74]. According to selected criteria, each of identified vulnerability groups is associated with a weighting coefficient that ranges from 0 to 1. These weighting coefficients are determined relying on worldwide information. They are proportionally assigned depending on potential damage possibly incurred due to the exploit of vulnerabilities of the corresponding group [10]. The overall flaws severity of the target asset is determined as a weighted sum of relative severity factors of considered vulnerabilities groups. This is summarized by the process presented in section §5. 4.2.

5.3.6 Controls effectiveness

The negative effect of different threats experienced by computing environment assets can be reduced relying on multiple categories of security controls. For federal organizations, NIST has imposed minimal required security controls. The basic control collections of federal organizations consist of managerial, operational and technical countermeasures [248], [249], [260]. Managerial or organizational controls focus on security policies, guidelines and standards that protect and preserve organization mission continuity. Operational controls mainly concern physical countermeasures that ensure consistency and uniformity in security operations. Technical safeguards are required to protect critical and sensitive information and information systems. Each of identified three categories supports preventive, corrective and recovery security controls.

Within this context and to ensure continuous monitoring and revision of the computing environment security state, NIST has presented a guideline for assessing security controls in federal information systems. This guideline helps organizations in evaluating proper implementation, operation and effectiveness of their security controls. Highly structured process of the guideline conveniently assists security practitioners in conducting safeguards assessment starting by the preparation and development of assessment plans until post-assessment and report analysis [323]. Other security audit methods such as CRAMM and

ISRAM allow measurement of countermeasures effectiveness as recommended in standards of the ISO-27000 series [85], [80], [115], [173], [175], [216], [367].

In assessing effectiveness of functional security controls different features, related to the safeguard or its environment, should be considered. Correctness attribute is concerned with implementation flaws of countermeasure. Strength of security control reflects both its capacity to resolve supported vulnerabilities and attacker effort and capability to reach his objective. Control function feature determines the types of ensured protection, preventive, corrective or recovery [38]. Other features can be considered in addition to those previously discussed including configuration flaws and dependence between safeguards. Such extended feature set ensures a stringent assessment that addresses countermeasures implementation, operation and returned services with consideration of confidentiality, integrity and availability requirements of the computing environment.

In this context, controls effectiveness estimation should not be based specifically on collected data either within the organization or from controls vendors. Both sets of collected data are complementary and should be considered to evaluate effectiveness of current security strategy and its controls. Moreover, assessment of security controls involving collected datasets and discussed features can be conducted manually or using automated tools.

Safeguards effectiveness can be determined using an automated tool such as ASSET of NIST. This tool is associated with the guideline of [341]. The recommendations of the guideline impose a normalized approach for security self-assessment for federal organizations in order to determine current status of their security programs and plan potential improvements. Moreover, ASSET is useful in evaluating technical controls even if their assessment process is more complicated than managerial and operational countermeasures. As stated in [151], [323] effectiveness of installed managerial, operational or technical security controls can be estimated relying on an assessment process, including ASSET process, in conjunction with an approved security plan of the organization. In fact, the security plan provides security assessor with detailed description of installed and planned security controls with respect to recommended security baseline [248], [260]. Another possible way to measure security controls effectiveness focuses on audit reports. This alternative is based on manual processing of detailed information on installed safeguards provided by audit reports.

Among discussed alternatives to gauge security controls effectiveness, the last one is the simplest and widely accepted regarding objectivity and confidence of audit reports. Moreover, it was mentioned in risk management and best practice standards [80], [175] not only due to

availability of audit reports even for private organizations, but also for multilevel assessment ensured by security audit mission. Additionally, control assessment in security audit is achieved by considering both benchmarks provided by developer and configuration of the target system, with respect to its security requirements. Actually, information security systems of any organization should be audited at least twice a year by internal and external certified auditors as imposed by national and international regulation [22], [118], [174], [282]. An audit report of each mission is composed of two main parts that focus respectively on nominal and technical audit [313]. Nominal audit is concerned with organizational and procedural aspects of managerial, operational and technical security controls. Technical audit instead addresses implementation, operation and outcomes of different categories of safeguards. Multiple penetration testing scenarios are experimented in this part to technically evaluate the resistance of deployed security strategy. Both parts of audit reports, in addition to security expert recommendations and worldwide information, are required for a coherent assessment of countermeasures effectiveness.

In the proposed risk model, we focus on security controls of different categories. The estimated effectiveness of controls are interpreted as their reduction factors that shrink the exploitability extent of supported vulnerabilities and decrease attackers opportunities to successfully execute intrusive actions. Reduction factor values range between 0 and 1. Based on available data, they are determined for different flaw categories as presented in table 5.2. This vulnerability-safeguards matrix involves flaws categories determined based on their exploit objectives.

Table 5.2: Security control effectiveness by flaw category

Flaws categories	Controls				
	s_1	…	s_l	…	s_L
V^1	$e_{1,1}$	…	$e_{l,1}$	…	$e_{L,1}$
⋮	…	…	…	…	…
V^r	$e_{1,r}$	…	$e_{l,r}$	…	$e_{L,r}$
⋮	…	…	...	…	…
V^R	$e_{1,R}$	…	$e_{l,R}$	…	$e_{L,R}$

Where, $e_{l,r}$ is the expected effectiveness of the security control s_l against vulnerabilities of the category V^r, $e_{l,r} \in [0, 1]$ and $e_{l,r} = 0$, if the control s_l is not applicable in reducing the exploit of vulnerabilities of the class V^r.

The relative effectiveness of single control s_l to defend against vulnerability set V_i of asset a_i, is evaluated by considering different categories that compose V_i. Assuming that $n_{i,R}$ flaw categories are included in V_i, the mean effectiveness of the security control s_l against exploits of vulnerabilities supported by the asset a_i is $e_{l/i}$:

$$e_{l/i} = 1/n_{i,R} \sum_{r}^{n_{i,R}} e_{l,r} \quad (5.4)$$

Assuming that the initial security strategy, $SS_{i,0}$, associated with the asset a_i is composed by l_i independent controls. Its relative effectiveness against supported vulnerabilities of a_i is estimated by $e_{S_{i,0}}$:

$$e_{SS_{i,0}} = 1 - \pi_{SS_{i,0}} \quad (5.5)$$

$\pi_{SS_{i,0}}$ corresponds to the bypass rate of the initial security strategy, $SS_{i,0}$, of the asset a_i regarding categories of its supported vulnerabilities. It is determined by the mean failure rate based on the above matrix as follows:

$$\pi_{SS_{i,0}} = 1/n_{i,R} \sum_{r}^{n_{i,R}} \pi_{i,r} \quad (5.6)$$

where $\pi_{i,r}$ assesses the failure rate of $SS_{i,0}$ with respect to single vulnerability category, V_i^r, of a_i. It is interpreted as the probability of potential exploit of any vulnerability, $v_j \in V_i^r$, of the victim. It is estimated using table 5.2 by:

$$\pi_{i,r} = \prod_{l}^{l_i} 1 - e_{l,r} \quad (5.7)$$

Additionally, an asset is targeted by an internal or external attackers depending mainly on its attractiveness and supported weaknesses. But, despite of its high profitability for attackers, it remains unreachable by them while supported vulnerabilities are appropriately fixed. Thus, the probability of targeting an asset depends roughly on the failure of its security strategy. Such probability can be estimated using the failure rate of deployed security control combination to protect the target asset as discussed above. The probability of targeting given asset, a_i, of the computing environment that supports security strategy $SS_{i,0}$ can be approximated by the overall probability of failure of all its controls.

5.4 Basic risk evaluation

The basic risk of the asset a_i, $R_{i,q}^{B}$, due to mounted attack c_q depends on its exposure, vulnerabilities severity and controls effectiveness. It is expressed by the function Φ as follows, given that: c_q : a threat targeting a_i,

V_i : a set of supported vulnerabilities by a_i,

W_i : a vector of weights associated to security concerns of a_i,

$SS_{i,0}$: the initial set of security controls deployed to protect a_i.

$$R_{i,q}^{B} = \Phi\left(X_{i,q}, Y_i, Z_{i,0}\right)$$
$$\begin{cases} X_{i,q} = f\left(I_i, L_{i,q}\right) \\ I_i = f_1\left(U_i, W_i, V_i\right) \\ Y_i = g\left(V_i, \lambda\right) \\ Z_{i,0} = h\left(V_i, SS_{i,0}\right) \end{cases} \qquad (5.8)$$

where:

$X_{i,q}$: the exposure of the asset a_i, given that the threat c_q takes place, determines the expect loss of the target asset due to the mounted attack. It is a function of the overall impact factor of the asset a_i and the likelihood of the threat c_q.

I_i : the overall impact factor of the asset a_i represents the expected loss due to potential exploit of supported vulnerabilities. It depends on asset value, U_i, its supported flaws, V_i, and considered security services, W_i.

$L_{i,q}$: threat likelihood reflects how likely an attack, c_q, has a negative effect on the victim, a_i, when occurring. Its estimation is based on the probability of occurrence of c_q and failure rates of deployed security controls, on the target asset, to defend against an exploit of searched flaws of a_i.

Y_i: severity of vulnerabilities of an asset expresses to which extent the mounted attack has the opportunity to inflict more damage. It is determined based on supported vulnerabilities, their CVSS scores and gravity weights, λ, of considered flaws groups.

$Z_{i,0}$: effectiveness of the current security strategy of the asset a_i represents the defensive capacity of the target asset. It is estimated using the relative effectiveness of installed security controls, $SS_{i,0}$, with consideration of supported vulnerabilities.

5.4.1 Risk exposure evaluation

Risk exposure arising from mounted attacks should be determined based on an appropriate combination of estimated threat likelihood and expected potential damage of the target asset.

In this work, the function *f* implements such combination. The simplest way to combine both risk parameters with respect to ALE standard is the product function. The potential expected loss due to mounted attack and its impact on the target asset is calculated as follows:

$$\begin{aligned} X_{i,q} &= f(I_i, L_{i,q}) \\ &= I_i \times L_{i,q} \end{aligned} \quad (5.9)$$

Considered risk elements, asset impact and threat likelihood, are determined based on identified parameters of respective sections 5.3.2 and 5.3.3. Their estimation steps are discussed in following two sections.

5.4.1.1 Impact estimation

The impact of an asset is determined based on (5.1) as previously discussed. It is estimated in terms of confidentiality, integrity and availability losses, as recommended in [175], [282], [340], with consideration of asset value and supported vulnerabilities.

$$I_i = f_I(U_i, W_i, V_i)$$

U_i: asset value is estimated based on business income, B_i, and operating cost P_i of the asset a_i

V_i: set of supported flaws by the asset a_i identified and reported by a vulnerability scanner

W_i : weights of security requirements for the asset a_i. Such vector can be estimated before determining the value of an asset relying on the security policy and asset classification, if exists.

a) Asset valuation process

In this step, asset valuation process supposes that the organization is structured into different business function each of which is composed of several business processes. Each business process uses a subset of assets associated with the business function to implement one of its activities. In this context, asset value can be determined based on its contribution in different business functions and by considering its business value. The annualized business value of a given asset a_i corresponds to the difference between its total annual revenue and total annual cost of acquisition and operation [216]. Additionally, in real time environment, the daily worth of an asset can be determined by dividing its annual business value by the number of business days in a year.

In the proposed valuation process, business income of an asset should be determined using annual revenue of the organization and assets importance coefficients. The annual revenue of an organization corresponds to the sum of the incomes of all its business functions. Within each business function, different weighting coefficients are assigned to assets to express their

importance in achieving its objectives and implementing its activities. The importance coefficient of each asset is determined by considering its weights over all business functions that implement the organization's mission. A simple process to appraise asset values based on chosen approach is given below:

B_i : total annual revenue of the asset a_i,
P_i : annual investment cost of the asset a_i,
B : total annual revenue of the organization,
BF : set of business functions of the organization,
η_f: importance or criticality of the business function f to the organization's mission,
A_f: set of assets required to achieve the business function f,
$\varepsilon_{i,f}$: weighting coefficient that assesses the degree of involvement and the importance of the asset a_i in fulfilling the business function f,
d: number of business days in a year,
U_i : annual business value of asset a_i,
U_i^D : daily business value of asset a_i,
ε_i : importance coefficient of asset a_i over all business functions in which it is involved.

Asset value determination process

1-determine normalized importance coefficient, ε_i^N of the asset a_i:

1-1: compute importance coefficient of a_i over all business functions $\varepsilon_i = \sum_f \eta_f \varepsilon_{i,f}$

1-2: compute normalized importance coefficient of a_i over all assets in $A = \bigcup_f A_f$,

$$\varepsilon_i^N = \varepsilon_i \Big/ \sum_j \varepsilon_j$$

2-evaluate $B_i = B \times \varepsilon_i^N$

3-determine $U_i = B_i - P_i$

4-compute $U_i^D = U_i / d$

U_i^D is useful for loss determination in our real time system after the estimation of the impact factor of mounted attack and number of days required to repair damaged asset. However, it is enormously difficult, in the information security domain, to estimate the number of days to recover normal function of the computing environment due to the complicated nature of mounted attacks, either if the victim is completely replaced.

In the proposed risk model, asset value is required to assess inflicted damage by the mounted attacks. It may be estimated based on the annual business value and operation cost of the asset. It merely corresponds to the difference between these monetary values as given by the following formula:

$$U_i = B_i - P_i \quad (5.10)$$

b) Impact factor determination

Different security services are required by assets of the monitored computing environment. Commonly considered security principles in security policies and security reports, such as audit or security plan or risk assessment, include confidentiality, integrity and availability (CIA). Other security services can be added to these including non-repudiation and authenticity depending on the target asset and the organization's mission. In this work, specifically, these three common security concerns are included in determining impacts of targeted assets. Their relative impact factors given by the vector $\Gamma_{i=<} \gamma_{i,1}, \gamma_{i,2}, \gamma_{i,3}>$, respectively for confidentiality, integrity and availability impact factors are estimated for each victim a_i. These factors express fractions of potential loss interpreted as a failure in fulfilling associated security concerns for the target asset. In our risk model, they are determined using impact scores given by CVSS for supported vulnerabilities, $v_j \in V_i$, $j=1..|V_i|$, as discussed in section §5.3.5. Impact scores present potential effects of considered vulnerabilities on CIA services using numerical values scaled between 0 and 10. Confidentiality, integrity and availability impact factors for a given asset a_i are estimated based on supported vulnerabilities as follows:

$$\gamma_{i,n} = \frac{1}{|V_i|} \sum_{v_j \in V_i} is_{j,n} \quad (5.11)$$

where:

$\gamma_{i,n}$: the impact factor of the asset a_i associated with the n^{th} security service

$| V_i |$: the size of V_i, set of supported vulnerabilities of the asset a_i

$is_{j,n}$: normalized impact scores in terms of n^{th} security principle, associated to the vulnerability j, (impact scores of vulnerability j on confidentiality, integrity and availability requirements divided by the maximum score). Scores, $is_{j,n}$, are available for each vulnerability v_j through the CVSS scoring system.

Security concerns are not uniformly important for all assets within the organization. But, their relative importance depends on the target assets and business functions involving these. Several assets require only integrity and availability such as DNS database either in e-banking or ecommerce systems. Others specifically need availability including printer and routers. Such context dependent aspect is considered in the proposed model. Different weight coefficients are associated to impact factors to bring out the relative importance of considered security concerns of the target asset. These importance coefficients are determined by experts

based on security requirement of the business function that involves the target asset. They correspond to the values within the interval [0,1] given by the vector W_i. The overall impact factor, τ_i, of the asset a_i based on its supported weaknesses and its requirements in terms of considered security principles is determined by the following formula:

$$\tau_i = \sum_n \gamma_{i,n} \, W_{i,n} \quad (5.12)$$

where :

$W_{i,n}$, a weight coefficient of n^{th} security service required by the asset a_i, n=1..3, for basic security principles of CIA.

The overall impact factor of an asset estimates the fraction of its value that can be lost due supported flaws. Based on the value and the impact factor of an asset, its potential loss due to flaws exploit can be approximated by a product function as follows:

$$\begin{aligned} I_i &= f_1(U_i, W_i, V_i) \\ &= \frac{1}{|V_i|} U_i \sum_n \sum_j^{|V_i|} W_{i,n} \; is_{j,n} \\ &= \tau_i \times U_i \end{aligned} \quad (5.13)$$

5.4.1.2 Threats likelihoods estimation

Threat likelihood, $L_{i,q}$, expresses how likely threat c_q occurs and impacts the asset a_i regarding its current security state. It can be interpreted as the probability that threat c_q has a negative effect on asset a_i when occurring. But, negative effect arising from any threat has always a single source which is exploiting supported flaws of the victim. In the proposed model, the likelihood of a given threat is thought of as its probability of affecting or causing damage to the target asset. It is expressed in terms of probabilities of occurrence of threats and their opportunities to affect target assets as follows:

$$\begin{aligned} L_{i,q} &= p(c_q \textit{ affects } a_i) \\ &= p(c_q \textit{ occurs }) \, p(c_q \textit{ affects } a_i \textit{ given that } c_q \textit{ has taken place}) \\ &= p(c_q) \, p(c_q \textit{ affects } a_i \,/\, c_q) \end{aligned}$$

$p(c_q)$: the probability that threat c_q takes place. This probability is determined by the multimodel analysis and detection engine that process events and monitor security of the target asset in the computing environment

$p(c_q \textit{ affects } a_i \,/\, c_q)$ expresses the opportunity of an attacker to bypass current security controls of the target asset and exploit at least one of its supported weaknesses. It can be estimated

relying on the failure rate of currently deployed security strategy, to eliminate supported flaws of the target asset a_i, with respect to potentially exploitable vulnerabilities of the identified threat c_q .

Supposing that threat c_q exploits different vulnerabilities of classes given by $V_k^C = \left\{ V_q^1,...,V_q^{n_{k,r}} \right\}$ and a_i supports flaws determined by classes of $V_i^C = \left\{ V_i^1,...,V_i^{n_{i,r}} \right\}$, potentially exploitable weaknesses by threat c_q when targeting a_i belong to classes given by $V_{i,q}^C = V_i^C \cap V_q^C$. By considering the current security strategy of the asset a_i, $SS_{i,0}$ which is composed by l_i security controls, and supposing independence between included vulnerability classes, the probability that c_q affects a_i when occurring can be gauged as follows:

$$p(\, c_q \ affects \ a_i \,/\, c_q \,) = \prod_{\forall V^r \in \ V_{i,q}^C} \pi_{i,r} \quad (5.15)$$

Where $\pi_{i,r}$ is the failure rate of currently deployed security strategy of a_i with consideration of vulnerabilities of class V^r . It can be estimated using the effectiveness of controls that compose the deployed security strategy using (5.7), as discussed in the previous section § 5.3.6.

In our risk model, threats likelihoods are determined as follows:

$$\begin{aligned} L_{i,q} &= p(c_q \text{ affects } a_i) \\ &= p(c_q) \prod_{\forall V^r \in V_{i,q}^C} \pi_{i,r} \end{aligned} \quad (5.16)$$

5.4.2 Vulnerabilities severity estimation

Vulnerabilities severity element of our risk model depends on supported flaws of the target asset and their severity scores, according the CVSS system, and actual remediation states. After discovering flaws of the considered asset, three main groups are identified. Patched and unpatched groups, respectively V_i^p and $V_i^{\overline{p}}$, focus on weather resolved vulnerability, by the manufacturer or developer, are effectively corrected or not in the target asset. Unresolved vulnerability group, V_i^u, includes flaws for which technical countermeasures still unavailable. According to [26] vulnerabilities of the second group are the most searched by attackers.

Vulnerabilities severity parameter allows partial information on the gravity of damage when an attacker has the potential to exploit more than a single weakness of the victim. It is determined using a weighted sum of scores over discriminated vulnerability groups. The main

steps of vulnerabilities severity computation process for the proposed risk model are presented by the activity diagram in figure 5.5.

In this process, vulnerabilities of each asset, V_i, can be discovered and reported using automated tools such as ISS, RealSecure or Cybercop. For each vulnerability, $v_j \in V_i$, its severity score is determined based on public databases such as NVDB and OSVDB that in turn save CVSS system assessments. Then, it is assigned to one of the considered groups in , V_i^G , namely patched, unpatched or unresolved, $V_i^G = \{ V_i^p , V_i^{\bar{p}} , V_i^u \}$, relying on its remediation state. The latter depends on availability and implementation state of recommended countermeasures. Identified vulnerabilities of different groups are involved in evaluating relative severity scores of considered groups. After that, a weight vector λ, $\lambda = < \lambda_p , \lambda_{\bar{p}} , \lambda_u >$, associated to V_i^G is determined by security experts. It reflects the world wide gravity of exploiting vulnerabilities of each group, $V^r \subset V_i^G$, *r=1..3*. Both, relative scores and weighting coefficients will determine severity scores of vulnerabilities of different groups as illustrated by figure 5.5.

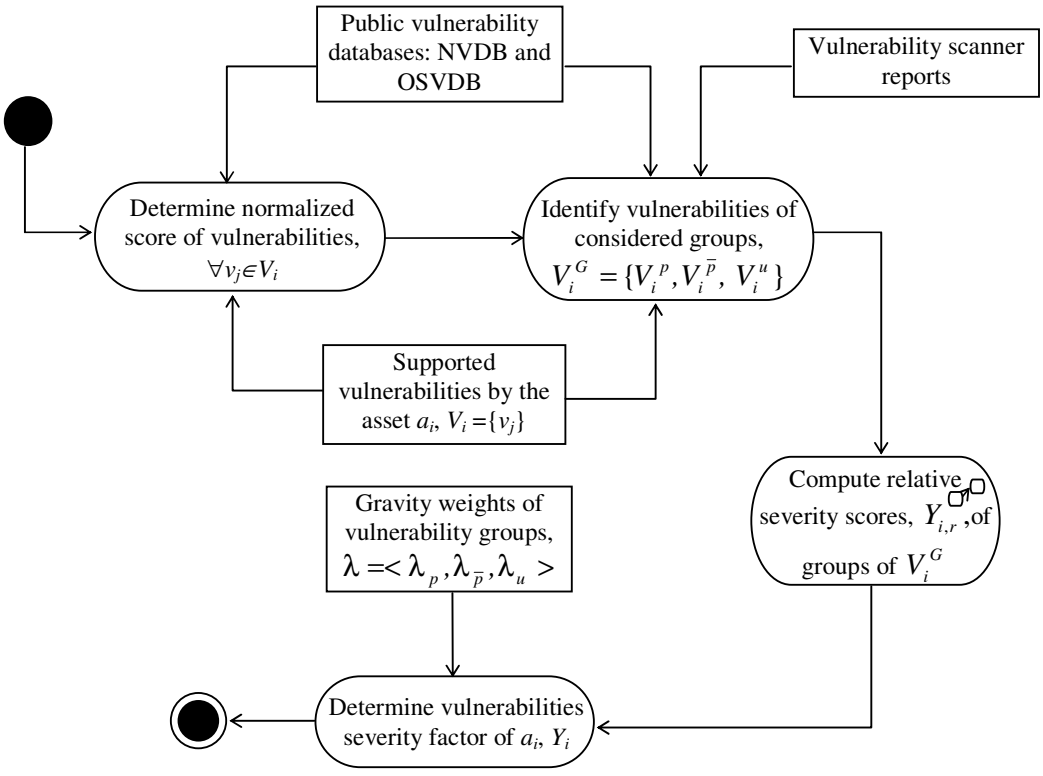


Figure 5.5: Vulnerabilities severity determination process

The severity factor of supported flaws of the target asset a_i is estimated by the latter process using the following formula:

$$Y_i = g(V_i, \lambda) = \sum_{V^r \subset V_i^G} \lambda_r \, Y_{i,r} \quad (5.17)$$

$$Y_{i,r} = \frac{1}{|V^r|} \sum_{j}^{|V^r|} sr_j$$

Where:

Y_i : vulnerabilities severity of the asset a_i

λ_r : weighting coefficient of flaws of the group V^r, determined based on world wide information including public databases, security reports and expert recommendations

sr_j :normalized severity score of vulnerability v_j of the group V^r, $v_j \in V^r$.

$Y_{i,r}$: mean normalized severity scores of flaws of the group V^r, $V^r \subset V_i^G$.

Y_i , λ_r, sr_j and $Y_{i,r} \in [0,1]$, and $\sum_r \lambda_r = 1$

Another group of secret vulnerabilities can be considered in the severity determination process. It concerns unpublished flaws either accidentally, due to lack of thorough test of self developed software, or deliberately because of fear to negatively influence customer trust. Weaknesses of this group can be easily identified using existing vulnerability scanners

Moreover, the vulnerability popularity feature is useful in measuring flaws severity. Such attribute was introduced by US-CERT and used in OSVDB and SANS [74], [394]. It addresses to flaws ranking by considering either the most searchable weaknesses identified by SANS or the most frequently checked vulnerabilities presented by OSVDB [284], [336]. Furthermore, elapsed time between disclose and resolution of vulnerability can serve as a good indicator of vulnerability severity because, as illustrated in [26], attack frequency remarkably increases, for a short time period, just after the publication of patches.

5.4.3 Controls effectiveness approximation

Controls effectiveness is a risk element that addresses the extent to which vulnerability severity and exploitability are reduced and attacker's opportunity of success is shrunken. The effectiveness of security strategy of a given asset is determined with consideration of different supported flaws. In the proposed risk model, it is estimated relying on the efficacy of each countermeasure of the deployed security strategy with respect to considered vulnerability categories. Effectiveness matrix of table 5.2 that links between vulnerability categories and operational controls is defined for this aim. It is involved in assessing the overall effectiveness

of current or planned security strategy with respect to selected controls and supported vulnerabilities.

In our risk model, controls effectiveness estimation takes account of:

$SS_{i,0}$: set of dedicated or shared security controls associated with the asset a_i, $a_i \in A$ and $SS_{i,0}=\{s_l, s_l \in SC\}$

V_i : set of vulnerabilities supported by a_i, that can be structured into $n_{i,r}$ different categories based on exploit objective, with respect to those initially considered

The effectiveness of deployed controls, $Z_{i,0}$, is expressed as follows:

$$Z_{i,0} = h(V_i, SS_{i,0}) = 1 - 1/n_{i,r} \sum_{r,\forall V^r \in V_i^c}^{n_{i,r}} \pi_{i,r} \qquad (5.18)$$

where $\pi_{i,r}$ represents the expected failure rate of deployed security strategy associated with the asset a_i regarding its vulnerabilities of category V^r. This rate is estimated, as discussed in section §5.3.6 using (5.7), by considering the single remediation effect of each control s_l of $SS_{i,0}$ on given vulnerability category as follows:

$$\pi_{i,r} = \prod_{l}^{l_i} (1 - e_{l,r})$$

where, l_i : the size of deployed countermeasure set, $SS_{i,0}$,

$e_{l,r}$: the expected single effectiveness of control s_l against supported vulnerabilities of the category V^r as determined by table 5.2.

5.4.4 Basic risk assessment

The basic risk of an asset or a computing environment evaluates its actual security state with consideration of unmanageable factors namely internal and external threats. In this model, basic risk of target asset a_i is estimated by combining its exposure, $X_{i,q}$, vulnerability parameter, Y_i, and effectiveness of current security strategy, $Z_{i,0}$. The three risk parameters on which is based our model are approximated, as discussed in previous sections, using following formulas:

$$\begin{aligned} X_{i,q} &= f(I_i, L_{i,q}) \\ &= I_i \times L_{i,q} \\ Y_i &= g(V_i, \lambda) \\ &= \sum_r \lambda_r y_{i,r} \\ Z_{i,0} &= h(V_i, SS_{i,0}) \\ &= 1 - 1/n_{i,r} \sum_r^{n_{i,r}} \pi_{i,r} \end{aligned} \quad (5.19)$$

However, the exposure parameter should be evaluated with respect to Q_i mounted attacks on the target asset a_i, determined by the sequence T_i, within a given time interval:

$$X_i = 1/Q_i \sum_{c_q \in T_i} X_{i,q} \quad (5.20)$$

The basic risk of an asset is estimated by a combination of identified parameters as expressed below by:

$$\begin{aligned} R_i^B &= \Phi(X_i, Y_i, Z_i) \\ &= X_i \times Y_i \times (1 - Z_{i,0}) \end{aligned} \quad (5.21)$$

Where $(1\text{-}Z_{i,0})$ expresses the estimated failure of deployed security strategy of a_i against detected threats. The overall basic risk of the computing environment that has N_t targeted assets, $A \supset A_t = \{a_1, ..., a_i, ..., a_{N_t}\}$, by different attacks is estimated by:

$$R^B = \sum_i^{N_t} R_i^B \quad (5.22)$$

The assessed basic risk of the monitored computing environment is then considered by the treatment component of the proposed risk management model. Treatment option selection and implementation are the main issues addressed by this second component as discussed in the following section.

5.5 Risk treatment

The risk management process is based mainly on two components. The risk assessment component focuses on analysis and evaluation of risks inflicted by different threats. It identifies risk elements and determines their variables to be considered in the evaluation step. The risk treatment component instead is concerned with post-evaluation steps. It addresses two main steps of treatment option selection and implementation [175], [325].

The risk treatment component of the proposed management process imposes risk prioritization before implementing the selected treatment option. Such step is required to guide the mitigation process and choose appropriate treatment activities to the identified risks. Priorities can be determined by the organization's managers and decision makers or merely by ranking evaluated risks. Other techniques such those discussed later in this section are useful in prioritizing risks of the computing environment.

Risks priorities and other factors such as expected investment costs and benefits are included in deciding which treatment option will be selected and implemented. Risk mitigation is possibly achieved by adopting different options including avoidance, transfer, retention or reduction. Avoidance option is selected when treated risk is excessively high and implementation costs of other options exceed their benefits. It imposes the elimination of risk causes or consequences or both [325]. Risk transference option ensures transfer of losses to other organizations, outsourcing companies or insurance agencies, which are capable to manage and reduce risks to meet an acceptable level (acceptance criterion). Risks are retained if their damage costs are manageable and do not exceed tolerated risk level of the organization. Reduction option involves the selection of appropriate managerial, operational and/or technical controls to reduce assessed risk and ensure its acceptance regarding the tolerance level of the concerned organization [80], [175].

In our work, risk treatment solely focuses on reduction option. The proposed risk mitigation process is capable to determine optimal security strategies to defend against detected attacks with respect to the tolerated risk level and available security budget of the organization. The selected security strategies consist of combinations of security controls. They are effective in impeding supported vulnerabilities exploit by mounted attacks. Selected security controls of these combinations are identified among those recommended by security experts with respect to security policy specification and the appendix A of the ISO 27001 [173]. Recommended subsets of countermeasures are saved by the idrs knowledge base. They concern different attack classes considered by our idrs framework.

Our mitigation process determined cost effective control combinations, based on recommended subsets. In previous works, mitigation processes identify appropriate security solutions in terms of residual risk [219], [229]. But if two or more security strategies have the same reduction effect on current risk, they will be identically rated according the single decision criterion used in [229]. The proposed risk mitigation process presents a refined solution to such problem. It includes another decision criterion, the security investment cost,

in its optimization function, in addition to that commonly considered. Thus, it sifts candidate security strategies to identify the best one that meets both decision criteria. Furthermore, identified security strategies minimize risk cost that concerns both residual risk and investment cost. In this step, other criteria that concern conflict and compatibility between controls can be considered to thoroughly identify appropriate combinations.

The cost effective control combination, presented by our incremental and iterative mitigation process, is identified by minimizing both residual risk and security investment cost. The residual risk of the computing environment corresponds to the unmitigated fraction of assessed risk by selected security strategy. It is estimated relying on the basic risk of the computing environment, R^B, and potentially deployed security strategy on it, SS_k. It is assessed using the following formula:

$$\begin{aligned} R_k^R &= \psi(R^B, SS_k) \\ &= R^B(1 - Z_k) \end{aligned} \quad (5.23)$$

Where Z_k corresponds to the estimated effectiveness of the security strategy SS_k in protecting against exploit of vulnerabilities supported by victim assets in the set A_t. It is estimated by the mean effectiveness overall target assets of A_t using:

$$Z_k = 1 - 1/N_t \sum_{i}^{N_t} \pi_{SS_{i,k}} \quad (5.24)$$

where $\pi_{SS_{i,k}}$ is an estimation of the failure rate of the security strategy $SS_{i,k} \subset SS_k$ in defending against attacks targeting any of supported vulnerabilities of the asset $a_i \in A_t$. It is determined based upon identified categories of a_i's vulnerabilities and elementary effectiveness of controls in $SS_{i,k}$, as presented above in section §5.3.6

Security investment cost, to mitigate current risk experienced by assets of the computing environment, corresponds to the cumulative expected costs of acquiring, installing and maintaining selected security controls. It is estimated using global expected costs of countermeasures of SS_k, composed by L_k safeguards, as follows:

$$\begin{aligned} \Delta(SS_k) &= \sum_{l=1}^{L_k} \delta(s_l) \\ &= \omega_k \end{aligned} \quad (5.25)$$

Where $\delta(s_l)$ is the expected cost of the security control s_l of SS_k.

The proposed process is based on the following optimization function to identify optimal control combinations:

$$\text{Minimize } \Psi(R^B, SS_k) = R_k^R + \omega_k$$

Subject to:

1) $R_k^R \leq \bar{\tau}$
2) $\omega_k \leq \beta$

Where $\bar{\tau}$ and β are respectively the tolerated risk of the target computing environment and the allocated security budget to secure it.

Our risk treatment process prioritizes risks, and consequently target assets, before designing candidate security strategies. It is based on prioritizing factor that ranks risks by jointly considering the target asset criticality and its damage degree. Such factor is useful in identifying the most serious risk to be immediately addressed. These risks are generally experienced by the most critical assets which are served in the first line by this process. Such aspect remains unreachable if using other ALE based risk management methodologies, even if supported by automated tools. These methodologies lack required flexibility to ensure a clear discrimination between catastrophic and manageable risks which is extremely important in mitigation step as discussed in section §5.3.4 and stated in [365]. Moreover, priority factor integrated in this mitigation process ensures treatment of all other preponderant risks before less significant ones.

Rather risks prioritization, our proposed treatment process takes account of global constraints. These constraints concern reached risk cost due to the designed security strategy. On one hand, they ensure low residual risk that shouldn't exceed the prespecified tolerance level, $\bar{\tau}$. On the other hand, they impose acceptance of selected controls cumulative costs with respect to the allocated security budget, β. The proposed treatment process imposes different tolerance levels respectively to assessed risks with respect to target assets. Each asset considered by the process is assigned with a tolerated risk level that reflects its importance, within the computing environment, and security needs. Determined tolerance levels are inversely proportional to computed priorities, such that, the most critical and damaged asset is associated with the lowest acceptance level. Such constraint is imposed in our mitigation process for many reasons. It ensures the maximum security to highly critical and damaged assets by incrementally designed security solution. Moreover, it allows the development of context dependent security strategy that serves serious risks in the first rank. Additionally, it

gradually treats low priority damages to overcome the overall risk of the computing environment and meet security budget criterion.

The proposed mitigation process is incremental and iterative. Each increment is concerned with the security requirements of a single asset; whereas, an iteration corresponds to a single execution of the optimization process. An increment treats security risk experienced by the considered asset with respect to its priority factor and security strategies of previous increments. High priority risks are mitigated first with consideration of low tolerance levels to simultaneously ensure maximum risk reduction and global constraint satisfaction. Their remediation security strategies, if applicable, are imposed on low priority damages. Within each increment, multiple iterations of the optimization process can be performed depending upon identified candidate security strategies, their costs and residual risks. Iterations of an increment focus on relaxations of tolerated risk constraint. Different tolerance levels below the overall acceptable risk are considered. Gradual checking of different tolerance levels by increment's iterations ensures thorough sweep across candidate solution space and increases the chance to meet global acceptance criteria, residual risk and security investment constraints. However, the collective failure of iterations, when mitigating high priority risks, means that no feasible security strategy exists, regarding the recommended control set. Then, risks of the computing environment are irreducible to the imposed acceptable level using the available security budget. In this situation, even the treatment of low priority risks has no chance to satisfy imposed constraints and preserve vital business functions of the organization because serious damages are left unmitigated. The main steps of the proposed risk reduction process and their associated activities are summarized by figure 5.6.

As depicted by the diagram of figure 5.6, the risk mitigation process consists of three main composite activities of initialization, increment risk treatment and security strategy evaluation and selection. The initialization activity focuses on prioritizing risks and determining tolerance levels. It takes account of assets in A_t, targeted by detected attacks. Each asset, $a_i \in A_t$, is associated with a basic risk and criticality respectively R_i^B and cr_i. In order to set up a global ranking of risks to be followed by the treatment process, risk priority factor, pf_i, is evaluated based on incurred damage amplitude and criticality of the target asset, $a_i \in A_t$, as follows:

$$pf_i = \frac{cr_i R_i^B}{\sum_i^{N_t} cr_i R_i^B} \quad (5.27)$$

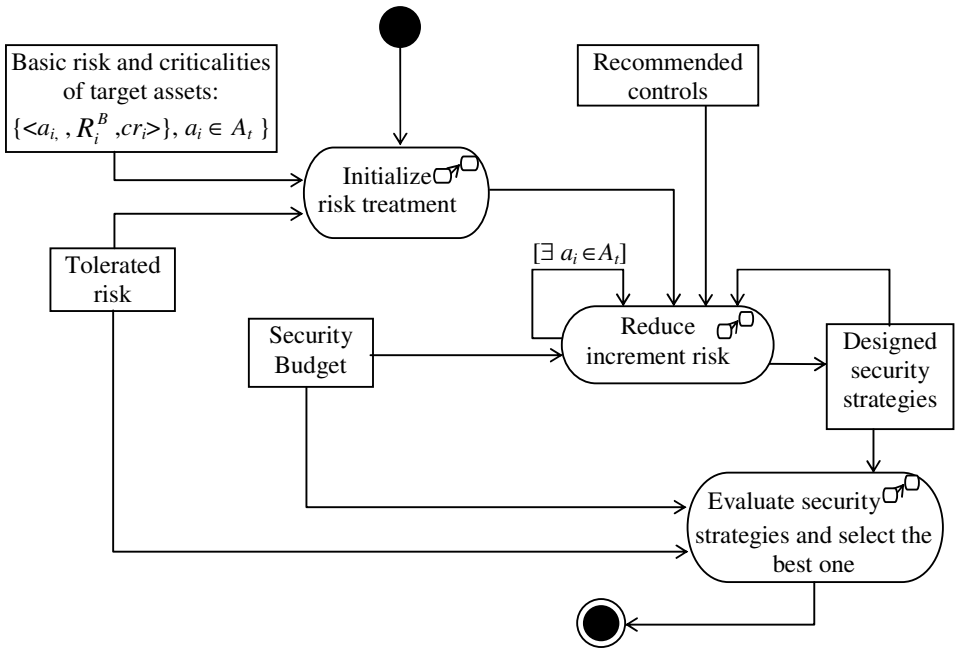


Figure 5.6: Risk reduction process

Determined priorities will serve to identify risks to be addressed at first ranks by the risk reduction process. To each prioritized risk is associated a tolerance level evaluated as follows:

$$\nu_i = \left(\frac{1 - pf_i}{N_t - 1} \right) \bar{\tau} \quad (5.28)$$

Where, $\bar{\tau}$ is the tolerated risk level of the monitored computing environment and ν_i is the tolerance level associated to the basic risk R_i^B of the asset a_i.

According to this, high priority risks are associated with low tolerance levels in order to ensure the selection of appropriate security strategies that remarkably reduce reached risk level and meet requirements of the treatment program. Different activities of the initialization step of the risk reduction process are depicted in figure 5.7.

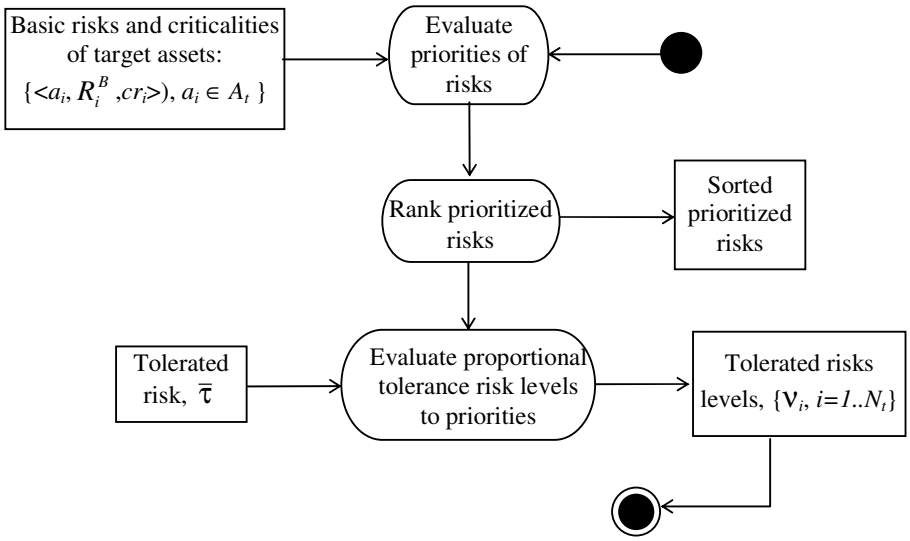


Figure 5.7: Risk reduction process initialization

Prioritized risks and their tolerance levels are then involved in the incremental and iterative risk reduction step of the process. In each increment, the reduction activity takes account of a single risk level. The initial increment treats the most serious risk associated with the highest priority, determined by R_{inc}^B, the cumulative basic risk until current increment *inc.* The other increments instead focus on high priority risks among those remaining unmitigated. Relaxed tolerance level, $\bar{\tau}_{inc}$, is iteratively determined for each increment with respect to $\bar{\tau}$. The main activities of increment risk mitigation are presented in figure 5.8.

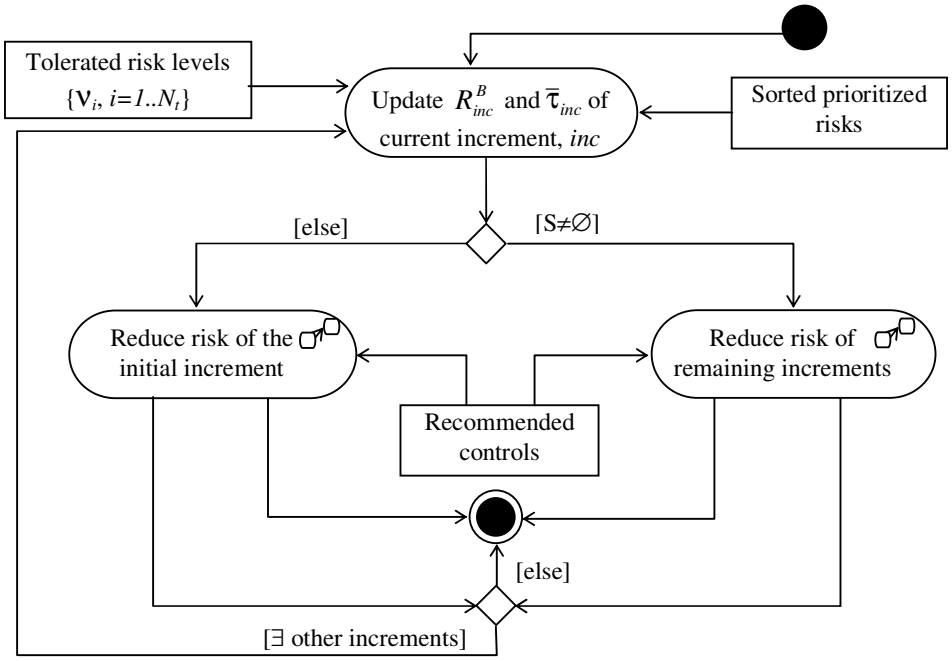


Figure 5.8: Increment risk treatment process

The most serious risk associated with highly critical asset or high damage cost or both is mitigated by the first increment of the proposed process. As illustrated by the diagram in figure 5.9, within this increment, a single iteration of the treatment program is performed with respect to determined tolerance level. When single or multiple control combinations are identified in this iteration, S_{inc}, as appropriate to treat the increment's risk, they are appended to the set of candidate security strategies, *S*. In the other case, the tolerance level associated with the initial increment, $\bar{\tau}_{inc}$, is relaxed by considering those of less prioritized risks. Another iteration of the treatment program is carried each time the tolerance level of the initial increment is revised. If current iteration identifies a solution then corresponding control combinations are included in S, as in the first alternative. However, when revised tolerance level reaches $\bar{\tau}$ and the corresponding iteration returns no solution, the treatment process is interrupted signaling that effective security strategies to the mitigation problem are unreachable with respect to current settings, recommended controls and imposed constraints. Different discussed activities of initial increment risk mitigation process are summarized in figure 5.9.

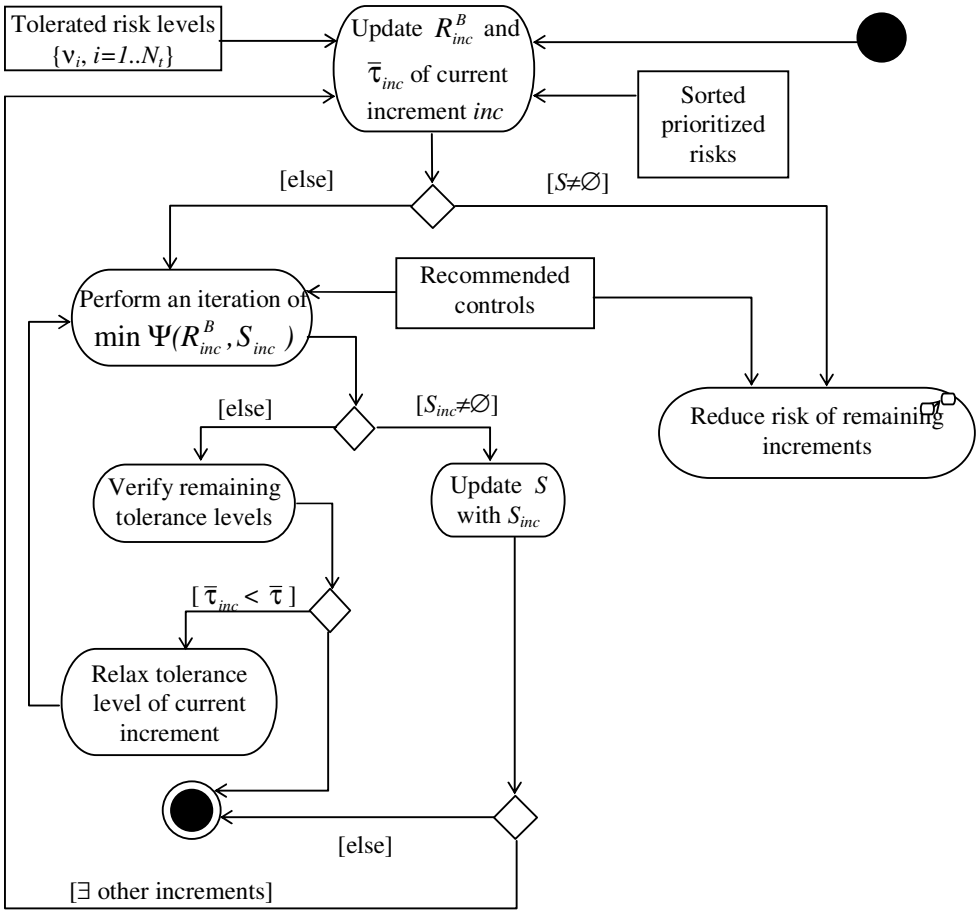


Figure 5.9: Initial increment risk treatment

Next increments of the treatment process concern risks associated with higher priority factors of those remaining. In each increment, efficacies of candidate control combinations in S are evaluated with consideration of the added increment's risk and revised tolerance level respectively R^B_{inc} and $\bar{\tau}_{inc}$. If all tested combinations of S effectively reduce cumulative risks, over all carried increments, to the fixed tolerance level, the current increment is interrupted and another one is initiated. Otherwise, combinations incapable to reduce cumulative risks, over treated increments including the current one, to the fixed tolerance level are assigned to S^N_{inc}, the set of ineffective candidate combinations. For each combination $CC_{inc,k} \subset S^N_{inc}$, a single iteration of the risk treatment program is performed with respect to the revised tolerance level, non treated risk and remaining security budget. Determined candidate combinations in S_{inc} by the current iteration and the considered ineffective candidate $CC_{inc,k}$ will produce new security strategies that appropriately mitigate reached risk. In the case of an empty set, S_{inc}, additional iterations of the treatment program are required after revising, each

time, the corresponding tolerance level with consideration of unmitigated risks. For these iterations, subsets of recommended controls are also revised depending on involved ineffective candidates.

Designed control combinations in the current increment, by boosting those previously ineffective, are reinserted in *S*. However, if no control combination is identified in this increment, then all ineffective candidates are eliminated from *S*. Depending on available candidate strategies in *S*, the decision to perform new increments or interrupt the mitigation process is taken. Main activities of the treatment process that concern mitigation of risks associated with remaining increments are depicted in figure 5.10.

In the case where appropriate control combinations are identified in S_{inc}, after performed iterations of the treatment process, they are involved in updating ineffective combination, $CC_{inc,k}$.. The updated control combination is then reinserted in the set of candidates *S* and its tolerance level is saved. However, if no update is available to boost effectiveness of $CC_{inc,k}$ and meet requirements of the current increment, then this candidate combination is eliminated from *S*.

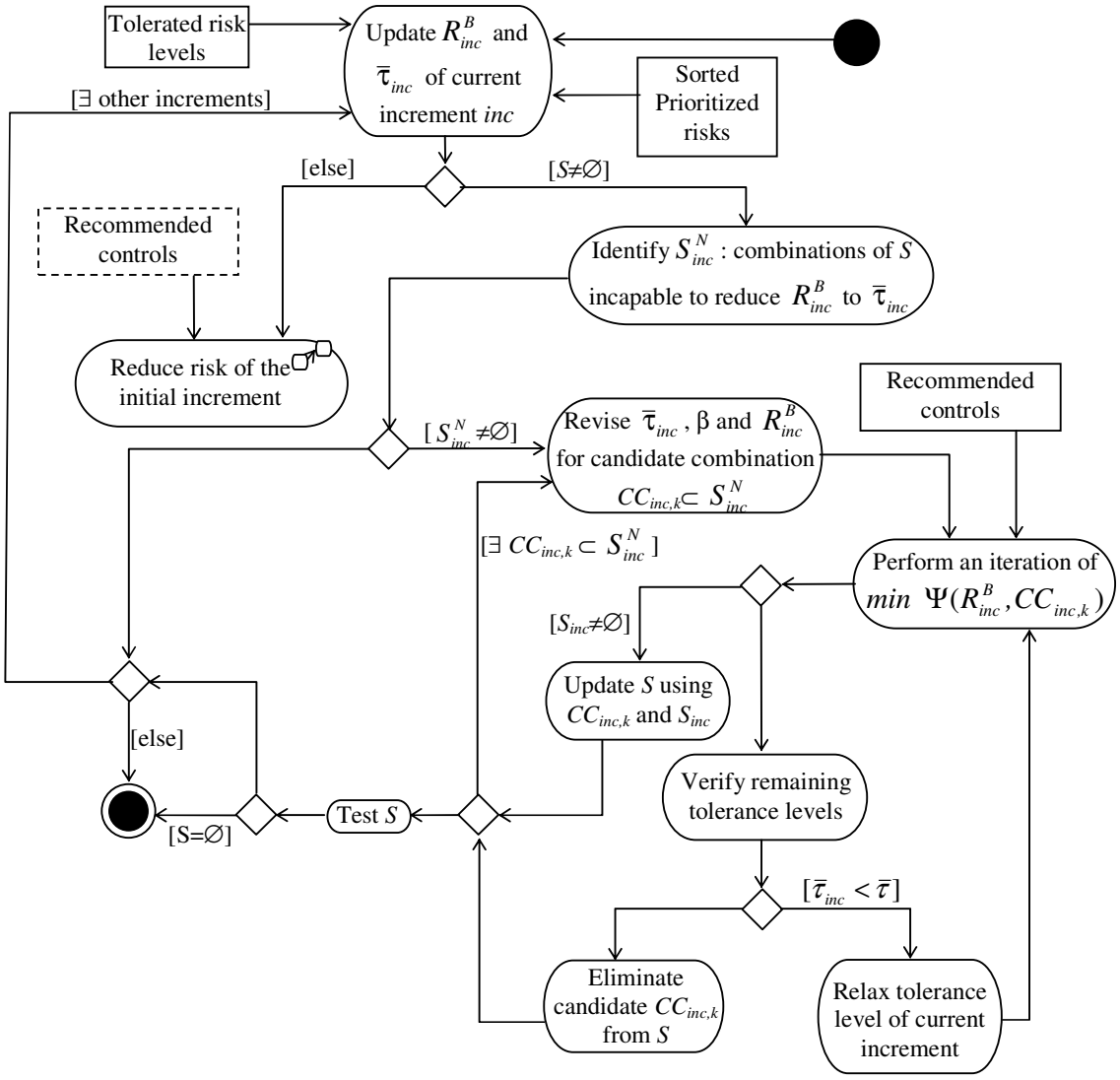


Figure 5.10: Remaining increments risks treatment process

After mitigation of prioritized risks of the computing environment, multiple control combinations, $SS_k \subset S$, are candidates to be deployed. To select the most appropriate control combination, residual risk of each candidate in S, R^R_k, is evaluated. Then, the trade-off between risk reduction and cost of each control combination SS_k is evaluated using the return on security investment ratio, $ROSI_k$, as follows:

$$ROSI_k = (R^B - R^R_k)/\omega_k \quad (5.29)$$

The return on security investment ratio, $ROSI_k$, assesses the global effectiveness of a candidate security strategy depending on its cost. Such ratio guides the selection of cost-effective security strategy for mitigating currently reached risk level. The most appropriate

security strategy of S, $SS^* \subset S$, ensures the highest return ratio. Different activities of the security strategy selection process are presented by the activity diagram in figure 5.11.

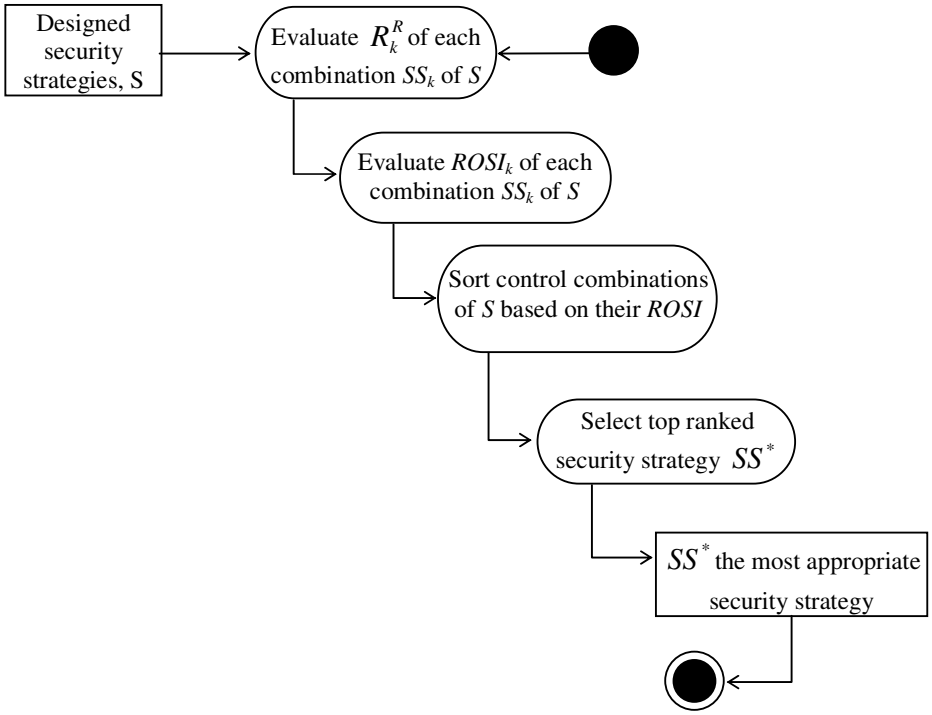


Figure 5.11: Security strategy selection process

The optimal control combination identified by the proposed risk mitigation process is then presented to the SSO. Additional information may be provided to the SSO including current security state report and deployment steps of included controls. Further improvements may be also added to designed response component to automatically deploy technical controls and assist the SSO in installing other managerial and operational countermeasures.

5.6 Conclusion

The designed risk model with respect to standards [149], [175] and [325], and recommendations of security guidelines, ISO-27001, was detailed in this chapter. Compared to existing generic models of ALE and NIST methodologies, it takes account of additional components, such as vulnerabilities and security controls components, in identifying parameters and assessing risks of the computing environment. Furthermore, it is structured into two parts as recommended by the ISO-27005. This is extremely useful and appropriately supports the risk driven response process of post-detection component in our idrs framework. The main process of the proposed response component consists of two sub-processes of risk

assessment and risk treatment. On one hand, risk assessment process identifies and determines considered risk elements and then estimates the current risk level of the computing environment. On the other hand, the risk treatment process aims at mitigating risks relying on its minimization program that focuses on risk cost. Selected security control combination through resolving associated minimization problem, in this step, reduces both of damage and security investment costs

Our proposed risk model includes different parameters neglected or partially considered by the majority of risk assessment methodologies but are extremely useful in this context such as the severity of vulnerabilities and the effectiveness controls. Moreover, it is well suited for multistage attacks that are merely treated by our model without any additional modification thanks to the vulnerabilities severity parameter. Furthermore, it is a generalization of existing information security risk models due to explicitly included components. Additionally, the single normalized quantitative model of ALE is also one of the special cases of the proposed model. However, the proposed risk model requires an extended knowledge base that concerns included risk parameters to ensure useful and realistic estimation of the current risk of an organization. Thus, enhancing idrs knowledge base using an incident database is recommended, for this model, because the latter combines all required information on attacks and possible countermeasures. But, the majority of existing organizations neither dispose of incident databases nor publish their security incidents. Even though incident databases are available for some organizations, they are trivially managed and maintained. However, as recommended by NIST, incident database is not only useful for the organization micro-environment but also for the macro-environment because it makes easier communication of security risks and participates in standardizing the means to combat them. Further promising extensions of designed risk driven response component may focus on the involvement of incident databases and the integration of created knowledge in different steps of the response process.

CONCLUSION AND FUTURE WORK

1 Conclusion

This thesis proposes an adaptive risk driven idrs framework. It mainly focuses on modeling adaptive detection and risk driven response components of the idrs framework respectively in chapters 4 and 5. Before these, the chapter 1 of this work introduces the foundations of information security domain. It mainly concerns different services and mechanisms of information security and attack taxonomies. Additionally, this chapter focuses on intrusion detection and response systems. It presents also their generic architecture, processing steps, classifications and normalization activities.

The chapter 2 reviews previous intrusion detection works and their proposed detection and response components. Surveyed analysis and detection components rely mainly on supervised or unsupervised learning techniques or both. Their design and implementation approaches and test results are also discussed in this chapter. Additionally, the latter revises the most widely known response taxonomy and its passive and active classes. Moreover, it summarizes several processes and cost factors of different subclasses of active response mechanisms. Main problems of reviewed analysis and response mechanisms are also discussed in this chapter.

Known the problems of existing mechanisms, an adaptive risk driven idrs framework was introduced in chapter 3. It aims at overcoming identified shortcomings and founding new requirements of future idrs generation. Critical requirements of the proposed framework including adaptive detection and risk based response are revised in this chapter. The latter presents also the extended architecture of the framework and discusses usefulness of the added model generation and knowledge base components. Moreover, it introduces the idrs life cycle and explains its processing steps. Main processes of the first and last life cycle steps that respectively concern model generation and knowledge base components are briefly illustrated in this chapter.

Additional steps of the life cycle concern analysis and response components of the idrs framework. The chapter 4 of this work focuses on the adaptive analysis and detection component. The latter takes account of several log types, some of them are reviewed in this chapter. Its process includes three main steps of selection, analysis and fusion instead of single one, as commonly encountered in existing components. The selection step starts by a

preliminary security analysis of current security state of the monitored system. Then, it involves an integrated criterion and various selection methods to identify best combinations of detection models. The integrated criterion and different steps of the selection process are presented in detail in this chapter. The analysis step formats next available log data examples and processes them using selected detection models. Afterward, outputs of involved models are fused using an evidential method to determine the combined decision on current security state. Analysis and fusion processes of the adaptive detection component and their steps are thoroughly studied in this chapter.

In this work, the chapter 5 focuses on the life cycle step that concerns real-time risk driven response component. The latter relies on the proposed risk management model with its two parts of assessment and treatment. The assessment part concentrates on the identification and determination of risk parameters. Then, it estimates identified parameters as well as the basic risk of the monitored system using appropriate processes. Identified risk parameters, their determined variables and assessment process are detailed in this chapter. The treatment part instead is specifically interested in mitigating assessed risks and reducing them to an acceptable level. The risk cost minimization program developed to this aim incrementally designs candidate combinations of security controls from those applicable. Afterwards, it determines the most appropriate combination based on ROSI criterion that takes account of both reduction effect and investment cost of designed security strategies. Different processes of the treatment part including security strategy construction and selection are also presented in detail in this chapter.

The last chapter of the thesis, chapter 6, thoroughly illustrates designed processes of the adaptive analysis and risk driven response components. Moreover, it gives a detailed comparison between the prototyped analysis mechanism and KDD winning strategy in terms of detection rates. All developed examples and conducted tests of the chapter are based on the DARPA 99 simulated environment and associated network traffic datasets, preprocessed in the KDD99. In this chapter, several log types and samples of vulnerabilities and security controls are considered. Additionally, different processing steps of the generation component are illustrated in order to build detection models and corresponding profiles. The discussed examples in this chapter concern specifically the detection and reaction against a DOS attack instance. Illustrative examples of multimodel analysis component take account of generated detection models and other inputs. They detail different computations performed respectively in selection and fusion processes of this component. Presented examples of the risk driven response component rely on several assumptions that mainly concern the target environment

and potential attacks. They illustrate different processing steps of the assessment part including risk parameters and basic risk estimation. Afterward, they discuss main settings of the adopted genetic technique to the treatment part as well the optimal security strategy determined by mitigation processes of the response component.

The proposed idrs framework in this work achieves initially fixed objectives of adaptive detection and risk driven response. To our knowledge, it is one of few works that focus on modeling complementarity between detection and response components and proposing required processes to implement this. Additionally, it is useful to deal with other intrusion detection and information security problems. Indeed, adaptive analysis and detection in the idrs framework reduces log processing cost relying on two means. On one hand, the preliminary security analysis using different checking steps identifies an extended number of system normal activities. In conducted experiment, more than 60% of normal instances are recognized at this stage of the analysis and detection process. On the other hand, dynamically selected combinations ensure that only appropriate subsets instead of all generated detection models are included in the current analysis task. Moreover, diverse log types considered by our idrs framework allow different traces of malicious and normal activities, and thus increase the chance of recognizing them even after slight changes. This is extremely useful for a host, network and application based ids. Adaptive analysis and detection is also appropriate for detecting and defending against distributed attacks. Additionally, it can be easily adapted to combine decisions of different passive or active ids and synchronize their reactions.

Risk driven reaction of the idrs framework introduces another class of active responses. It is relies on the developed quantitative risk model. The latter complies with risk standards, such as ISO-27005, FIPS-65 and NIST-39, and meets recommendations of security guidelines. It consists of two main parts of respectively risk assessment and treatment as recommended in the ISO-27005. The assessment part focuses on risk parameters identification and determination and basic risk estimation. It takes account of commonly considered parameters that concern target assets and mounted attacks. Additionally, it involves vulnerability and countermeasure dependent parameters that were implicitly included in previous risk models. Thus, it is perceived as a generalization of the ALE based assessment methodologies.

The treatment part solely focuses the mitigation option of risk reduction. It dynamically revises deployed security solutions. In addition, it takes advantage of optimization techniques to implement the mitigation option and hence reduce risk cost. The latter concerns both effectiveness, against realization of exploits, and deployment cost of selected security

controls. Therefore, best security strategies determined based on ROSI criterion in this part should satisfy acceptance levels of respectively residual risk and investment cost.

The proposed risk management model can be considered as a well thought out interpretation and implementation of the ISO-27005. It is suitable to design appropriate security strategies against detected attacks. Furthermore, it is extremely useful to deal with risk problems in different security projects including strategic security plans and information security management systems.

The idrs framework introduced in this work relies also on an expandable and well adapted architecture to the requirements of existing and future idrs generations. Moreover, the life cycle in the framework explicitly determines performed tasks of each participating component. The knowledge base (K-boxes) and detection model generation (P-boxes) components that extend classical idrs architecture are critical to the adaptive analysis as well cost effective reaction. The K-boxes save updatable detection profiles, supported flaws and recommended security controls respectively for log types, system assets and attack classes. They provide idrs components with required knowledge to their processes. Moreover, they may be enhanced by additional knowledge about detection environments, security best practices and guidelines in order to reinforce cooperation between idrs and other systems. P-boxes instead offer the possibility to generate new detection models and periodically update existing ones. Furthermore, they are involved in revising knowledge that concerns detection models in K-boxes. These new components introduced by the framework architecture and life cycle are critical for adaptive detection and cost effective response. Furthermore, they emphasize commonly neglected idrs requirements of detection model generation and security knowledge management.

The prototyped multimodel detection mechanism has shown promising initial results as illustrated by experiments of different groups of chapter 6. For combinations including two detection models, the prototyped detection engine outperforms KDD winner. As presented in table 6.32, it ensures a detection rate of 93.23%, instead of 92.71% for the KDD winner. It is also suitable to recognize DOS, U2R and R2L attack instances better than the KDD winner. Conducted experiments on combinations involving three detection models as well present good results of the prototyped detection mechanism. For instance, preliminary results in the second and third experiment group using combinations of three detection models show that the adaptive analysis mechanism is also capable to exceed 93% of detection. Different experiments of the adaptive analysis and detection mechanism prove its usefulness and illustrate its improvements specifically in recognizing DOS and rare attacks.

Detection results of carried experiments of the adaptive analysis mechanism, using combinations of two or three detection models, can be also enhanced if data problems are resolved. On one hand, data problems, as stated in [40], [57], [381], concern confusing data examples and low representation of U2R and R2L classes in training sets. The problem of low represented classes is specifically identified in experiments of the first group, as illustrated by reached detection results. Additionally, confusing instance problem is associated with experiments involving the initially designed train sets, as presented in tables 6.25 and 6.26. Indeed, these data sets are reduced comparatively the whole KDD train set. They include less than 10% of the original DARPA set. The prototyped multimodel detection mechanism using these training sets is also tested with an extended set of unknown attacks. Thus, it ensures low detection rates of normal and attack classes in these experiments.

On the other hand, data problems are related to identifying appropriate log data types that better trace given output class. In fact, normal or attack classes are better traced in some log types than others. For instance, detailed detection results by log type of the prototyped detection mechanism using combinations of two top ranked detection models selected relying on their accuracies of the second experiment group, are depicted in table 7.1.

Table 7.1: Detailed detection results of the KDD winner and prototyped detection engine using combinations of two detection models

Analysis and detection engines		Detection results by output class					Global detection results
		DOS	U2R	R2L	Probe	Normal	
KDD winner		97.12%	13.16%	8.40%	83.32%	99.45%	92.71%
Log type	intrinsic	97.24%	19.74%	10.19%	65.00%	93.27%	91.45%
	traffic	22.70%	3.51%	3.37%	2.47%	60.61%	28.79%
	content	0.00%	16.23%	7.76%	0.00%	60.62%	12.22%
Combined detection results		97.73%	18.42%	11.84%	65.55%	96.48%	92.53%

According to these results, selected detection model combination of the intrinsic log type outperforms those of other log types in detecting normal and attacks instances. It appropriately recognizes instances of these classes because they are better traced on intrinsic than on traffic or content log types.

At this level, selective combination rules could be deduced. They are useful to improve the fusion process and boost the detection performance of the adaptive analysis component. Furthermore, they appropriately traduce specificities of included log types and selection methods. An example of selective combination rule for discussed case is expressed as follows:

If (first level combined decision of intrinsic log is Normal) then (recombine intrinsic and content decisions only at the second fusion level)

Additional other rules are determined for experimented selection methods with respect to considered log types and output classes. Such rules will implement selective fusion in the proposed process. Moreover, they will reduce the set of considered log types at the combination step to two instead of all included log types. By applying deduced selective fusion rules, detection results of the multimodel detection component in table 7.1 are improved, as illustrated by table 7.2.

Table 7.2: Detailed detection results of the prototyped detection engine using two detection model combinations and selection rules

Performance criteria	Selection method	Testing results by class					Global testing result
		DOS	U2R	R2L	Probe	Normal	
ACC	Top ranked	97.60%	15.35%	11.77%	76.38%	97.36%	92.74%

Selective fusion rules have improved detection rates of normal and Probe attack classes. They have also boosted the global detection result of the adaptive analysis component in comparison to the KDD winner.

Tested prototype of the risk driven response component is capable to reach appropriate results. As illustrated in chapter 6, the prototyped risk mitigation component is able to design optimal security strategies that reduce assessed risk to an acceptable level and preserve security budget. However, experiments of the prototyped risk driven response mechanism lack a suitable test environment and detailed results of previous works. Therefore, the designed security strategies in these experiments can't be objectively compared to those of other works. Additionally, absence of clear and common evaluation criteria remains one of the major drawbacks in comparing the proposed risk driven response component to existing ones. Moreover, response mechanisms require thorough detection results to be more efficient, in that within the same class, different attacks of same type may need diverse corrective actions. Thus, detailed information about detected attacks appropriately guides corrective action selection and increases effectiveness of designed response strategy. At this level, a toolbox that offers to the SSO multiple techniques to resolve risk mitigation problem is extremely practical, specifically when dealing with the physical constraints of the monitored environment (limited resource computing environment).

Despite detailed experiments and thorough presentations of designed idrs components, this work has certain limitations. In fact, conducted experiments, in this work, focus on a reduced subset of detection techniques. Specific attention is also given to traffic log type in discussed illustrative examples. Additional other detection techniques and log types could be included in this study. As discussed in chapter 3, techniques including clustering, genetic algorithm, fuzzy expert system, chi-square, weighted moving average and evidential decision trees need thorough studies to be involved in processing steps of idrs components [273], [417], [418]. Moreover, host and application based log types, as stated in section §4.3, and others such as routers, firewalls and proxies logs may be considered by the idrs framework. Greater investigation effort is required to identify subsets of compatible log types that meet the requirement of diverse tracing of reported events, as imposed in the idrs framework. However, different aspects should be worked out and additional processes should be developed when dealing with complex log types, ids alerts for instance. Furthermore, conducted experiments should include detailed illustrative examples that concern different log types and their preprocessing steps in generating detection models as well analyzing the security state of the monitored system.

Although improved idrs architecture was proposed in this work, the added components, namely, detection model generation and knowledge base were briefly presented. Therefore, detailed designs of these components of the idrs framework are required. P-boxes processes including log data preprocessing and formatting, feature selection and detection model generation and updating for considered log types need to be thoroughly developed and tested with respect to considered detection techniques. Additionally, detailed definitions of different steps of detection profiles generation and revision processes should be done for P-boxes. K-boxes instead require well designed knowledge management processes as well an explicit revision policy. Moreover, an appropriate representation of detection, response and domain knowledge is also an essential requirement of this component.

Additionally, the adaptive analysis and risk driven response components of the idrs framework may be enhanced through resolved limitations of their designed processes. In the adaptive detection component, selection, analysis and fusion processes require additional revisions in order to reinforce their decisions. The selection process of this component includes initially three checking steps. First two steps focus on system normal behavior, while the last one concerns different attack classes. Decisions of first steps about currently processed security state are crucial to the whole idrs process. They enormously reduce processing cost of the idrs if normal security states are correctly recognized. However, they

may cause damage to the monitored system when normal and intrusive states are confused. Therefore, these checking steps need to be reinforced by others to thoroughly control log data and ensure precise decisions. Additionally, the last checking step in the selection process, which focuses on relevant feature subsets of different attack classes, requires appropriate techniques to determine reference vectors and control intervals independently to log data problems. Indeed, the sample mean technique adopted in this work is sensitive to outliers. Detailed experiments should be also conducted to test different revisions of checking steps and validate their reached results.

Remaining steps of the selection process focus respectively on the evaluation and ranking of candidate detection models and identification of best combinations of these. They are based on scores of candidate detection models. These scores are expressed using data and model dependent factors. However, additional factors may be useful to determine scores, depending on the target environment. For instance, computation and storage loads are relevant to detection model scores, specifically when dealing with adaptive analysis in embedded systems. Ranked subsets of detection models based on evaluated scores serve next to identify best combinations using different methods. Selected combinations in this work include a fixed number of detection models. But, variable size combinations may be also appropriate to adaptive analysis and detection mechanisms. Despite the partial tests using the coverage constraint, detailed experimentations are required to objectively compare variable and fixed size combinations in adaptive intrusion detection. Another alternative to implement the selection process that focuses on filtering and allows variable size combinations may be also discussed and tested at this level. Moreover, a detailed comparison between filtering and score based selection is extremely useful particularly in the case of extended subsets of detection models.

In the analysis step of the detection process, outputs of selected detection models are assumed at the abstract level. However, multiple detection models built using diverse artificial intelligence and data mining techniques are considered by the analysis and detection component. Additionally, they allow different outputs. Therefore, in the analysis step, processes are required to unify representation of selected detection models outputs. They transform and format different outputs of selected detection models as required by the fusion step of the detection process. At this level, different output types can be considered and appropriate transformation processes to required representation by the fusion step can be developed.

The last step of the detection process, namely detection models decisions fusion, uses Denœux distance based approach to evaluate belief of detection models and combine these using Smets's conjunctive rule. At this level, several extensions to the fusion process, not included or experimented in this work, may be implemented. Initially, the belief evaluation process proposed in this work requires additional other to appropriately estimate involved parameters including normalization factors and reliability coefficients. Although, these parameters are determined as recommended in [109], [431], adequate estimation processes that take account of revised performances and updated learning sets of detection models are also very interesting to the designed detection component. Afterward, the proposed process to update learning sets of generated detection models was not included in conducted experiments of chapter 6. Therefore, detailed tests of this process are useful to all steps of the multimodel detection process. They serve to evaluate and compare reached results using updated and static learning sets. Furthermore, belief evaluation in the proposed fusion process uses the distance based approach and assumes abstract outputs for all detection models. However, different detection models may allow probabilistic, fuzzy or other output types. Thus other belief evaluation approaches are applicable in this step unless output unifying processes discussed above are considered. In this context, experimentation of other evidential combination methods is also extremely useful to our fusion process. In fact, conducted experiments involving several evidential fusion rules will illustrate differences and similarities between them in managing conflicts, and hence identify the most appropriate one to the intrusion detection field. Additional other fusion methods including fuzzy, probabilistic and meta-learning should be included to the idrs framework. They yield diverse adaptive analysis and detection mechanisms. They are useful to conduct a thorough comparison of capabilities of these mechanisms.

Risk driven response component designed in this work supports also other insufficiencies that need to be addressed to determine well adapted post detection reactions and appropriately meet security requirements of the monitored computing environment. The proposed risk model for this component includes several difficult to estimate parameters. Therefore, external entities such as domain and security experts are actively involved in determining these risk parameters. For instance, required security services, their importance factors and criticalities of assets are parameters determined depending upon organization's mission, information owner decisions and domain and security experts knowledge. Additional other parameters including severity coefficients of vulnerability classes and effectiveness of security controls are also estimated by security experts relying on available public or

proprietary databases. At this level, another tricky problem linked to unpublished vulnerabilities and how treating these within the proposed risk model arise. It puts emphasis on the difficulties of assessing risks of the computing environment supporting private flaws known that impact factors, severity scores and possible remediation controls are not available for these. Easing such problem requires further extensions that concern not only proposed risk model but also CVSS system and NVDB database. On one hand, an extension module should be integrated to CVSS and NVDB in order to assist the SSO in updating and revising database of inherited flaws of the monitored system. On the other hand, required processing steps of private vulnerabilities should be considered to determine exposure, flaws severity and controls effectiveness parameters of the risk model.

The designed risk driven response component may take advantages of attacking scenarios, incident databases and other optimization techniques to improve risk assessment and treatment processes. Attacking scenarios are determined relying on vulnerability graphs. They concern different graph paths that represent potential sequences of exploits to reach attackers objectives. Attacking scenarios are extremely useful to damage assessment in the proposed risk model. They may ensure precise and more realistic estimations of incurred damages by target assets due to mounted attacks. Besides, they may be very interesting to the multimodel detection component because they offer required means to check and predict implemented actions by attackers.

Incident databases are also useful for assessment and mitigation processes of the proposed risk model. They provide required information about detected intrusions and exploited vulnerabilities in order to conduct thorough risk assessment and derive precise estimation of inflicted damages to assets of the monitored system. Moreover, they allow detailed reports on deployed controls and their appropriateness in defending against detected threats. Thus, it is very advantageous to include them in identifying and revising applicable control sets of the risk mitigation program. Furthermore, the latter was solely based on genetic algorithms. Additional other optimization techniques including tabu search and simulated annealing can be applied in resolving risk cost program. Multiple criteria may be also considered by the risk cost minimization problem. Safeguards applicability, incompatibility and complementarity are among criteria to be included in identifying control combinations and selecting the most appropriate of these to reduce assessed risks of the monitored system to an acceptable level. At this level, a detailed study of the risk mitigation program and adopted technique is required because illustrated example is limited to the simple case of a system asset target by a DOS attack. Illustrative examples of complex cases including several threats targeting multiple

assets are extremely useful for elucidating how designed processes perform. Additionally, a thorough study of designed mitigation and adaptive analysis and detection processes performances, computation time and complexity, is needed to evaluate the corresponding components as well adapt these to their deployment environments.

2 Future work

This work has presented and detailed different solutions to some of challenging problems of existing and future idrs generations. Although proposed solutions have been focused both on architecture and operating steps of idrs systems, they require additional extensions to better meet requirements of new idrs generation. Initial enhancements are interested in expanding the proposed idrs framework to include other detection techniques and fusion methods, as discussed above. Further revisions of designed processes are also required due to appended techniques. Moreover, potential improvements of this work concern three main axes that address to analysis and detection component, response component and synchronization between these. Analysis and detection component enhancements specifically concentrate on unknown attack detection. Response component potential extensions focus on designed risk model and proposed mitigation program. Additional improvements of both detection and response components concern attack stages prediction and proactive reaction.

In this work, the prototyped multimodel analysis and detection component is able to recognize a reduced subset of unknown attack instances relying on selected detection models. Detection models, in turn, fail to correctly detect unseen attacks instances. Although, they assign true labels to some of these instances, their assessed beliefs are nearly null. Thus, their false positive rates are increased, and hence the combined detection model. Such situation is commonly encountered due to two main reasons. On one hand, treated instances of attack types are not included in the DARPA training set and learning sets of selected detection models. On the other hand, the designed analysis and detection component assumes the closed world hypothesis that the frame of discernment is exhaustive and covers all output labels of treated data instances. This assumption may be confirmed when dealing with low granularity level such that in the proposed detection component that focuses on five output labels. However, in the case of increased granularity, the assumption is invalid because the frame of discernment does not cover all attack labels, except those involved in generating detection models. Therefore, the open world assumption may be useful to cope with the unknown attack detection problem in our idrs framework. It supposes that some labels of treated data instances are covered by hypotheses of the frame of discernment but not others. Uncovered labels

concern unknown attack instances and may be represented by a specific hypothesis of the extended frame as discussed in [360]. Additionally, the open world hypothesis with increased attack class granularity should be considered at different steps of the intrusion detection process including detection model generation and decision fusion. Moreover, detailed experiments are required to compare results of the adaptive analysis and detection mechanism under open and closed world hypotheses.

The risk driven response component in this work may be also extended depending on new requirements of idrs systems and their environments. These extensions mainly concern loss estimation, cooperation reinforcement and real-time response. In the proposed risk model, loss estimation may be improved if additional information about involved parameters and historical database on mounted attacks is available. Incident databases and vulnerability graphs are extremely useful to the designed risk driven response component. Incident databases allow required historical data on detected attacks and implemented responses. They ensure precise estimations of attacks likelihoods based on available prior knowledge. Furthermore, they are appropriate to dynamically design and assess security strategies relying on history of implemented response. Vulnerability graphs are also useful to our risk driven response component. They determine, based on supported flaws, sequences of potential exploits to reach attackers' objectives. These sequences are required to better estimate victim impact, flaws severity and controls effectiveness parameters. In fact, the identification of the most probable path followed by an attacker to reach a fixed objective allows better estimations of victim impacts because the target flaws subset is involved, instead of the whole set. In addition, other paths of the graph determine potential threats to the victim due to currently detected attack, and thus they are useful to objectively estimate vulnerabilities severity factor of the risk model. Furthermore, vulnerability subsets of attacking scenarios are also required to conduct a detailed assessment of deployed security control efficacy.

Reinforcing cooperation between idrs and other management systems is also a promising future research direction in this field. It merely aims at enhancing the security of the monitored computing environment through cooperation and interaction between different systems. For instance, asset management system (AMS) that keeps history of existing assets is very useful to the risk driven response component. It may provide this component with up-to-date information to ensure realistic estimates of asset values and the exposure of the monitored system. Moreover, it is extremely useful to identify appropriate deployment locations of security controls. Information security management system (ISMS) instead assists the SSO in the deployment of selected controls. Before that, it can be involved in

synchronizing between risk driven response mechanism and other security systems in order to design a defense strategy that reinforces the global security of the monitored computing environment. AMS and ISMS are also required in updating knowledge respectively about supported vulnerabilities and security controls regarding the security policies, best practices, guidelines, standards and regulations.

The risk treatment program proposed in this work also needs greater improvement effort to meet security and performance requirements of increasingly evolving computing environments. Indeed, in grid, cloud and high performance computing environments, the implemented version of the risk mitigation program may be ineffective due to new constraints. Thus, an adapted implementation of this program that does not affect performance and resource availability in these environments is required. Additionally, it should ensure real-time response against mounted attacks, as imposed by the idrs life cycle. A system model that presents resources of the computing environment and dependencies between these may be also considered by this version of the mitigation program. It serves to illustrate effects of designed and deployed security strategies on system resources. Moreover, additional constraints that concern controls, assets and the target computing environment may be involved in the improved version of the mitigation program.

In this work, event sequence of an attack is considered by the detection mechanism as a single block that allows an intruder to reach a fixed objective. Moreover, defense actions against these intrusive events are implemented all together, just after the detection moment. Such detection and response strategies are commonly adopted. As such, a promising idea that focuses on attack stages detection and proactive responses may considerably enhances capabilities of designed detection and response components as well idrs systems of future generations.

Indeed, intrusive actions of an attacker may be structured into different stages that characterize progress levels of the mounted attack. Each of these stages is also determined by different states identified with the assistance of security experts and depending upon considered log types and selected detection techniques. Detection models are initially trained to recognize these states. Then, the analysis mechanism selects detection models to predict states and consequently attack stage. It next forwards information about detected attack stage, its potential type and progress level to the response component. The latter evaluates risk level of the monitored system due to detected attack attempt with respect to the determined progress level. Afterward, it decides whether to implement corrective actions or wait for additional pieces of information regarding the assessed risk level and progress level. In the

former case, it designs and deploys the most appropriate security strategy to defend against detected attempt or attack. However, in the latter case, it waits for additional information about true type of the predicted attack stage and its progress level.

Even though attack stages and proactive reactions are very interesting extensions to analysis and response components, they lead to several designing problems. On one hand, many issues that concern the analysis and detection component including preprocessing techniques, state representation and attack stage prediction should be addressed. On the other hand, response component problems that mainly focus on revision of the determined risk level and implementation decision should be resolved. Additional difficulties are also encountered in generating detection models, however, attack stage prediction and proactive response are very promising research topics and merit further thorough explorations.

REFERENCES

[1] J. Adeva, U.C. Beresi and R.C. Calvo, "Accuracy and Diversity in Ensembles of Text Categorisers," *Latin-amer. Center for Informatics Studies (CLEI) Electron. J.*, vol. 8, no. 2, pp. 1-12, Dec. 2005.

[2] M. Aghamohammadi and M. Analoui, "A Comparison of Support Vector Machine and Multi-level Support Vector Machine on Intrusion Detection," *World of Comput. Sci. and Inform. Technol. J.*, vol. 2, no. 7, pp. 215-219, 2012.

[3] R. Agrawal, T. Imielinski and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," *Proc. of the ACM SIGMOD Conf. on Management of Data*, Washington, D.C., 1993, pp. 207-216.

[4] J. Aguillar, "Adapted Fusion Scheme for Multimodal Biometric Authentication," Ph.D. dissertation, Dept. Signal, Syst. and Radio Commun., Univ. Polytechnic Madrid, Madrid, Spain, 2006.

[5] I. Ahmad, A. B. Abdullah and A. S. Alghamdi, "Application of Artificial Neural Network in Detection of DOS Attacks," *2nd Inter. Conf. on Security of Inform. and Networks*, North Cyprus, 2009, pp. 229-234.

[6] I. Ahmad, A. B. Abdullah and A. S. Alghamdi, "Artificial Neural Network Approaches to Intrusion Detection: A Review," *Proc. of the 8th WSEAS Inter. Conf. on Telecommun. and Informatics*, Istanbul, Turky, 2009, pp. 200-205.

[7] I. Ahmad, A. B. Abdullah and A. S. Alghamdi, "Towards the Selection of best Neural Network System for Intrusion Detection," *Inter. J. of the Physical Sci.*, vol. 5, no. 12, pp. 1830-1839, Oct. 2010.

[8] M. Ahmad, "Intrusion Prevention System Based on Secure Mobile Agents," M.S. thesis, Dept. of Comput. and Syst. Sci., Kungl Tekniska Högskolan Univ., Stockholm, Sweden, 2006.

[9] M. Ahmed, E. Al-Shaer and L. Khan, "A Novel Quantitative Approach For Measuring Network Security," *23th IEEE Inter. Conf. on Comput. Commun.*, Phoenix, AZ, USA, 2008, pp. 1957-1965.

[10] M. Ahmed, E. Al-Shaer and L. Khan, M.M. Taibah and M. Abedin, "Dynamic Risk Measurement and Mitigation for Proactive Security Configuration Management," *Proc. of IEEE Network Operations and Management Symp.*, Salvador, Bahia, 2008, pp. 722-725.

[11] S. Akbar, K. Nageswara Rao and J. A. Chandulal, "Intrusion Detection System Methodologies Based on Data Analysis," *Inter. J. of Compu. Appl.*, vol. 5, no. 2, pp. 10-20, Aug. 2010.

[12] M. Aksela and J. Laaksonen, "Using Diversity of Errors for Selecting Members of a Committee Classifier," *Pattern Recognition*, vol. 39, no. 4, pp. 608-23, 2006.

[13] M. Aksela, "Comparison of Classifier Selection Methods for Improving Committee Performance," *Proc. of Multiple Classifier Syst. Conf.*, Guildford, UK, 2003, pp. 84-93.

[14] A. Al-Ani and M. Deriche, "A New Technique for Combining Multiple Classifiers using The Dempster-Shafer Theory of Evidence," *J. of Artificial Intell. Research*, vol. 17, no. 1, pp. 333-361, 2002.

[15] V. Alarcon-Aquino, C. A. Oropeza-Clavel, J. Rodriguez-Asomoza, O. Starostenko and R. Rosas-Romero, "Intrusion Detection and Classification of Attacks in High-Level Network Protocols Using Recurrent Neural Networks," *Int. Joint Conf. on Comput., Inform. and Sys. Sci. and Eng.*, 2009, pp. 129-134.

[16] M. Albalate, "Data Reduction Techniques in Classification Processes," Ph.D. dissertation, Dept. of Languages and Comput. Syst., Univ. of Jaume 1, Spain, 2007.

[17] C. Alberts and A. Dorofee, *Managing Information Security Risks: The OCTAVE^{SM} Approach*, 1st ed., Addison Wesley, 2002.

[18] D. Alessandri, C. Cachin, M. Dacier, K. Julisch, Brian Randel, J. Riordan, A. Tscharner, A. Wespi and C. Wüest, "Towards a Taxonomy of Intrusion Detection Systems and Attacks, Malicious- and Accidental-Fault Tolerance for Internet Applications," MAFTIA project, Rep. D3, Sep. 2001.

[19] K. Ali, "Algorizmi: A Configurable Virtual Testbed to Generate Datasets for Offline Evaluation of Intrusion Detection Systems," M.S. thesis, School of Comput. Sci., Univ. of Waterloo, Ontario, Canada, 2009.

[20] J. Allen, *The CERT Guide to System and Network Security Practices*, 1st ed., Addison Wiley Professional, 2001, ch1, pp.1-18.

[21] X. An, D. Jutla and N. Cercone, "Privacy Intrusion Detection Using Dynamic Bayesian Networks," *Proc. Int. Conf. on Electron. Commerce*, Fredericton, Canada, 2006, pp. 208-215.

[22] ANSI Tunisie, Décret: Décret n° 1250 - 2004 du 25 mai 2004, L'obligation d'audit de sécurité des systèmes d'information en Tunisie. Available: www.ansi.tn/fr/audit/lois_audit.html, last accessed Sep. 2012.

[23] N. Anuar, M. Papadaki, S. Furnell and N. Clarke, "An Investigation and Survey of Response Options for Intrusion Response Systems (IRSs)," *Conf. on Inform. security for South Africa*, Johannesburg, S.A., 2010, pp. 1-8.

[24] D. Ariu and G. Giacinto, "HMMPayl: an Application of HMM to the Analysis of the HTTP Payload," *Workshop on Appl. of Pattern Analysis*, Cumberland Lodge, UK, 2010, pp. 81-87.

[25] D. Ariu, "Host and Network based Anomaly Detectors for HTTP Attacks," Ph.D. dissertation, Dept. of Elect. and Electron. Eng., Univ. of Cagliari, Cagliari, Italy, 2010.

[26] A. Arora, A. Nandkumar and R. Telang, "Does Information Security Attack Frequency Increase with Vulnerability Disclosure? An Empirical Analysis," *Inform. Syst. Frontier*, vol. 8, no. 5, pp. 350-362, 2006.

[27] T. Aslam, "A Taxonomy of Security Faults in the UNIX Operating System," M.S. thesis, Dept. of Comput. Sci., Purdue Univ., West Lafayette, IN, 1995.

[28] Y. Aslandogan, G. A. Mahajani and S. Taylor, "Evidence Combination in Medical Data Mining," *IEEE Int. Conf. on Inform. Technol. Coding and Computing*, LasVegas, NV, 2004, pp. 465-469.

[29] E. Awad and H. M. Ghaziri, *Knowledge Management*, 2ed ed., Prentice Hall, 2004.

[30] S. Axelsson, "Intrusion Detection Systems: A Survey and Taxonomy," Chalmers Univ. of Technol., Goteborg, Sweden, Rep. T.R. 99-15, Mar. 2000.

[31] J. Baayer and B. Regragui, "New Cost-Sensitive Model for Intrusion Response Systems Minimizing False Positive,", *Int. J. of Modern Eng. Res.*, vol. 2, no. 5, pp 3473-3478, 2012.

[32] S. Balamurugan and R. Rajaram, "Effective and Efficient Feature Selection for Large-scale Data Using Bayes'Theorem," *Int. J. of Automation and Computing*, vol. 6, no. 1, pp. 62-71, Feb. 2009.

[33] J. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, E.H. Spafford and D. Zamboni, "An Architecture for Intrusion Detection using Autonomous Agents," Purdue Univ., West Lafayette, IN, Rep. COAST TR 98-05, 1998.

[34] P. Baldi and S. Brunak, *Bioinformatics: The Machine Learning Approach*, 2ed ed., Bradford Books, 2001.

[35] I. Balepin, S. Maltsev, R. Rowe and K. Levitt, "Using Specification-based Intrusion Detection for Automated Response," *Proc. of the Int. Symp. on Research in Attacks, Intrusion and Defense*, Pittsburgh, PA, USA, 2003, pp. 136-154.

[36] L. Baum and T. Petrie, "Statistical Inference for Probabilistic Functions of Finite State Markov Chains," The *Ann. of Math. Statist.*, vol. 37, no. 1, pp. 164-171, 1966.

[37] D. Baumgartner, "Global-Local Hybrid Classification Ensembles: Robust Performance with a Reduced Complexity," M.S. thesis, College of Eng., Univ. of Toledo, Ohio, USA, 2009.

[38] G. Beauchemin and G. Dansereau, "Harmonized Threat and Risk Assessment Methodology, Communications Security Establishment," Can. government's Commun. Security Establishment, Canada, Rep. T.R.A-1, Oct. 2007.

[39] C. Bellefeuille, "Quantifying and Managing the Risk of Information Security Breaches to the Supply Chain," M.S. thesis, Dartmouth College, MIT, Cambridge, MA, 2005.

[40] N. Ben Amor, S. Benferhat and Z. Elouedi, "Naive Bayes vs Decision Trees in Intrusion Detection Systems," *Symp. of Appl. Computing*, Nicosia, Cyprus, 2004, pp. 420-424.

[41] N. Ben Amor, S. Benferhat and Z. Elouedi, "Réseaux Bayésiens Naïfs et Arbres de Décision dans les Systèmes de Détection d'intrusions," *Conf. sur la Reconnaissance de Forme et l'Intell. Artificielle*, Toulouse, France, 2004, pp. 1175-1184.

[42] S. Benfarhat and K. Tabia, "Integrating Anomaly-Based Approach into Bayesian Network Classifiers," *Int. Conf. on e-Business and Telecommun.*, Porto, Portugal, 2008, pp. 127-139.

[43] S. Benfarhat, A. Boudjelid and H. Drias, "An Intrusion Detection Approach Based on Tree Augmented Naive Bayes and Expert Knowledge," *1ière Doctorialies STIC 09*, Alger, 2009.

[44] J. Bezdek and L. Kuncheva, "Nearest Prototype Classifier Designs: An experimental Study," *Int. J. of Intell. Syst.*, vol. 16, no. 12, pp. 1445-73, 2001.

[45] K. Bharti, S. Shukla and S. Jain, "Intrusion Detection using Clustering," *Special Issues of Int. J. of Compu. and Commun. Technol.*, vol. 1, no. 2, 3, 4, pp. 158-165, Aug. 2010.

[46] S. Bian and W. Wang, "On Diversity and Accuracy of Homogeneous and Heterogeneous Ensembles," *Int. J. of Hybrid Intell. Syst.*, vol. 4, no. 2, pp.103-128, 2007.

[47] C. Bishop, *Pattern Recognition and Machine Learning*, 1st ed., Springer, 2006.

[48] M. Bishop, "A Standard Audit Format," *18th Nat. Inform. Syst. Security Conf.*, Baltimore, Maryland, 1995, pp. 136-145.

[49] M. Bishop, "A Taxonomy of UNIX System and Network Vulnerabilities," Univ. of California, Davis, USA, Rep. TR-CSE-95-10, May 1995.

[50] A. Bivens, C. Palagiri, R. Smith, B. Szymanski, and M. Embrechts, "Network-Based Intrusion Detection using Neural Networks," *Proc. Intelli. Eng. Syst. through Artificial Neural Networks*, Missouri, 2002, pp. 579-584.

[51] I. Bloch, "Fusion d'Informations Numériques : Panorama Méthodologique," *Journées Nat. de la Recherche en Robotique*, Guidel, France, 2005, pp. 79-88.

[52] B. Boser, I. Guyon and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," *5th Annu. Workshop on Computational Learning Theory*, San Mateo, CA, 1992, pp. 144-152.

[53] Y. Bouzida and R. Cuppens, "Neural Networks vs. Decision Trees for Intrusion Detection," *IEEE 1st Workshop on monitoring, Attack Detection and Mitigation*, Tuebingen, Germany, Sep. 2006.

[54] P. Brazdil and C. Soares, "A Comparison of Ranking Methods for Classification Algorithm Selection," *Proc. of the European Conf. on Mach. Learning*, Barcelona, Spain, 2000, pp. 63-74.

[55] J. Broder, *Risk Analysis and Security Survey*, 3rd ed., city, Elsevier Sci. and Technol., 2006.

[56] G. Brown, J. Wyatt, R. Harris and X. Yao, "Diversity Creation Methods: A Survey and Categorization," *Int. J. of Inform. Fusion*, vol. 6, no. 1, pp. 5-20, 2005.

[57] A. Bsila, S. Gombault and A. Belghith, "Improving Traffic Transformation Function to Detect Novel Attacks," *Proc. of the 4th Int. Conf. on Sci. of Electronics, Technologies of Inform. and Telecommun.*, Hammamet, Tunisia, Mar. 2007.

[58] T. Buchheim, M. Erlinger, B. Feinstein, G. Matthews, R. Pollock, J. Betser and A. Walther, "Implementing the Intrusion Detection Exchange Protocol," *17th Annu. Comput. Security Appl. Conf.*, Louisiana, 2001, pp. 1-32.

[59] Bugtrack database. Available: seclists.org/bugtrack/, last accessed Mar. 2010.

[60] K. Burbeck and N.T. Simin, "ADWICE- Anomaly Detection With fast Incremental Clustering," *7th Int. Conf. on Security and Cryptology*, Seoul, Korea, 2005, pp. 407-424.

[61] C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Microsoft Research, Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121-167, Jan. 1998.

[62] J. Cannady, "Applying CMAC-Based On-line Learning to Intrusion Detection," *Proc. of the 2000 IEEE Int. Joint Conf. on Neural Networks*, Como, Italy, July 2000, pp. 405-410.

[63] J. Cannady, "The Application of Artificial Neural Networks to Misuse Detection: Initial Results," *Proc. of the Int. Symp. on Research in Attacks, Intrusions and Defense*, Louvain la Neuve, Belgium, 1998, pp. 31-47.

[64] J. Cannady, B. C. Rhodes and S.A. Mahaffey, "Multiple Self Organizing Maps for Intrusion Detection," *Proc. of 23rd Nat. Inform. Syst. Security Conf.*, Baltimore, MD, 2000, pp. 16-19.

[65] R. Caruana and A. Niculescu-Mizil, "An Empirical Evaluation of Supervised Learning for ROC Area," *1st Int. Workshop on ROC Analysis in Artificial Intell.*, Valencia, Spain, 2004, pp. 1-8.

[66] R. Caruana and A. Niculescu-Mizil, "Data Mining in Metric Space: an Empirical Analysis of Supervised Learning Performance Criteria," *Proc. of the 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Seattle, Washington, USA, 2004, pp. 69-78.

[67] C. Carver and W.P. Pooch, "An Intrusion Response Taxonomy and its Role in Automatic Intrusion Response," *Proc. IEEE Workshop on Inform. Assurance and Security*, United States Military Academy, West Point, NY, 2000.

[68] F. Cate, "Information Security Breaches and the Threat to Consumers," the Center for Inform. Policy Leadership, Hunton & Williams LLP, USA, Sep. 2005.

[69] CERT. (2004). CERT/Coordination Center: Overview of Attack Trends. Pittsburgh, USA. [online] Available: www.cert.org/archive/pdf/attack_trends.pdf, last accessed Mar. 2007.

[70] CERT. (2006). CERT/CC Statistics 1988-2005. [online] Available: http://www.cert.org/stats/cert_stats.html, last accessed Mar. 2007.

[71] I. Chairunnisa, I., Lukas, C. and H.D. Widiputra, "Clustering based Intrusion Detection for Network Profiling using K-means, ECM and K-nearest Neighbour Algorithms," *Konferensi Nasional Sistem dan Informatika*, Bali, Indonesia, 2009.

[72] V. Chandola, A. Banerjee and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1-58, Sep. 2009.

[73] Y. Chen and Y. M. Chen, "Combining Incremental Hidden Markov Model and Adaboost Algorithm for Anomaly Intrusion Detection," the *ACM SIGKDD Workshop on Cyber Security and Intell. Informatics*, Paris, France, 2009, pp.3-9.

[74] Y. Chen, B. Boehm and L. Sheppard, "Value Driven Security Threat Modelling Based on Attack Path Analysis," *40th Hawaii Int. Conf. on Syst. Sci.*, Big Island, Hawaii, 2007.

[75] Y. Chen, K. Diao, I. Orci and M. Astrom, "Normalizing Security Audit Log in XML format," Dept. of Business Admin. and Social Sci., Division of Syst. Sci., Sweden, Rep. TR-2004-13/SE, 2004.

[76] T. Chiang, J. S. Kouh and R. I. Chang, "Ontology-based Risk Control for the Incident Management," *Int. J. of Comput. Sci. and Network Security*, vol. 9, no. 11, pp. 181-9, Nov. 2009.

[77] M. Choras, R. Kozik, A. Flizikowski and R. Renk, "Ontology-Based Decision Support for Security Management in Heterogeneous Networks," *Int. Conf. in Intell. Computing*, South Korea, 2009, pp. 920-927.

[78] T. Chou, K. K. Yen and J. Luo, "Network Intrusion Detection Design Using Feature Selection of Soft Computing Paradigms," *Int. J. of Inform. and Mat. Sci.*, vol. 4, no. 3, pp. 196-208, 2008.

[79] P. Cisar and S. M. Cisar, "Model based Algorithm for Statistical Intrusion Detection," *10th Int. Symp. of Hungarian Researchers on Computational Intell. and Informatics*, Hunguria, 2009, pp. 625-631.

[80] Code of Practice for Information Security Management, ISO, Information Technology, Security Techniques, BS ISO-IEC 27002:2005, Jul. 2007.

[81] F. Cohen, *Protection and Security on the Information Superhighway*, 1st ed., New York, John Wiley and Sons, 1995.

[82] *Complete Guide to the Common Vulnerability Scoring System Version 2.0,* NIST, 2007.

[83] Computer Economics, "The 2005 Malware Report: Executive Summary," Computer Economics, Irvine, CA, Jan. 2005.

[84] D. Corsar and D. Sleeman, "Reusing JessTab rules in Protégé," *Knowledge-Based Syst.*, vol. 19, no. 5, pp. 291-297, 2006.

[85] CRAMM user's guide version 2.0, Central Computer Telecommunication Agency, UK, 1991.

[86] N. Cuppens-Boulahia and F. Cuppens, "Specifying Intrusion Detection and Reaction Policies: An Application of Deontic Logic," *9th Int. Conf. on Deontic Logic in Comput. Sci.*, Luxembourg, 2008, pp. 65-80.

[87] CVE, Common Vulnerability and Exposure. Available: cve.mitre.org, last accessed Mar. 2010.

[88] CVSS, Forum of Incident Response and Security Teams (FIRST): Common Vulnerability Scoring System (CVSS). Available: http://www.first.org/cvss, last accessed Mar. 2010.

[89] M. D'silva and D. Vora, "Comparative Study of Data Mining Techniques to Enhance Intrusion Detection," *Int. J. of Eng. Res. and Appli.*, vol. 3, no. 1, pp.1267-1275, 2013.

[90] DARPA Information Systems Technology, Intrusion Detection Evaluation Project. Available: www.ll.mit.edu/IST, last accessed Mar. 2010.

[91] DARPA, The 1999 DARPA Intrusion Detection Evaluation Program. Available: http://www.ll.mit.edu/IST/ideval, last accessed Mar. 2010.

[92] K. Das, "Attack Development for Intrusion Detection Evaluation," M.S. thesis, Dept. of Elect. Eng. and Comput. Sci., MIT, Cambridge, MA, 2000.

[93] H. Debar, An Introduction to Intrusion-Detection Systems, IBM Zurich Research Laboratory, Switzerland, 2002.

[94] H. Debar, M. Dacier and A. Wespi, "Revised Taxonomy of Intrusion Detection Systems," *Ann. des Télécommun.*, 2000, pp. 361-378.

[95] D. Delmotte and P. Smets, "Target Identification based on the Transferable Belief Model Interpretation of Dempster-Shafer Model," *IEEE Trans. on Syst., Man, and Cybernetics*, vol. 34, no. 4, pp. 457-471, Jul. 2004.

[96] A. DeMontigny-Leboeuf, "Flow Attributes For Use in Traffic Characterization," Commun. Research Center, Ottawa, Canada, Rep. CRC-TN-2005-003, Dec. 2005.

[97] S. Démotier, W. Schon and T. Denœux, "Risk Assessment Based on Weak Information using Belief Functions: A Case Study in Water Treatment," *IEEE Trans. on Syst., Man and Cybernetics*, vol. 36, no. 3, pp. 382-396, May 2006.

[98] T. Denœux, "A k-nearest Neighbour Classification Rule based on Dempster-Shafer Theory," *IEEE trans. on Syst. Men and cybernetics*, vol. 25, no. 5, pp. 804-813, 1995.

[99] T. Denœux, "A Neural Network Classifier based on Demspter-Shafer Theory*," IEEE Trans. on Syst., Man and Cybernetics*, vol. 30, no. 2, pp. 131-150, 2000.

[100] T. Dietterich, "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting and Randomization," *Mach. Learning*, vol. 40, no. 2, pp. 139-157, 2000.

[101] A. Dobra, "Scalable Classification and Regression Tree Construction," Ph.D. dissertation, Dept. of Comput. Sci., Cornell Univ., Ithaca, NY, 2003.

[102] W. DongLiang and W. Hongxin, "The Application and Research of IDS Model Based on Multi-technique Fusion" *Proc. of the 2009 Int. Symp. on Web Inform. Syst. and Appl.*, Nanchang, China, 2009, pp. 148-151.

[103] A. Doruk, "Standards and Practices Necessary to Implement a Successful Security Review Program for Intrusion Management Systems," M.S. thesis, Dept. of Comput. Eng., Izmir Inst. of Technol., Izmir, Turkey, 2002.

[104] L. Douglas, *The Security Risk Assessment Handbook: a Complete Guide for Performing Security Risk Assessments*, 2ed ed., FL, Taylor & Francis, 2005.

[105] R. Duda, P. O. Hart and D. G. Stork, *Pattern Classification*, 2ed ed., NJ, Wiley & Sons, 2001.

[106] G. Elahi, E. Yu and N. Zannone, "A Modelling Ontology for Integrating Vulnerabilities into Security Requirements Conceptual Foundations," *28th Int. Conf. of Conceptual Modelling*, Gramado, Brazil, 2009, pp. 99-114.

[107] Y. EL-Manzalawy and V. Honavar, "WLSVM: Integrating LibSVM into Weka Environment," Unpublished. Available: http://www.cs.iastate.edu/~yasser/wlsvm, last accessed May 2010.

[108] N. El-Moussadi, A. Toumanari and M. El Azhaei, "Intrusion Detection Based On Clustering Algorithm," *Int. J. of Electron. and Compu. Sci. Eng.*, vol. 2, no. 3, pp.1059-1064, 2013.

[109] Z. Elouedi, E. Lefèvre and M. Mercier, "Discountings of Belief Function using a Confusion Matrix," *22th IEEE Int. Conf. on Tools with Artificial Intell.*, Arras, France, 2010, pp. 287-294.

[110] E. Eskin, "Anomaly Detection over Noisy Data using Learned Probability Distributions," *Proc. of 7th Int. Conf. of Mach. Learning, Stanford*, CA, USA, 2000, pp. 255-262.

[111] P. Evangelista, "Computer Intrusion Detection Through Statistical Analysis and Prediction Modelling," M.S. thesis, Dept. of Comput. Sci., Fac. of Rensselaer Polytechnic Inst., Troy, NY, 2005.

[112] F. Farahmand, "Developing a Risk Management System for Information Systems Security Incidents," Ph.D. dissertation, College of Computing, Georgia Inst. of Technol., Georgia, USA, 2004.

[113] D. Farid, N. Harbi and M. Z. Rahman, "Combining Naïve Bayes and Decision Tree for Adaptive Intrusion Detection," *Int. J. of Network Security & Its Appl.*, vol. 2, no. 2, pp. 12-25, 2010.

[114] D. Farid, N. Harbi, S. Ahmmed, M. Z. Rahman, and C. M. Rahman,, "Mining Network Data for Intrusion Detection through Naïve Bayesian with Clustering," *World Academy of Sci., Eng. and Technol.*, vol. 4, no. 6, pp. 280-284, 2010.

[115] B. Farquhar, "One Approach to Risk Assessment," *Comput. Security*, vol. 10, no. 1, pp. 21-23, 1991.

[116] T. Fawcett, "An Introduction to ROC Analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861-874, 2006.

[117] T. Fawcett, "ROC Graphs: Notes and Practical Considerations for Researchers," HP Labs, CA, USA, Rep. TR HPL-2003-4, 2003.

[118] F. Federal Information System Control Manual, FISCAM, General Accounting Office, GOA/AIMD-12.19.6, Jan. 1999.

[119] L. Feng, W. Wang, L. Zhu and Y. Zhang, "Predicting Intrusion Goal using Dynamic Bayesian Network with Transfer Probability Estimation," *J. of Network and Comput. Appl.*, vol. 32, no. 1, pp. 721-732, Jan. 2009.

[120] S. Fenz, G. Goluch, A. Ekelhart, B. Riedl and E. Weippl, "Information Security Fortification by Ontological Mapping of the ISO/IEC 27001 Standard," *Proc. of 13th IEEE Pacific Rim Int. Symp. on Dependable Computing*, 2007, pp. 381-388.

[121] B. Fessi, M. Hamdi, S. Benabdallah and N. Boudriga, "Automated Intrusion Response System: Surveys and Analysis," *Security and Management Conf.*, Las Vegas, NV, USA, 2008, pp. 149-155.

[122] B. Foo, M. W. Glause, G. M. Howard, Y.S. Wu, S. Bagchi and E.H. Spafford, "Intrusion Response Systems: A Survey," Purdue Univ., West Lafayette, IN, Rep. CERIAS TR 2008-4, 2008.

[123] B. Foo, Y. S.Wu, Y. C. Mao and E.H. Spafford, "ADEPTS: Adaptive Intrusion Response using Attack Graphs in an E-Commerce Environment," *Proc. of the Int. Conf. on Dependable Syst. and Network*, Yokohama, Japan, 2005, pp. 508-517.

[124] S. Forrest, S. Hofmeyr, A. Somayaji and E.H. Spafford, "A Sense of Self for UNIX Processes," *IEEE Symp. on Security and Privacy*, Oakland, CA, USA, 1996, pp. 120-128.

[125] T. Furlong, "Tools, Data, and Flow Attributes for Understanding Network Traffic without Payload," M.S. thesis, Carleton Comput. Security Lab, Carlton Univ., Ottawa, Canada, 2007.

[126] B. Gao, H. Y. Ma and Y. H. Yang, "HMMS (Hidden Markov Models) based Anomaly Intrusion Detection Method," *Proc. of the First Int. Conf. on Mach. Learning and Cybernetics*, Beijing, 2002, pp. 4348-4351.

[127] M. Gao, J. Tian and M. Xia, "Intrusion Detection Method Based on Classify Support Vector Machine," *2ed Int. Conf. on Intell. Computation Technol. and Automation*, Changsha, China, 2009, pp. 391-394.

[128] S. Garcia, J. Derrac, J. R. Cano and F. Herrera, "Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study," *IEEE Trans. on Pattern Analysis and Mach. Learning*, vol. 34, no. 3, pp. 417-435, Mar. 2012.

[129] F. Gasse and V. Haarslev, "DLRule: A Rule Editor plug-in for Protégé," *4th Int. Workshop OWL Experiences and Directions*, Washington DC, 2008.

[130] C. Gates, J. J. McNutt, J. B. Kadane and M.I. Kellner, "Scan Detection on Very Large Networks Using Logistic Regression Modelling," *IEEE Symp. on Comput. and Commun.*, Pula-Cagliari, Sardinia, Italy, 2006, pp. 402-407.

[131] J. Gennari, M. A. Musen, R. W. Fergerson, W.E. Grosso, M. Crubézy, H. Eriksson, N.F. Noy and S.W. Tu, "The Evolution of Protégé: An Environment for Knowledge-Based Systems Development," Int. *J. of Human-Comput. Study*, vol. 58, no. 1, pp. 89-123, Jan. 2003.

[132] M. Ghasemzadeh and M. Sarram, "Upgrading Intrusion Detection Systems through Hybrid Algorithms," *World Appl. Sci. J.*, vol. 9, no. 4, pp. 393-397, 2010.

[133] A. Ghosh and A. Schwartzbard, "A Study in using Neural Networks for Anomaly and Misuse Detection," *Usenix Security Symp.*, Washington, DC, 1999.

[134] A. Ghosh, A. Schwartzbard and M. Schatz, "Learning Program Behavior Profiles for Intrusion Detection," *Workshop on Intrusion Detection and Network Monitoring*, Santa Clara, California, USA, 1999.

[135] G. Giacinto and F. Roli and L. Didaci, "Fusion of Multiple Classifiers for Intrusion Detection in Computer Networks," *Pattern Recognition Letters*, vol. 24, no.12, pp. 1795-1803, 2003.

[136] G. Giacinto and F. Roli, "Dynamic Classifier Selection," *1st Int. Workshop of Multiple Classifier Syst.*, Cagliari, Italy, 2000, pp. 177-189.

[137] L. Gigstad, "Reducing False Positives in Intrusion Detection by Means of Frequent Episodes," M.S thesis, Dept. of Comput. Sci. and Media Technol., Gjøvik Univ. College, Norway, 2008.

[138] S. Gilpin and D. M. Dunlavy, "Relationships between Accuracy and Diversity in Heterogeneous Ensemble Classifiers," SANDIA Lab, SANDIA, California, Rep. 2009-6940C, Sep. 2009.

[139] P. Gogoi, D.K. Bhattacharyya, B. Borah and J.K. Kalita, "MLH-IDS: A Multi-Level Hybrid Intrusion Detection Method," *The Compu. J.*, vol. 57, no. 4, pp. 602-623, 2013.

[140] J. Gordon, "Sharing the Unverifiable: Prediction Exchange," infectionvectors.com, NY, USA, SANSFIRE, 2006.

[141] L. Gordon, M. P. Loeb, W. Lucyshyn and R. Richardson, "2004 CSI/FBI Computer Crime and Security Survey," Comput. Security Inst., San Francisco, CA, USA, Rep. 2004 CSI/FBI, 2004.

[142] L. Gordon, M. P. Loeb, W. Lucyshyn and R. Richardson, "2005 CSI/FBI Computer Crime and Security Survey," Comput. Security Inst., NY, USA, Rep. 2005 CSI/FBI, 2005.

[143] A. Goscinski, *Distributed Operating Systems: the Logical Design*, 1st ed., MA, Addison-Wesley, 1991, ch 11 and ch12.

[144] Y. Gu, Y. Shi and J. Wang, "Efficient Intrusion Detection Based on Multiple Neural Network Classifiers with Improved Genetic Algorithm," *J. of Software*, vol. 7, no. 7, pp. 1641-1648, Jul. 2012.

[145] Y. Guan, A. A. Ghorbani and N. Belacel, "Y-Mean: A Clustering method For Intrusion Detection," *Can. Conf. on Elect. and Comput. Eng.*, Montréal, 2003, pp. 1083-1086.

[146] Guide to Computer Log Management, NIST SP-800-92, 2006.

[147] Guide to Intrusion Detection and Prevention Systems, NIST SP800-94, 2012.

[148] Guide for Assessing the Security Controls in Federal Information Systems, Building Effective Security Assessment Plans, NIST SP800-53-A, Jul. 2008.

[149] Guideline for Automatic Data Processing Risk Analysis, Federal Information Processing Standards, FIPS 65, Aug. 1979.

[150] Guideline for Mapping Types of Information and Information Systems to Security Categories, NIST SP-800-60, Aug. 2008.

[151] Guideline for the Analysis of Local Area Network Security, Federal Information Processing Standards, FIPS191, Nov. 1994.

[152] D. Gunter, B. L. Tierney, A. Brown, J. Bresnahan and M. Schopf, "Log Summarization and Anomaly Detection for Troubleshooting Distributed Systems," *Int. Conf. on Grid computing*, Austin, Texas, 2007, pp. 226-234.

[153] K. Gupta, S. Nath and K. Ramamohanarao, "User Session Modelling for Effective Application Intrusion Detection," *IFIP Tc11 23rd Int. Inform. Security Conf.*, Milan, Italy, 2008, pp. 269-284.

[154] J. Haines, R. P. Lippmann, D. J. Fried, M.E. Zissmen, E. Tran and S.B Boswell, "1999 DARPA Intrusion Detection Evaluation: Design and Procedures," MIT, Cambridge, MA, Rep. TR-1062, Feb. 2001.

[155] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2ed ed., San Francisco, CA, Morgan Kaufmann, 2006.

[156] S. Han and S. B. Cho, "Detecting Intrusion with Rule-based Integration of Multiple Models," *Comput. & Security*, vol. 22, no. 7, pp. 613-623, 2003.

[157] *Handbook of Applied Cryptography*, A. Menzes, P. V. Oorshot and S. Vanstone, CRC press, 1997.

[158] S. Hansman, "A Taxonomy of Network and Computer Attack Methodologies," Honours Project Report, Dept. of Comput. Sci. and Software Eng., Univ. of Canterbury, New Zealand, Sep. 2003.

[159] K. Heller, K. M. Svore, A. D. Keromytis and and S. Stolfo, "One Class Support Vector Machines for Detecting Anomalous Windows Registry Accesses," *Proc. of ICDM Workshop on Data Mining for Comput. Security*, Melbourne, Florida, 2003.

[160] G. Helmer, J. S. K. Wong, V. Honavar, L. Miller and Y. Wang, "Lightweight Agents for Intrusion Detection," *J. Syst. and Software*, vol. 67, no. 2, pp. 109-122, Aug. 2003.

[161] L. Ho, "Reduction of IDS False Alarms using KNN Classifiers," M.S. thesis, Dept. of Comput. Sci., City Univ. of Hong Kong, Hong Kong, 2007.

[162] S. Hofmeyr, S. Forrest and A. Somayaji, "Intrusion Detection using Sequences of System Calls," *J. of Comput. Security*, vol. 6, no. 3, pp. 151-181, Aug. 1998.

[163] M. Hosseinkhani, E. Tarameshloo and B. Sadeghiyan, "A Two Dimensional Approach for Detecting Input Validation Attacks Based on HMM," *Int. Conf. on Databases and Data Mining*, China, 2011.

[164] J. Howard, "An Analysis of Security Incidents on the Internet," Ph.D. dissertation, Dept. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, Pennsylvania, 1997.

[165] B. Hu, R. Hea and X. Yuanc, "Information-Theoretic Measures for Objective Evaluation of Classifications," *Acta Automatica Sinica*, vol. 38, no. 7, pp. 1169-1182, 2012.

[166] J. Hu, H. H. Chen and X. Yu, "A Simple and Efficient Hidden Markov Model Scheme for Host-based Anomaly Intrusion Detection," *IEEE Networking*, vol. 23, no. 1, pp. 42-47, Oct. 2009.

[167] L. Hu, L. Nurbol, Z. Liu, J. He and K. Zhao., "A Time-stamp Frequent Pattern based Clustering Method for Anomaly Detection," *IETE Tech. Review*, vol. 27, no.3, pp. 220-227, 2010.

[168] L. Huang, "A Study on Masquerade Detection," M.S. thesis, Dept. of Comput. Sci., San Jose State Univ., San Jose, Canada, 2010.

[169] N. Hubballi, S. Biswas and S. Nandi, "An Efficient Data Structure for Storing Network Intrusion Detection Dataset," *2^{ed} Int. Symp. on Advanced Networks and Telecommun. Syst.,* Mumbai, 2008, pp. 1-3.

[170] A. Hung-Ren Ko, R. Sabourin and A. J. De Souza Britto, "Combining Diversity and Classification Accuracy for Ensemble Selection in Random Subspaces," *Int. Joint Conf. on Neural Networks*, Vancouver, BC, 2006, pp. 2144-2151.

[171] K. Hwang, M. Cai, Y. Chen and M. Qin, "Hybrid Intrusion Detection with Weighted Signature Generation over Anomalous Internet Episodes," *IEEE Trans. on Dependable and Secure Computing*, vol. 4, no. 1, pp. 1-15, 2007.

[172] L. Ibrahim, "Artificial Neural Network for Misuse Detection," *J. of Commun. and Comput.*, vol. 7, no. 6, pp. 457-471, Jun. 2010.

[173] Information Security Management System Requirements, ISO, Information Technol., Security Techniques, BS ISO-IEC 27001:2005, Oct. 2005.

[174] Information Security Risk Assessment Practices and Leading Organizations, IFSRAPLO, General Accounting Office, GOA/AIMD-99-139, Aug. 1999.

[175] Information Security Risk Management, ISO, Information Technology, Security Techniques, BS ISO-IEC 27005:2008, Jun. 2008.

[176] Intrusion Detection Systems, NIST SP800-31, Aug. 2001.

[177] Y. Jain and U. Pendra, "An Efficient Intrusion Detection based on Decision Tree Classifier using Feature Reduction," *Int. J. of Scientific and Res. Publication*, vol. 1, no. 6, pp.1-6, 2012.

[178] F. Jensen, *An Introduction to Bayesian Networks*, 1st ed., London, UCL Press, 1996.

[179] Z. Jia and S. Jin, "The Application of Data Mining Technology in the Intrusion Detection System," *3rd Int. Symp. on Comput. Sci. and Computational Technol.*, Jiaozuo, P. R. China, 2010, pp. 208-211.

[180] E. Johnson, *Managing Information Risk and the Economics of Security*, 2009 ed., Springer, 2009.

[181] A. Jones and D. Ashenden, *Risk Management for Computer Security Protecting your Network and Information Assets*, 1st ed., Betterworth-Heinemann, 2005.

[182] M. Joshi, "On Evaluating Performance of Classifiers for Rare Classes," *Proc. of the 2ed IEEE Int. Conf. on Data Mining*, Japan, 2002, pp. 641-644.

[183] K. Karunanidhi, "ARROS: Distributed Adaptive Real-time Network Intrusion Response," M.S. thesis, Dept. of Comput. Sci., Russ College of Eng. and Technol., Ohio Univ., USA, 2006.

[184] C. Katar, "Combining Multiple Techniques for Intrusion Detection," *Int. J. of Comput. Sci. and Network Security,* vol.6, no. 2B, pp. 208-218, Feb. 2006.

[185] C. Katar, B. Raggad and E. J. Collar, "Transferable Belief Model in Computer Security," *J. of Computing E-Syst.*, vol.1, no.1, pp. 39-48, Jan. 2008.

[186] C. Katar, "Etude, Analyse et Comparaison de Protocoles d'Authentification Forte Basée sur le Mot de Passe," Mémoire de DEA, ISG, Tunis, Tunisie, Jui. 2002.

[187] V. Katkar and R. Mathew, "One Pass Incremental Association Rules Detection Algorithm for network Intrusion Detection," *Int. J. of Eng. Sci. and Technol.*, vol. 3, no. 4, pp. 3055-3060, Apr. 2011.

[188] H. Kayacık, "Hierarchical Self Organizing Map based IDS on KDD Benchmark," M.S. thesis, Dept. of Comput. Sci., Dalhousie Univ., Nova Scotia, Canada, 2003.

[189] KDD DATABASE. Available: kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, last accessed Sep. 2009.

[190] K. Kendall, "Database of Computer Attacks for the Evaluation of Intrusion Detection Systems," M.S. thesis, Dept. of Elect. Eng. and Comput. Sci., MIT, Cambridge, MA, 1999.

[191] F. Kerschbaum, E. H. Spafford and D. Zamboni, "Using Embedded Sensors for Detecting Network Attacks," *1st ACM Workshop on Intrusion Detection Syst.*, 2000, pp. 1-16.

[192] V. Kettnaker, "Time-dependent HMMs for Visual Intrusion Detection," *IEEE Workshop on Event Mining Detection and Recognition of Event in Video*, Madison, Wisconsin, USA 2003.

[193] L. Khan, M. Awad and B. Thuraisingham, "A New Intrusion Detection System using Support Vector Machine and Hierarchical Clustering," *Int. J. of Very Large Data Base*, vol. 16, no. 4, pp.507-521, Oct. 2007.

[194] R. Khanna and H. Liu, "Control Theoretic Approach to Intrusion Detection using Distributed Hidden Markov Model," *IEEE Wireless Commun.*, vol. 15, no. 4, pp. 24-33, Aug. 2008.

[195] K. Killourhy, R. A. Maxion and K. M. C. A Tan, "Defense-Centric Taxonomy Based on Attack Manifestations," *Int. Conf. on Dependable Syst. & Networks*, Florence, Italy, 2004, pp. 102-111.

[196] T. Kohonen, "Self Organization Map," *Proc. of the IEEE*, vol. 78, no. 9, pp. 1464-1480, Sep. 1990.

[197] L. Kokorina, "The Use of Frequent Episodes in Intrusion Detection," M.S. thesis, Dept. of Comput. Sci. and Media Technol., Gjøvik Univ. College, Norway, 2009.

[198] G. Kołaczek and K. Juszczyszyn, "Traffic and Attack Pattern Analysis for Multi-agent Distributed Intrusion Detection System," *Int. Conf. on Intell. Syst. and Knowledge Eng.*, Chengdu, China, 2007.

[199] I. Krsul, "Software Vulnerability Analysis," Ph.D. dissertation, Dept. Comput. Sci., Purdue Uni., West Lafayette, IN, 1998.

[200] C. Kruegel, D. Mutz, W. Robertson and F. Valeur, "Bayesian Event Classification for Intrusion Detection," *Annu. Comput. Security Appl. Conf.*, Las Vegas, NV, USA, 2003, pp. 14-23.

[201] C. Kruegel, F. Valeur and G. Vigna, "Intrusion Detection and Correlation: Challenges and Solutions," *Advances in Inform. Security*, Springer, vol. 14, pp. 1-118, 2005.

[202] C. Kruegel, G. Vigna and W. Robertson, "A Multi-model Approach to the Detection of Web-based Attacks," *Comput. Networks*, vol. 48, no. 5, pp. 717-738, Aug. 2005.

[203] S. Kumar, "Classification and Detection of Computer Intrusion," Ph.D. dissertation, Dept. Comput. Sci., Purdue Univ., West Lafayette, IN, USA, 1995.

[204] A. Kumaravel, "Comparison of Two Multi-Classification Approaches for Detecting Network Attacks," *World Appl. Sci. J.*, vol. 27, no. 11, pp.1461-1465, 2013.

[205] L. Kuncheva and C. J. Whitaker, "Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy," *Mach. Learning*, vol. 51, pp. 181-207, 2003.

[206] L. Kuncheva, "On Combining Multiple Classifiers," *Proc. of 7th Int. Conf. of Inform. Processing and Management of Uncertainty*, Paris, France, 1998, pp. 1890-1891.

[207] L. Kuncheva, C. Whittaker, C. Shipp and R. Duin, "Is Independence Good for Combining Classifiers," *Proc. of the 15th Int. Conf. on Pattern Recognition*, Barcelona, Spain, 2000, pp. 168-171.

[208] L. Kuncheva, *Combining Pattern Classifiers, Methods and Algorithms*, John Wiley & sons, 2005.

[209] L. Kuncheva, M. Skurichina and R. P. W. Duin, "An Experimental Study on Diversity for Bagging and Boosting with Linear Classifiers," *Inform. Fusion*, vol. 3, no. 2, pp. 245-258, 2002.

[210] K. Labib and R. Vemuri, "NSOM: A Real-Time Network-Based Intrusion Detection System Using Self-Organizing Maps," Dept. of Appl. Sci., Univ. of California, Davis, USA, Rep. Technical report, 2002.

[211] C. Landwehr, A. Bull, J. P. McDermott and W.S. Choi, "A Taxonomy of Computer Program Security Flaws," *ACM Computing Surveys*, vol. 26, no. 3, pp. 211-254, Sep. 1994.

[212] U. Larson, "Aspects of Adapting Data Collection to Intrusion Detection," Dept. of Comput. Sci., Chalmers Univ. of Technol., Gothenburg, Sweden, Rep. TR-25-L 2006.

[213] U. Larson, E. Jonsson and S. Lindskog, "Structured Overview of Data Collection with a Focus on Intrusion Detection," Karlstad Univ., Karlstad, Sweden, Rep. TR-2008-19, 2008.

[214] N. Lavesson and P. Davidsson, "A Multi-dimensional Measure Function for Classifier Performance," *2ed IEEE Conf. on Intell. Syst.*, 2004, pp. 508-513.

[215] N. Lavesson and P. Davidsson, "Analysis of Multi-Criteria Metrics for Classifier and Algorithm Evaluation," *24th Annu. Workshop of the Swedish Artificial Intell. Soc.*, 2007.

[216] T. Layton, *Information Security Design, Implementation, Measurements and Compliance*, 1st ed., Auerbach Publishers, 2006.

[217] A. Lazarevic, V. Kumar and J. Srivastava, "Intrusion Detection: a Survey," *Managing Cyber Threats, Massive Computing*, vol. 5, no. 1, pp. 19-78, 2005.

[218] J. Lee, S. G. Sohn, J. H. Ryu and T.M. Chung, "Effective Value of Decision Tree with KDD 99 Intrusion Detection Dataset for Intrusion Detection System," *Int. Conf. on Advanced Commun. Technol.*, India, 2008, pp. 1170-1175.

[219] T. Lee, D. Kim and H. Peter, "Maximizing Return on Security Safeguard Investment with Constraint Satisfaction," *Int. Conf. on Inform. Security and Assurance*, Busan, Korea, 2008, pp. 172-175.

[220] W. Lee, and S. J. Stolfo, "A Framework for Constructing Features and Models for Intrusion Detection Systems," *ACM Trans. on Inform. and Syst. Security*, vol. 3, no. 4, pp. 227-261, Nov. 2000.

[221] W. Lee, and S. J. Stolfo, "Data Mining Approaches for Intrusion Detection," *7th USENIX Security Symp.*, San Antonio, TX, 1998.

[222] W. Lee, S. J. Stolfo and K. Mok, "Adaptive Intrusion Detection: a Data Mining Approach," *Artificial Intell. Review*, vol. 14, no. 6, pp. 533-567, 2000.

[223] W. Lee, S. J. Stolfo and K. Mok, "Mining in a Data-Flow Environment: Experience in Network Intrusion Detection," *Proc. of the Fifth Int. Conf. on Knowledge Discovery and Data Mining*, San Diego, CA, 1999, pp. 114-124.

[224] W. Lee, S. J. Stolfo and F. K. Chan, "A Data Mining Approach Building Cost-Sensitive and Light Intrusion Detection Models," Colombia Univ., NY, Rep. AFRL-IF-RS-TR-2004-84, Mar. 2004.

[225] W. Lee, W. Fan, M. Millerand and E. Zadok, "Toward Cost-Sensitive Modelling for Intrusion Detection and Response," *J. of Comput. Security*, vol. 10, no. 1/2, pp. 5-22, 2000.

[226] Y. Lee, Y. Lin and G. Wahba, "Multi-category Support Vector Machines," Dept. of Statistics, Univ. of Wisconsin-Madison, Wisconsin, USA, Rep. TR-1043, Sep. 2001.

[227] E. Lefevre, O. Colot, P. Vannoorenberghe and et D. de Brucq, "Informations et Combinaison: les Liaisons Conflictuelles," *Traitement du Signal*, vol. 18, no. 3, pp. 161-177, 2001.

[228] F. Lefevre, "Fusion Adaptée d'Informations Conflictuelles dans le Cadre de la Théorie de l'Evidence, Application au Diagnostic Médical," Thèse de doctorat, Institut Nat. des Sci. Appliquées, Rouen, France, 2001.

[229] A. Lenstra and T. Voss, "Information Security Risk Assessment, Aggregation, and Mitigation," *Proc. of the 9th Australian Conf. on Inform. Security and Privacy*, Sydney, Australia, 2004, pp. 391-401.

[230] Y. Liao and V. R. Vemuri, "Using K-Nearest Neighbour Classifier for Intrusion Detection," *Comput. and Security*, vol. 21, no. 5, pp. 439-448, 2002.

[231] P. Lichodzjewski, A. N. Zincir-Heywood and M. Heywood, "Dynamic intrusion detection using self-organizing maps," *Proc. of the 14th Annu. Can. Inform. Technol. Security Symp.*, Ottawa, Canada, May 2002, pp. 428-433.

[232] U. Lindqvist and E. Jonsson, "How to Systematically Classify Computer Security Intrusions," *IEEE Security and Privacy Symp.*, 1997, pp. 154-163.

[233] U. Lindqvist, "On the Fundamentals of Analysis and Detection of Computer Misuse," PhD. dissertation, Dept. of Comput. Eng., Chalmers Univ. of Technol., Göteborg, Sweden, 1999.

[234] F. Liu, "Constructing Enterprise Information Network Security Risk Management Mechanism by Using Ontology," *21st Int. Conf. on Advanced Inform. Networking and Appl.*, Niagara Falls, Canada, 2007, pp. 929-934.

[235] S. Liu, C. Cheung and L. Kwok, "A Knowledge Framework for Information Security Modelling," *4th Australian Inform. Security Management Conf.*, Western Australia, 2006, pp. 81-90.

[236] T. Löfström, "Utilizing Diversity and Performance Measures for Ensemble Creation," Licentiate thesis, School of Sci. and Technol., Univ. of Boräs, Boräs, Sweden, 2009.

[237] T. Löfström, U. Johansson and H. Boström., "On the Use of Accuracy and Diversity Measures for Evaluating and Selecting Ensembles of Classifiers," *7th Int. Conf. on Mach. Learning and Appl.*, San Diego, CA, 2008, pp. 127-132.

[238] J. López, "Prototype Selection Methods," *Computación y Sistemas J.*, vol. 13, no. 4, pp. 449-462, 2010.

[239] W. Lu and I. Traore, "Unsupervised Anomaly Detection using an Evolutionary Extension of k-means Algorithm," *Int. J. on Inform. and Comput. Security*, vol. 2, no. 2, pp. 107-139, Aug. 2008.

[240] C. Lui, T. C. Fu and T.Y. Cheung, "Agent based Network Intrusion Detection System using Data Mining Approaches," *3rd Int. Conf. on Inform. Technol. and Appl.*, Caims, Australia, 2005, pp. 131-136.

[241] A. Lumini and L. Nanni, "A Clustering Method for Automatic Biometric Template Selection," *Pattern Recognition*, vol. 39, no. 3, pp. 495-497, Mar. 2006.

[242] J. Luo and S. M. Bridges, "Mining Fuzzy Association Rules and Fuzzy Frequent Episodes for Intrusion Detection," *Int. J. of Intell. Syst.*, vol. 15, no. 1, pp. 687-703, Aug. 2000.

[243] J. Luo and S. M. Bridges and R. B. Vaughn, "Fuzzy Frequent Episodes for Real-Time Intrusion Detection," *10th IEEE int. conf. on fuzzy Syst.*, 2001, pp. 368-371.

[244] F. Maggi, M. Matteucci and S. Zanero, "Detecting Intrusions through System Call Sequence and Argument Analysis," *Trans. on Dependable and Secure Computing*, vol. 7, no. 4, pp. 381-395, Aug. 2007.

[245] A. Mahmood, C. Leckie and P. Udaya, "Echidna: Efficient Clustering of Hierarchical Data for Network Traffic Analysis," *Proc. of the 5th int. IFIP-TC6 conf. on Networking 2006*, 2006, pp. 1092-1098.

[246] M. Mahoney and P. K. Chan, "An Analysis of the 1999 DARPA/Lincoln Lab Evaluation Data for Network Anomaly Detection," *Symp. of Recent Advances in Intrusion Detection*, 2003, pp. 220-237.

[247] K. Makkithaya, N.V. Subba Reddy and U. Dinesh Acharya, "Improved C-fuzzy Decision Tree for Intrusion Detection," *Int. Conf. on Artificial Intell. and Pattern Recognition*, Orlando, Florida, USA, 2008, pp. 140-145.

[248] Managing Information Security Risks, NIST SP800-39, 2011.

[249] Managing Risk from Information Systems: An Organizational Perspective, NIST SP800-39, Apr. 2008.

[250] H. Mannila, H. Toivonen and A. I. Verkamo, "Discovering Frequent Episodes in Sequences," *Proc of the 1st Int. Conf. on Knowledge Discovery in Databases and Data Mining*, Montreal, Canada, 1995, pp. 259-289.

[251] Z. Mao, V. Sekar, O. Spatscheck, J. Van Der Merwe and R. Vasudevan, "Analyzing Large DDoS Attacks Using Multiple Data Sources," *ACM SIGCOMM Workshop on Large-Scale Attack Defense*, Pisa, Italy, 2006, pp. 161-68.

[252] V. Markam and L. S. D. Dubey, "A General Study of Associations rule mining in Intrusion Detection System," *Int. J. of Emerging Technol. and Advanced Eng.*, vol. 2, no. 1, pp. 347-356, Jan. 2012.

[253] O. Mazhelis, "One-Class Classifiers: A Review and Analysis of Suitability in the Context of Mobile-Masquerader Detection," *South African Comput. J. - SA Comput. J.*, vol. 36, pp. 29-48, 2006.

[254] C. Mazzariello, "Multiple Classifier Systems for Network Security from Data Collection to Attack Detection," Ph.D. dissertation, Dept. Comput. Sci. and Syst., Univ. of Napoli, Napoli, Italy, 2007.

[255] A. Medaglia and E. Gutiérrez, "JGA: An Object-Oriented Framework for Rapid Development of Genetic Algorithms," *Handbook of Research on Nature Inspired Computing in Economics and Management*, IGL Global, 2006.

[256] P. Mell, K. Scarfone and S. Romanosky, "Common Vulnerability Scoring System (CVSS) and Its Applicability to Federal Agency Systems," NIST, USA, Rep. IR-7435, Aug. 2007.

[257] D. Mercier, G. Cron, T. Denœux and M. Masson, "Fusion of Multi-level Decision Systems using the Transferable Belief Model," *Proc. of the 8th Int. Conf. on Inform. Fusion*, Philadelphia, USA, 2005, pp. 655-658.

[258] A. Mewada, P. Gedam, S. Khan and M.U. Reddy, "Network Intrusion Detection Using Multiclass Support Vector Machine," *Int. J. of Comput. and Commun. Technol.*, vol. 1, no. 3, pp. 536-543, Aug. 2010.

[259] Microsoft Corporation, MS Security Response Center Security Bulletin Severity Rating System, Nov. 2002. Available: http://www.microsoft.com/technet/security/bulletin/rating.mspx, last accessed Jun. 2009.

[260] Minimum Security Requirements for Federal Information and Information Systems, FIPS 200, Mar. 2006.

[261] T. Mitchell, *Machine Learning*, 1st ed., McGraw-Hill, 1997.

[262] A. Mitrokotsa and C. Douligeris, "Detecting Denial of Service Attacks Using Emergent Self-Organizing Maps," *IEEE Int. Symp. on Signal Processing and Inform. Technol.*, Athens, Greece, 2005, pp. 375-380.

[263] H. Mousa, "A Survey and Analysis of Neural Network Approaches to Intrusion Detection," SANS inst., GIAC practical repository, 2002.

[264] Z. Muda, W. Yassin, M. N. Sulaiman and N.I. Udzir, "K-means and Naïve Bayes Learning Approach for Better Intrusion Detection," *Inform. Technol. J.*, vol. 10, no. 3, pp. 648-655, 2011.

[265] S. Mukkamala, G. I. Janoski and A. H. Sung, "Intrusion Detection using Support Vector Machine," *Proc. of Advanced Simulation Technologies Conf.*, San Diego, CA, 2002, pp. 178-193.

[266] S. Mukkamala, G. I. Janoski and A. H. Sung, "Intrusion Detection using Neural Networks and Support Vector Machine," *Proc. of IEEE Int. Joint Conf. on Neural Networks*, Honolulu, Hawaii, 2002, pp. 1702-1707.

[267] S. Mulay, P. R. Devale and G. V. Garje, "Intrusion Detection System using Support Vector Machine and Decision Tree," *Int. J. of Comput. Appli.*, vol. 3, no. 3, pp.40-43, Jun. 2010.

[268] G. Munz, S. Li and G. Carle, "Traffic Anomaly Detection Using K-Means Clustering," Comput. Networks and Internet, Univ. of Tuebingen, Germany, 2007.

[269] R. Musehane, F. Netshiongolwe, F. V. Nelwamondo, L. Masisi and T. Marwala, "Relationship between Structural Diversity and Performance of Multiple Classifiers

for Decision Support," *Annu. Symp. of the Pattern Recognition Association of South Africa*, Cape Town, South Africa, 2008.

[270] R. Naoum and Z. N. Al-Sultani, "Learning Vector Quantization and k-Nearest Neighbor for Intrusion Detection," *World of Comput. Sci. and Inform. Technol. J.*, vol. 2, no. 3, pp. 105-109, 2012.

[271] I. Naseem, "Combining Classifiers Using the Dempster-Shafer Theory of Evidence," M.S. thesis, King Fahed Univ. of Petroleum and Minerals, Dhahran, Saudi Arabia, 2005.

[272] P. Natesan, P. Balasubramanie and G. Gowrison, "Improving the Attack Detection Rate in Network Intrusion Detection using Adaboost Algorithm," *J. of Comput. Sci.*, vol. 8, no. 7, pp. 1041-1048, 2012.

[273] D. Neill and G. F. Cooper, "A Multivariate Bayesian Scan Statistic for Early Event Detection and Characterization," *Mach. Learning*, vol. 79, no. 3, pp. 261-282, Nov. 2009.

[274] P. Neumann and D. B. Parker, "A Summary of Computer Misuse Techniques," *Proc. of 12th Nat. Comput. Security Conf.*, Baltimore, USA, 1989, pp. 396-407.

[275] H. Nguyen and Y. Choi, "Application of Data Mining to Network Intrusion Detection: Classifier Selection Model," *Asia-Pacific Network Operations and Management Symp.*, 2008, pp. 399-408.

[276] H. Nguyen and Y. Choi, "Proactive Detection of DDoS Attacks Utilizing k-NN Classifier in an Anti-DDos Framework," *Int. J. of Elect. and Electron. Eng.*, vol. 4, no. 4, pp. 247-252, 2010.

[277] J. Nieves, "Data Clustering for Anomaly Detection in Network Intrusion Detection," Research Alliance in Math and Sci. Program, Polytechnic Univ. of Puerto Rico, PR, Aug. 2009. Available: http://info.ornl.gov/sites/rams09/j_nieves_rodrigues/Documents/report.pdf

[278] A. Novokhodko, "Neural Networks with categorical valued inputs and applications to intrusion detection," Doctoral dissertation, Laboratory of Appl. Computational Intell., Univ. of Missouri-Rolla, Rolla, MO, USA, 2004.

[279] NSL-KDD database. Available: http://nsl.cs.unb.ca/NSL-KDD, last accessed Sep. 2009.

[280] NVDB database, National Institute of Standards and Technology: National Vulnerability Database. Available: http://nvd.nist.gov/nvd.cfm , last accessed Mar. 2010.

[281] J. O'Brien, *Management Information Systems with Additional Cases for NJIT*, 1st ed., McGraw-Hill, 2004.

[282] Office of Management and Budget, Management of Federal Information Resources, Security of Federal Automated Information Resources, SFAIR, NY, USA, Rep. Circular A-130, Feb. 1996.

[283] I. Onut and A. A. Ghorbani, "A Feature Classification Scheme for Network Intrusion Detection," *Int. J. of Network Security*, vol. 5, no. 1, pp. 1-15, Jul. 2007.

[284] OSVDB, Open Source Vulnerability Database (OSVDB): Open Source Vulnerability Database. Available: http://osvdb.org, last accessed Mar. 2010.

[285] V. Pachghare and P. Kularni, "Network Security Based on Pattern Matching: An Overview," *Int. J. of Comput. Sci. and Network Security*, vol. 8, no. 10, pp. 314-318, Oct. 2008.

[286] M. Panda and M. R. Patra, "Some Clustering Algorithms to Enhance Performance of the Network Intrusion Detection Systems," *J. of Theoretical and Appl. Inform. Technol.*, vol. 4, no. 8, pp. 795-801, 2008.

[287] M. Panda and M. R. Patra, "A Novel Classification via Clustering Method for Anomaly Based Network Intrusion Detection System," *Int. J. of Recent Trends in Eng.*, vol. 2, no. 1, pp. 1-6, Nov. 2009.

[288] M. Panda and M. R. Patra, "Ensemble Voting System for Anomaly Based Network Intrusion Detection," *Int. J. of Recent Trends in Eng.*, vol. 2, no. 5, pp. 8-13, Nov. 2009.

[289] M. Panda and M. R. Patra, "Network Intrusion Detection using Naïve Bayes," *Int. J. of Comput. Sci. and Network Security*, vol. 7, no. 12, pp. 258-263, Dec. 2007.

[290] M. Papadaki, S. Furnell, B. Lines and P. Reynolds, "Operational Characteristics of an Automated Intrusion Response System," *Int. Federation for Inform. Processing, Commun. and Multimedia Conf.*, 2003, pp. 65-75.

[291] C. Parikh, M.J. Pont and N.B. Jones, "Application of Dempster-Shafer Theory in Condition Monitoring Systems: A case study," *Pattern Recognition Letters*, vol. 22, no. 6-7, pp. 777-785, 2001.

[292] Y. Park, "A Statistical Process Control Approach for Network Intrusion Detection," Ph.D. dissertation, School of Industrial and Syst. Eng., Georgia Inst. of Technol., Georgia, USA, 2005.

[293] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, 1st ed., Morgan Kaufmann, 1988.

[294] S. Peddabachigari, A. Abraham, C. Grosan and J. Thomas, "Modelling Intrusion Detection System using Hybrid Intelligent Systems," *J. of Network and Comput. Appl.*, vol. 30, no. 1, pp. 114-132, 2007.

[295] T. Peltier, *Information Security Risk Analysis*, 2ed ed., CRC Press, 2005.

[296] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto and W. Lee, "McPAD : A Multiple Classifier System for Accurate Payload-based Anomaly Detection Computer Networks," *Traffic Classification and Its Appl. to Modern Networks*, vol. 5, no. 6, pp. 864-888, 2009.

[297] I. Perona, O. Arbelaitz, I. Gurrutxaga, J. I. Martin, J. Muguerza and J.M. Pérez, "Unsupervised Anomaly Detection System for nIDSs based on payload and Probabilistic Suffix Trees," *Proc. of IADIS Int. Conf. Appl. Computing*, Rome, Italy, 2009, pp.11-18.

[298] S. Pervez, I. Ahmad, A. Akram and S.U. Swatt, "A Comparative Analysis of Artificial Neural Network Technologies in Intrusion Detection Systems," *6th WSEAS Int. Conf. on Multimedia, Internet & Video Technologies*, Lisbon, Portugal, 2006, pp. 84-89.

[299] B. Pfahringer, "Winning the KDD99 Classification Cup: Bagged Boosting," *ACM SIGKDD Explorations Newsletter*, vol.1, no.1, pp. 65-66, Jan. 2000.

[300] P. Pfleeger, *Security in Computing*, 4th ed., Prentice Hall, 2006.

[301] P. Pichon, "Belief Functions: Canonical Decompositions and Combination Rules," Thèse de doctorat, Univ. de Technologie de Compiègne, Compiègne, France, 2009.

[302] J. Pinkston, J. Undercoffer, A. Joshi and T. Finin, "A Target-Centric Ontology for Intrusion Detection," *18th Int. Joint Conf. on Artificial Intell.*, Barcelona, Spain, 2003.

[303] L. Plasencia, C. E. Garcia-Reyes, R. P. Duin and M. Orozco-Alzate, "Prototype Selection Methods for Dissimilarity Space Classification," Center of Appl. and Advanced Technol. CENATVA, Cuba, Rep. TR 24-032010, 2010.

[304] Ponemon Institute. (2012, Oct.), 2012 Cost of Cyber Crime Study. Available: www.ponemon.org/data-security, last accessed Mar. 2010.

[305] K. Pooloth and R. Jetley, "Integrating the CLIPS Rule Engine with Protégé," *Conf. on Artificial Intell. and Mach. Learning*, Cairo, Egypt, 2005.

[306] P. Porras and P. G. Neumann, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances," *20th Nat. Inform. Syst. Security Conf.*, Baltimore, MD, 1997, pp. 353-365.

[307] L. Portony, E. Eskin and S. Stolfo, "Intrusion Detection with Unlabelled Data using Clustering," *Proc. of ACM CSS Workshop on Data Mining Appl. to Security*, Philadelphia, PA, USA, 2001.

[308] F. Pouget and M. Dacier, "Alert Correlation: Review of the State of the Art," Institut Eurecom, Sophia Antipolis, France, Rep. RR-03-093, Nov. 2003.

[309] Protégé Ontology Editor and Knowledge Acquisition System, Stanford Univ. Available: http://protege.stanford.edu/ , last accessed Mar. 2009.

[310] J. Quinlan, "Induction of decision trees," *Mach. Learning*, vol. 1, no. 1, pp. 81-106, 1986.

[311] J. Quinlan, *C4.5: Programs for Machine Learning*, 1st ed., San Francisco, CA, Morgan Kaufmann, 1993.

[312] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Application in Speech Recognition," *Proc. of the IEEE*, vol. 77, no. 2, pp. 257-286, 1989.

[313] B. Raggad and E. J. Collar, "The Simple Information Security Audit Process: SISAP," *Int. J. of Comput. Sci. and Network Security*, vol. 6, no. 6, pp. 189-198, Jun. 2006.

[314] B. Raggad, "Corporate Vital Defense Strategies, A Framework for Information Assurance," *23th Nat. Inform. Syst. Security Conf.*, Baltimore, MD, USA, 2000.

[315] B. Raggad, *Information Security Management Concepts and Practice*, 1st ed., Taylor & Francis, 2010.

[316] S. Ramachandrani, R. C. Mundada, A. K. Bhattacharjee, C. S. Murthy and R. Sharma, "Classifying Host Anomalies: Using Ontology in Information Security Monitoring," in *Cyber Security, Cyber Crime and Cyber Forensics: Appl. and Perspectives*, IGI Global 2011, Dec. 2010, pp. 70-86.

[317] R. Ranawana and V. Palade, "Multi-Classifier Systems - A Review and Roadmap for Developers," *Int. J. of Hybrid Intell. Syst.*, vol. 3, no. 1, pp. 35-61, 2006.

[318] V. Raskin, C. F. Hempelmann, K. E. Triezenberg and S. Nirenburg, "Ontology in Information Security: A Useful Theoretical Foundation and Methodological Tool," *Proc. of ACM New Security Paradigms Workshops*, Germany, 2001, pp. 53-59.

[319] A. Rauber, D. Merkl and M. Dittenbach, "The Growing Hierarchical Self-Organizing Map: Exploratory Analysis of High Dimensional Data," *IEEE Trans. on Neural Networks*, vol. 13, no. 6, pp. 1331-1341, 2002.

[320] P. Ravi Kiran Varma and V. Valli Kumari, "Feature Optimization and Performance Improvement of a Multiclass Intrusion Detection System using PCA and ANN," *Int. J. of Comput. Appl.*, vol. 44, no. 13, pp. 4-9, Apr. 2012.

[321] T. Ravindra Babu and M. Narasimha Murty, "Comparison of Genetic Algorithm based Prototype Selection Schemes," *Pattern Recognition*, vol. 34, no. 2, pp. 523-525, Apr. 2001.

[322] L. Rayon, "A Method for Classifying Attack Implementations based upon its Primary Objective," M.S. thesis, Graduate College, Iowa State Univ., Iowa, USA, 2004.

[323] Recommended Security Controls for Federal Information Systems, NIST SP800-53, Jul. 2008.

[324] R. Richardson, "2010/2011 CSI/FBI Computer Crime and Security Survey," Comput. Security Inst., NY, Rep. 2010/2011 CSI/FBI, 2011.

[325] Risk Management Guide for Information Technology Systems, NIST SP800-30, Jul. 2002.

[326] D. Ruta and B. Gabrys, "An Overview of Classifier Fusion Methods," *Computing and Inform. Syst.*, vol. 7, no. 1, pp.1-10, 2000.

[327] D. Ruta and B. Gabrys, "Classifier Selection for Majority Voting," *Inform. Fusion*, vol. 6, no. 1, pp. 63-81, Mar. 2006.

[328] I. Ruthven and M. Lalmas, "Using Dempster-Shafer's Theory of Evidence to Combine Aspects of Information Use," *J. of Intell. Inform. Syst.*, vol. 19, no. 3, pp. 267-301, 2002.

[329] J. Ryan, M. Lin and R. Mikkulainen, "Intrusion Detection with Neural Networks," *Advances in Neural Inform. Processing Syst.*, MIT Press, vol. 10, pp. 943-949, 1998.

[330] M. Sabhnani and G. Serpen, "Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context," *Mach. Learning, Technologies and Appl.*, Las Vegas, Nevada, USA, 2003, pp. 209-215.

[331] R. Sadoddin and A. A. Ghorbani, "A Comparative Study of Unsupervised Machine Learning and Data Mining Techniques for Intrusion Detection," *5th Int. Conf. in Mach. Learning and Data Mining in Pattern Recognition*, Leipzig, Germany, 2007 pp. 404-418.

[332] M. Sahinoglu, *Trustworthy Computing: Analytical and Quantitative Engineering Evaluation*, 1st ed., NY, John Wiley & sons, 2007.

[333] M. Sammany, M. Sharawi, M. El-Beltagy and I. Saroit, "Artificial Neural Networks Architecture for Intrusion Detection Systems and Classification of Attacks," *5th Int. INFOS 2007 Conf.*, Giza, Egypt, 2007.

[334] R. Sandhu and P. Smaratti, "Authentication, Access Control and Intrusion Detection," in *Comput. Sci. and Eng. Handbook*, GMU, 1997.

[335] P. Sangkatsanee, N. Wattanapongsakorn and C. Charnsripinyo, "Real-time Intrusion Detection and Classification," *Nat. Comput. Sci. and Eng. Conf.* (NCSEC), Bangkok, Thailand, 2009.

[336] SANS Institute, SANS Critical Vulnerability Analysis Archive. Available: http://www.sans.org/newsletters/cva/, last accessed Jun. 2009.

[337] S. Schechter, "Computer Security Strength & Risk: A Quantitative Approach," Ph.D. dissertation, Harvard Univ., Cambridge, MA, USA, 2004.

[338] S. Schechter, "Toward Econometric Models of the Security Risk from Remote Attacks," *IEEE Security & Privacy*, vol. 3, no .1, pp. 40-44, Feb. 2005.

[339] P. Scherer, M. Vicher, M. Dráždiová, J. Dvorskŷ and V. Snášel, "Using SVM and Clustering Algorithms in IDS Systems," *Proc. of Annu. Int. Workshop, on Databases, Texts Specifications and Objects*, Pisek, Czech Republic, 2011, pp.108-119.

[340] Security Categorization of Federal Information and Information Systems, Federal Information Processing Standards, FIPS 199, Feb. 2004.

[341] Security Self-Assessment Guide for Information Technology Systems, NIST SP800-26, Aug. 2001.

[342] S. Selvakani and R. Rajesh, "Escalate Intrusion Detection using GA - NN," *Int. J. Open Problems Comput. Sci. and Mathematics*, vol. 2, no. 2, pp. 272-284, Jun. 2009.

[343] G. Shafer, *A Mathematical Theory of Evidence*, Princeton Univ. Press, 1976.

[344] B. Shah and B. H. Trivedi, "Artificial Neural Network based Intrusion Detection System: A Survey," *Int. J. of Comput. Appl.*, vol. 39, no. 6, pp. 13-18, Feb. 2012.

[345] A. Shameli-Sendil, N. Ezzati-jivan, M. Jabbarifar and M. Dagenais, "Intrusion Response Systems: Survey and Taxonomy," *Int. J. of Comput. Sci. and Network Security*, vol. 12, no. 1, pp. 1-14, Jan. 2012.

[346] N. Sharma and S. Mukherjee, "A Layered Approach to Enhance Detection of Novel Attacks in IDS," *Int. J. of Advances in Eng. and Technol.*, vol. 4, no. 2, pp. 444-55, Sep. 2012.

[347] M. Sheikhan and M. Sharif Rad, "Misuse Detection Based on Feature Selection by Fuzzy Association Rule Mining," *World Appl. Sci. J.*, vol. 10, pp. 32-40, 2010.

[348] M. Sheikhan and Z. Jadidi, "Misuse Detection Using Hybrid of Association Rule Mining and Connectionist Modeling," *World Appl. Sci. J.*, vol. 7, pp. 31-37, 2009.

[349] B. Sheng, Q. Li, W. Mao and W. Jin, "Outlier Detection in Sensor Networks," *Proc. of the 8th ACM Int. Symp. on Mobile Ad Hoc Networking and Computing*, Montréal, Québec, Canada, 2007, pp. 219-228.

[350] H. Shirazi, "Anomaly Intrusion Detection System Using Information Theory, K-NN and KMC Algorithms," *Australian J. of Basic and Appl. Sci.*, vol. 3, no. 3, pp. 2581-2597, 2009.

[351] B. Sierra, E. Lazakano, I. Inza, M. Merino, L. Larranaga and J. Quirogo, "Prototype Selection and Feature Subset Selection by Estimation of Distribution Algorithms, a

Case Study of Survival in Cirrhotic Patients Treated with TIPS," *Proc. of the 8th Conf. on Artificial Intell in Medicine*, Cascais, Portugal, 2001, pp. 20-29.

[352] L. Sinclair and S. Matzner, "An Application of Machine Learning to Network Intrusion Detection," *Annu. Comput. Security Appl. Conf.*, Phonix, Arizona, 1999, pp. 371-377.

[353] P. Smets and R. Kennes, "The Transferable Belief Model," *Artificial Intell.*, vol. 66, pp. 191-234, 1994.

[354] P. Smets, "Analyzing the Combination of Conflicting Belief Functions," *Inform. Fusion*, vol. 8, no. 4, pp. 387-412, 2007.

[355] P. Smets, "Data Fusion in Transferable Belief Model," *3rd Int. Conf. Inform. Fusion*, PS21-33, Paris, France, 2000.

[356] P. Smets, "Decision Making in the TBM: the Necessity of the Pignistic Transformation," *Int. J. Approximate Reasoning*, vol. 38, no. 2, pp. 133-147, 2005.

[357] P. Smets, "The Application of the Matrix Calculus to Belief Functions," *Int. J. Approximate Reasoning*, vol. 31, no. 1-2, pp. 1-30, 2002.

[358] P. Smets, "The Axiomatic Justification of the Transferable Belief Model," Univ. Libre de Bruxelles, Bruxelles, Belgium, Rep. TR/IRIDIA/1995-8.1, 1995.

[359] P. Smets, "The Transferable Belief Model and other Interpretations of Dempster-Shafer's Model," *6th Annu. Conf. on Uncertainty in Artificial Intell.*, Amsterdam, 1991, pp. 375-383,

[360] P. Smets, "The Transferable Belief Model for Quantified Belief Representation," *Handbook of Defeasible Reasoning and Uncertainty Management Syst.*, vol. 1, pp. 267-301, 1998.

[361] M. Sokolova and G. Lapalme, "A Systematic Analysis of Performance Measures for Classification Tasks," *Inform. Processing and Management*, vol. 45, no. 4, pp. 427-437, 2009.

[362] M. Sokolova, N. Japkowicz and S. Szpakowicz, "Beyond Accuracy, F-score and ROC: A Family of Discriminant Measures for Performance Evaluation," *Proc. of the Australian Joint Conf. on Artificial Intell.*, Hobart, Australia, 2006, pp. 1015-1021.

[363] J. Song, H. Takakura, Y. Okabe, D. Inoue, M. Eto and K. Nakaoa, "A Comparative Study of Unsupervised Anomaly Detection Techniques Using Honypot Data," *IEICE Trans. on Inform. & Syst.*, vol. E93-D, no. 9, pp. 2544-2554, Sep. 2010.

[364] Y. Song, A.D. Keromytis and S. Stolfo, "Spectrogram: A Mixture-of-Markov-Chains Model for Anomaly Detection in Web Traffic," *16th Annu. Network & Distributed Syst. Security Symp.*, San Diego, CA, USA, 2009.

[365] K. Soo Hoo, "How much is Enough? A Risk Management Approach to Computer Security," *Workshop on Economics and Inform. Security*, Berkeley, CA, 2002.

[366] E. Spafford and D. Zamboni, "Intrusion Detection using Autonomous Agents," *Comput. Networks*, vol. 34, n. 4, pp. 547-570, Oct. 2000.

[367] *SSADM-CRAMM Subject Guide*, Central Comput. and Telecommun. Agency, IT Security and Privacy Group, London, 1991.

[368] N. Stakhanova, S. Basu and J. Wong, "A Taxonomy of Intrusion Response Systems," *Int. J. of Inform. and Comput. Security*, vol. 1, no. 1/2, pp. 169-184, 2007.

[369] N. Stakhanova, S. Basu and J. Wong, "Cost-Sensitive Model for Preemptive Intrusion Response Systems," *21st Int. Conf. on Advanced Networking and Appl.*, Niagara Falls, Canada, 2007, pp. 428-435.

[370] D. Stepanova, S. E. Parkin and A. Van Moorsel, "A Knowledge Base for Justified Information Security Decision-Making," Univ. of Newcastle, England, Rep. CS-TR-1137, Feb. 2009.

[371] C. Strasburg, "A Framework for Cost-sensitive Automated Selection of Intrusion Response," M.S. thesis, Dept. of Comput. Sci., Iowa State Univ., Ames, Iowa, USA, 2009.

[372] C. Strasburg, S. Basu, N. Stakhanova and J. Wong, "A Methodology for Evaluating Response Cost for Intrusion Response Systems," *J. of Comput. Security*, vol. 20, no. 2-3, pp. 169-198, 2012.

[373] M. Su, G.J. Yu and C. Y. Lin, "A Real-time Network Intrusion Detection System for Large-scale Attacks based on an Incremental Mining approach," *Comput. and Security J.*, vol. 28, no. 5, pp. 301-309, 2009.

[374] M. Su, K. C. Chang, H.F. Wei and C.Y. Lin, "Feature Weighting and Selection for Real-Time Network Intrusion Detection System based on GA with KNN," *Intell. and Security Informatics Workshops*, 2008, pp. 195-204

[375] X. Su, "An Overview of Economic Approaches to Information Security Management," Univ. of Twente, Enshede, Netherlands, Rep. TR-CTIT-06-30, Jun. 2006.

[376] B. Suh and I. Han, "The IS Risk Analysis based on a Business Model," *Elsevier Sci., Inform. and Management*, vol. 41, no. 2, pp. 149-158, Dec. 2003.

[377] A. Sultana, A. Hamou-Lhaj and M. Couture, "An Improved Hidden Markov Model for Anomaly Detection using Frequent Common Patterns," *IEEE Int. Conf. on Commun., Symp. of Commun. and Inform. Syst. Security*, Ottawa, Canada, 2012, pp. 1113-1117.

[378] Sun Microsystem, Java Development Kit, JDK. Available: www.sun.com, last accessed Sep. 2010.

[379] Symantec, "Internet Security Threat Report," Symantec Enterprise Security, Symantec Corporation, USA, Oct. 2005.

[380] Symantec, Symantec Security Response - Assessing the Severity of Threats, Events, Vulnerabilities, Security Risks, Symantec, USA, Feb. 2006.

[381] K. Tabia, "Modèle Graphiques et Approches Comportementales pour la Détection d'Intrusions," Thèse de doctorat, Univ. d'Artois, Lens, France, 2008.

[382] V. Takkellapati and G. V. Prasad, "Network Intrusion Detection based on Feature Selection and Triangle Area Support Vector Machine," *Int. J. of Engineering Trends and Technol.*, vol. 3, no. 4, pp. 466-470, 2012.

[383] L. Talavera, "An Evaluation of Filter and Wrapper Methods for Feature Selection in Categorical Clustering," *6th Int. Symp. on Intell. Data Analysis*, Madrid, Spain, 2005, pp. 440-451.

[384] X. Tang, C. N. Manikopoulos and S. G. Ziavras, "Generalized Anomaly Detection Model for Windows-based Malicious Program Behavior," *Int. J. of Network Security*, vol. 7, no. 3, pp. 428-35, Nov. 2008.

[385] M. Tavallee, E. Bagheri, W. Lu and A. Ghorbani, "A Detailed Analysis of the KDD Cup 99 Data Set," *IEEE Symp. on Computational Intell. in Security and Defense Appl.*, Ottawa, Canada, 2009.

[386] S. Timms, C. Potter and A. Beard, "Information Security Breaches Survey 2004," PriceWaterHouseCoopers, UK, Rep. Technical report, 2004.

[387] F. Toro-Negro, P. García-Teodoro, J. E. Díaz-Verdejo and G. Maciá-Fernández, "Networking Analysis for Signature Based Intrusion Detection System Methodologies," *Int. Conf. Appl. Computing*, Samalanca, Spain, 2007, pp. 469-473.

[388] T. Toth and C. Kruegel, "Evaluating the Impact of Automated Intrusion Response Mechanisms," *Proc. of 18th Annu. Comput. Security Appl. Conf.*, Washington DC, USA, 2002, pp. 301-311.

[389] D. Tran, M. Ma and D. Sharma, "Automated Feature Weighting for Network Anomaly Detection," *Int. J. of Comput. Sci. and Network Security*, vol. 8, no. 2, pp. 173-179, Feb. 2008.

[390] Q. Tran, H. Duan and X. Li, "One-class Support Vector Machine for Anomaly Network Traffic Detection," *2nd Network Research Workshop of 18th Asia-Pacific Advanced Network*, Cairns, Australia, 2004.

[391] F. Tsai, "Network Intrusion Detection using Association Rules," *Int. J. of Recent Trends in Eng.*, vol. 2, no. 2, pp. 202-204, Dec. 2009.

[392] M. Tuba and D. Bulatovic, "Design of an Intrusion Detection System based on Bayesian Networks," *WSEAS Trans. on Comput.*, vol. 8, no. 5, pp. 799-809, May 2009.

[393] S. Tulyakov, S. Jaeger, V. Govindaraju and D. Doermann, "Review of Classifier Combination Methods," *Studies in Computational Intell.*, 2008, pp. 361-386.

[394] United States Computer Emergency Readiness Team (US-CERT). US-CERT Vulnerability Note Field Descriptions, 2006. Available: http://www.kb.cert.org/vuls/html/fieldhelp, last accessed Jun. 2009.

[395] R. Vaarandi, "Real-time Classification of IDS Alerts with Data Mining Techniques," *IEEE Int. Conf. for Military Commun.*, Boston, 2009, pp. 1786-1792.

[396] R. Vaarandi, "Tools and Techniques for Event Log Analysis," Ph.D. dissertation, Dept. of Comput. Eng., Tallinn Univ. of Technol., Estonia, 2005.

[397] M. Van Loggenberg, "Computer Vulnerability Risk Analysis," M.S. thesis, Dept. of Comput. Sci., Rand Afrikaans Univ., Johannesburg, South Africa, 2003.

[398] P. Vannoorenberghe and T. Denœux, "Likelihood based vs Distance based Evidential Classifiers," *10th Int. Conf. on Fuzzy Syst.*, Melbourne, Australia, 2001, pp. 320 - 323.

[399] V. Vapnik, *Statistical Learning Theory*, Adaptive and Learning Systems for Signal Processing, Communications and Control Series, JohnWiley & Sons, 1998.

[400] G. Vigna and C. Kuregel, "Host based Intrusion Detection," *The Handbook of Inform. Security*, vol. 3, John Wiley & Sons, Dec. 2005.

[401] L. Vokorokos, A. Balaz and M. Chovanec, "Intrusion Detection System using Self Organizing Map," *Acta Electrotechnica et Informatica* , vol. 6, no. 1, pp. 1-6, 2006.

[402] W. Wang, F. Masseglia, T. Guyet and R. Quiniou, "A General Framework for Adaptive and Online Detection of Web attacks," *Proc. of the 18th Int. World Wide Web Conf.*, Madrid, Spain, 2009, pp. 1141-1142.

[403] Y. Wang, *Statistical Techniques for Network Security: Modern Statistically-Based Intrusion Detection and Protection, Inform. Sci. Reference*, Inform. Sci. Reference, IGI Global, 2009.

[404] W. Wanga, X. Guana, X. Zhangc and L. Yanga, "Profiling Program Behavior for Anomaly Intrusion Detection based on the Transition and Frequency Property of Computer Audit Data," *Comput. & Security*, vol. 25, no. 7, pp. 539-550, 2006.

[405] C. Warrender, S. Forrest and B. Perlmutter, "Detecting intrusions using system calls: Alternative data models," *Proc. of the 1999 IEEE Comput. Soc. Symp. on Research in Security and Privacy*, Los Alamitos, CA, 1999, pp. 133-145.

[406] *WEKA Manual for Version 3-6-2*, 1st ed., Univ. of Waikato, Hamilton, New Zealand, 2010.

[407] P. Wennerberg, "Ontology Based Knowledge Discovery in Social Networks," Joint Research Center, European Commission, Rep. T.P.267 I - 21020 Ispra, Apr. 2005.

[408] J. Whitten, L. D. Bentley and V.M. Barlow, *Systems Analysis and Design Methods*, 2nd ed., Richard D. Irwin, 1996.

[409] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, 2ed ed., San Francisco, Morgan Kaufmann, 1999.

[410] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2ed ed., Morgan Kaufmann, 2005.

[411] Y. Wu, H. R. Tseng, W. Yang and R.H. Jan, "DDoS Detection and Traceback with Decision Tree and Grey Relational Analysis," *Int. J. of Ad Hoc and Ubiquitous Computing*, vol. 7, no. 2, pp. 121-136, 2011.

[412] L. Xu, A. Krzyzak and C. Y. Suen, "Methods of Combining Multiple Classifiers and their Applications to Handwriting Recognition," *IEEE Trans. on Syst., Man and Cybernetics*, vol. 22, no. 3, pp. 418-435, 1992.

[413] X. Xu, "Adaptive Intrusion Detection based Machine Learning : Feature Selection, Classifier Construction and Sequential Pattern Prediction," *Int. J. of Web Services Practices*, vol. 2, no. 1-2, pp. 49-58, 2006.

[414] N. Ye and Q. Chen, "An Anomaly Detection Technique based on Chi-square Statistic for Detecting Intrusion into Information Systems," *Quality and Rel. Eng.* Int., vol. 17, no. 2, pp. 105-112, 2001.

[415] N. Ye, "A Markov Chain Model of Temporal Behavior for Anomaly Detection," *Proc. of Workshop on Inform. Assurance and Security*, West Point, NY, 2000.

[416] N. Ye, C. M. Borror and D. Parmar, "Scalable Chi-Square Distance versus Conventional Statistical Distance for Process Monitoring with Uncorrelated Data Variables," *Quality Rel. Eng. Int*., vol. 19, no. 6, pp. 505-515, 2003.

[417] N. Ye, *Secure Computer and Network Systems Modelling, Analysis and Design*, 1st ed., John Wiley & Son, 2008.

[418] N. Ye, T. Ehiabor and Y. Zhang, "First Order versus High Order Stochastic Models for Computer Intrusion Detection." *Quality and Rel. Eng. Int.*, vol. 18, no. 3, pp. 243-250, 2002.

[419] N. Ye, X. Li and S. M. Emran, "Decision Trees for Signature Recognition and State Classification." *Man and Cybernitics Inform. Assurance & Security Workshop*, West Point, NY, 2000, pp. 194-199.

[420] K. Yendrapalli, S. Mukkamala, A. H. Sung and B. Ribeiro, "Biased Support Vector Machine and Kernel Methods for Intrusion Detection," *World Congress Eng.*, vol. 1, pp. 321-325, Jul 2007.

[421] D. Yeung and Y. Ding, "User Profiling for Intrusion Detection using Dynamic and Static Behavioral Models," *6th Pacific Asia Conf. on Knowledge Discovery and Data Mining*, Melbourne, Australia, 1998, pp. 494-505.

[422] C. Yina, S. Tiana and S. Shaomin Mua, "High-order Markov Kernels for Intrusion Detection," *Neurocomputing*, vol. 71, no. 16-18, pp. 3247-3252, Oct. 2008.

[423] J. Yu, H. Lee, M. S. Kim and D. Park, "Traffic Flooding Attack Detection with SNMP MIB using SVM," *Comput. Commun. J.*, vol. 31, no. 17, pp. 4212-4219, 2008.

[424] J. Yu, Z. Li, H. Chen and X. Chen, "A Detection and Offense Mechanism to Defend Against Application Layer DDoS Attacks," *Int. Conf. on Networking and Services*, Athens, Greece, 2007.

[425] L. Zadeh, "On the Validity of Dempster's Rule of Combination of Evidence," Univ. of California, Berkeley, Rep. ERL Memorandum M79/24, 1979.

[426] M. Zaghdoud and M.S. Al-Kahtani, "Contextual Fuzzy Cognitive Map for Intrusion Response System", *Int. J. of Compu. and Inform. Technol.*, vol. 2, no. 3, pp. 471-478, 2013.

[427] D. Zamboni, "Using Internal Sensors for Computer Intrusion Detection," Ph.D. dissertation, Dept. Comput. Sci., Purdue Univ., West Lafayette, IN, 2001.

[428] X. Zang, A. Tangpong, G. Kesidis and D.J. Miller, "Botnet Detection Through Fine Flow Classification," Dept. of Comput. Sci. an Elect. Eng., Pennsylvania State Univ., PN, USA, Rep. CSE-TR11-001, Jan. 2011.

[429] T. Zhang, R. Ramakrishnan and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases," *ACM SIGMOD Int. Conf. of Management of Data*, Montreal, Canada, 1996, pp. 103-114.

[430] R. Zhong and G. Yue, "DDOS Detection System Based on Data Mining," *2ed Int. Symp. on Networking and Network Security*, 2010, pp. 062-065.

[431] L. Zouhal and P. Denœux, "An Evidence Theoretic k-NN Rule with Parameter Optimization," *IEEE Trans. on Syst., Man and Cybernetics*, vol. 28, no. 2, pp. 263-271, 1998.

APPENDICES

Appendix A

Adaptive multimodel analysis process: steps pseudo code

Preliminary security evaluation step

Preliminary security evaluation step: Evaluation(D_t)
Initialization:
D_t ={ $x_{t,j}$, j=1..J}collected and preprocessed log data at time t,
$DP_{j,Q+1}$: set of norm detection models($DP_{j,Q+1} \subset DP_j$) of log type j, j=1,…,J,
O_t: Set of outputs of selected norm detection models, each output $o_{i,j}$ is generated by $M_i \in DM_{j,Q+1}$ when processing data example, $x_{t,j} \in D_{tj}$, $o_{i,j} \in$ {Normal, Anomalous},
Set_{Q+1} : set of selected norm detection models.
Output: CO_t: combined decision of selected norm detection models over J log types
Begin: Evaluation
While (∃ log type)
Sort models of $DM_{j,Q+1}$ based on their relative scores, $sr_{i,j}$
Insert top ranked model M_i to Set_{Q+1}
Evaluate $o_{i,j}=M_i(x_{t,j})$While($\exists M_{i,j} \in Set_{j,K+1}$) do
Update O_t by $o_{i,j}$
Enddo
C$O_t \leftarrow$ Combine(O_t)

End Evaluation
-Combine({d_j}): aggregates decisions of different detection models over considered J log types.

Control of norm relevant features

Norm relevant features control step: FCP-1st step(D_t)
Initialization:
D_t ={ $x_{t,j}$, j=1..J}collected and preprocessed log data at time t,
$U_{j,Q+1}$: subset of norm relevant features of category F^j that exhibit unacceptable changes, $U_{j,Q+1}$=∅,
FNF_t: set flagged norm relevant features subsets over all log types j=1..J,
$F^C_{j,Q+1}$: common relevant features to normal behavior over detection models based on F^j,
$RF_{j,Q+1}$: reference vector of system expected behavior according $F^C_{j,Q+1}$.

Output : FNF_t
Begin: FCP-1st step
While(∃$x_{t,j}$ ∈ D_t) do
While (∃ $f_{,l}$ ∈ $F^C_{j,Q+1}$) do
If(¬verify($x_{t,j}$ (l), $RF_{j,Q+1}$(l)))
Insert flagged feature $f_{,l}$ to $U_{j,Q+1}$
Enddo
Update FNF_t with $U_{j,Q+1}$
Enddo
End FCP-1st step

-Verify(x,y): verify whether x is within an interval centered on y.

Control of attack classes relevant features

Attack classes relevant features control step: FCP-2ed Step ($x_{t,j}$, $U_{j,Q+1}$)
Initialization:
$x_{t,j}$: preprocessed log data instance of type j at time t,
$U_{j,Q+1}$: set flagged norm relevant features of log type j,
$U_{j,q}$: subset of relevant features of category F^j that exhibit changes within control interval based on $RF_{j,q}$of attack c_q, $U_{j,q} = \varnothing$,
U_j^T : set of flagged feature subsets of category F^j over all attack classes in C,
$F_{j,q}^C$: common relevant features to attack class c_q over detection models based on F^j,
$RF_{j,q}$: reference vector of attack class c_q according $F_{j,q}^C$.
U_j: set of flagged abnormal features for log type j over all output classes $c_q \in$ C, $q=1..Q+1$.
Output : U_j
Begin: FCP-2ed Step
While ($\exists\, c_q \in C / q \leq Q$) do
While ($\exists\, f_l \in F_{j,q}^C$) do
If(verify($x_{t,j}$ (l), $RF_{j,q}(l)$))
Insert checked feature f_l to $U_{j,q}$
Enddo
Update U_j^T with $U_{j,q}$
Enddo
Determine U_j using $U_{j,Q+1}$ and U_j^T
End FCP-2ed Step

Security analysis process

Security analysis: security evaluation and feature control processes: E&C(D_t)
Initialization:
D_t ={ $x_{t,j}$, j=1..J}collected and preprocessed log data of J log types at time t
$U_{j,Q+1}$: subset of norm relevant features of category F^j that exhibit unacceptable changes, $U_{j,Q+1}$=∅,
FNF_t: set flagged norm relevant features over all log types j=1..J,
CO_t: combined output of selected norm detection models,
FFS_t: set of flagged feature subsets over all output classes and log data types

Output : FFS_t
Begin: E&C

CO_t ←Evaluation(D_t)
FNF_t←FCP-1st step(D_t)
If($CO_t \neq$ Normal $\vee$ $FNF_t \neq \emptyset$)
While (∃ log type) do
Determine $U_{j,Q+1,}$ the j^{th} subset of FNF_t set
U_j←FCP-2ed step ($x_{t,j}$, $U_{j,Q+1}$)
Update FFS_t by U_j
Enddo
End E&C

Base detection models evaluation, ranking and selection

Base detection models evaluation, ranking and selection: Selection (DP_j, U_j)
Initialization:
DP_j,: subset of detection profiles of models generated using log type j, DP_j={P_i, i=1..$N_{i,j}$}, where each profile corresponds to single detection model Mi, its selected feature set, Fi, relative score, sri,j, and other information, P_i=<M_i, F_i, $sr_{i,j}$ …>
U_j: set of flagged abnormal features for log type j,
K: number of selected detection for each log type,
CS_j: set of candidate detection models to process current log data example of type j, CS_j =∅
$S_{t,j}$: selected combination of detection models to process current preprocessed log data instance of type j.
Output: $S_{t,j}$
Begin: Selection
While(∃ P_i∈ DP_i) do
Determine $\delta_{i,j}$ =F_i∩U_j
If($\delta_{i,j}$≠∅)
Update CS_j with M_i
Evaluate global score $sg_{i,j}$ of M_i
enddo
Sort detection models of CS_j based on their $sg_{i,j}$
Insert K top ranked detection models of CS_j in $S_{t,j}$
Update S_t with $S_{t,j}$
End Selection

Appendix B

Complete list of features of included log types

Table B.1: Intrinsic log type features

Index	Feature name	Description
1	duration	length (number of seconds) of the connection
2	protocol_type	type of the protocol, e.g. tcp, udp, etc.
3	service	network service on the destination, e.g., http, telnet, etc.
4	src_bytes	number of data bytes from source to destination
5	dst_bytes	number of data bytes from destination to source
6	flag	normal or error status of the connection
7	land	1 if connection is from/to the same host/port; 0 otherwise
8	wrong_fragment	number of “wrong” fragments
9	urgent	number of urgent packets

Table B.2: Content log type features

Index	Feature name	Description
1	hot	number of “hot” indicators
2	num_failed_logins	number of failed login attempts
3	logged_in	1 if successfully logged in; 0 otherwise
4	num_compromised	number of “compromised” conditions
5	root_shell	1 if root shell is obtained; 0 otherwise
6	su_attempted	1 if “su root” command attempted; 0 otherwise
7	num_root	number of “root” accesses
8	num_file_creations	number of file creation operations
9	num_shells	number of shell prompts
10	num_access_files	number of operations on access control files
11	num_outbound_cmds	number of outbound commands in an ftp session
12	is_hot_login	1 if the login belongs to the “hot” list; 0 otherwise
13	is_guest_login	1 if the login is a “guest” login; 0 otherwise

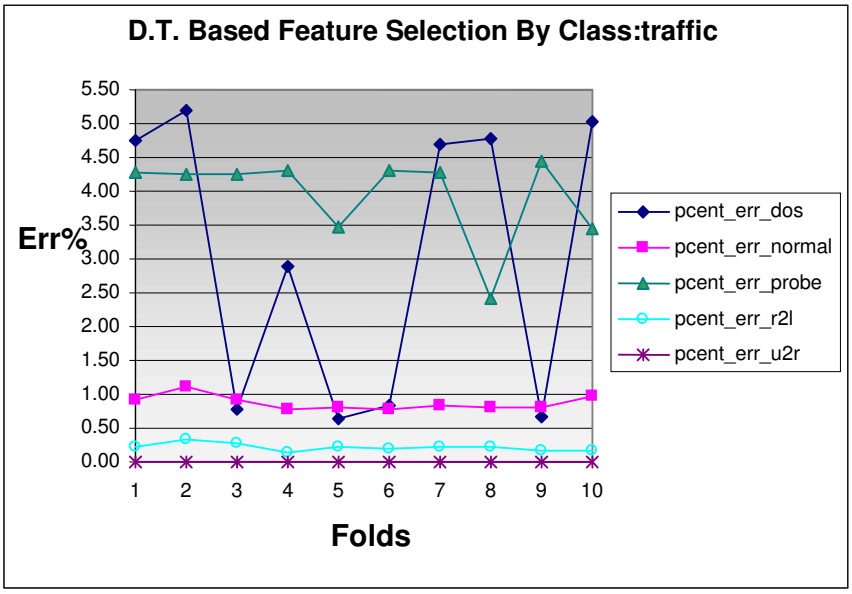
Table B.3: Traffic log type features

Index	Feature name	Description
1	count	number of connections to the same host as the current connection in the past two seconds
2	srv_count	number of connections to the same service as the current connection in the past two seconds
3	serror_rate	Percentage of host connections that have “SYN” errors
4	srv_serror_rate	Percentage of service connections that have “SYN” errors
5	rerror_rate	Percentage of host connections that have “REJ” errors
6	srv_rerror_rate	Percentage of service connections that have “REJ” errors
7	same_srv_rate	Percentage of connections to the same service
8	diff_srv_rate	Percentage of connections to different services
9	srv_diff_host_rate	Percentage of connections to different hosts
10	dst_host_count	Count of connections having same destination host
11	dst_host_srv_count	Count of connections having the same destination host and using same service
12	dst_host_same_srv_rate	Percentage of connections having the same destination host and using same service
13	dst_host_diff_srv_rate	Percentage of different service on the current host
14	dst_host_same_src_port_rate	Percentage of connections to the current host having same src port
15	dst_host_srv_diff_host_rate	Percentage of connections to the same service coming form different hosts
16	dst_host_serror_rate	Percentage of connections to the current host that have an S0 error
17	dst_host_srv_serror_rate	Percentage of connections to the current host and specified service that have an S0 error
18	dst_host_rerror_rate	Percentage of connections to the current host that have an RST error
19	dst_host_srv_rerror_rate	Percentage of connections to the current host and specified service that have an RST error

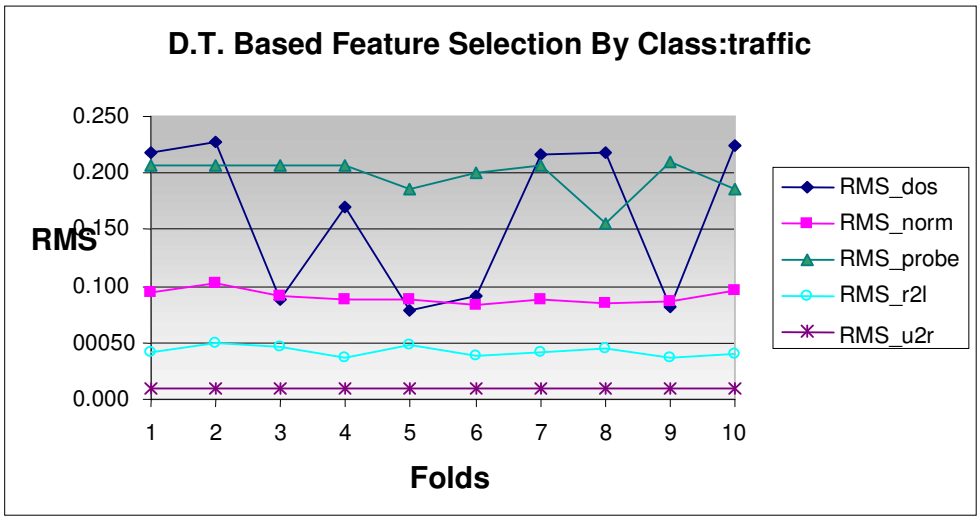
Appendix C

Sample results of DT based feature selection

Detection error of DT based feature selection for different output classes using traffic log type



Root Mean Square Error (RMS) of DT based feature selection for different output classes using traffic log type



Appendix D

Risk driven response component settings

Table D.1: List of supported vulnerabilities and their scores

CVE ID	Vulnerability category*	Vulnerability state**	Global. score	Impact scores		
				C	I	A
CVE-1999-1035	0	R	5.0	0	0	0.275
CVE-1999-0153	0	R	5	0	0	0.275
CVE-1999-0667	0	UR	10	0.66	0.66	0.66
CVE-1999-1199	0	UR	10	0.66	0.66	0.66
CVE-1999-0107	0	UR	5	0	0	0.275
CVE-1999-0016	0	UR	5	0	0	0.275
CVE-1999-0250	0	UR	10	0.66	0.66	0.66
CVE-1999-1504	0	R	5	0	0	0.275
CVE-1999-0116	0	UR	5	0	0	0.275
CVE-1999-0128	0	UR	5	0	0	0.275
CVE-1999-0377	0	UR	5	0	0	0.275
CVE-1999-1423	0	UR	2.1	0	0	0.275
CVE-1999-0513	0	UR	5	0	0	0.275
CVE-2002-1024	0	R	7.1	0	0	0.66
CVE-1999-1321	0	R	5.5	0.275	0.275	0.275
CVE-1999-0223	0	UR	2.1	0	0	0.275
CVE-2004-0230	0	R	5	0	0	0.275
CVE-1999-0015	0	UR	5	0	0	0.275
CVE-1999-0103	0	UR	5	0	0	0.275
CVE-2004-0718	2	R	7.2	0.66	0.66	0.66
CVE-2003-1475	2	R	6.8	0.275	0.275	0.275
CVE-2004-1317	2	R	7.5	0.275	0.275	0.275
CVE-2000-0597	2	UR	7.5	0.275	0.275	0.275
CVE-1999-1333	2	R	7.5	0.275	0.275	0.275
CVE-2000-0034	2	UR	5	0.275	0	0
CVE-1999-0002	2	UR	10	0.66	0.66	0.66
CVE-1999-1298	2	R	7.5	0.275	0.275	0.275
CVE-1999-0527	2	R	10	0.66	0.66	0.66
CVE-1999-200	2	UR	10	0.66	0.66	0.66
CVE-2001-1087	2	UR	7.5	0.275	0.275	0.275
CVE-1999-0005	1,2	R	10	0.66	0.66	0.66
CVE-1999-0009	1,2	R	10	0.66	0.66	0.66
CVE-1999-0260	2	UR	7.5	0.275	0.275	0.275
CVE-1999-0047	0,1,2	R	10	0.66	0.66	0.66
CVE-1999-1321	0,2	R	7.5	0.275	0.275	0.275
CVE-1999-0034	2,3	UR	7.2	0.66	0.66	0.66
CVE-1999-0525	2,3	UR	7.5	0.275	0.275	0.275
CVE-1999-0376	1	UR	4.6	0.275	0.275	0.275
CVE-1999-1317	1	R	4.6	0.275	0.275	0.275
CVE-1999-0975	1	UR	4.6	0.275	0.275	0.275
CVE-1999-0344	1	UR	4.6	0.275	0.275	0.275

CVE-2000-0258	1	R	5	0	0	0.275
CVE-1999-0449	1	UR	7.8	0	0	0.66
CVE-1999-1084	1	UR	4.6	0.275	0.275	0.275
CVE-1999-0027	1	UR	7.2	0.66	0.66	0.66
CVE-1999-0109	1	UR	7.2	0.66	0.66	0.66
CVE-1999-1586	1	UR	7.5	0.275	0.275	0.275
CVE-1999-0509	1	UR	7.5	0.275	0.275	0.275
CVE-1999-0301	1	R	7.2	0.66	0.66	0.66
CVE_1999-0126	1	R	7.2	0.66	0.66	0.66
CVE-1999-0119	3	R	10	0.66	0.66	0.66
CVE-1999-1035	3	UR	10	0.66	0.66	0.66
CVE-1999-0153	3	UR	7.5	0.275	0.275	0.275
CVE-1999-0667	3	UR	5	0	0	0.275
CVE-1999-1199	3	UR	5	0	0	0.275
CVE-1999-0107	3	UR	10	0.66	0.66	0.66
CVE-1999-0016	3	UR	5	0	0	0.275
CVE-1999-0250	3	UR	7.6	0.66	0.66	0.66

*Vulnerability categories:
0: service degradation
1: privilege elevation
2: remote exploit
3: information disclose
**Vulnerability state:
R: publicly resolved flaw
UR: unresolved vulnerability

Setting files for risk treatment using genetic algorithm

```
# --------------------------------
# JGARiskConfig.ini settings
# Created: july 23, 2011
# Updated: August 19, 2011
# --------------------------------
POPSIZE             =  100
MAXGEN              =  500
MUTRATE             =   0.2
CROSSRATE           =   0.8
SEED                =   1
CRITERIA            =   MIN
GENOTYPE            = edu.uniandes.copa.jga.BinaryGenotype
PHENOTYPE           = edu.uniandes.copa.jga.SingleFitnessPhenotype
FITNESSFCTN         = RiskEvalFunction
# defined fitness function for our risk treatment problem
MUTATION            = edu.uniandes.copa.jga.ExchangeBinaryMutation
CROSSOVER           = edu.uniandes.copa.jga.SinglePointBinaryCrossover
```

```
# ---------------------------------
# RiskSettings.ini settings
# Created: july 23, 2011
# Updated: August 19, 2011
# ---------------------------------
SEED         =   1
CONTROLS     =  7
# Genotype size: number of recommended controls for detected attack
BASIC_RISK   = 48996
# Basic risk evaluated by our risk assessment component
TOLERATED_RISK = 15000
SECURITY_BUDGET = 150000
# These two parameters are fixed by information owner and experts
COST_FILE    =  risk122.txt
# Data file that contains required details about 7 included controls
```

```
#-------------------------------------------------
# File: risk122.txt
# Problem name: risk treatment
# Format: control #, control id, effectiveness, cost
#-------------------------------------------------
0 ,A.12.6.1 ,0.7,18500
1 ,A.11.4.5 ,0.7 ,19750
2 ,A.11.4.6 ,0.6 ,18000
3 ,A.11.5.6 ,0.5 ,12000
4 ,A.12.4.1 ,0.7 ,14000
5 ,A.11.5.5 ,0.45 ,1250
6 ,A.11.6.2 ,0.6 ,18500
```