Alert correlation towards an efficient response decision support

Yosra Ben Mustapha

THÈSE DE DOCTORAT CONJOINT TELECOM SUDPARIS et L'UNIVERSITE PIERRE ET MARIE CURIE

Spécialité Informatique et réseaux

École doctorale : Informatique, Télécommunications et Electronique de Paris

Présentée par

Yosra BEN MUSTAPHA

Pour obtenir le grade de
DOCTEUR DE TELECOM SUDPARIS

Corrélation d'alertes : un outil plus efficace d'aide à la décision pour répondre aux intrusions

Soutenue le 30 Avril 2015 devant le jury composé de :

Thèse No : 2015TELE0007

PHD THESIS TELECOM SUDPARIS IN PARTNERSHIP WITH PIERRE ET MARIE CURIE UNIVERSITY

Speciality : Informatics and Networks

Doctoral School : Informatique, Télécommunications et Électronique de Paris

Presented by

Yosra BEN MUSTAPHA

To obtain the degree of
DOCTOR OF TELECOM SUDPARIS

Alert Correlation Towards an Efficient Response Decision Support

Presented on 04 30^{th}, 2015 with the Jury composed by :

Thèse No : 2015TELE0007

I dedicate this work to my parents, the soul of my grandmother, my sister, my brother and my brother in law. I am most grateful to them for their great patience, and encouragement during these three long years.

Abstract

Security Information and Event Management (SIEM) systems provide the security analysts with a huge amount of alerts. Managing and analyzing such tremendous number of alerts is a challenging task for the security administrator. Alert correlation has been designed in order to alleviate this problem. Current alert correlation techniques provide the security administrator with a better description of the detected attack and a more concise view of the generated alerts. That way, it usually reduces the volume of alerts in order to support the administrator in tackling the amount of generated alerts. Unfortunately, none of these techniques consider neither the knowledge about the attacker's behavior nor the enforcement functionalities and the defense perimeter of the protected network (Firewalls, Proxies, Intrusion Detection Systems, etc). It is still challenging to first improve the knowledge about the attacker and second to identify the policy enforcement mechanisms that are capable to process generated alerts.

Several authors have proposed different alert correlation methods and techniques. Although these approaches support the administrator in processing the huge number of generated alerts, they remain limited since these solutions do not provide us with more information about the attackers' behavior and the defender's capability in reacting to detected attacks.

In this dissertation, we propose two novel alert correlation approaches. The first approach, which we call honeypot-based alert correlation, is based on the use of knowledge about attackers collected through honeypots. The second approach, which we call enforcement-based alert correlation, is based on a policy enforcement and defender capabilities' model.

Résumé

Les SIEMs (systèmes pour la Sécurité de l'Information et la Gestion des Evénements) sont les cœurs des centres opérationnels de la sécurité. Ils corrèlent un nombre important d'événements en provenance de différents capteurs (anti-virus, pare-feux, systèmes de détection d'intrusion, etc), et offrent des vues synthétiques pour la gestion des menaces ainsi que des rapports de sécurité. La gestion et l'analyse de ce grand nombre d'alertes est une tâche difficile pour l'administrateur de sécurité. La corrélation d'alertes a été conçue afin de remédier à ce problème.
Des solutions de corrélation ont été développées pour obtenir une vue plus concise des alertes générées et une meilleure description de l'attaque détectée. Elles permettent de réduire considérablement le volume des alertes remontées afin de soutenir l'administrateur dans le traitement de ce grand nombre d'alertes. Malheureusement, ces techniques ne prennent pas en compte les connaissances sur le comportement de l'attaquant, les fonctionnalités de l'application et le périmètre de défense du réseau supervisé (pare-feu, serveurs mandataires, Systèmes de détection d'intrusions, etc).
Dans cette thèse, nous proposons deux nouvelles approches de corrélation d'alertes. La première approche que nous appelons corrélation d'alertes basée sur les pots de miel utilise des connaissances sur les attaquants recueillies par le biais des pots de miel. La deuxième approche de corrélation est basée sur une modélisation des points d'application de politique de sécurité.

Acknowledgement

I would like to express my sincerest gratitude and thanks to my supervisor Pr. Hervé Debar who provided support and guidance. I deeply appreciate his guidance and valuable assistance in the working out and completion of this study. It was an opportunity to work under his direction.

My deepest gratitude goes to Dr. Gregory Blanc for his continuous help and guidance. His presence since the second year of my PhD was of a great importance. Without his motivation and encouragement, I would not have accomplished important and critical steps.

I also wish to express my appreciation to Dr. Grégoire Jacob for his guidance and valuable advice during the first year of my PhD.

I am eternally indebted to my loving parents and all my family members specially my father Béchir, my mother Ahlem, my sister Maha, my brother in law Mehrez and my brother Majdi. They readily and selflessly tried to provide the best conditions, advices and support to avoid stressful periods, to perform my work as better as I can...

I would like to thank my grandmother for her prayers and best wishes for me to success my studies. Your soul still exists around me.

I would also thank my dear Vincent for his love and his motivation. I want to thank his family for their encouragement.

Finally, I would like to thank my dear friends who made the three years of study more joyful. They all know who they are ;-)

Contents

Chapter 1

Introduction

SECURITY ISSUES are becoming increasingly severe to organizations. Thus, it is vital to apply appropriate security measures in order to protect both users and organizations. Security mechanisms are deployed in a network in order to prevent attacks and to react to them when detected. In the literature, multiple security mechanisms have been designed to meet the needs of information systems security. These needs can be summarized by security properties such as availability, authentication, confidentiality, integrity and privacy.

The main impact of cyber attacks is financial. According to a joint study performed by Microsoft Digital Crimes Unit (DCU) and National University of Singapore [1], organizations spend around 500 billion US dollars dealing with security issues and data breaches due to malware. Indeed, this economic aspect represents one of attackers motivations. However, many other motivations exist, in particular accessing confidential data and stealing bank accounts.
In recent decades, attackers have developed techniques and exploits that are increasingly sophisticated and complex. This makes both of intrusion detection and reaction difficult. In addition, considering the existing limitations of these systems (intrusion detection and reaction), the role of the security administrator is becoming more and more challenging.

Security Information and Event Management (SIEM) systems have been developed to assist the security administrator in achieving his role. SIEM combines two distinct aspects: Security Information Management (SIM) and Security Event Management (SEM) [2]. It focuses on:

- logs collection and normalization;
- events analysis and correlation;
- events aggregation and reporting.

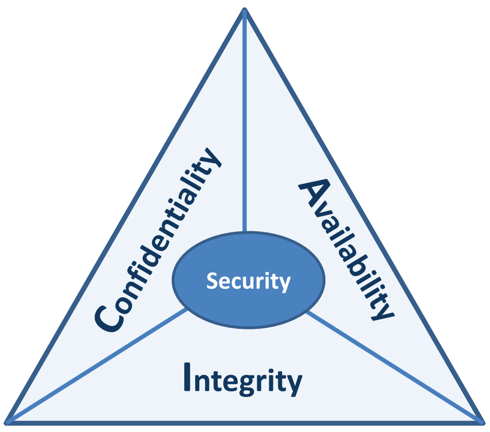


FIGURE 1.1 - Core principles of computer security: Confidentiality, Integrity, Availability (C, I, A).

Tarala [3] shows that SIEM systems are able to provide the data required by organizations in order to maintain and improve their security policy.
We now review basic definitions related to computer security (Section 1.1). Then, in Section 1.2, we give the problem statement, objective and contributions of this dissertation. Finally, the dissertation road-map is detailed in Section 1.3.

1.1 Basic *Definitions*

Research on alert correlation is a domain where many different related domains such as intrusion detection, intrusion response decision, security policy enforcement, are combined. For a better understanding, a consistent terminology is used along this dissertation. In this section, the basic definitions are given, most of them have been taken from [4]. The rest of the definitions can be found in the glossary of terms of this dissertation.

Information security is the protection of information and its critical elements based on three key factors: Confidentiality, Integrity and Availability (C, I, A) (Figure 1.1), as defined by the International Organization for Standardization (ISO) [5] :

- **Confidentiality (C):** it protects the data against unauthorized disclosure. Only authorized users are allowed to access the information;
- **Integrity (I):** it ensures the accuracy and the consistency of the data during its entire life-cycle. It also prevents from unauthorized modifications that can be brought to the information;
- **Availability (A):** it ensures the authorized access to the data when this latter is needed.

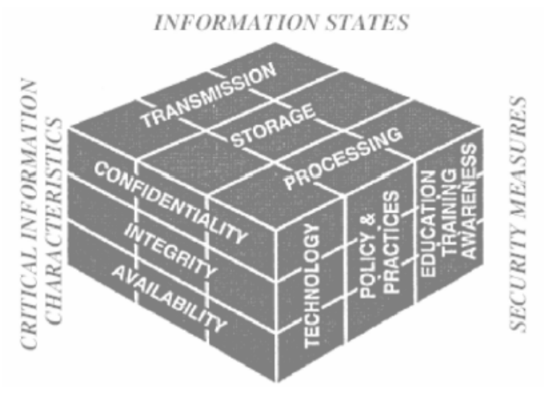


FIGURE 1.2 - Security properties as represented by the McCumber's cube. Source: [6]. This cube contains three blocks: the first block corresponds to the critical information characteristics (C, I, A), the second block corresponds to the information states, and the third block corresponds to the security measures.

Although this core factor of IT security (C, I, A) is still considered as a sine-qua-none for computer security, it is closely related to the protection of information. Due to the rapid growth of IT infrastructures and networks, researchers extended these principles in order to handle contemporary security requirements.
In [6], the author includes additional properties to the (C, I, A) principles. The proposed model in [6] is known as the McCumber's Cube which is shown in Figure 1.2. The McCumber's cube extends the (C, I, A) properties (block in the cube shown in Figure 1.2) with two blocks: *i)* information states (transmission, storage and processing) and *ii)* security measures (technology, policy and practices, and education, training and awareness). The McCumber's cube is proposed in order to adapt computer security to the evolution of technology, networks and attacks.
In information security, a *threat* is a circumstance or event with the potential to adversely impact organizational operations (e.g., mission, function and reputation), as well as organizational assets, individuals, other organizations, and even the Nation through unauthorized access, destruction, disclosure, modification of information, and/or denial of service.
A *vulnerability* is a weakness in one (or more) of the following: information systems, system security procedures, internal controls and software implementation. It can be exploited by attackers.
A *risk* is the level of impact on organizational operations, assets, or individuals resulting from tampering with the operation of the information system. The notion of risk covers two aspects: the potential impact of a threat and the likelihood of such threat occurring.
An *attack* is the instance or the realization of a potential *threat.* An attack is defined as:

- an attempt to gain unauthorized access to system services, resources, information, or;
- an attempt to compromise the integrity of a system.

In an organization, identifying *threats* is usually based on discovering systems' vulnerabilities. Discovered vulnerabilities are generally referred by the MITRE organization [1]. It defines the Common Vulnerabilities and Exposures (CVEs) dictionary, [2], known as CVE-IDs. CVEs are common identifiers for publicly known information-security vulnerabilities in publicly released software packages. In addition, known vulnerabilities are also collected in the Open Source Vulnerability Database (OSVDB) [3]. OSVDB is an independent and open sourced web-based vulnerability which provides in-depth information about technical security information and vulnerabilities.
In addition to threats associated to system's vulnerabilities, threats can be associated with external and physical weaknesses of the information system. For instance, keeping critical servers in an insecure environment is a potential threat which can be exploited in order to have direct access to the server.
To mitigate the effects of a given attack, one needs to implement security measures that are known as *Countermeasures*. *Countermeasures* are security actions required to:

- oppose an attack by eliminating or preventing it [4];
- minimize the harm caused by the attack.

In this dissertation, we assume that the implementation of a countermeasure always results into an update of the security policy's implementation. *A Security Policy*, as defined in [7], defines and constrains the activities of a data processing facility in order to maintain the security of systems and data. Security policies are enforced by *Policy Enforcement Points* (PEP). A PEP is a logical entity that enforces policies for admission control and decisions in response to a request from a user seeking to access a resource or a service on a computer or a network server. For instance, the "Authentication, Authorization, Accounting" (AAA) protocol, [8], is implemented by PEPs such as access control systems (radius servers, network firewalls, application firewalls, etc).

[1]MITRE, `http://www.mitre.org/`
[2]Common Vulnerability Exposure: CVE, `http://www.cve.mitre.org/`
[3]Open Source Vulnerability Database: OSVDB, `http://www.osvdb.org/`

1.2 Problem Statement, Objectives and Contributions

This dissertation proposes two different alert correlation approaches allowing a more precise diagnosis of SIEM alerts.
The first approach, called "honeypot-based alert correlation", is based on improving the content of these alerts by exploiting the knowledge about the attackers. This knowledge can be gathered using specific attack learning techniques which are the honeypots. Since honeypots are capable to interact with attackers, they expose the attacker capabilities and behavior. In addition, honeypots allow reassessment of vulnerabilities by providing information about attacks' impact.
The second approach, called "enforcement-based alert correlation", is based on the knowledge about policy enforcement capabilities deployed in the information system. This knowledge includes information about the defender capabilities which is deployed in the network to be protected from attacks.

In [9], Chapter 3, the author states: *"One who knows the enemy and knows himself will not be endangered in a hundred engagements. One who does not know the enemy but knows himself will sometimes be victorious. Sometimes meet with defeat. One who knows neither the enemy nor himself will invariably be defeated in every engagement."*

Following this citation, the two proposed approaches are complementary, in the sense that the security administrator is able to improve network security by enhancing his knowledge about:

- the attackers: information about the attacker includes details about his behavior and the attack vectors and exploits. This information is convenient for the security administrator to rank the importance of threats exploited by attackers;
- the defender: information about the defender include details about his *capabilities* in enforcing the security policy of the supervised information system. This information is convenient in order to identify the capacity of the defender to mitigate attacks and limit their impacts.

Problem statement :

Current Security Information and Event Management (SIEM) systems process a huge amount of security events without having a better knowledge neither about the attacker nor the defender capabilities.

Thesis statement:

Attack impact's evaluation and efficient response decision support could significantly benefit from a better knowledge about the attacker and the intrusion response capabilities deployed in the network. Improving our knowledge about the attacker allows a better understanding of alerts and then a more efficient alert correlation. Identifying the policy enforcement capabilities - deployed in the network - contributes to apply efficient response decision support.

In order to tackle this problem and validate the thesis statement, the objectives of this dissertation can be defined as follows:

- *Objective 1*: designing and experimenting a novel alert correlation methodology which benefits from information collected by honeypots;

- *Objective 2*: proposing an Alert Correlation methodology based on the proposed PEP's Responsibility Domain;

- *Objective 3*: applying the proposed enforcement-based alert correlation approach on a real case study.

Contributions: The proposed alert correlation approaches developed in this dissertation rely on existing data provided by both honeypots and the configurations of deployed PEPs.
In our "honeypot-based alert correlation" proposed approach, we study the application of attackers' information provided by honeypots in order to correlate alerts. Moreover, since information about attackers is not generally sufficient to improve response decision, we propose an "enforcement-based alert correlation" approach which is based on the policy enforcement capabilities deployed in the network.
The contributions on this dissertation are detailed as follows:

- Identification of appropriate data that can be provided by honeypots and that can improve our knowledge about generated alerts *(Objective 1)*;
- Design of an alert enrichment process which properly improves the information raised in generated alerts and enhances alert correlation *(Objective 1)*;
- Identification of honeypot limitations in improving information about alerts and enhancing alert correlation *(Objective 1)*;
- Design of a PEP model which illustrates the role of a PEP in an intrusion response decision process. Our model is based on the proposed notion of Responsibility Domain of a PEP *(Objective 2)*;
- Development of an enforcement-based alert correlation approach based on the notion of responsibility domain of PEP *(Objective 2)* and the analysis of its application on the security patterns commonly used in security architecture

(Objective 3).

1.3 Dissertation road-map

This dissertation is organized as follows:

Chapter 2 - Computer Security, Intrusion Detection and Response Decision: a State Of the Art describes the state of the art in intrusion detection and response as well as alert representation, honeypot techniques and alert correlation. From the identified limitations of intrusion detection and alert correlation techniques, we propose two alert correlation approaches as introduced above: honeypot-based alert correlation and enforcement-based alert correlation.

Chapter 3 - Alert Correlation based on honeypots introduces the usability of honeypot datasets in order to improve information about generated alerts. In this chapter, we detail our honeypot-based alert correlation. We demonstrate how honeypots can be deployed in order to improve our knowledge about generated alerts. We then propose an alert enrichment process based on honeypot datasets. We detail our experimental results and discuss the different limitations related to the application of honeypots in our proposed alert correlation. Chapter 3 fulfils to *Objective 1.*

Chapter 4 - PEP Model formalizes the proposed PEP model. In this chapter, we show how responsibility domain of a PEP can be approximated. We also propose five approximation methods that can be applied in alert correlation. We evaluate these approximations and analyse their impact on alert correlation. Chapter 4 contributes to *Objective 2*

Chapter 5 - Enforcement-based alert correlation framework defines our proposed enforcement-based alert correlation approach. We investigate the application of the proposed PEP model in correlating alerts. Chapter 5 contributes to *Objective 2*

Chapter 6 - Applications and interpretations shows two use cases on single and multiple PEP environment and demonstrates the applicability of our model in real scenarios. Chapter 6 fulfils *Objective 3*

Chapter 7 - Conclusions and Perspectives concludes the dissertation with a summary of contributions and presents the perspectives for future work.

Chapter 2

Computer Security, IDS and IRS: a State Of the Art

Contents

2.1 Introduction

Security Information and Event Management (SIEM) systems represent today an essential tool to meet the security requirements of complex networks. It supports security administrators in identifying, understanding and reacting to intrusions. Moreover, it contributes to maintaining and enforcing the security policy of the network. Thus, a SIEM strongly depends on Intrusion Detection Systems (IDS), Intrusion Response Systems (IRS) and Policy Enforcement Points (PEP).
In this chapter, we review the most important areas of security research that are related to our proposed approaches. In Section 2.2, we first give an overview of computer security policy and policy enforcement. Then, Section 2.3 and Section 2.4 describe the different intrusion detection techniques: from traditional IDS to recent honeypot tools. Section 2.5 details the Intrusion Detection and Message Exchange Format (IDMEF) which allows a standard representation of security events and alerts generated by IDS and other security probes. In Section 2.6, we review existing alert correlation approaches. Finally, in Section 2.7, we briefly define Intrusion Response Systems.

2.2 Computer security policy and policy enforcement

2.2.1 Computer security policy

Implementing a security service and service arrangement can be complex [10]. The security services defined in Section 1.1 are characterized by their costs and risks. There must be a trade-off between the security services and the organization's security objective. An organization deploys security services depending on the near-term costs and the long-term risks related to the system vulnerabilities and their evolution. The decision maker in this context should study the impact and risk of each potential threat and decide about the security policy to be deployed.
Therefore, the security administrator, when defining a security policy, should be aware of several parameters and constraints. One of the most important constraints are the funding allocated to deploy the security policy, the technical and architectural obstacles, the organizational and personnel issues, etc. The ISO 27001 standard [7] specifies the requirements for establishing, implementing, maintaining

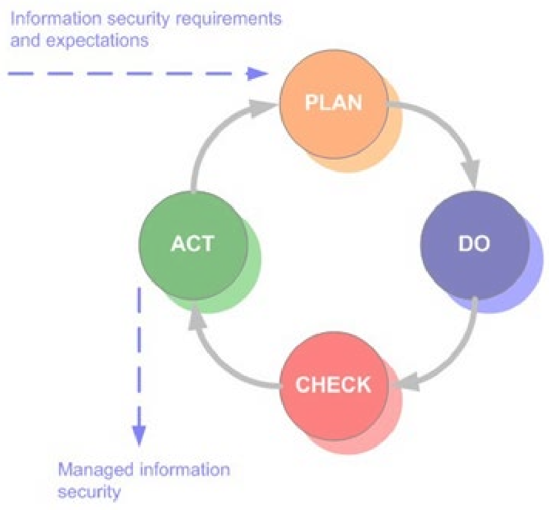

Information security requirements and expectations
PLAN
DO
CHECK
ACT
Managed information security

FIGURE 2.1 - PDCA: Security policy life-cycle, source [7]. It represents the four steps (represented by colored circles) of the security life cycle that are: Plan (orange circle), Do (blue circle), Check (red circle) and Act (green circle).

and continually improving an information security management system within the context of the organization. It also includes requirements for the assessment and treatment of information security risks tailored to the needs of the organization. Thus, it formally specifies the design of an Information Security Management System (ISMS). It defines the most critical steps of the security policy life-cycle which are summarized in the so-called PDCA model shown in Figure 2.1. The PDCA model defines how information security requirements and expectations are processed in order to carry out managed information security. PDCA stands for:

- Plan (P): In this step, the security objectives are identified in accordance with the organization overall policies and objectives. The plan step is depicted by the orange circle in Figure 2.1;
- Do (D): the security policy is implemented in this step following the previously defined objectives. The do step is depicted by the blue circle in Figure 2.1;
- Check (C): it is important to continuously control the conformity of the implemented security policy with the organization's security objectives. In this step, the ISMS is audited and assessed. The check step is depicted by the red circle in Figure 2.1;
- Act (A): this step includes both of the corrective and the preventive actions based on the results of the check step. The objective of the act action is to continuously preserve the security of the organization. The act step is depicted by the green circle in Figure 2.1.

In practice, the PDCA life cycle is established through the deployment of different techniques and mechanisms, known in general as policy enforcement.

2.2.2 Policy enforcement

Policy enforcement is the application of access control mechanisms. It fulfils the security policy objectives along its life-cycle as detailed in Figure 2.1. In particular, check and act actions ensure the security policy enforcement through setting up and updating configurations on Policy Enforcement Points (PEP).
The deployment of modern information systems and networks is strongly associated with Access Control technologies that are located at critical points of the network. Access control technologies are usually provided by Policy Enforcement Points (PEP). The term of PEP was introduced in [11] as an entity that performs access control by making decisions requests and *enforcing* authorization decisions made by the Policy Decision Point (PDP). Referring to the latest version of [11], a PEP comes in many forms. It can be part of a remote-access gateway, part of a web server or email user-agent, etc.
In [12], the PEP is defined as the most security critical component, which protects the resources and enforces the PDP's decision. Generally, the PDP and the PEP are combined to control access and enforce the security policy. According to [13], PEPs are defined as modules which reside on the managed devices and are responsible for installing and enforcing the Security Policy. *"the Policy Decision Points (PDPs) process Access Control policies, along with other data such as network state information, and take policy decisions regarding what policies should be enforced and how this will happen. These policies are sent as configuration data to the appropriate Policy Enforcement Points (PEPs), which reside on the managed devices and are responsible for installing and enforcing them"*. In [14], authors define a PEP as a network entity which is responsible of controlling the traversal of packets across network segments.
In [15], authors specifies that the logical level integration allows the enforcement of policies through a PEP (Policy Enforcement Point). Usually, in applications where multiple architecture is layered, the PEP is located between the presentation layer and the logical layer, intercepting the application requests, using the controls, and in case the application requests is authorized, it can be executed.
As defined in Section 1.1, the PEP can be either a dedicated product (e.g., firewalls, intrusion detection systems), or embedded modules (e.g., patch modules, use of vlans) in larger products or systems.
Since policy enforcement is usually distributed along the different communication layer of the system, we categorize the PEPs based on the communication layer on which it is able to enforce decisions:

- Network-level PEP: iptable, checkpoint, fortinet, snort;
- Application-level PEP: ModSecurity, anti viruses;

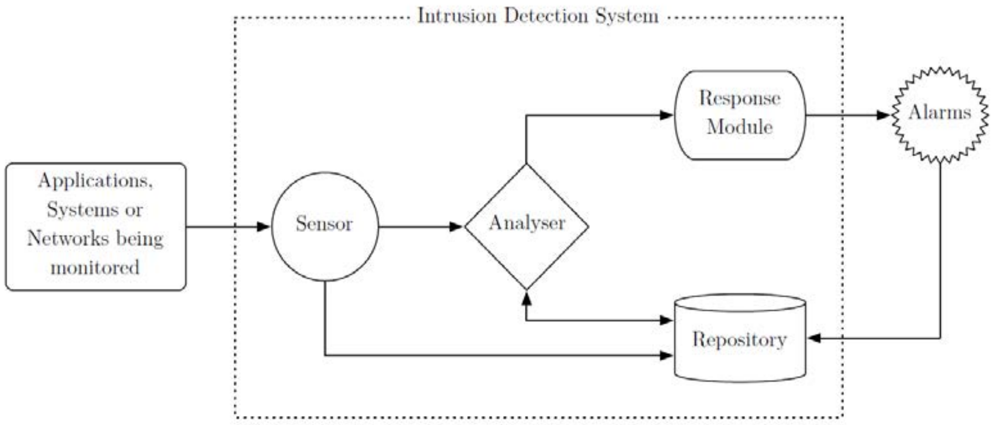

Intrusion Detection System
Applications, Systems or Networks being monitored
Sensor
Analyser
Response Module
Alarms
Repository

FIGURE 2.2 - Basic intrusion detection system architecture, source [19]. It represents the main components of an IDS that are: a sensor, an analyzer, a response module, and a repository

- Information-level PEP: MySQL (security mechanisms of SQL language);
- Identity Access Control PEP: LDAP.

The deployment of these PEPs enforces the security of the supervised system along its different communication layer. PEPs are deployed in a cooperative way in order to satisfy the security policy objectives.

2.3 Intrusion Detection System (IDS)

2.3.1 IDS Definition

Authors in [16] and [17] are the first researchers who tackled the problem of intrusion detection. They defined the foundations of intrusion detection systems. Intrusion detection has a critical role to preserve and control the safety of the organization. In the literature, we may find different intrusion detection definition. Definition given in [18] is the most suitable one where IDS is defined as *the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network.* Figure 2.2 represents a generic architecture of intrusion detection systems. It is composed of: a sensor, an analyzer, a response module, and a repository [19]. Once the sensor have collected recorded events from applications, systems and networks being monitored, it sends events to the analyzer and to the repository. The analyzer is then responsible of qualifying these events (malicious or not malicious). Consequently, it

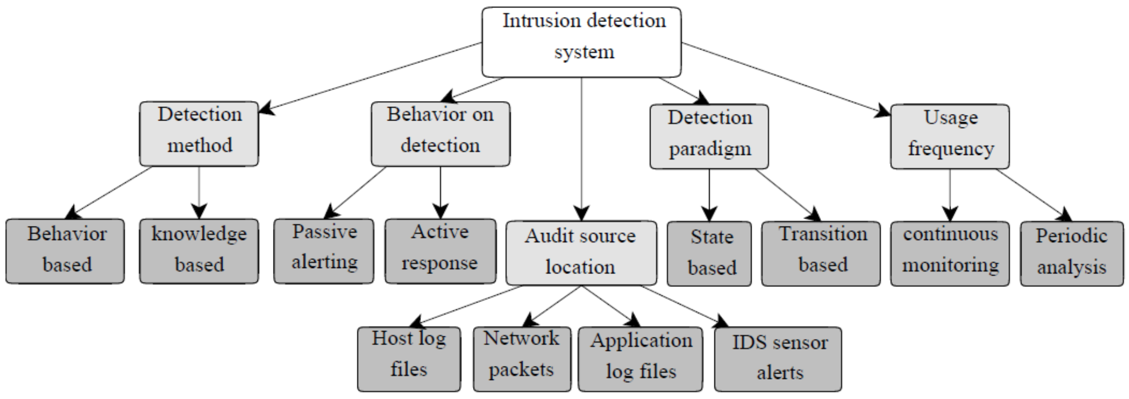


FIGURE 2.3 - Intrusion Detection taxonomy, source [24]. The categorization depends on the detection method, behavior on detection, detection paradigm, usage frequency.

sends information about detected attacks to the Response Module which is in charge of reporting this received information mainly alerts to the human operators.
An IDS can be either centralized or distributed. The former supposes that all monitoring, detection and reporting is controlled directly from a central location. The latter consists in using an agent-based approach for monitoring and detection, so the response decisions are made at the point of analysis.
Henceforth, the main goals of these systems are to identify, preferably in real time, unauthorized use, misuse, and abuse of computer systems by both internal and external accesses. These automated systems are used in blending with the traditional mechanisms of security to yield better security for computer systems. In fact, it can be placed into the control points of the network, for example close to the router, or outside and/or inside the firewall.

2.3.2 IDS Taxonomy

Along years, researchers has been continuously developing new detection techniques and methods. There is a number of existing literature that survey IDS or propose taxonomies [17,20–32]. In [24], authors reviews the existing IDS techniques. Figure 2.3 shows the proposed IDS taxonomy as it is proposed in [24]. The classification of intrusion detection system may be conditioned by the mechanisms of detection used (detection method), the behavior on detection, the nature of the analyzed information (audit source location), and the different operational modes where it is ran (usage frequency):

- Behaviour on detection: According to the kind of the intrusion, the IDS performs the appropriate detection methods. For more details about these types, the interested reader would refer to [17] and [24]. In [24], authors specifies two types of IDSs. The first type is the passive IDSs which only generate alerts when detecting unexpected behavior. These systems do not contribute on attack mitigation since they do not interleave with these activities. The second type is the active IDSs which make possible countering ongoing attacks. Active IDSs are capable to act on the detected threat.

- Audit source location: it refers to where an IDS look for intrusive behavior. Three categories are discerned: host based, network based, and application based.
    - Application-based IDS refers specific applications, using application's transaction logs as main sources of data. It monitors the interaction between user and application, which is not afforded by the other two categories of IDS. Moreover, this system allows the targeting of finer grained activities on the system.
    - Host based IDS monitors and process activities related to a software environment associated with a specific host. It looks also at the communications traffic in and out of a single computer.
    - Network-based IDS monitors the entire network environment. It uses audit trails of multiple hosts and network traffic as main sources of data to identify the intrusion signatures. It performs local analysis of the traffic, via the use of a set of single-purpose sensors placed at various points of the network, then reports attacks to a central management console.
- Detection method: there are two well known approaches of intrusion detection: misuse detection and anomaly detection. In [33], Chapter 2, authors review the existing detection approaches. Hereafter, we detail the most commonly used detection approaches:
    - Misuse detection: is based on signatures of known attacks. It analyzes system activity and audit data stream, looking for events or sets of events that match predefined pattern of events. It relies on a database of attack signature knowledge. Therefore, IDSs using misuse detection are not capable to detect zero-day attacks for which the signatures are not yet published. Moreover, they do not take into account environmental characteristics of the supervised network.
    - Anomaly detection: is based on normal activity profile of a system (e.g., CPU usage, job execution time, system calls, etc). Any action that significantly deviates from the normal behavior is considered intrusive. Depending on the type of the analysis approach, various processing techniques are employed for analyzing data to perform the detection of harmful events as described in [34]. Several anomaly detection approaches has been proposed in the literature such as [24, 35–37]. In contrast to the misuse detection IDS, anomaly detection IDS are capable to detect zero-day attack. But, they require a definition of the legitimate system behavior. This latter

changes over time and makes the anomaly detection IDS obsolete.

2.3.3 IDS Issues

Although IDS provides us with several advantages to enforce existing security mechanisms, researchers in the field on intrusion detection has been continuously working on alleviating limitations of IDSs [38]. In fact, IDSs presents distinct drawbacks related to their detection techniques, attack knowledge, alerts reporting format, etc. One of the most critical IDSs limitations are the alert representation format, false positive rate, huge number of generated raw alerts, etc. Moreover, IDSs usually require frequent updates of their attack knowledge and detection techniques aiming at detecting new attacks and thus decreasing false negative. Thus, the deployment of an IDS, its configuration and its maintenance represents a costly and time consuming task. Hereafter, we detail the aforementioned drawbacks:

Alerts Representation

In [39–42], authors highlight one of the major drawbacks of developed IDSs which is their reporting format. Both commercial (e.g. Corero [1]) and non-commercial (e.g. Snort [2]) IDS solutions use different reporting formats. In general, each IDS solution use its proper alerting format. Heterogeneity of generated alerts format makes challenging cooperation between IDSs. Tenable network security [3] reviews the different alerts and logs representation. It proposes 7 types of logs: Application Execution logs, Authentication logs, DNS logs and Passive Monitoring, Firewall logs, Network Intrusion Detection logs, Netflow and Network Monitoring logs and web server logs.
In order to efficiently analyze attacks, IDSs interoperability becomes a requirement. To alleviate this issue, alert normalization has been proposed by several researcher in the context of alert correlation. These works will be detailed in Section 2.6. The standard Intrusion Detection Message and Excheange Format (IDMEF) has been proposed by Intrusion Detection Exchange Format Working Group (IDWG) in IETF. We detail IDMEF in Section 2.5.

False Positives

Every IDS, independent of its detection technique, generates false positive and false negative alerts. False negative corresponds to non detected attacks and then to

[1]Corero: First Line of Defense, `www.corero.com`

[2]Snort, `www.snort.org`

[3]Log Correlation Engine: Best Practices,
`http://www.tenable.com/whitepapers/log-correlation-engine-best-practices`

non generated alerts. However, false positive corresponds to non malicious behavior reported as malicious. Unless false negative is much more critical for a vendor of IDSs, false positive are more serious from a user's point of view, [43]. In fact, an important number of false positive can hide real attacks and then makes the response decision faulty. For instance, due to false positive, a legitimate user can be blocked and then the service will no longer be available for him.
In [43–47], authors examine the causes of false positive and how can be reduced. In [43, 45], authors presents a study on statistical analysis of false positive and false negative from real traffic with IDSs. They propose techniques to identify false positive. They demonstrated that around 90% of false positive are using HTTP and are due to IDS's policy and not security issue. In [46], the notion of Usual False Positive was proposed in order to reduce false positive in an IDS. In the literature, several false positive reduction approaches has been developed such as [47]. The existing approaches are based on data mining and artificial intelligence techniques, fuzzy logic, machine learning, etc.
In our work, false positive reduction will not be addressed. As a consequence, our work will come on the top of these false positive techniques.

Duplicated Alerts

Duplicated alerts are an IDS issue similar to the false positive in the sense that they overwhelm the security operator when analyzing alerts. Multiplied alerts are due to the limited visibility of the IDS throughout the environment and its contextual parameters. For instance, IDSs do not generally aggregate same alerts related to a same attack. In [48], authors explain the multiplied alerts by the fact that IDS do not take into account the context of the attack. They point out that an attack can cause multiplied alerts over a period of time and they propose a method which is able to eliminate duplicate alerts. In [49], authors proposes an alert aggregation technique based on a dynamic and probabilistic model of the current attack. A meta-alert model was proposed. They proved how their developed approach is able to achieve a reduction rate of up to 99,6%.

Information about Attacks

Another IDS issue which has been pointed in [48] is the semantic of information provided by IDS in generated alerts. Usually, IDS alerts includes low level of information. These events often lack precise and concise information [50]. This fact prevent the security operator from obtaining a consistent view of the ongoing attacks. In order to tackle such issue, alert enrichment techniques has been proposed. In [51], authors exposes the issues of alert's data quality in processing generated alerts. They propose an Intrusion Alert Quality Framework (IAQF) which is responsible of enriching alerts with quality criteria such as correctness, accuracy, reli-

ability, etc. Unless this work attempts to enrich alerts, it was not able to complete knowledge about detected attack. Such knowledge can be gathered from honeypots. Honeypots techniques are detailed hereafter in section 2.4.
Moreover, in order to tackle this issue in particular and IDS issues in general, Security Information and Event Management (SIEM) systems has been developed. It constitutes the central platform of modern security operating centres. They gather events from various sensors (intrusion detection systems, anti-virus, firewalls, etc.), correlate these events, and deliver synthetic views for threat handling and security reporting.

2.4 Honeypot

2.4.1 Honeypot Definition

Discovering and analysing attacker behavior allows better protecting the supervised network and choosing appropriate response decisions. Therefore, honeypots are valuable solution that has been designed to gather and analyze information about attackers.
In [52], the author gives a classical definition of honeypots which affirm that *a honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource.* In [53] and [54], a honeypot consists in an environment where vulnerabilities have been deliberately introduced in order to observe attacks and intrusions and to facilitate in-depth analysis of attackers strategies. It provides its operators with intelligence about threats and network exploits. In recent years, honeypots are proved to be a relevant mean for collecting and analyzing information about attackers behavior. In [55], authors demonstrate how honeypots enable the security administrator to understand the attackers, their techniques, tactics, intentions, and motivations. Honeypots are non-production systems. In fact, they do not belong to any user in the network and do not impact publicly available services.
Honeypots sensors interact with attackers in different ways, classified according to three types of interaction level as listed in [52, 56].

2.4.2 Low-interaction Honeypot

It emulates the presence of different network services on a host rather than a complete system. In general, low-interaction honeypots are based on simple system processes and services. It allows monitoring attack attempts. Unless low-interaction honeypot are useful to detect new attack attempts, they are not capable to emulate more complex environment. HoneyD is an example of low-interaction honeypot [57].

2.4.3 Medium-interaction Honeypot

It is a more sophisticated honeypot than low-interaction one but it is still not emulating a complete operating system like high-interaction honeypots. It emulated a collection of services and more complex operations. Thus, the attacker has more attack entry points. Nepenthens, [58] is an example of medium-interaction honeypots that deploys honeypot modules. It offers a large-scale malware collection. A recent evolution of medium-interaction honeypot research field is proposed in [59] and integrates different tools such as ScriptGen [60], Argos [61] and Nepenthes [58]. In [59], authors propose a distributed system of honeypots in order to gather more details about attacker's activity against several victim machines and networks.

2.4.4 High-interaction Honeypot

Unlike low-interaction honeypot and medium-interaction honeypot, a high-interaction honeypot emulates a complete real system. It usually runs a full implementation of an Operating System and installed applications. It is able to ensure a real environment with which the attacker will interact. This technique gathers more details about the modus-operandi of attackers by allowing a high interactivity level with the attacker. It allows discovering the attacker behavior in order to gain root access in a machine. Capture C [62] is an example of high-interaction honeypot. Argos [61] and Minos [63] and are also other examples of high-level interaction honeypots.

2.4.5 Discussion

Honeypots are still being an active research topic because of their deployment differences and techniques. In Table 2.1, we summarize the pros and cons of these honeypot technologies (in the second and the third column on the table) and we give examples of each category (in the fourth column) and more details about the information collected by them (in the last column).
A common drawback of honeypots is that their field of view is limited and they only gather attacker's activity which interacts with them.
In [64], authors demonstrate how honeypot databases offer significant data to study malware propagation and to get a deeper understanding of their evolution. In this dissertation, we explore such information about the attacker behavior collected by selected honeypots.

Table 2.1 - Honeypot interaction levels. These levels that are low-interaction, high-interaction and medium-interaction are represented in the raw of the table.

Interaction level	**Advantages**	**Disadvantages**	**Examples**	**Collected Information**
Low-interaction honeypot	- simple implementation - easy to use and maintain - low risk of penetration	- emulate specific services - lower interaction performance - no real interaction - limited scope of attacker's activity - detection of known attacks - detectable by advanced attackers - capture only activity that directly interact with them	Specter [65], HoneyD [57]	- limited to the level of emulation - time and date of the attack - protocol - source and destination IP address - source and destination ports
High-interaction honeypot	- full implementation of an Operating System - real interaction with attacker	- complex implementation and maintaining - time-consuming to design - increased risk - capture only activity that directly interact with them	Honeynet [66], Argos [61], Minos [63]	- information about the attacker's behaviour - possible information about zero-day attacks
Medium-interaction honeypot	- simulated services are more complicated - low risk of penetration - better interaction performance	- do not emulate a complete real Operating System - capture only activity that directly interact with them	nepenthes [58], mwcollect [67], honeytrap [68], SGNET [59]	- information about more complex attacks such as botnets, multi-step attacks, etc.

2.5 Representation of Alerts: Intrusion Detection and Message Exchange Format (IDMEF)

Independently from the intrusion detection techniques described above, the aforementioned IDSs generate alerts using different logging format. For instance, for the same suspicious behavior, two IDSs generate two different alerts with the same meaning. This alert format heterogeneity avoid automatic cooperation between IDSs as explained above. The proposed Intrusion Detection and Message Exchange Format (IDMEF) [69] aims at alleviating this problem. Its main purpose is *to define date formats and exchange procedures for sharing information of interest to intrusion detection and response systems and to the management systems that may need to interact with them.* In [70] authors specifies the requirements for IDMEF and also for a communication protocol for communicating IDMEF.
IDMEF ensures both of syntactic and semantic interoperability between IDSs. Syntactic interoperability means that a data generated by two different systems is syntactically analysed with the same manner. Semantic interoperability ensures getting the same meaning of a same data generated by different systems. IDMEF uses DTD XML format in order to represent alerts and uses an object-oriented representation. We detail in Figure 2.4 the alert model as defined in IDMEF.
As depicted in Figure 2.4, IDMEF message is the top-level class of the IDMEF data model. It is composed of two sub-classes: Alert and Heartbeat. Alert class includes information about the detected event by an analyzer. It is essentially composed of 9 main classes. Hereafter, we detail the most significant classes used within the Alert class.

Analyzer : This component analyzes the collected data by sensors in order to detect suspicious activity. Usually, the analyzer and the sensor are included in the same component in existing IDSs. In the alert class, the analyzer field includes information about this component such as its name, version, Operating System (OS) type, etc;

DetectTime : The time when the attack is detected;

Source : It contains information about the possible sources generating the alert. It is represented in Figure 2.4 by black written term. It is composed of four main sub-classes: Node, User, Process and Services;

Target : It contains information about the possible target that are exploited by the detected attack. It is represented in Figure 2.4 with the same form as Source class. Target class is composed of different sub-classes and it reuses the sub-classes of Source class with additional other sub-classes detailed in [69];

Classification : This class provides the name of the alert which usually depends on the deployed IDS product. It includes a reference associated to the alert.

The reference is an information which points to an external documentation detailing the alert.

Assessment : This class informs the security administrator about the analyzer's assessment of the event. For instance, it provides the impact of the detected attack, its confidence level, actions that should be taken in order to respond to the attack, etc.

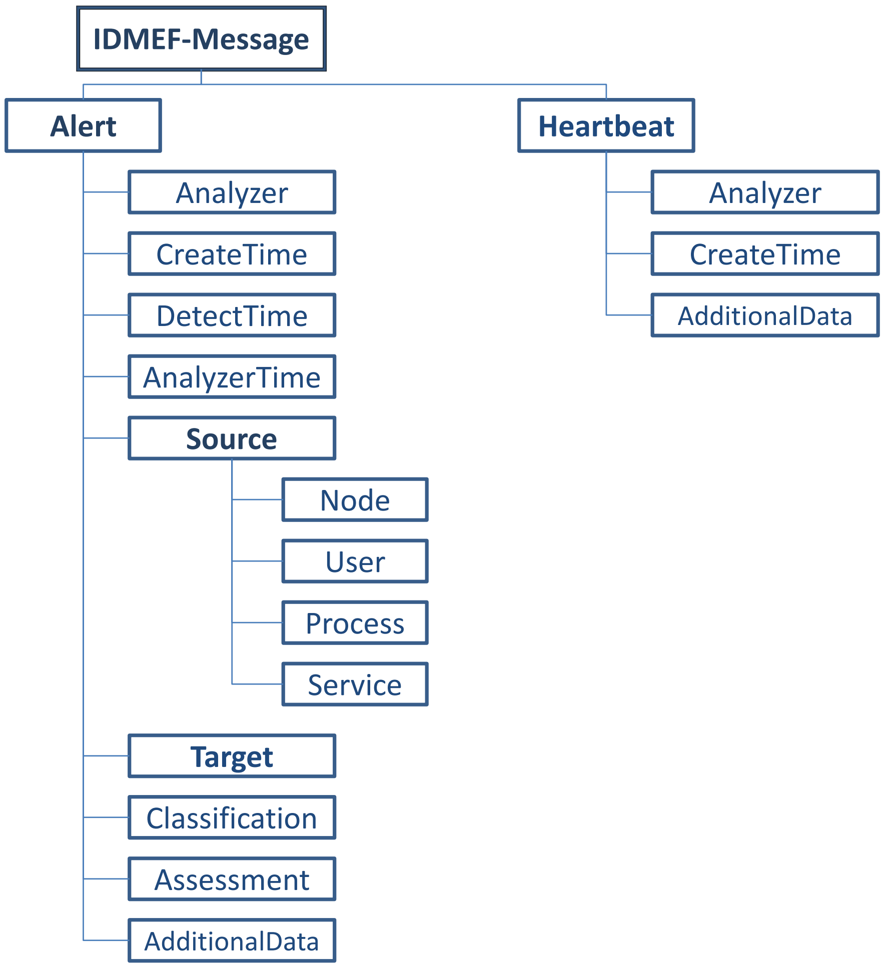


FIGURE 2.4 - IDMEF Alert model. It is composed of two major classes that are the alert class and the heartbeat class. The alert class includes information about the detected attack. The heartbeat class indicates information about the current analyzer's status to managers

2.6 Alert Correlation

2.6.1 Alert Correlation Definition

In practice, IDSs are deployed in a distributed manner. Each single IDS has a limited scope in terms of capabilities and coverage. In fact, an intrusion detection sensor is not able to observe total information about an ongoing attack. This results in two issues:

- high number of generated alerts because of the multiple deployed IDSs. In addition, IDSs generate multiple alerts associated to the same root-cause. This issue prevents the administrator to prioritize the most critical alerts.
- generated alerts include partial information about the ongoing attacks. This issue makes more challenging the enforcement policy decision in order to fill security policy objectives.

Alert correlation has been designed in order alleviate these IDSs issues. To satisfy this aim, it usually verifies the relationships that may exist between alerts following a correlation criteria. Alert correlation approaches have also been developed to improve the accuracy of alerts and attack understanding [71].
In the literature, we may find different correlation techniques. Hereafter, in paragraph 2.6.2, we categorize these techniques following our proposed taxonomy.

2.6.2 Alert Correlation Taxonomy

In [72], we review different Alert Correlation approaches. The main categories of Alert Correlation techniques are similarity-based, knowledge-based which is divided into scenario-based and rule-based approaches, and model-based correlation approaches.

2.6.2.1 Similarity-based Alert Correlation

This technique aims at clustering alerts which have the closest similarity values between alert attributes. Most considered attributes are source and destination IP address and ports as well as timestamp. Distance-based function and probabilistic function, as proposed in [73], are computed to evaluate similarities between alerts. The goal of this technique is to group alerts with high similarity in a new *meta-alert*. Thus, similarity-based techniques are able to correlated alerts while ignoring the attack type.

Discussion

Such correlation approaches cannot entirely satisfy the security administrator's needs in responding to detected attacks. While they are useful to cluster similar alerts, they are not useful to neither evaluate to causality links between alerts nor provide information about alerts' root-cause. First, they do not provide additional information about the attacker. Second, they do not take into account information about the supervised network.

2.6.2.2 Knowledge-based Alert Correlation

Knowledge-based alert correlation is based on a priori knowledge on malicious activities and attacks scenarios and exploits. It requires not only expertise rules and heuristics to correlate heterogeneous alerts but also a consistent information related to alerts. We distinguish between two main sub-categories of knowledge-based techniques:

2.6.2.2.1 Rule-based Alert Correlation Known as prerequisites and consequences, this technique is designed to correlate alerts following their causality relationships. It is a specific technique that serves for scenario-based alert correlation. But, it does not require a prior knowledge of attack scenarios. Rule-based alert correlation approaches take advantage of the JIGSAW attack description language [74] to model the conditions of an attack to occur and its steps.
These latter are constructed once the attack conditions are satisfied. Hence, if some alerts are missed, the attack steps reconstruction remains incomplete. This limitation was addressed by the MIRADOR correlation method which does not require full satisfaction of all attack conditions, [75].

Discussion Rule-based alert correlation approaches explore the causality links between generated alerts. Therefore, the security administrator can adjust the response decision based on detected causality links. He also has the possibility to process rule-based correlated alerts more efficiently. In fact, it is possible to react to these correlated alerts based on the discovered intrusion objective. However, such correlation approaches do not refer to extra knowledge about defense capabilities deployed in the supervised system.

2.6.2.2.2 Scenario-based Alert Correlation A single alert reflects often to an elementary step of an attack scenario. Consequently, various attack scenario templates are required to correlate those alerts. Modelling language such as LAMBDA (Language to Model a Database for Detection of Attacks), [76] and AdeLe (Attack Description Language) have been used for attack scenarios specification. In [76],

authors define a formal attack description language called *LAMBDA*. Respecting LAMBDA language, an attack is characterized by its pre-conditions (pre) and post-conditions (post). Pre-conditions are the conditions satisfied by the computer system and which make the attack feasible. Post-conditions are the consequences of the successful execution of the attack. Using LAMBDA formalism, it is possible to discover ongoing attack scenarios. In fact, an attack scenario is a succession of elementary attacks which aim to achieve an intrusion objective. Two elementary attacks att_1 and att_2 belong to the same scenario if $post(att_1) = pre(att_2)$. $post(att_1)$ and $pre(att_2)$ are respectively the post-conditions and pre-conditions of att_1 and att_2. Similar to [76], in [77], authors propose to correlate alerts based on deduction rules.
Furthermore, attack scenarios recognition techniques were developed like statistical Granger Causality Test, chronicles formalism and other machine learning techniques to enlarge the set of scenarios.

Discussion Within scenario-based alert correlation, alerts are correlated based on predefined scenarios. Thus, such correlation approaches are based on a priori knowledge about the attacker's behavior. Using techniques such as honeypots is a key fact in order to enhance these correlation techniques. In fact, as we explained in Section 2.4, attacker's behavior can be gathered through honeypots. Moreover, scenarios are predefined independent from the supervised system. Thus, such approach may not support the administrator in defining adequate intrusion response decision.

These aforementioned alert correlation techniques are essentially based on the information contained in the alert itself and on static databases including information about vulnerabilities such as the Common Vulnerabilities and Exposure CVE database, National Vulnerability Database, Open Source Vulnerability Database, etc.

2.6.2.3 Model-based Alert Correlation

This approach aims at supporting alert correlation at analysing security alerts with a view on the relevant context. M2D2, [78], was the first attempt to offer a formal representation of sensor capabilities in order to decide whether an alert was false positive or not. This was enhanced by the proposed data model M4D4 [79,80], which covers contextual and topology information. This model is a shared model that is developed in order to process in a cooperative way while correlating alerts. It represents the different relationships among system entities and components to facilitate the correlation process by a cooperative analysis of heterogeneous information.

Discussion Model-based alert correlation are the first correlation techniques that explore additional information about the supervised network. Otherwise, neither information about the attacker's behavior nor the defenser's capability are taken into account when correlation alerts.

2.6.3 Discussion

In Table 2.2, we summarize the advantages and drawbacks of each alert correlation method. Alert correlation methods are listed in the raws of Table 2.2. Advantages and drawbacks are respectively listed in the second and third columns. We remind some examples of correlation technique developed in the literature in the last column of the table. In [81], authors show that the combination of different alert correlation methods is beneficial for effective alert correlation. For instance, similarity-based correlation method could be improved by the knowledge-based correlation method, since this latter are able to correlate alerts based on their causality link.

Table 2.2 - Advantages and drawbacks of alert correlation methods represented in the table's lines.

Alert correlation method	**Pros**	**Cons**	**Examples**
Similarity-based alert correlation	- easy to implement - no need for external knowledge	- unable to discover causality link between alerts	Probabilistic Alert Correlation [73]
Rule-based alert correlation	- alerts are correlated based on their pre-defined causality link - dynamic attack scenario construction	- complex implementation and maintaining of inference rule - time-consuming to design inference rules - need for expert knowledge	Managing alerts in a multi-intrusion detection environment [75]
Scenario-based alert correlation	- alerts are correlated based on their pre-defined causality link	- complex maintaining of defined scenarios - need for a prior knowledge on attack description	LAMBDA [76]
Model-based alert correlation	- consideration of context information and external knowledge	- complex to implement and maintain	M2D2 and M4D4 [78–80]

In our work, we propose complementary approaches that tackle two main drawbacks of alert correlation techniques. The first drawback is the lack of information about the attacker behavior and the detected attack. The second drawback is the role of alert correlation in policy enforcement. To the best of our knowledge, no previous study has proposed solutions to alleviate these issues.

2.7 Intrusion Response Systems (IRS)

Intrusion response system represents the decision tool of an IDS, that takes actions to mitigate attacks and ensures safety in computing systems. It includes a set of countermeasures that aims to protect computer resources. Generally, when speaking about intrusion response, intrusion detection is automatically considered as it gives the main inputs for the response engine (attack type, time, etc.). Intrusion response decisions are actions, devices, procedures, or techniques that meet or oppose (i.e., counters) a threat, a vulnerability, or an attack by eliminating or preventing it, by minimizing the harm it can cause, or by discovering and reporting it so that corrective action can be taken [4].
The focus on the literature shows that several IRSs are developed recently. These systems are found to be lacking in one or more dimensions that make them unsuitable for protecting dynamic and complex distributed systems. Some of the commonly observed shortcomings are:

- the system may have a static mapping of symptoms from the detector to the response;
- it may not take feedback into account for determining future responses;
- it may assume perfect detectors with no missed and no false alarm;
- or it may assume perfect success rate for a deployed response.

As we can point out, intrusion response decision is a challenging step to ensure the security of the supervised system. It comes on top of intrusion detection and alert correlation. Therefore, it is important to design appropriate correlation techniques that can improve the effectiveness of IRS.

2.8 Conclusion

Detection techniques have been extensively studied in the literature. Honeypot techniques are complimentary tool to intrusion detection. They have been developed in order to learn about the attacker's behavior. In our research, we assume that intrusion detection techniques are mature enough to satisfy the need of the security administrator in detecting attacks. We also assume that these IDSs generate IDMEF alerts. Honeypot techniques are still an active research topic. Meanwhile, we assume that they are capable to provide us with appropriate knowledge about attackers' behavior. In this dissertation, we explore the knowledge gathered by honeypots.

Furthermore, we present a definition and a classification of alert correlation methods. We discuss the advantages and drawbacks of correlation techniques. These techniques should be improved by the consideration of attacker behavior knowledge and the policy enforcement capabilities of the supervised network.
The next chapter will be focused on the application of honeypot datasets in alert correlation and how they can enhance the information about generated alerts.

Chapter 3

Honeypot-based Alert Correlation approach

Contents

3.1 Introduction

Existing correlation techniques such as [50,71,73–75,78–80,82] are based on local alert datasets which include locally deployed IDS generated alerts.

The view over the local alert datasets is limited by functional and structural boundaries of the monitored system.
In this context, alert correlation does not provide methods to determine if the alerts, locally detected, are part of more global threat phenomena. Thus, honeypot technology is a valuable instrumentation technique to automatically collect and learn information about server-based exploits and global threat phenomena.
In this chapter, we describe a novel honeypot-based correlation approach capable of analyzing causality relationships between local alerts detected in the monitored system level and the global threat phenomena observed by honeypot sensor deployment. This approach is beneficial to reassess the attack severity and to re-evaluate the attack impact. In fact, analyzing information about the global threat phenomena attributed to the locally detected alerts apprise us of severity of the global threat phenomena, its propagation strategy, its capabilities, etc. Our approach takes advantage of the data collected by four honeypot databases to enrich our knowledge about alerts.

3.2 Framework Description

Recent works [64, 83, 84] have shown the usefulness of gathering experimental data to model and better understand the threats due to attackers. The deployment of honeypots in several locations of the IP space has underlined the fact that different blocks of addresses are attacked differently. Such in-depth information about global attack phenomena makes possible identification of the causality links with local detected attacks.
Our honeypot-based alert correlation approach exploits the information provided by honeypot datasets in order to enhance correlation of local alerts. The most important step is the enrichment of local alerts by the information in honeypot datasets.

Framework workflow

In Figure 3.1, we describe the workflow of our honeypot-based alert correlation approach. The honeypot-based alert correlation process is composed of three main steps:

- Alert enrichment process: The objective of this step is to enrich the content of locally generated alerts by information about the attacker. The output of this step is an enriched alert denoted by *Er_ alert*. This step will be detailed in Section 3.4;
- Analyze Er_Alert: At this stage, the content of Er_alert is analyzed in order to reassess attack severity, re-evaluate attack impact and intrusion objective;

- Correlate Er_Alert: This step aims at correlating enriched alerts using existing alert correlation techniques. It is also possible to combine different correlation techniques as explained in Section 2.6.3. This step results in correlated alerts.

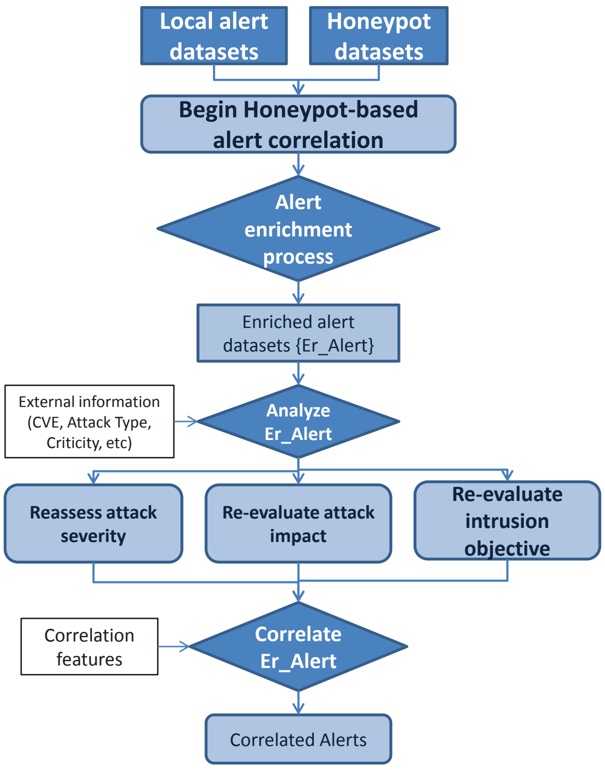

Local alert datasets
Honeypot datasets
Begin Honeypot-based alert correlation
Alert enrichment process
Enriched alert datasets {Er_Alert}
External information (CVE, Attack Type, Criticity, etc)
Analyze Er_Alert
Reassess attack severity
Re-evaluate attack impact
Re-evaluate intrusion objective
Correlation features
Correlate Er_Alert
Correlated Alerts

FIGURE 3.1 - Honeypot-based alert correlation workflow.

3.3 Information Sources

The proposed honeypot-based correlation operates on heterogeneous security-related information gathered from two different views: local view and global view.
Information in the local view comes from analyzers such as Intrusion Detection Systems (IDS), Firewalls, etc. They are deployed to report traces of malicious activity affecting the local network. For instance, IDSs monitor the activity of the network for the occurrence of malicious activities and generate alerts triggered by their signatures. From the alerts, we can retrieve the timestamp of the malicious activity, the source and target IP address, used port, etc. We call these attributes *selectors*.
Information in the global view comes from honeypot data. As we described in paragraph 2.4.1, the primary goal of a honeypot is the study of the attacker behavior while interacting with the sensor. The data collected through honeypot sensor

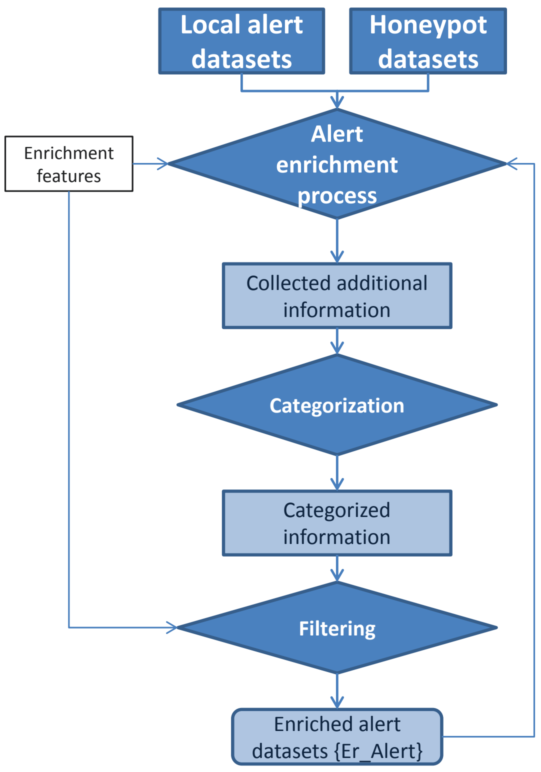


FIGURE 3.2 - Alert Enrichment Process.

includes information about malware behavior, propagation vectors used by malware, the propagation strategy, relationships between exploits, attacker's location, source and destination characterization, origin and relevance of zero-day attacks, etc [56, 59].

3.4 Alert Enrichment Process

In the context of honeypot-based alert correlation, we enrich our local knowledge about alerts with information about the global threat landscape. Hence, it is essential to define an alert enrichment strategy that improves the alert-related knowledge, especially with appropriate external information related to the occurrence of the exploit. Figure 3.2 describes an overview of the alert enrichment process. As shown in figure 3.2, we first *collect* information respecting appropriate *enrichment features* that are detailed hereafter. For instance, based on the originating source of the observed alert, we collect global knowledge about the tracked server and evaluate its security profile and evolution over time. After the enrichment is performed, we *categorize* the collected information following the defined categories presented later and we propose a *filtering* process which is composed of three types of filters: Tem-

poral, Semantic and configuration filters. Each filter performs on a corresponding category of information.
The result of the enrichment process is an *Er_ alert* which appends the elementary alert's information with additional information about the security state of the source, detected attack, in-depth information about corresponding exploits. We then analyze the relationship that may exist between the threat which has been reported on the specific server and the locally reported alerts.
The *Er_ alert* is then processed to request for more general information based on a generalization strategy of the enrichment feature. In fact, honeypots have limited view on the threat landscape. The more honeypots are distributed, the more information about threats phenomena is gathered. Therefore, it is possible to request generalized information in order to gather more knowledge about the attacker's behavior. For instance, to analyze the environment of the originating source of the local threat, it is possible to request information about its localization; the classes of its IP address (class A, B or C).

3.4.1 Enrichment selectors

It is essential to analyze local alert selectors that must be considered in the enrichment process since we manipulate information from two different views: the local view and the global view.
At the local victim-side, Intrusion Detection Systems (IDS)s monitor the activity of the network for the occurrence of malicious activities and generate alerts of detected malicious activity including elementary information about the infection. Following the Intrusion Detection Message Exchange Format (IDMEF) [69], the alert is composed of several aggregate classes. The following aggregation classes can act as selectors for querying honeypot datasets:

Source: class includes the IP address of the originating source of the detected threat. As mentioned in paragraph 2.4.1, honeypot databases log information related to the source IP address of malicious activity.

DetectTime: represents the time when the attack was detected by the local analyzer. In honeypot databases, captured activities are usually timestamped and allow us to analyze the security evolution of the alert's originating source within a specific time window.

Classification: represents the alert semantic. For instance, Intrusion Detection Systems categorize alerts respecting a set of alert classification which inform us about the alert's type. This may be linked to the threat type detected by honeypot sensors.

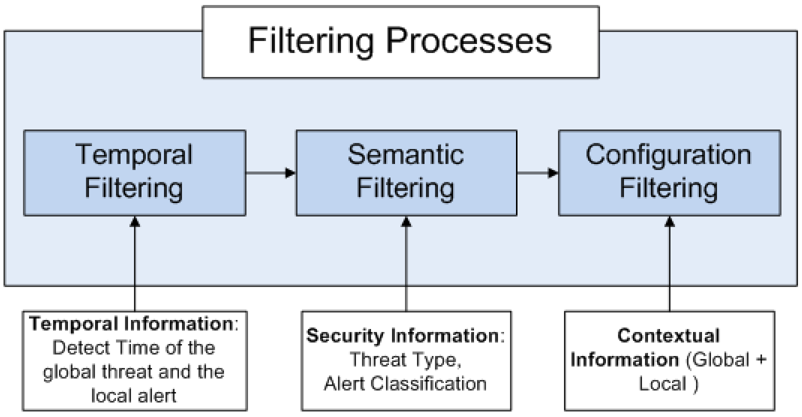

Filtering Processes
Temporal Filtering
Semantic Filtering
Configuration Filtering
Temporal Information: Detect Time of the global threat and the local alert
Security Information: Threat Type, Alert Classification
Contextual Information (Global + Local)

FIGURE 3.3 - Filtering Processes.

3.4.2 Filtering Process

We propose to categorize the collected information aiming at simplifying the filtering processes. We define three major categories:
- *Security Information:* This category includes security-related information such as md5, type of the threat, security states of the originating sources, exploited vulnerabilities by the attacker, etc.
- *Temporal Information :* when detecting a malicious activity, honeypots' logged information is timestamped. Honeypot datasets' objects include temporal attributes about the detection and analysis time.
- *Contextual Information :* this category includes generic type of information: spatial information, whois information, generic information. This set of attributes allows us to build a more general picture about the environment of the object being analyzed.
The objective of the enrichment process is to increase the accuracy of the knowledge related to the alert. To fulfill this need, we set up 3 filters (as shown in figure 3.3). The objective of these filters is to eliminate data which is less likely to link the local alert to the corresponding global threat phenomena.

Temporal Filtering: Honeypot sensors track the activity of attackers and the evolution of the threat landscape referring to a time settings. Honeypot datasets include a timeline of events for each tracked source. This timeline is composed of different time spans defined by the first and last time (resp. t_{first_seen} and t_{last_seen}) of the observed activity generated from tracked source. t_{first_seen} and t_{last_seen} are timestamps expressed by seconds.
We denote by $T\_th_{IPaddr} = [t_{first\_seen}, t_{last\_seen}]$ the time span of an observed threat on a specific source. Within the defined honeypot-based alert correlation approach, it is important to operate on time spans that are close to the *DetectTime* of the local alert. Obviously, it is not significant to attribute

local alerts to a global threat which no longer exists. Therefore, we use sliding windows to avoid investigating the correlation of local alerts with old reported global threats. We denote by $\alpha = \mathit{DetectTime} - t_{last\_seen}$ the delay that exist between the last time of the last activity observed on a source and the detection time of the local alert. We expect α to be of the order of a day.

Semantic Filtering: As local alerts are usually generated by IDS systems referring to attacks' signature, global threat is also assigned to specific types of attack phenomena. By setting up a semantic filtering, we eliminate non-relevant global threat relatively to the classification of the local alert. The objective of semantic filter is to keep only threats that are highly likely causing the local alert.
The observed global threat is classified respecting a high level description of the threat defined by a *threat_class*. This classification is extremely helpful to characterize threats and the modus operandi of attackers behind them. Honeypot datasets use their own set of threat classes to characterize detected global threats. Therefore, the semantic filtering is able to compare the threat class against the alert type with the condition that the threat classes used by honeypots are based on the same dictionary of local alerts classification. The semantic filtering takes into account intermediate cross-references in order to map these different sets of threat classes and alert types.

Configuration Filtering: Cross-view correlation also takes into account the contextual knowledge of the monitored network. This knowledge is composed of topological and cartographic data. It represents hosts' characteristics, their interconnections, software products, their vulnerabilities, etc. As mentioned in [80], comparing the affected configuration of vulnerability with the actual set of products of a given host is valuable for alert correlation.

3.5 Experimentations and Evaluations

Alert enrichment is the most important step in our honeypot-based alert correlation. It is responsible of enriching our knowledge about locally generated alerts with appropriate information gathered through honeypots. In this section, we detail the alert enrichment experimentation results. We then discuss these results and their impact on our honeypot-based alert correlation.

3.5.1 Local Information Source

Our analysis is conducted on real world alerts generated by a snort v2.8 NIDS sensor running on our University Network for 4 months. The University Network is composed of hundreds of machines. For the experiment, we use the latest version

of the signature rule-sets available at the time of the experiment procedure which began on January 2012.
During these 4 months, snort generated 183170 alerts originating by 2499 unique IP address source. We summarize these alerts in Table 3.1, sorted by their classification (detailed in the first column of the table). In the second column of the table, we give the number of generated alerts during these four month for each classification. In the third column, we identify the number of unique IP source addresses in generated alerts. The two last columns show the number of filtered alerts and their corresponding unique IP source addresses. The alert enrichment experimentation is based on these filtered alerts.

Table 3.1 - Classification of alerts generated by the local deployed IDS sensor. In this table, we identify the total number of unique IP sources for both unfiltered alerts and filtered alerts.

Classification	**Number of alerts**	**Total Number of unique IP sources**	**Number of filtered alerts**	**Number of filtered unique IP sources**
attempted-recon	156338	2198	870 (0,5%)	132 (6%)
attempted-dos	1	1	1 (100%)	1 (100%)
attempted-user	1540	28	1540 (100%)	28 (100%)
misc-activity	839	174	23 (3%)	21 (12%)
trojan-activity	3	3	3 (100%)	3 (100%)
bad-unknown	22573	34	51 (0,2%)	9 (26,5%)
unclassified	1876	61	1876 (100%)	61 (100%)
Global	183170	2499	4364 (2,5%)	255 (10%)

Several alerts have been filtered. In fact, in order to increase the performance of the enrichment process, we eliminate several alerts since snort IDS is known by its high rate of false positive as explained in Chapter 2. Hereafter, we consider classifications for which alerts have been filtered:

- **Attempted reconnaissance alerts** are usually preliminary intrusion steps aiming at collecting information about the network. Most of these alerts are, in general [45], alerts for normal network activity. In fact, such alerts are more significant if they are part of a global attack sequence. Therefore, we filter alerts whose IP source address is not present in other alerts having different signature. More than 99% of attempt-recon alerts have been filtered (ref. Table 3.1).
- **Misc-activity and Bad-unknown alerts** include large number of ICMP alerts which can be considered as a Usual False Positive referring to [46]. These alerts are usually generated due to misconfigured hosts, topology of the network,

normal activity of network services, etc. As shown in Table 3.1, more than 99% of misc-activity and bad-unknown alerts have been filtered.

In the last line of the Table 3.1, we compute the number of alerts generated by our Snort sensor and the number of corresponding unique IP source address. We then aggregate the total number of filtered alerts, with its average referring to the totality of alerts. Finally, we give the number of corresponding filtered IP source address that will be considered during our experiments.
Table 3.2 shows more detailed information about the exploited vulnerabilities, protocol and ports reported in the generated alerts. In the second column of Table 3.2, the signature used by the IDS while generating alerts. In the third column, we specify the corresponding vulnerabilities that are identified by their CVE reference [1]. The fourth column specifies the protocol and port used by the detected exploits. In the last column of the table, the number of generated alerts for each signature is given.

3.5.2 Experimental Honeypot Datasets

The experimentation uses four honeypot databases provided by Symantec in the context of the Vis-Sense Project. These datasets include information about malware characterization and security profile of suspicious web-based servers.

3.5.2.1 HARMUR datasets

HARMUR v1, the Historical ARchive of Malicious URLs, [85], is a security dataset that aims at exploring the dynamics of the security and contextual information associated to web-related threats. HARMUR extract several security information tracked web-based servers where hosted suspicious domains. It is possible to retrieve threats that were reported on the tracked server. Like SGNET, HARMUR's developers improve several functionalities of HARMUR and they developed the HARMUR v2.

3.5.2.2 SGNET datasets

SGNET v1, [59], is a distributed honeypot deployment which benefits from different tools, namely ScriptGen [60], Argos [61] and Nepenthes [58]. It collects information about the malware propagation strategies as well as information providing as better understanding about global threat landscape. Recently, an enhanced version of SGNET has been developed and called SGNET v2.

[1]Common Vulnerability Exposure: CVE, `http://www.cve.mitre.org/`

TABLE 3.2 - Summary of reported signatures, their vulnerabilities, protocols and ports.

Classification	**Signature**	**CVE Reference**	**Protocol and Port**	**Nb. of alerts**
attempted-recon	SNMP request tcp	CVE-2002-0012/0014	TCP:80	3
	SNMP AgentX/tcp request	CVE-2002-0013	TCP:80, 31337	2
	SCAN FIN/SCAN SYN FIN	–	TCP	860
attempted-dos	DoS Teardrop attack	CVE-1999-0015	UDP	1
attempted-user	MS-SQL probe response overflow	CVE-2003-0903	UDP:55989, 56538, 41376, 36845, 64439, 4974	1335
	Web-Client Windows Media Player directory traversal via Content-Disposition	CVE-2003-0228	TCP:80	1
misc-activity	ICMP PING CyberKit 2.2 Windows	–	ICMP	20
	BAD-TRAFFIC tcp port 0 traffic	–	TCP:0	3
trojan-activity	BACKDOOR typot trojan traffic	–	TCP:44086	3
bad-unknown	ICMP Source Quench	–	ICMP	50
	DNS SPOOF query response with TTL of 1 min, and no authority	–	UDP:53	1

These honeypot databases have been developed initially within the WOMBAT project. A public API called WAPI (Wombat API) [86], has been developed in order to query information from advanced honeypot databases. Information collected in these datasets is object oriented. The specification of the WAPI protocol relies on four different concepts: objects, attributes, methods and references. Aggregation of these concepts offer information on a security object (e.g. an IP address) that is generated by a set of different datasets (SGNET honeypots, HARMUR web servers, ...).

Table 3.3 - Experimental Results using alert's IP source address. The results shown in this table are based on IP sources of filtered alerts detailed in table 3.1.

Classification	**Number of filtered IP source address**	**SGNET** v1	**SGNET** v2	**HARMUR** v1	**HARMUR** v2
attempted-recon	132	–	–	**1** (SNMP AgentX/TCP)	–
attempted-dos	1	–	–	–	–
attempted-user	28	–	–	–	–
misc-activity	21	–	–	–	–
trojan-activity	3	–	–	–	–
bad-unknown	9	**1** (DNS SPOOF)	–	–	**1** (DNS SPOOF)
unclassified	61	–	–	**1** (tcp portscan) & **1** (tcp portscan,tcp portsweep)	**1** (tcp portscan)

3.5.3 Experimental Results

In our experiments, the enrichment process operates alert attributes such as source address or port. For instance, based on the IP address of the alert's source, the alert enrichment process allows us to gather more details about potential root causes. Additional features would be considered within the enrichment process such as the hash of potentially detected malwares.
During our experiments, we check if the originating source reported in our set of alerts has been analyzed by one of the honeypot sensors. We analyze the security

evolution of alert's source and evaluate and quantify the threat phenomenon which infects the originating source of the local detected alert.

Table 3.4 - Analysis of identified IP source address objects from Table 3.3.

Signature involved	SNMP AgentX/TCP	DNS SPOOF		tcp portscan	tcp portscan & tcp portsweep	tcp portscan
Number of alerts	1	1		4	203 & 103	1
Honeypot Dataset	HARMUR v1	SGNET v1	HARMUR v2	HARMUR v1	HARMUR v1	HARMUR v2
Temporal Information:						
first_seen, last_seen attributes	**unfilled**	**unfilled**	**unfilled**	**unfilled**	**unfilled**	**unfilled**
Security Information:						
Number of hosted Domains	41	–	1	1	2	1
Security current color of hosted domains	28 green, 8 grey, 4 orange & 1 red	–	1 grey	1 green	2 orange	1 red
Number of corresponding threat objects	25	no threat	–	no threat	4	1
Threats rating	14 unknown & 11 red	–	–	–	4 unknown	–
Type of the threat	Virus, Browser Exploit, Generic google safebrowsing	–	–	–	Virus	–
Number of satisfied content reference	**missing**	–	**missing**	**missing**	**missing**	**missing**
number of filled help attribute	13	–	–	–	4	–
information about affected systems	Windows: 98, 95, XP, Me, NT, 2003, 2000	–	–	–	Windows XP, Vista, NT, Server 2003, 2000	–

Table 3.3 summarizes the results obtained during the alert enrichment process. In this table, we represent the number of alerts' IP source address having corresponding server object analyzed by one of the honeypot sensor. The primary step of the enrichment process is concentrated on the identification of the set of alerts that are systematically enriched.
Based on the result shown in Table 3.3, we conclude that a high number of alerts' IP source address have not been analyzed by honeypot sensors. Thus, we conclude that many alerts will not be automatically enriched. Moreover, the examples in the experiment honeypot datasets which are linked to the alerts are difficult to explore in the context of our honeypot-based alert correlation.
In Table 3.4, we detail the information gathered from each IP source address object found in the honeypot datasets.

Enrichment : As shown in Table 3.3, the enrichment process operates on a small set of originating sources. 5 IP source addresses out of 255 IP source addresses have been identified in the experiment honeypot datasets. Due to this reduced number of identified IP source addresses, the performance of the enrichment process is highly reduced. Knowledge about the originating source of the local alerts is not entirely enriched.

Categorization : Experiment honeypot datasets set temporal attributes for each object that are composed of last_seen, first_seen. Moreover, security information includes information about the threat reported in each source, their types, their rating, a link to a more detailed description about the threat, the security color of the hosted domains in the source (e.g. red color means that the domain is infected). In the case of the source identified in SGNET v1, all the attributes and references are missing.

Filtering : Temporal information is the main input for temporal filtering. Normally, this information is of the order of seconds, which is a sufficient granularity in the context of our alert enrichment process. Due to such information in the results, it does not allow the filtering process to be completely performed. A part from this, threat type values (e.g. Virus, Browser Exploit, Generic google safebrowsing) listed in Table 3.4 do not offer a standard representation of the threat type which avoid the semantic filtering to be automatically executed.

3.5.4 Honeypot Limitations in Alert Correlation

Four major limitations have been identified during the experiments. Two limitations are related to the characteristics of a honeypot sensor implementation. In addition, we identify two other limitations which are related to the design of a honeypot datasets data representation.

3.5.4.1 Coverage Limitations

When requesting the corresponding object of a specific originating source of a local alerts, around 95% of locally detected alerts do not have a corresponding originating source reference in the explored databases (as deduced from Table 3.3). The main cause of this result is explained by the interaction level of the honeypot sensors. Even if honeypot sensors are able to interact with attackers and emulate network protocols, they would be unable to cover a large variety of activities like a real interaction between the attacker and the system. For instance, Web-Client Windows Media Player directory traversal via Content-Disposition attacks (cf. CVE from Table 3.2) require a high-interaction level with the attacker and this is not always realized.

Discussion The coverage limitation is also one of the IDS issues as discussed in Section 2.3. Each intrusion detection node has a limited view on the supervised network as shown in [87]. This prevents alert correlation in general and knowledge-based alert correlation in particular of efficiently study the causality link between alerts and then correlate alerts.

3.5.4.2 Unfilled Attributes and references Limitations

An important number of threats have unfilled attributes such as the help attribute which contain details about the threat type, the vulnerabilities that can be exploited, etc. We conduct a statistical evaluation on 73699 threat objects from HARMUR datasets and we observe that over than 85% of these objects are not referenced to a content object [86]. This latter includes in-depth information related to the threat. This pitfall prevent us essentially from applying semantic and configuration filters described in 3.4.2. Moreover, from Table 3.4, due to the absence of temporal attributes; our temporal filtering is not capable to accomplish the filtering process. During our enrichment process, we identify the problem of unfilled attributes of datasets' objects which does not guarantee the automatic reasoning and the continuity of the collection process.

Discussion Consistency of alerts' related information has been the subject of numerous researches in IDS and alert correlation. It has been shown that such related information is necessary for better alert correlation [80]. Several works have proposed data mining to tackle this problem [88]. Unless these techniques have been proved to be efficient in enhancing alerts' content, it becomes more complex when it is jointly applied on both of alerts datasets and honeypots datasets. Moreover, it requires more processing time and effort. Therefore, it impacts the overall quality of our honeypot-based alert correlation.

3.5.4.3 Lack of Standard representation of honeypot data

During our experiments, we observe that information gathered about threats lacks a proper and standard representation. Unfortunately, this limitation makes difficult qualitative evaluation of the exploits and semantic filter. Within HARMUR threats object, each object has a generic threat type as phishing page, virus, browser exploit, etc. Such threat types do not provide clear understanding and in-depth details about the exploit and construct an obstacle for semantic filtering. For instance, even if honyepot datasets report a threat in a specific server, neither the threat type representation nor the help attribute allow us identifying if a causality relationship exist between the locally detected alert and this global threat phenomena.

Discussion We have shown in Section 2.3.3 the importance of security event's representation. In the literature, security databases such as CVE and OSVDB offer a standard representation of vulnerabilities. IDMEF is also a standard format for representing alerts. In our context, the use of standard representation and references allow the design of inference rules in order to facilitate the causality analysis between local and global events. In [89], the use of standard representation and references in security events allow authors developing a cooperative module for alert correlation.

3.5.4.4 Cross-references Limitations

Since honeypot databases lack of cooperation and coordination between them, it was a tedious task to request in-depth information of malware infection reported in a specific web-based server from SGNET that includes information about the malware characterization and behavior. For instance, it would be interesting to gather in-depth information about a threat reported in HARMUR by requesting SGNET. In the context of our approach of honeypot-based alert correlation and enrichment, it is required that we take advantage of a complete global view of the threat phenomena such as the infected servers, their security evolution over time, characterization and specificity of exploits.

3.6 Conclusion

This chapter introduces an application of honeypot databases to enhance alert correlation techniques. We describe a honeypot-based alert correlation that aims at considering external security information collected through the deployment of honeypot sensors in order to enrich the local knowledge of detected intrusions with in-depth details about the security profile of the originating source, exploits to which our network is exposed to, etc. We then analyze and explain experimental results and limitations encountered when dealing with the explored honeypot databases to

ensure the honeypot-based alert correlation. Although honeypot technologies have proved to be a valuable mean to analyze the threat ecosystem, our experiments demonstrate several limitations of the deployment of current honeypot databases to improve alert correlation. For instance, the lack of precise information and standard representation do not ensure correlation analysis and automatic reasoning. This pitfall can be alleviated by conceiving a unified and standardized framework for honeypot data storage and representation. Assigning standard reference such as CVE to observed exploits and threats could be of a great interest while exploring honeypot databases. These suggestions would ensure that honeypot-based alert correlation is executed. It would make the navigation and the cross-references between these different honeypot databases easier.
Honeypot techniques are still evolutionary and can be ameliorated to cover additional security analysis application. In the context of our honeypot-based alert correlation, it is important to ensure concise and complete information that offer to the security analyst a good understanding of threat landscape ecosystem to efficiently identify causality relationships between the local detected alerts and observed threat phenomena in the global view.

Chapter 4

Policy Enforcement Point Model

Contents

4.1 Introduction

Existing correlation approaches such as [50, 71, 73–75, 78–80, 82]) operate independently of the security enforcement capabilities of the network. This is due to the lack of a unified model of policy enforcement mechanisms capabilities. In fact, modeling and identifying the enforcement coverage of each security policy enforcement mechanism is necessary to deploy it effectively when processing alerts and reacting to ongoing attacks.
Policy Enforcement Points (PEP), once deployed and configured, are responsible to apply rules on specific requests. Intrusion response decisions and countermeasures are generally modifications that should be performed on PEP's configuration. Thus, it is important to easily identify the appropriate PEP to set up these modifications and produce the desired effect. In our work, we concentrate on this desired effect. If alerts are correlated based on these capabilities, it would be more efficient to process them.
We propose to model these policy enforcement capabilities in order to not only have a good understanding of deployed policy enforcement capabilities but also tackle several issues in security policy management and intrusion detection. We represent this model by the PEP Responsibility Domain ($RD(PEP)$). The main objective of $RD(PEP)$ is to build a consistent view of the deployed policy enforcement capabilities that contribute in defining the appropriate response decision.
We first give basic definitions that are mandatory to our PEP model. Then, we propose a definition of a *Policy Enforcement Point Responsibility Domain* $RD(PEP)$. Second, we expose several approximation approaches of the RD(PEP). Third, we evaluate the differences between these approximations. Finally, we describe the application of the proposed PEP model on two use cases.

4.2 Definitions and axioms

An essential step to be performed when modeling a concept is detailing the basic definitions and axioms. In this section, we enumerate the basic definitions and axioms that are used in our PEP model.

4.2.1 Definitions

A PEP is a set of enforcement rules denoted by $r_1, r_2, \ldots, r_m$ (m is the total number of rules applied by the PEP). In Equation 4.1, we give an algebraic representation characterizing a PEP.

$$PEP = \{r_j\}_{j \in [1 \ldots m]} \tag{4.1}$$

A PEP is a security entity capable to apply on the received requests, the enforcement decisions represented by $\{d_1, d_2, d_3, \ldots, d_p\}$ (p is the total number of all decisions that can be applied by the PEP class such as AAA servers that can either authorize or refuse access for a specific service or entity [90].).
As defined in access control models [12, 15, 91–94], an access control rule r_j has usually the following general form:

$$\begin{array}{l} r_j : \ \{conditions\}_j \longrightarrow \{decisions\}_j \quad \forall j \in [1, \ldots, m] \\ r_j : \ C_j \longrightarrow D_j \quad \forall j \in [1, \ldots, m] \end{array}$$

In general, C_j is a conjunctive set of condition attributes which we call *Selectors* and denote by $\{S_i\}_{i \in [1 \ldots n]}$. Hence, a rule r_j is represented following Equation 4.2.

$$r_j : \ \underset{i \in [1 \ldots n]}{\times s_i} \longrightarrow \{d_k\}_{k \in [1 \ldots p]} \quad \forall j \in [1, \ldots, m] \tag{4.2}$$

s_i ($i \in [1 \ldots n]$) are instantiation of selectors. In Section 4.3, we define and characterize these selectors.

4.2.2 Axioms

In the rest of this dissertation, the proposed PEP model and enforcement-based alert correlation approach are developed based on the following axioms:

- **Axiom 1** The verification of false positive alerts is out of the scope of our work. In the literature, false positive reduction has been studied by many researchers. It is considered as a complex task. Therefore, we assume that our processed alerts are not false positives, are written following the IDMEF (Intrusion Detection and Message Exchange Format) standard described in [69] and are well formed. This axiom implies that all the IDMEF attributes are instantiated in the processed alerts.
- **Axiom 2** All considered PEPs have a finite set of rules. In practice, security administrators configure on each PEP a finite set of rules which apply the security policy guidelines.
- **Axiom 3** We ignore the default rule of the PEP. Usually, since the default rule includes the entire selectors' domains, it does not inform us about the configuration specification of the PEP. Thus, only configured rules with specific selectors' instantiations are taken into account.

- **Axiom 4** We assume that all the selectors have finite domains. As a consequence, it becomes possible to decompose the selecotrs' domains.
- **Axiom 5** We also ignore the decision/action that can be applied by the PEP. In fact, the PEP' enforcement decisions do not include information about the PEP's responsibility domain. This axiom limits the identification of modifications that should be performed on PEP's configuration and in particular decisions applied by the PEP.
- **Axiom 6** The definition of the Responsibility Domain should only consider the intrinsic characterization and deployment of the PEP. This axiom does not exclude taking into account common domain knowledge that is easily accessible from standard documents, public dictionnaries, etc.

4.3 Selector Characterizations

In this section, we define a basic notion of our proposed PEP model which is the selectors.

4.3.1 Selector Definition

The security policy enforcement is usually based on a set of decision/enforcement criteria known as "Selectors". Referring to axiom 4 listed in Section 4.2, selectors have finite domains. This latter is denoted by $D(S)$. We denote by $\mid D(S) \mid$ the cardinality of D(S).

Selector Type

Each selector has a defined ***Selector Type*** denoted by $S \centerdot Type$ which we define in Equation 4.3.

$$S \centerdot Type = \{(Type(S), D(S))\} \tag{4.3}$$

$Type(S)$ represents the type of the Selector S. It can be for example *integer, real, binary, string, timestamp, etc.* For instance, the selector *IP address* (ip_addr) is an *integer* in $[0, 2^{32}-1]$. We write its type as follow: $ip\_addr \centerdot Type = \{integer, [0, 2^{32}-1])\}$.

Selector Category

Each *Selector* has its role in a Security Policy and its significance. It can identify either the *Source* and the *Target* of the considered flow or the type of the interaction

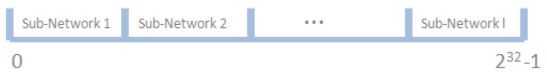


FIGURE 4.1 - Example of src/dst_ip selector taxonomy.

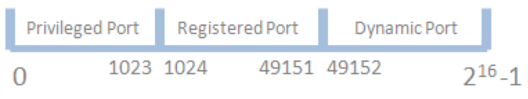


FIGURE 4.2 - Example of src/dst_port selector taxonomy.

between them, mainly the *Action*. We define $S.Category$ the **Selector Category**. We consider two categories that are:

- **identST**: This category of selectors enable us to identify the **S**ource and the **T**arget of an interaction flow. For example, the *ip_ src* and *ip_ dst* are examples of this category. LDAP selectors such as common name *cn* are example of identST selectors.
- **identA**: This category of selectors identifies the type of interaction (*Action*) between the *Source* and *the Target*. For instance the *port_ src*, the *port_ dst* and the *protocol* are selectors that specify the type of the connection between the Source and the Target.

4.3.2 Selector Domain Decomposition

As defined in paragraph 4.3.1, $D(S)$ represents the range of all the possible values which can be affected to Selector S. $D(S)$ can be split into a finite number l of generalized sub-domains denoted by $\delta(S)$ (Eq 4.4). Those sub-domains are totally disjoint.

$$\begin{aligned} D(S) &= \bigcup_{k \in [1 \ldots l]} \delta_k(S) \\ where & \quad \forall k, k' \in [1 \ldots l], \delta_k(S) \cap \delta_{k'}(S) = \emptyset \end{aligned} \tag{4.4}$$

These sub-domains $\delta_k(S)$ identifies specific ranges/elements of selector values. In our approach, we assume the these sub-domains are defined by the security administrator.

For example, the domain of selectors *src_ ip* and *dst_ ip* can be composed of different subset of IP addresses such as the subsets of an IP subnetwork range, as shown in Figure 4.1.

As another example, the domain of selectors *src_ port* and *dst_ port* can be composed of port categories such as privileged ports, registered ports and dynamic and/or private ports, as shown in Figure 4.2. Following Equation 4.4, the domain of ports is written as indicated in Equation 4.5. Such decomposition is defined in the RFC 6335 [95].

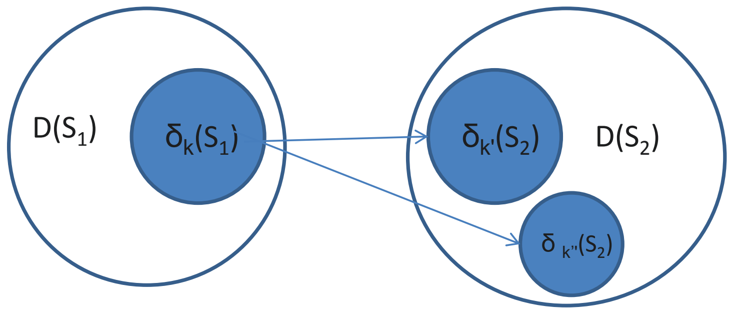
$$\begin{aligned} D(src/dst\_port) &= \bigcup_{k\in[1\ldots 3]} \delta_k(src/dst\_port) \\ &= [0\ldots 1023] \cup [1024\ldots 49151] \\ &\quad \cup[49152\ldots 65535] \end{aligned} \tag{4.5}$$

4.3.3 Selectors Combinability

A selector may have a different meaning and may impact the security policy when used in a combination with other selectors. Therefore, it should not be considered independently while instantiating the Security Policy Rules. Hereafter, we analyze and study the specific relations between selectors and their instantiations. In fact, there exist several functional constraints and ineffective and insignificant combinations. For instance:
i) Selectors with $S\centerdot Category = identA$ have functional constraints such as mutually exclusive sets of values. For example, the *protocol* when it is set to *udp*, it implies the use of a valid sub-domain of $D(port\_src)$ and $D(port\_dst)$.
ii) Selectors with $S\centerdot Category = identST$ have several ineffective and insignificant combinations that may figure between this type of selectors. For example, the use of *wildcard* for both source and destination and combination between the same value for these selectors has no significance.
We define the *Selectors Combinability* Boolean function which we denote by the $SCombinability$ defined in Equation 4.6:

$$\begin{aligned} SCombinability: \quad & D(S_{j_1}) \times D(S_{j_2}) \times \ldots \times D(S_{j_p}) \longrightarrow \{True, False\} \\ & < s_{j_1}, s_{j_2}, \ldots, s_{j_p} > \longmapsto \{True, False\} \\ \text{where} \quad & j_1, j_2, \ldots, j_p \in [1\ldots n] \end{aligned} \tag{4.6}$$

This function verifies the combinability between instantiation of selectors (i.e. $s_{j_1}, s_{j_2}, \ldots, s_{j_p}$). It verifies the **mutually exclusive** values of selectors.
In [96], authors represent a complete analysis of configuration rules aiming at ensuring reliable network security policies. The main objective of the work proposed in [96] is the discovery and the removal of anomalies of configuration rules. This work is based on a prior knowledge about the network topology and architecture. However, in our axiom **6**, we do not take into account this external knowledge in our proposed PEP model. Since we restrict our proposed model to intrinsic characterization of a PEP, we manually define combinable selector sub-domains. We suppose that, for each generalized sub-domain $\delta_k(S_i)$ ($k \in [1\ldots l]$) of a selector domain $D(S_i)$ ($i \in [1\ldots n]$), we identify combinable sub-domains in $D(S_{i'})$ ($i' \in [1\ldots n]$). We denote a combinable sub-domain of $D_{S(i')}$ associated to a generalized sub-domain $\delta_k(S_i)$ ($k \in [1\ldots l]$) by $D_{Comb\_\delta_k(S_i)}(S_{i'})$. In Equation 4.7, we give a representation



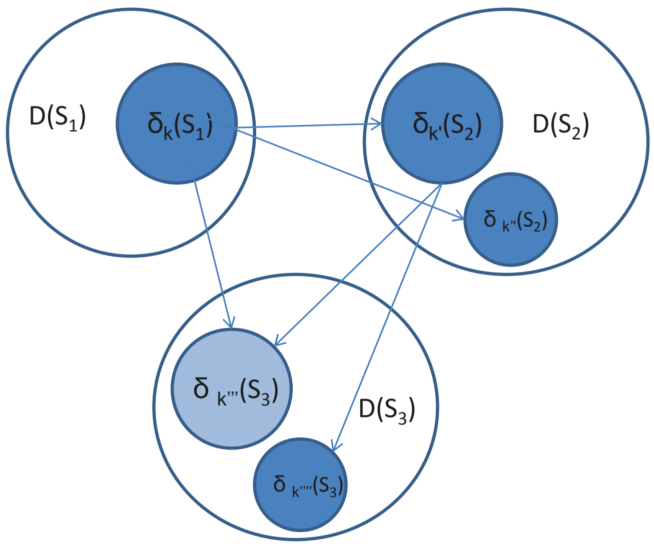


FIGURE 4.4 - Combinability between multiple selectors $D(S_1)$, $D(S_2)$ and $D(S_3)$.

of $D_{Comb\_\delta_k(S_i)}(S_{i'})$.

$$
\begin{array}{l}
\text{for } k \in [1 \ldots l] \text{ and } \forall i,\ i' \in [1, \ldots, n] \\
D_{Comb\_\delta_k(S_i)}(S_{i'}) = \{\delta_{k'}(S_{i'}),\quad k' \in [1 \ldots l']\} \\
\text{such that } \forall \quad s_{i'} \in D_{Comb\_\delta_k(S_i)}(S_{i'}) \text{ and } s_i \in \delta_k(S_i), \\
SCombinability(s_i, s_{i'}) = True.
\end{array}
\tag{4.7}
$$

In Figure 4.3, we give a representation of $D_{Comb\_\delta_k(S_1)}(S_2)$ where we show how the combinability between two selectors S_1 and S_2 is defined. In this case, the sub-domain $\delta_k(S_1)$ of $D(S_1)$ is combinable with $\delta_{k'}(S_2)$ and $\delta_{k''}(S_2)$ of $D(S_2)$. Then we have $D_{Comb\_\delta_k(S_1)}(S_2) = \{\delta_{k'}(S_2), \delta_{k''}(S_2)\}$.
In Figure 4.4, we show how the SCombinability is evaluated for three selectors S_1, S_2 and S_3.
For an instance of selectors $< s_1, s_2, s_3 >$, we evaluate the intersection $D_{Comb\_\delta_k(S_1)}(S_2)$

$\cap D_{Comb\_\delta_{k'}(S_2)}(S_3)$. In the example shown in Figure 4.4, we have $D_{Comb\_\delta_k(S_1)}(S_3)$ $\cap D_{Comb\_\delta_{k'}(S_2)}(S_3) = \{\delta_{k'''}(S_3)\}$.
For multiple selectors S_1, S_2, ... and S_n, in order to have combinable selectors, we should have $D_{Comb\_\delta_k(S_{k_1})}(S_{k_3})$ $\cap D_{Comb\_\delta_{k'}(S_{k_2})}(S_{k_3}) \neq \{\varnothing\}$ for all k_1, k_2, k_3 $\in [1 \ldots n]$.

4.4 Running Example

In order to illustrate our proposed PEP model, we first detail the running example shown in Figure 4.5. It represents a network with two zones (D1 and D2) connected to the Internet and protected by a *Firewall*.

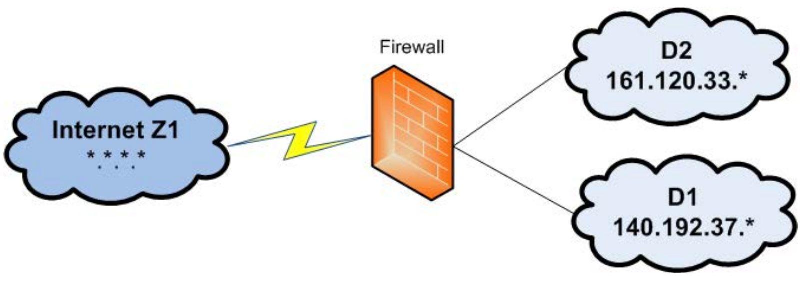


FIGURE 4.5 - Running Example: a network with two zones.

The configuration of this firewall is detailed in Table 4.1. All the HTTP flow is allowed to server 161.120.33.40 in the zone D2 except the traffic originated from the zone D1 (line 1 and 2 in Table 4.1). HTTP and FTP flows are respectively denied from servers 140.192.37.20 and 140.192.37.30 to access zone D2 and the Internet (line 3 and 5). FTP and HTTP flow is allowed from zone D1 to access zone D2 and the Internet (line 4 and 6). DNS flow destined to zone D2 is allowed (line 9). All the udp flow from zone D1 to zone D2 is allowed (line 10). Line 8 and line 11 in Table 4.1 represent the default rule of the firewall which are responsible of blocking all the tcp and udp traffic.

Table 4.1 - Configuration of Firewall in Figure 4.5.

	Source		**Destination**		**Action**
Protocol	**Address**	**Port**	**Address**	**Port**	**Action**
1:tcp	140.192.37.*	*	161.120.33.40	80	deny
2:tcp	*.*.*.*	*	161.120.33.40	80	accept
3:tcp	140.192.37.20	*	*.*.*.*	80	deny
4:tcp	140.192.37.*	*	*.*.*.*	80	accept
5:tcp	140.192.37.30	*	*.*.*.*	21	deny
6:tcp	140.192.37.*	*	*.*.*.*	21	accept
7:tcp	140.192.37.*	*	161.120.33.40	21	accept
8:tcp	*.*.*.*	*	*.*.*.*	*	deny
9:udp	*.*.*.*	*	161.120.33.40	53	accept
10:udp	140.192.37.*	*	161.120.33.*	*	accept
11:udp	*.*.*.*	*	*.*.*.*	*	deny

This example of a PEP can be represented as follow:

$$Firewall = \{r_j\}_{j\in[1\ldots 11]}$$

The configured rules shown in Table 4.1 are represented as follow:

$$r_j : \underset{i\in[1\ldots5]}{\times} s_i \longrightarrow \{d_k\}_{k\in[1\ldots2]} \quad \forall j \in [1,\ldots,11]$$

where:

$S_1 = ip\_src,\ S_2 = port\_src,\ S_3 = ip\_dst,\ S_4 = port\_dst,\ S_5 = p$

$d_1 = \text{accept}, d_2 = \text{deny}$

We remind that selector's domain decomposition is defined by the security administrator as mentioned in section 4.3.2. Therefore, for the running example, the domain of ip source and destination selectors is extracted from the network topology and detailed as follow :

$$\begin{aligned} D(src/dst\_ip) &= \bigcup_{k\in[1\ldots3]} \delta_k(src/dst\_ip) \\ &\quad \delta_1(src\_ip) \cup \delta_2(src\_ip) \cup \delta_3(src\_ip) \\ &= [161.120.33.0/24] \cup [140.192.37.0/24] \\ &\quad \cup[2^{32}-1] \setminus \delta_1(src\_ip) \cup \delta_2(src\_ip) \end{aligned}$$

The domain of port source and destination selectors is extracted from common domain knowledge which is the RFC 6335 [95] and detailed as follow:

$$\begin{aligned} D(src/dst\_port) &= \bigcup_{k\in[1\ldots3]} \delta_k(src/dst\_port) \\ &= [0\ldots1023] \cup [1024\ldots49151] \\ &\quad \cup[49152\ldots65535] \end{aligned}$$

The domain of protocol selector is extracted from common domain knowledge which is the RFC 6335 [97] and detailed as follow:

$$\begin{aligned} D(p) &= \bigcup_{k\in[1\ldots2]} \delta_k(p) \\ &= \{tcp\} \cup \{udp\} \end{aligned}$$

In such example, the SCombinability function allows us evaluate the effectiveness of the configured firewall policy. For instance, a request (in this example, it is a network packet) could not contain the same ip source and destination. Hence, we have $D_{Comb\_\delta_1(src\_ip)}(dst\_ip) = D(dst\_ip) \setminus \delta_1(dst\_ip)$. We also have all values of src_ip selector combinable with all dst_port selector's values. Then, we write $D_{Comb\_\delta_k(src\_ip)}(dst\_port) = D(dst\_port)$ for all $k \in [1, 2, 3]$.
SCombinability could inform us about rules that include same ip source and destination. These rules are not totally effectively and then are over estimated.

4.5 PEP class

We differentiate between PEPs based on the communication layer: Network-level PEP (iptables, juniper, checkpoint, snort ...), Application-level PEP (ModSecurity, Anti viruses, ...), Information-level PEP (MySQL, ...), Identity-Access Control PEP (LDAP, ...). Hereafter, we introduce the notion of a *PEP class.*

Definition 1 ***PEP class:*** *It is a family of PEPs which share common functional characteristics and enforce the policy based on a common (sub)set of selectors.*

Usually, attacks perform actions on a specific communication layer. A PEP class enables the security administrator to identify the capability of a PEP to respond to these attacks.

PEP class properties

Following the Definition 1, each class of PEP is characterized by an identical **core** set of *Selectors* denoted by $\{S_1, S_2, S_3, \ldots, S_n\}$ where n is the total number of selectors characterizing the PEP class. n represents the *Dimension* of the PEP class defined in Equation 4.8. as shown in Equation 4.8.

$$\begin{aligned} Dim(PEP) &= \mid (\{S_j\}_{j\in[1\ldots n]}) \mid = n \\ & \quad where \ \mid \ \mid \ is\ the\ cardinality\ of\ the\ set. \end{aligned} \tag{4.8}$$

In some cases, as shown in figure 4.6, we may have some optional selectors that are not mandatory to characterize the PEP Class. Optional selectors are not necessary

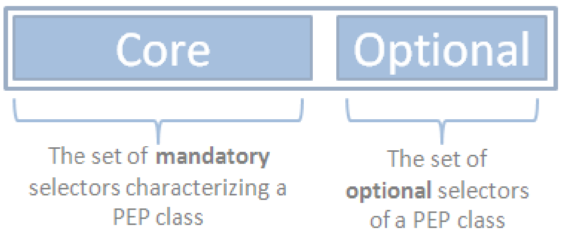

Core
Optional
The set of mandatory selectors characterizing a PEP class
The set of optional selectors of a PEP class

FIGURE 4.6 - Characterization of a PEP Class.

TABLE 4.2 - Network-level Firewall common Selectors Domains.

Selector S	**Meaning**	$S.Type$	**Characterization**
src_ip / dst_ip	source/destination IP address	$\{integer, \subset [0, 2^{32}-1]\}$	Core
src_port / dst_port	source/destination Port	$\{integer, \subset [0, 2^{16}-1]\}$	Core
p	protocol	$\{string, \in \{tcp, udp, icmp\}\}$	Core
tcp_flag	TCP Flag	$\{string, \in \{syn, rst, ack, rst, psh, urg, ecn, fin\}\}$	Optional

to apply the security policy but allow the definition of finer grained enforcement rules.
Example: The Network-level firewall class, denoted by netFW, commonly has a set of five selectors which are { IP source address, source port, IP destination address, destination port, protocol }. The types and domains of these selectors are shown in Table 4.2. The *TCP flag* selector is one of the optional selectors for a netFW class. The *Decision* set of the netFW class is reduced to either Accept (a), i.e. authorize the data flow or Deny (d), i.e. reject it. The dimension of the netFW class is defined as follow:

$$\begin{aligned} Dim(netFW) &= \mid \{src\_ip, src\_port, dst\_ip, dst\_port, protocol\} \mid \\ &= 5 \end{aligned} \tag{4.9}$$

4.6 Responsibility Domain (RD) of a PEP's rule RD(r)

4.6.1 Definition of RD(r)

As defined in Section 4.2.1, a configured rule has usually the following form:

$$\begin{array}{ll} r_j : & C_j \longrightarrow D_j \quad \forall j \in [1, \dots, m] \\ \text{where:} & \\ C_i & = \{r_i(S_j) = s_{ij}, \forall i \in [1 \dots n]\} \\ D_j & \in \{d_1, d_2, d_3, \dots, d_p\} \end{array}$$

For each request vector, the access control rule r_i, $\forall\ i \in [1 \dots m]$ applies the decision D_i when conditions on selectors C_i are satisfied. A rule can apply several decisions such as denying and logging. Every rule r_i defined in the configured PEP is *responsible* of enforcing the decision on the corresponding set of flows.

Definition 2 ***Responsibility Domain of a rule :*** *It represents the Applicability Domain of the rule. It is the count of all the instantiated selectors combinations. It is based on the set of C_i configured for each* Selector *of the PEP. It includes all the requests on which the rule's decision(s) may be applied and enforced. We denote it by $RD(r_i)$.*

We write the Responsibility Domain of a rule $RD(r_i)$ as follow:

$$\begin{aligned} RD(r_i) &= < C_i >, \forall i \in [1 \dots m] \\ &= < s_{ij} >_{j \in [1 \dots n]}, \forall i \in [1 \dots m] \end{aligned} \tag{4.10}$$

Since $s_{ij} \subseteq D(S_j)$, one rule may include multiple selectors combination. We define hereafter the $RD(r_i)$ coverage.

Consequence 1 *$RD(r_j)$* ***Coverage:*** *$RD(r_i)$ Coverage is the number of all the selectors combination defined in the rule r. It is expressed in Equation 4.11.*

$$\mid RD(r_i) \mid \ = \prod_{j \in [1 \dots n]} \mid s_{ij} \mid \tag{4.11}$$

Example Consider the following rule of a Network-level Firewall:

$$\begin{aligned} r:\ & src\_ip = 140.192.37.* \wedge src\_port = * \wedge \\ & dst\_ip = 161.120.33.40 \wedge dst\_ip = 80 \wedge protocol = tcp \\ & \quad \rightarrow deny \end{aligned} \tag{4.12}$$

Its corresponding Responsibility Domain is:

$$RD(r) \ =< 140.192.37.*, *, 161.120.33.40, 80, tcp > \tag{4.13}$$

and the coverage of $RD(r)$ is:

$$\begin{aligned} \mid RD(r) \mid \ &=\mid 140.192.37.* \mid, \mid * \mid, \mid 161.120.33.40 \mid, \mid 80 \mid, \mid tcp \mid> \\ &= (2^8 - 1) \times (2^{16} - 1) \times 1 \times 1 \times 1 \end{aligned} \tag{4.14}$$

4.6.2 Characterization of relations between Responsibility Domain of rules

In [98], authors define five relations that may exist between rules. They demonstrate that these relations are unique and that can be applied to define the different conflicts and anomalies that may figure between rules.
We adopt these relationships and define them between Responsibility Domain of rules. Overlaps between rules result in overlaps between their Responsibility Domains. Hereafter, we detail the relationships that may exist between the Responsibility Domains of rules.

- $RD(r_1)$ and $RD(r_2)$ are ***Completely Disjoint*** (CD) and we write $CD(RD(r_1), RD(r_2))$, iff

$$\begin{array}{rl} \forall j \in [1 \ldots n], & r_1(S_j) \not\trianglerighteq r_2(S_j) \\ where & \trianglerighteq \in \{\subset, \supset, =\} \end{array} \tag{4.15}$$

- $RD(r_1)$ and $RD(r_2)$ are ***Exactly Matched*** (EM) and we write $EM(RD(r_1), RD(r_2))$, iff

$$\forall j \in [1 \ldots n],\ r_1(S_j) = r_2(S_j) \tag{4.16}$$

- $RD(r_1)$ and $RD(r_2)$ are ***Inclusively Matched*** (IM) and we write $IM(RD(r_1), RD(r_2))$, iff

$$\begin{array}{ll} \forall j \in [1 \ldots n], & r_1(S_j) \subseteq r_2(S_j) \\ and\ \exists j'\ such\ that & r_1(S_{j'}) \neq r_2(S_{j'}) \end{array} \tag{4.17}$$

- $RD(r_1)$ and $RD(r_2)$ are ***Partially Matched*** (PM) and we write $PM(RD(r_1), RD(r_2))$,iff

$$\begin{array}{rl} \exists j',\ j'' \in [1 \ldots n], & r_1(S_{j'}) \trianglerighteq r_2(S_{j'}) \\ & r_1(S_{j''}) \not\trianglerighteq r_2(S_{j''}) \\ where & \trianglerighteq \in \{\subset, \supset, =\} \end{array} \tag{4.18}$$

- $RD(r_1)$ and $RD(r_2)$ are ***Correlated*** (C) and we write $C(RD(r_1), RD(r_2))$,iff

$$\begin{array}{rl} \forall j \in [1 \ldots n], & r_1(S_j) \trianglerighteq r_2(S_j) \\ and\ \exists\ j',\ j''\ \in [1 \ldots n] & such\ that\ r_1(S_{j'}) \subset r_2(S_{j'}) \\ & and\ r_1(S_{j''}) \supset r_2(S_{j''}) \\ where & \trianglerighteq \in \{\subset, \supset, =\} \end{array} \tag{4.19}$$

In [98], these relationships are used in order to detect conflicts which are a topic out of the scope of our work. Different from [98], these relationships are used in order to define approximation inferences that will be detailed in Section 4.7.

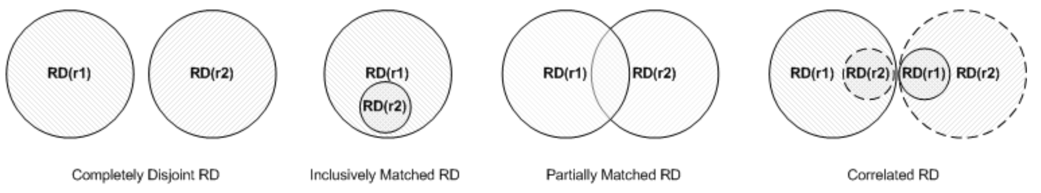


FIGURE 4.7 - Relations between Responsibility Domains of two rules.

4.7 Responsibility Domain of PEP ($RD(PEP)$)

Definition 3 ***Responsibility Domain of PEP:*** *Each PEP, once deployed in the network, has a finite range of applicability which we call "Responsibility Domain". We denote it by $RD(PEP)$. This domain is an abstraction over the PEP implementation and configuration and its intrinsic enforcement capabilities.*

The Responsibility Domain of the PEP informs us about the enforcement coverage of the PEP across the network.

Definition 4 *The Responsibility Domain is multi dimensional domain and its dimension is $Dim(PEP)$.*

Following Definition 4, $RD(PEP)$ is a bounded multi-dimensional domain and we identify two types of $RD(PEP)$ bounds:

- The first type of bounds considers specifically the $S.Types$ and it is an intrinsic characteristic of the PEP class. We respectively denote the minimum bound and the maximum bound of this first type by $RD_{Min}(PEP)$ and $RD_{Max}(PEP)$. $RD_{Min}(PEP)$ is the lower bound of the first type. It represents the minimum number of selectors to be used in order to implement effective rules. $RD_{Max}(PEP)$ is the upper bound of the first type. It includes the entire set of selectors characterizing the PEP class.
- The second type of bounds is related to the instantiation of the PEP class. We respectively denote by $RD_{inf}(PEP)$ and $RD_{sup}(PEP)$, the inferior bound and the superior bound of the $RD(PEP)$ once the PEP is deployed.

  Definition 5 *$RD_{inf}(PEP)$ is the union of the entire set of Responsibility Domains of configured rules in the PEP's instantiation.*
  $RD_{sup}(PEP)$ considers environmental constraints on the deployed PEP. The identification of this bound requires external knowledge related to the topological visibility of the deployed PEP.

Since we assume in axiom 4 the non availability of such knowledge, $RD_{sup}(PEP)$ will not be identified in this work.

As policy enforcement is, in most cases, distributed along the different PEPs, it is important to model their enforcement coverage, $RD(PEP)$, in order to support the administrator in selecting the most appropriate ones. An appropriate PEP is a PEP which is able to process a considerable number of alerts while ensuring a low reaction cost. Thus, the definition and identification of an appropriate approximation of the $RD(PEP)$ must be well defined.
Hereafter, we first give an identification of $RD_{inf}(PEP)$ and then detail several approximations of the $RD(PEP)$.

4.7.1 Definition of $RD_{inf}(PEP)$

The configuration matrix $Conf_{selectors}(PEP)$ defined in equation 4.20 represents not only the configuration of the PEP but also the identification the $RD_{inf}(PEP)$. $RD_{inf}(PEP)$ is the union of the entire set of Responsibility Domains of configured rules in the PEP's instantiation.

$$Conf_{selectors}(PEP) = \begin{array}{c} \\ r_1 \\ r_2 \\ \vdots \\ r_m \end{array} \begin{array}{c} \begin{array}{cccc} S_1 & S_2 & ... & S_n \end{array} \\ \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1n} \\ s_{21} & s_{22} & \cdots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{m1} & s_{m2} & \cdots & s_{mn} \end{bmatrix} \end{array} \tag{4.20}$$

$RD_{inf}(PEP)$ is the union of $Conf_{selectors}(PEP)$ rows. The definition of $RD_{inf}(PEP)$ takes into account the entire set of the different combinations between selectors defined in configured rules.

$$\begin{aligned} RD_{inf}(PEP) &= \bigcup_{i\in[1...m]} RD(r_i) \\ &= \bigcup_{i\in[1...m]} < s_{ij} >_{j\in[1...n]} \end{aligned} \tag{4.21}$$

Running example : Based on the configuration table of the firewall, the $RD_{inf}(Firewall)$ will be written as follow:

$$\begin{aligned} &RD_{inf}(Firewall) = \\ &\quad \{< 140.192.37.*, *, 161.120.33.40, 80, tcp >, \\ &\quad < *.*.*.*, *, 161.120.33.40, 80, tcp >, \\ &\quad < 140.192.37.20, *, *.*.*.*, 80, tcp >, \\ &\quad < 140.192.37.*, *, *.*.*.*, 80, tcp >, \\ &\quad < 140.192.37.30, *, *.*.*.*, 21, tcp >, \\ &\quad < 140.192.37.*, *, *.*.*.*, 21, tcp >, \\ &\quad < 140.192.37.*, *, 161.120.33.40, 21, tcp >, \\ &\quad < *.*.*.*, *, 161.120.33.40, 53, udp >, \\ &\quad < 140.192.37.*, *, 161.120.33.*, *, udp >\} \end{aligned} \tag{4.22}$$

We remind that we ignore the default rule while identifying the $RD_{inf}(PEP)$.

4.7.2 Objective of $RD(PEP)$ approximations and methodology

Our objective at this stage is to analyze different possibilities of a comprehensive and appropriate approximation of $RD(PEP)$ without loosing the specificities of the deployed PEP. The $RD_{inf}(PEP)$ is considered as the unique starting point for all the approximations. Based on $RD_{inf}(PEP)$, we define inferences operations to build different versions of approximated Responsibility Domain denoted as $RD_{apprx}(PEP)$. These operations consider the relations between rules and characterizations of selectors, their combination properties. We detail them in the two next paragraphs.
The different approximations that we propose can be split in two major categories:

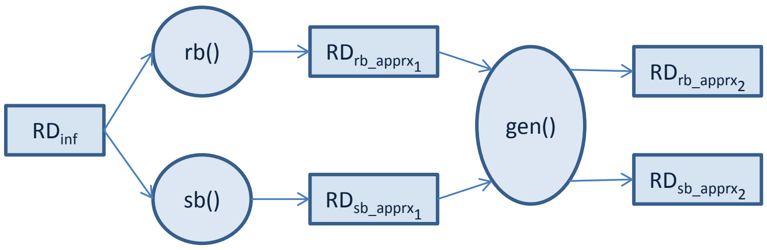


FIGURE 4.8 - Overview of Responsibility Domain Approximations.

Figure 4.8 represents an overview of the different steps and approximation functions.

$$\begin{aligned} rb() : \quad & \mathcal{U} \longrightarrow \mathcal{U} \\ & RD_{inf}(PEP) \longmapsto RD_{rb\_apprx_1(PEP)} \end{aligned} \tag{4.23}$$

The first rule-based approximation $RD_{rb\_apprx_1}(PEP)$ is the result of the application of the function $rb(RD_{inf})$ to all vectors of $RD_{inf}(PEP)$.

$$\begin{aligned} sb() : \quad & \mathcal{U} \longrightarrow \mathcal{U} \\ & RD_{inf}(PEP) \longmapsto RD_{sb\_apprx_1}(PEP) \end{aligned} \tag{4.24}$$

The first selector-based approximation $RD_{sb\_apprx_1}(PEP)$ is the result of the application of the function $sb(RD_{inf})$ to all vectors of $RD_{inf}(PEP)$.

We respectively detail these functions: $rb()$ and $sb()$ in Sections 4.7.3.1 and 4.7.4.1.

$$\begin{aligned} gen():\quad & \mathcal{U} \longrightarrow \mathcal{U} \\ & < s_j, j \in [1\dots n] > \longmapsto < \delta_{k_j}, j \in [1\dots n] > \end{aligned} \tag{4.25}$$

This function, gen(), refers to a generalization process which considers the Selector Domain Partition defined in paragraph 4.3. For each selector instantiation, s_j, gen() identifies the corresponding partition of s_j. The resulting vector will be a tuple of these generalized partitions.

4.7.3 Rule-based approximations of $RD(PEP)$

4.7.3.1 First rb_approximation of $RD(PEP)$

$RD_{rb\_apprx_1}(PEP)$ is the result of $rb(RD_{inf}(PEP))$ which performs several inference operations as explained before.

Inference operations performed by $rb()$

- If $RD(r_1)$ and $RD(r_2)$ are ***Completely Disjoint***, we include both of these rules. In the approximation algorithm described below, this operation is represented by $\leftarrow$.

- If $RD(r_1)$ and $RD(r_2)$ are ***Exactly Matched***, we keep only one rule since they represent the same Responsibility Domain.

- If $RD(r_1)$ and $RD(r_2)$ are ***Inclusively Matched***, then:

$$\exists\ j' \in [1\dots n]\ such\ that\quad r_1(S_{j'}) \subset r_2(S_{j'}) \tag{4.26}$$

  As shown in figure 4.7, in this relation, there is a rule with a larger Responsibility Domain. In our approximation, we only retain this rule. In the approximation algorithm described below, this operation is represented by $\leftarrow$.
- If $RD(r_1)$ and $RD(r_2)$ are ***Partially Matched***, then:

$$\exists\ i' \in [1\dots n]\ such\ that\quad r_1(S_{i'}) \not\supseteq r_2(S_{i'}) \tag{4.27}$$

  As shown in figure 4.7, in this relation, the two rules have a common area. We merge these two Responsibility Domains while verifying the **SCombinability** function. In the approximation algorithm described below, this operation is represented by $+$. If the SCombinability function is not verified, we keep both Responsibility Domains.

- Case of Correlated Responsibility Domains:
  If $RD(r_1)$ and $RD(r_2)$ are ***Correlated***, then:

$$\exists\ j', j'' \in [1 \ldots n]\ such\ that \begin{array}{l} r_1(S_{j'}) \subset r_2(S_{j'}) \\ r_1(S_{j''}) \supset r_2(S_{j''}) \end{array} \tag{4.28}$$

  We merge these two Responsibility Domains while verifying the **SCombinability** function

Algorithm 1 $RD_{rb\_apprx_1}(PEP) \leftarrow$ `Approximation_RD(PEP)_1`$(RD_{inf}(PEP))$

```
1: Input: $RD_{inf}(PEP)$
2: Output: $RD_{rb\_apprx_1}(PEP)$ /* first rb_approximation result of the $RD(PEP)$ */
3: $k = 1$
4: $RD(r'_1) = RD(r_1)$
5: $RD_{rb\_apprx_1}(PEP) = \{RD(r'_1)\}$
6: for all $i \in [2 \ldots m]$ do
7:   for all $k \in [1 \ldots lentgh(RD_{rb\_apprx_1}(PEP))]$ do
8:     $R \leftarrow verifyR(RD(r_i), RD(r'_k))$
9:     if $R = IM(RD(r'_k), RD(r_i))$ & $SCombinability(s_j, j \in [1 \ldots n]) =$ true then
10:      $RD(r'_k) \leftarrow RD(r_i)$
11:    else if $R = PM(RD(r'_k), RD(r_i))$ & $SCombinability(s_j, j \in [1 \ldots n]) =$ true then
12:      $RD(r'_k) \leftarrow RD(r'_k) + RD(r_i)$
13:    else if $R = CD(RD(r'_k), RD(r_i))$ & $SCombinability(s_j, j \in [1 \ldots n]) =$ true then
14:      $RD(r'_k) \leftarrow RD(r'_k) + RD(r_i)$
15:    else if $R = C(RD(r'_k), RD(r_i))$ & $SCombinability(s_j, j \in [1 \ldots n]) =$ true then
16:      $RD(r'_k) \leftarrow RD(r'_k) + RD(r_i)$
17:    else if $R = EM(RD(r'_k), RD(r_i))$ & $SCombinability(s_j, j \in [1 \ldots n]) =$ true then
18:      $RD(r'_k) \leftarrow RD(r_i)$
19:    else
20:      $RD_{rb\_apprx_1}(PEP) \leftarrow RD_{rb\_apprx_1}(PEP) + RD(r_i))$
21:      $k \leftarrow k + 1$
22:    end if
23:  end for
24: end for
25: return $RD_{rb\_apprx_1}(PEP)$
```

$RD_{rb\_apprx_1}(PEP)$ **Algorithm:** $RD_{rb\_apprx_1}(PEP)$ is a set of vectors $RD(r'_k)$ derived from $RD_{inf}(PEP)$.
Our approximation algorithm takes into account the relationships between rules and the constraints of combinability on selectors defined above. Algorithm 1 shows the different steps of the approximation algorithm. The $RD_{rb\_apprx_1}(PEP)$ is first initialized by $RD(r_1)$. Then, we iterate the $RD_{inf}(PEP)$ to check the existence of the defined relationships between $RD(r_i)$ and $RD(r'_k)$. For each discovered relationship, we append the $RD_{rb\_apprx_1}(PEP)$ with the corresponding elementary RD. We denote R the discovered relationships between $RD(r_i)$ and $RD(r'_k)$.
VerifyR is a function which identifies the relationship that exists between two different Responsibility Domains. $length(RD(PEP))$ is the number of vectors in the responsibility domain.

Running example In this paragraph, we detail the application of the first rb-approximation of $RD(PEP)$ on our running example. $m = 9$ represents the number of configured rules while ignoring the default rules. In Algorithm 2, we detail the application of Algorithm 1 and we indicate for each action the corresponding line number in algorithm 1.
The result of $RD_{rb\_apprx_1}(Firewall)$ is represented in Equation 4.29:

$$\begin{aligned} &RD_{rb\_apprx_1}(Firewall) = \\ &\qquad \{< *.*.*.*, *, 161.120.33.40, \{80, 21, 53\}, \{tcp, udp\} >, \\ &\qquad < 140.192.37.*, *, *.*.*.*, \{80, 21\}, tcp >, \\ &\qquad < 140.192.37.*, *, 161.120.33.*, *, udp >\} \end{aligned} \tag{4.29}$$

$RD_{rb\_apprx_1}(Firewall)$ is a concise representation of $RD_{inf}(Firewall)$.

Algorithm 2 $RD_{rb\_apprx_1}(Firewall) \leftarrow$ `Approximation_RD(Firewall)_1`$(RD_{inf}(Firewall))$

```
**Input:** $RD_{inf}(Firewall)$
**Output:** $RD_{rb\_apprx_1}(Firewall)$
$k = 1$ and $i = 1$
**4:** $RD(r'_1) = RD(r_1) =< 140.192.37.*, *, 161.120.33.40, 80, tcp >$
$RD_{rb\_apprx_1}(Firewall) = \{RD(r'_1)\}$
**9:** for $i = 2$ $R = IM(RD(r_2), RD(r'_1))$ &
$SCombinability(s_j, j \in [1 \dots n]) =$ **true**
$RD(r'_1) \leftarrow RD(r_2)$
**19-21:** for $i = 3$ $R = C(RD(r_3), RD(r'_1))$ &
$SCombinability(s_j, j \in [1 \dots n]) =$ **false**
$RD_{rb\_apprx_1}(Firewall) \leftarrow RD_{rb\_apprx_1}(Firewall) + RD(r_3))$
$RD(r'_2) = RD(r_3)$ $RD_{rb\_apprx_1}(PEP) = \{RD(r'_1), RD(r'_2)\}$
$k \leftarrow k + 1$
for $i = 4$ and for $k = 1$, $R = C(RD(r_4), RD(r'_1))$ &
$SCombinability(s_j, j \in [1 \dots n]) =$ **false**
**9:** for $k = 2$, $R = IM(RD(r'_2), RD(r_4))$ &
$SCombinability(s_j, j \in [1 \dots n]) =$ **true**
**10:** $RD(r'_2) \leftarrow RD(r_4)$
for $i = 5$ and for $k = 1$, $R = PM(RD(r_5), RD(r'_1))$ &
$SCombinability(s_j, j \in [1 \dots n]) =$ **false**
**11:** for $k = 2$, $R = PM(RD(r_5), RD(r'_2))$ &
$SCombinability(s_j, j \in [1 \dots n]) =$ **true**
**12:** $RD(r'_2) \leftarrow RD(r'_2) + RD(r_5)$
$RD(r'_2) =< 140.192.37.*, *, 161.120.33.40, \{80, 21\}, tcp >$
for $i = 6$ and for $k = 1$, $R = C(RD(r_6), RD(r'_1))$ &
$SCombinability(s_j, j \in [1 \dots n]) =$ **false**
**9:** for $k = 2$, $R = IM(RD(r_6), RD(r'_2))$
**10:** $RD(r'_2)$
**11:** for $i = 7$ and for $k = 1$, $R = PM(RD(r_7), RD(r'_1))$ &
$SCombinability(s_j, j \in [1 \dots n]) =$ **true**
**12:** $RD(r'_1) \leftarrow RD(r'_1) + RD(r_7)$
$RD(r'_1) =< *.*.*.*, *, 161.120.33.40, \{80, 21\}, tcp >$
**11:** for $i = 8$ and for $k = 1$, $R = PM(RD(r_8), RD(r'_1))$ &
$SCombinability(s_j, j \in [1 \dots n]) =$ **true**
$RD(r'_1) \leftarrow RD(r'_1) + RD(r_8)$
**12:** $RD(r'_1) =< *.*.*.*, *, 161.120.33.40, \{80, 21, 53\}, \{tcp, udp\} >$
**19-21:** for $i = 9$ and for $k = 1$, $R = C(RD(r_9), RD(r'_1))$ &
$SCombinability(s_j, j \in [1 \dots n]) =$ **false**
**19-21:** for $k = 2$, $R = C(RD(r_9), RD(r'_2))$ &
$SCombinability(s_j, j \in [1 \dots n]) =$ **false**
**20:** $RD_{rb\_apprx_1}(Firewall) \leftarrow RD_{rb\_apprx_1}(Firewall) + RD(r_9))$
$RD(r'_3) = RD(r_9)$ $RD_{rb\_apprx_1}(PEP) = \{RD(r'_1), RD(r'_2), RD(r'_3)\}$
**21:** $k \leftarrow k + 1$
**return 25:** $RD_{rb\_apprx_1}(Firewall)$
```

4.7.3.2 Second rb_approximation of $RD(PEP)$

A drawback of the first approximation is that it closely depends, by construction, on the configuration of the PEP. This causes a limited view of the real enforcement coverage of the PEP, since we only consider the configured values of selectors in rules and their combinations. But, the enforcement coverage of the PEP may be larger than inspecting only specific values and it may be capable to control a larger range of values.
Our second approximation of $RD(PEP)$ relies on selector taxonomy defined in paragraph 4.3.1. The objective at this stage, is the relaxation of the estimated $RD_{rb\_apprx_1}(PEP)$ in order to get more realistic view of the PEP's enforcement coverage.
We perform $gen()$ function on the set of vectors of $RD_{rb\_apprx_1}(PEP)$. For example, if the selector has a specific host IP address, then, we replace it with its corresponding sub-network. This operation is performed by the line 8 of the algorithm 3 where we **identify** the corresponding generalized domain $\delta_k(S_j)$ such that $r_i(S_j) \subset \delta_k(S_j)$. Hereafter, we detail the second approximation algorithm 3.
We denote by $m_{rb\_apprx_1}$ the number of rules in $RD_{rb\_apprx_1}(PEP)$.

Algorithm 3 $RD_{rb\_apprx_2}(PEP) \leftarrow$ `Approximation_RD(PEP)_2`$(RD_{rb\_apprx_1}(PEP))$

```
1: Input: $RD_{rb\_apprx_1}(PEP)$
2: Output: $RD_{rb\_apprx_2}(PEP)$ /* second rb_approximation result of the $RD(PEP)$ */
3: for all $i \in [1 \dots m_{rb\_apprx_1}]$ do
4:    for all $j \in [1 \dots n]$ do
5:       while $r_i(S_j) \neq D(S_j)$ and $r_i(S_j) \neq \delta_k(S_j)$ do
6:          identify $\delta_k(S_j)$ such that $r_i(S_j) \subset \delta_k(S_j)$
7:          $r_i(S_j) \leftarrow \delta_k(S_j)$
8:       end while
9:    end for
10: end for
11: $RD_{rb\_apprx_2}(PEP) \leftarrow \bigcup_{i \in [1 \dots m_{rb\_apprx_1}]} r_i(S_j), j \in [1 \dots n]$
12: return $RD_{rb\_apprx_2}(PEP)$
```

Example We consider the example listed above: $RD_{rb\_apprx_1}(Firewall)$ and we perform the second approximation algorithm as detailed in Algorithm 4.

Algorithm 4 $RD_{rb\_apprx_2}(Firewall) \leftarrow$ `Approximation_RD(Firewall)_2` $(RD_{rb\_apprx_1}(Firewall))$

```
Input: $RD_{rb\_apprx_1}(Firewall)$
Output: $RD_{rb\_apprx_2}(Firewall)$
for $i = 1$
for $j = 1$
5: $r_1(S_1) = D(S_1)$
for $j = 2$
5: $r_1(S_2) = D(S_(2))$
for $j = 3$
6: $r_1(S_3) \subset \delta_1(S_3) = [161.120.33.0/24]$
7: $r_1(S_3) \leftarrow \delta_1(S_3)$
for $j = 4$
6: $r_1(S_4) \subset \delta_1(S_4) = [0 \ldots 1023]$
7: $r_1(S_4) \leftarrow \delta_1(S_4)$
for $j = 5$
5: $r_1(S_5) = D(S_5)$
for $i = 2$
for $j = 1$
5: $r_1(S_1) = \delta_2(S_1)$
for $j = 2$
5: $r_1(S_2) = D(S_(2))$
for $j = 3$
5: $r_1(S_3) = D(S_(3))$
for $j = 4$
6: $r_1(S_4) \subset \delta_1(S_4) = [0 \ldots 1023]$
7: $r_1(S_4) \leftarrow \delta_1(S_4)$
for $j = 5$
5: $r_1(S_5) = \delta_1(S_5)$
for $i = 3$
for $j = 1$
5: $r_1(S_1) = \delta_2(S_1)$
for $j = 2$
5: $r_1(S_2) = D(S_(2))$
for $j = 3$
5: $r_1(S_3) = \delta_1(S_3)$
for $j = 4$
5: $r_1(S_4) = D(S_(4))$
for $j = 5$
5: $r_1(S_5) = \delta_2(S_5)$
11: $RD_{rb\_apprx_2}(Firewall) \leftarrow \bigcup_{i \in [1 \ldots m_{rb\_apprx_1}]} r_i(S_j), j \in [1 \ldots n]$
return $RD_{rb\_apprx_2}(Firewall)$
```

The result of $RD_{rb\_apprx_2}(Firewall)$ is represented in Equation 4.30:

$$\begin{aligned} &RD_{rb\_apprx_2}(Firewall) = \\ &\qquad \{< *.*.*.*, *, 161.120.33.*, [0\ldots1023], \{tcp, udp\} >, \\ &\qquad < 140.192.37.*, *, *.*.*.*, [0\ldots1023], tcp >, \\ &\qquad < 140.192.37.*, *, 161.120.33.*, *, udp >\} \end{aligned} \tag{4.30}$$

$RD_{rb\_apprx_2}(Firewall)$ has a dimension which represents most of the real enforcement coverage of the *Firewall*. The advantage of this second approximation is that it enables us to estimate the flow that may pass through the Firewall in a *blind manner*.

4.7.4 Selector-based approximations of $RD(PEP)$

4.7.4.1 First sb_approximation of $RD(PEP)$

This approximation is based on the instantiated values for each selector along $RD_{inf}(PEP)$ denoted as $D_{sb\_apprx_1}(S_i)$. We consider the set of all the valid selectors combinations where $SCombinability(s_j, j \in [1\ldots n]) = True$. Algorithm 5 displays the different operations to build the first sb_approximation of $RD(PEP)$. Within this approach, the wildcard is taken into account but is processed differently following the verification of *SCombinability* function. In fact, if the administrator does not set a specific value for a selector S along the configuration of the PEP, we ignore this selector. In this case, the wildcard is used for all the vectors in $RD_{inf}(PEP)$. Therefore, the dimension of the PEP will be decreased by one as described in Equation 4.31:

$$\begin{array}{ll} if\ \exists i' & such\ that\ S_{i'} = * \ \forall\ j \in [1\ldots m] \\ then & Dim(PEP) \leftarrow n-1 \\ & n \leftarrow n-1 \end{array} \tag{4.31}$$

Algorithm 5 $RD_{sb\_apprx_1}(PEP) \leftarrow$ `Approximation_RD(PEP)_3`$(RD_{inf}(PEP))$

```
1: Input: $RD_{inf}(PEP)$
2: Output: $RD_{sb\_apprx_1}(PEP)$ /* first sb_approximation result of the $RD(PEP)$ */
3: if $\exists i' such\ that\ S_{i'} = * \ \forall\ j \in [1 \ldots m]$ then
4:     then $n \leftarrow n - 1$
5: end if
6: for all $i \in [1 \ldots n]$ do
7:     $D_{sb\_apprx_1}(S_i) = r_1(S_i)$
8: end for
9: for all $i \in [1 \ldots n]$ do
10:     for all $j \in [2 \ldots m]$ do
11:         if $r_j(S_i) \neq D_{sb\_apprx_1}(S_i)$ or $(D_{sb\_apprx_1}(S_i) \subset r_j(S_i)$ and $r_j(S_i) \neq *)$ then
12:             $D_{sb\_apprx_1}(S_i) \leftarrow D_{sb\_apprx_1}(S_i)\ +\ r_j(S_i)$
13:         else if $r_j(S_i) = *$ then
14:             $D_{sb\_apprx_1}(S_i) \leftarrow D_{sb\_apprx_1}(S_i)\ +\ r_j(S_i)$
15:         end if
16:     end for
17: end for
18: $RD_{sb\_apprx_1}(PEP) \leftarrow \{\underset{i=1 \ldots n}{\times} D_{sb\_apprx_1}(S_i),\ such\ that\ SCombinability(s_i, i \in [1 \ldots n]) = \textbf{true}\}$
19: return $RD_{sb\_apprx_1}(PEP)$
```

Example We consider the example listed above: $RD_{inf}(Firewall)$ and we perform the third approximation algorithm. We notice that along all the rows of $RD_{inf}(Firewall)$, the wildcard is assigned to selector *src_port*. Therefore, the dimension of the Firewall is decreased by one.
In Algorithm 6, we detail the application of Algorithm 5 for the first selector which is *src_ip*.

Algorithm 6 $RD_{sb\_apprx_1}(Firewall) \leftarrow$ `Approximation_RD(Firewall)_3`$(RD_{inf}(Firewall))$

```
**Input:** $RD_{inf}(Firewall)$
**Output:** $RD_{sb\_apprx_1}(Firewall)$
**3:** $S_3 = src\_port = * \; \forall \; j \in [1 \ldots 9]$
**4:** then $5 \leftarrow 4$
**for all** **6:** $i \in [1 \ldots 4]$ **do**
    **7:** $D_{sb\_apprx_1}(S_i) = r_1(S_i)$
**end for**
for $i = 1$
for $j = 2$
**13:** $r_2(S_1) = *$
**14:** $D_{sb\_apprx_1}(S_1) \leftarrow D_{sb\_apprx_1}(S_1) + *$
for $j = 3$
$r_3(S_1) \subset D_{sb\_apprx_1}(S_1)(1)$
for $j = 4$
$r_4(S_1) = D_{sb\_apprx_1}(S_1)(1)$
for $j = 5$
$r_5(S_1) \subset D_{sb\_apprx_1}(S_1)(1)$
for $j = 6$
$r_6(S_1) = D_{sb\_apprx_1}(S_1)(1)$
for $j = 7$
$r_7(S_1) = D_{sb\_apprx_1}(S_1)(1)$
for $j = 8$
$r_2(S_1) = *$
for $j = 9$
$r_9(S_1) = D_{sb\_apprx_1}(S_1)(1)$
**18:** $RD_{sb\_apprx_1}(Firewall) \leftarrow \{ \underset{i=1\ldots4}{\times} D_{sb\_apprx_1}(S_i),\ such\ that\ SCombinability(s_i, i \in [1 \ldots 4]) = \textbf{true}\}$
**return** **19:** $RD_{sb\_apprx_1}(Firewall)$
```

Hereafter in Equation 4.32, we detail the result of the different $D_{sb\_apprx_1}$ domains of the Firewall selectors.

$$\begin{array}{lcl} D_{sb\_apprx_1}(src\_ip) & = & \{140.192.37.*,\ *.*.*.*\} \\ D_{sb\_apprx_1}(dst\_ip) & = & \{161.120.33.*,\ *.*.*.*\} \\ D_{sb\_apprx_1}(dst\_port) & = & \{21,\ 53,\ 80,\ *\} \\ D_{sb\_apprx_1}(p) & = & \{tcp,\ udp\} \end{array} \tag{4.32}$$

The result of $RD_{sb\_apprx_1}(Firewall)$ is represented in Equation 4.33:

$$\begin{aligned} RD_{sb\_apprx_1}(Firewall) = \ & \{D_{sb\_apprx_1}(src\_ip) \times D_{sb\_apprx_1}(dst\_ip) \\ & \times D_{sb\_apprx_1}(dst\_port) \times D_{sb\_apprx_1}(p), \\ & such\ that: \\ & SCombinability\ (s_j, j \in [1 \dots 5]) = True\} \end{aligned} \tag{4.33}$$

4.7.4.2 Second sb_approximation of $RD(PEP)$

At this stage of approximations, we apply the generalization function, $gen()$, on $D_{sb\_apprx_1}(S_j)$ domains. As a consequence, we consider larger range of values for selectors referring to its domain decomposition.

$$\begin{aligned} RD_{sb\_apprx_2}(PEP) \leftarrow \{ \underset{j=1 \dots n}{\times} gen(D_{sb\_apprx_1}(S_j)),\ such\ that \\ SCombinability(s_j, j \in [1 \dots n]) = True\} \end{aligned} \tag{4.34}$$

Algorithm 7 $RD_{sb\_apprx_2}(PEP) \leftarrow$ `Approximation_RD(PEP)_2`$(RD_{sb\_apprx_1}(PEP))$

```
1: Input: $RD_{sb\_apprx_1}(PEP)$
2: Output: $RD_{sb\_apprx_2}(PEP)$ /* second sb_approximation result of the $RD(PEP)$ */
3: for all $j \in [1 \dots n]$ do
4:     $identify\ \delta_k(S_j)\ such\ that\ D_{sb\_apprx1}(S_j) \subset \delta_k(S_j)$
5:     $D_{sb\_apprx_2}(S_j) \leftarrow \delta_k(S_j)$
6: end for
7: $RD_{sb\_apprx_2}(PEP) \leftarrow \{ \underset{j=1 \dots n}{\times} D_{sb\_apprx_2}(S_j),\ such\ that\ SCombinability(s_j, j \in [1 \dots n]) = \mathbf{true}\}$
8: return $RD_{sb\_apprx_2}(PEP)$
```

4.8 Analysis of $RD(PEP)$ Approximations and Interpretations

4.8.1 $RD(PEP)$ Approximations Properties

As explained above, all the approximations closely depends on the configuration of the deployed PEP. Therefore, as depicted in figure 4.9, several relations would exist between these approximations:

- **Totally Inclusive Approximations**: The application of $gen()$ function results in a generalization of the selectors values.

$$\begin{aligned} & RD_{rb\_apprx_1}(PEP) \subseteq RD_{rb\_apprx_2}(PEP) \\ \Rightarrow\ & \mid RD_{rb\_apprx_1}(PEP) \mid \leqslant \mid RD_{rb\_apprx_2}(PEP) \mid \end{aligned} \tag{4.35}$$

$RD_{rb\_apprx_2}(PEP)$ includes additional combination with values of a common generalized sub-domains $\delta_k(S)$ for one selector. Thus, as shown in Equation 4.35, the number of vectors, i.e. the cardinality of $RD_{rb\_apprx_1}(PEP)$, is lower than the cardinality of $RD_{rb\_apprx_2}(PEP)$.
Proof:

$$\begin{aligned} \mid RD_{rb\_apprx_1}(PEP) \mid &= \sum_{k\in[1\ldots m_{rb\_apprx1}]} \mid RD(r'_k) \mid \\ &= \sum_{k\in[1\ldots m_{rb\_apprx1}]} \prod_{j\in[1\ldots n]} \mid s'_{kj} \mid \\ &\leq \sum_{k\in[1\ldots m_{rb\_apprx1}]} \prod_{j\in[1\ldots n]} \mid gen(s'_{kj}) \mid \\ &\leq \mid RD_{rb\_apprx_2}(PEP) \mid \end{aligned}$$

$RD_{rb\_apprx_1}(PEP)$ and $RD_{sb\_apprx_1}(PEP)$ are also Inclusive domains because of the generation of additional selector values combinations when building $RD_{sb\_apprx_1}(PEP)$. Thus, as shown in Equation 4.36, the cardinality of $RD_{rb\_apprx_1}(PEP)$ is lower than the cardinality of $RD_{sb\_apprx_1}(PEP)$.

$$\begin{aligned} & RD_{rb\_apprx_1}(PEP) \subset RD_{sb\_apprx_1}(PEP) \\ \Rightarrow\ & \mid RD_{rb\_apprx_1}(PEP) \mid \leqslant \mid RD_{sb\_apprx_1}(PEP) \mid \end{aligned} \tag{4.36}$$

Moreover, since the approximation $RD_{sb\_apprx_2}(PEP)$ includes all the possible combinations between values of selectors generalized sub-domains, it will include $RD_{rb\_apprx_1}(PEP)$,
$RD_{rb\_apprx_2}(PEP)$ and $RD_{sb\_apprx_1}(PEP)$.

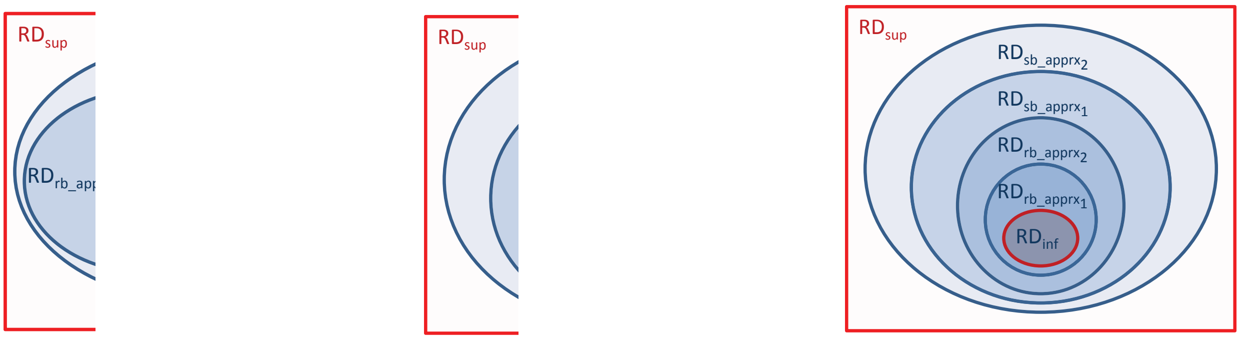
Depending on the configured values s_{ij} in $Conf_{selectors}(PEP)$, it is possible that $RD_{sb\_apprx_1}(PEP)$ is included in $RD_{rb\_apprx_2}(PEP)$, Figure 4.9 (b) or $RD_{rb\_apprx_2}(PEP)$ is included in
$RD_{sb\_apprx_1}(PEP)$, Figure 4.9 (c).
Proof:

$$\begin{aligned} & if\ \forall j \in [1\ldots n],\ \exists k_j \in [1\ldots l_j]\ such\ that \\ & \qquad D_{sb\_apprx_1}(S_j) \subseteq \delta_{k_j} \\ & \text{then} \underset{j=1\ldots n}{\times} D_{sb\_apprx_1}(S_j) \subseteq \underset{j=1\ldots n}{\times} \delta_{k_j} \\ & \text{then} RD_{sb\_apprx_1}(PEP) \subseteq RD_{rb\_apprx_2}(PEP) \end{aligned}$$

- **Partially Joint Approximations**:
Both of $RD_{rb\_apprx_2}(PEP)$ and $RD_{sb\_apprx_1}$ may have a common set of vectors

which is at least the $RD_{rb\_apprx_1}$ as shown in Figure 4.9 (a).
Proof:

[illegible]



(c) Third Case

FIGURE 4.9 - Relationships between the proposed approximations.

4.8.2 Qualitative Analysis: Approximation Accuracy metric

Use case

In this Section, we consider the Running Example defined in Section 4.4.

Identification of $RD_{sup}(Firewall)$: As mentioned in the Definition 5, the RD_{sup} of a deployed PEP includes the set of all possible vectors characterizing the flow that may pass through the PEP. We denote by $D_{sup}(S)$ the *real* domain of a Selector S. It is identified by considering the topological information about the network.

$$\begin{aligned} D_{sup}(src\_ip) &= \{140.192.37.*,\ 161.120.33.*,\ *.*.*.*\} \\ D_{sup}(dst\_ip) &= \{140.192.37.*,\ 161.120.33.*,\ *.*.*.*\} \\ D_{sup}(p) &= \{tcp,\ udp,\ icmp\} \end{aligned} \tag{4.37}$$

$$\begin{aligned} RD_{sup}(Firewall) = \ & \{D_{sup}(src\_ip) \\ & \times D_{sup}(dst\_ip) \times D_{sup}(p), \\ & such\ that: \\ & SCombinability\ (s_j, j \in [1 \ldots n]) = True\} \end{aligned} \tag{4.38}$$

Table 4.3 - Evaluation of Approximation Accuracy Metric of the running example

	RD_{inf}	$RD_{rb\_apprx_1}$	$RD_{rb\_apprx_2}$	$RD_{sb\_apprx_1}$	$RD_{sb\_apprx_2}$
$\Lambda(RD_{apprx}(PEP))$	$4 * 10^{-11}$	$2,5 * 10^{-8}$	$1,3 * 10^{-3}$	$3,9 * 10^{-3}$	$3,9 * 10^{-3}$

In Section 4.7, we identified the approximations of $RD(Firewall)$ based on $RD_{inf}(Firewall)$. These approximations correspond to the case of Figure 4.9 (c).

Approximation Accuracy has been introduced in several mathematical theories such as approximation theory, rough set, fuzzy set, etc. In our approach, we propose to apply this metric in order to evaluate how accurate the approximations are regarding the *real* Responsibility Domain of the PEP. We adapt the Approximation Accuracy expression defined in *Rough Set Theory* [99]. In [99], the author define the Accuracy Approximation as a measure to express the *quality* of the approximation. In Equation 4.39, we define the Approximation Accuracy of $RD_{apprx}(PEP)$ which we denote as $\Lambda(RD_{apprx}(PEP))$ as:

$$\Lambda(RD_{apprx}(PEP)) = \frac{|RD_{apprx}(PEP)|}{|RD_{sup}(PEP)|} \tag{4.39}$$

Obviously, $0 < \Lambda(RD_{inf}(PEP)) \leq \Lambda(RD_{apprx}(PEP)) \leq 1$ for any $RD_{apprx}(PEP)$. Following the definition of the approximation of the Responsibility Domain, it is more accurate when $\Lambda(RD_{apprx}(PEP))$ is closer to 1.
In Table 4.3, we evaluate this metric for the different approximations of the running example. Based on results shown in Table 4.3, the Approximation Accuracy of approximations $RD_{sb\_apprx_1}$ and $RD_{sb\_apprx_2}$ is 10^8 times bigger than the $\Lambda(RD_{inf}(PEP))$. In this case, selector-based approximations are more appropriate than rule-based approximations.

4.9 Applications

The proposed PEP model can be used in several security applications. Hereafter, we detail some of these applications.

4.9.1 Policy anomalies detection

Security Policy anomalies are one of the most critical security issue in a network. Rule misconfiguration induce security policy conflicts either within a single policy (intra-policy conflicts) or between policies in different devices (inter-policy conflicts), [14]. In [14], authors present a taxonomy of security policy conflicts. The most common policy conflict occurs when different access control decisions are applied on the same flow.

Conflict detection can be considered as an easy task to cope with in an environment with a single PEP. However, it becomes a challenging task and error-prone in an environment with distributed PEP. Referring to our model, conflict detection can be defined as overlapping domains between different $RD_{inf}(PEP)$ on which different decisions (d_{k_1} and $d_{k_2} \in \{d_k\}_{k\in[1\ldots p]}$) are applied. Our proposed PEP model is abstract enough to allow us to identify these overlapping domains between several PEPs. In Equation 4.40, we express the inter-PEP conflict detection process using our PEP model.

$$\begin{array}{ll} if & RD_{inf}(PEP_1) \cap RD_{inf}(PEP_2) \neq \oslash \\ \Rightarrow & if\ \exists\ d_{k_1},\ d_{k_2} \in [d_1 \ldots d_p], such\ that\ d_{k_1} \neq d_{k_2} \\ & and\ corresponds\ to\ common\ rules\ RD_{inf}(PEP_1) \cap RD_{inf}(PEP_2) \\ & \Rightarrow Conflict(PEP_1, PEP_2) = True \end{array} \tag{4.40}$$

Conflict(PEP_1, PEP_2) is a Boolean function which returns True if there is an inter-conflict between PEP_1 and PEP_2. Otherwise, it returns False.

4.9.2 Intrusion Detection Assessment

Scenario-based IDS usually rely on attack signatures when generating alerts. Attack signatures allow IDSs examining the events monitored in the supervised network by providing attack description specification. Cuppens et Al. [76] define LAMBDA, an attack description language. In LAMBDA, an attack is a combination of actions with additional statements related to the supervised network. LAMBDA language specifies the *pre-conditions* that should be satisfied by the supervised network for an attack to be feasible [76]. It also specifies the *post-conditions* of an attack. Post-conditions are the effects of a successful execution of an attack. In Figure 4.10, we show an example of a LAMBDA attack description. Pre-conditions and post-conditions are a set of logical conditions respectively denoted by C_{pre} and C_{post}. From Figure 4.10, $remote\_access(A, H)$ is an example of a condition C_{pre}. $knows(A, use\_serviceH, portmapper))$ is an example of C_{post}. In practice, pre and post conditions are elements that can be identified based on the attack signature and more specifically based on the exploited vulnerability [1]. IDS relies on these signatures in order to detect attacks and generate alerts. Alerts are messages including information about the potential attack instantiation. Following the PEP function given in Equation 4.2 and the LAMBDA description language, we model network

[1]Open Source Vulnerability Database: OSVDB, `http://www.osvdb.org/`

$$
\begin{array}{l}
\textbf{attack } \mathit{rpcinfo}(\mathit{Target\text{-}IP}) \\
\quad \textbf{pre} : \overbrace{\mathit{remote\_access}(A,H)}^{C_1} \wedge \overbrace{\mathit{ip\_address}(H,\mathit{Target\text{-}IP})}^{C_2} \\
\qquad \wedge \underbrace{\mathit{use\_service}(H,\mathit{portmapper})}_{C_3} \wedge \underbrace{\mathit{use\_service}(H,\mathit{mountd})}_{C_4} \\
\quad \textbf{post} : \underbrace{\mathit{knows}(A,C_3)}_{P_1} \wedge \underbrace{\mathit{knows}(A,C_4)}_{P_2} \\
\quad \textbf{scenario} : E_1 \\
\qquad \textbf{where } \mathit{action}(E_1) = \texttt{rpcinfo -p } \mathit{Target\text{-}IP} \\
\qquad\quad \wedge\ \mathit{actor}(E_1) = A \\
\quad \textbf{detection} : F_1 \\
\qquad \textbf{where } \mathit{action}(F_1) = \mathit{detect}(E_1) \\
\quad \textbf{verification} : G_1 \\
\qquad \textbf{where } \mathit{action}(G_1) = \mathit{test\_service}(\mathit{portmapper})
\end{array}
$$

FIGURE 4.10 - LAMBDA attack description: an example [76].

IDS systems (denoted by netIDS) in Equation 4.41:

$$
\begin{array}{lll}
netIDS: & < D(A_{att}) \times D(pre) > \longrightarrow \{d_{k,k\in[1\ldots p_{ids}]}), D(post)\} & \\
 & & \\
\text{where:} & D(A_{att}) \ =< D(src\_ip) \times D(src\_port) \times D(dst\_ip) & \\
 & \qquad\qquad \times D(dst\_port) \times D(p) > & (4.41) \\
 & D(pre) = \{C_{pre,i_{pre}}, i_{pre} \in [1 \ldots p_{pre}]\} & \\
 & D(post) = \{C_{post,i_{post}}, i_{post} \in [1 \ldots p_{post}]\} &
\end{array}
$$

$D(A_{att})$ represents selectors domain identifying the attack characteristics. It includes a common set of selectors as defined in the example of $netFW$. $D(pre)$ is the domain of the pre-conditions that are defined based on the signature database of netIDS. $D(post)$ is the domain of the post-conditions of the successful detected attacks. $d_k, k \in [1 \ldots p_{IDS}]$, is the intrusion detection decision (generate alert or not) that is applied by the netIDS. Note that post-conditions are considered to be one of the decisions taken by the netIDS. In fact, once an attack is detected, the netIDS identifies the real consequences of this attack by considering the attack signature allowing the detection of the attack and the properties of the supervised network.
Using the model expressed in Equation 4.41, it becomes straightforward to evaluate the capability of the netIDS in detecting different attack scenarios. In fact, if $\exists\ C_{post} \in D(post)$ such that $C_{post} \in\subset D(pre)$, then, it is possible to detect elementary attacks that potentially belong to a same attack scenario.
Our model allows the security administrator to identify IDSs that are capable to collaborate in detecting attack scenarios. We note $netIDS_1$ and $netIDS_2$ two different IDSs and $(D_1(pre), D_1(post))$ and $(D_2(pre), D_2(post))$ their corresponding domains of pre-conditions and post-conditions. $netIDS_1$ and $netIDS_2$ are capable to collaborate in detecting attack scenarios if $\exists\ C_{post} \in D_2(post)$ such that $C_{post} \in\subset D_1(pre)$.

4.10 Conclusion

We introduce a novel concept to model Policy Enforcement Point by their Responsibility Domain, $RD(PEP)$. We first characterize the PEP by the set of selectors. Then, we define the Responsibility Domain of a configured rule $RD(r)$. We analyze the relationships that may exist between these domains and define a set of approximation inferences. Based on the different properties that exist between $RD(r)$ and the characterization of selectors, we give different approximations of the $RD(PEP)$. We have shown that these approximations have different properties and multiple use cases. The advantage of our methodology to approximate $RD(PEP)$ is the performance in a 'blind manner'. Also, the consideration of the PEP configuration makes the approximations more useful for response decision.

Chapter 5

Enforcement-based Alert Correlation Framework

Contents

5.1 Definition and Framework Description

5.1.1 Definition and Objectives

Alerts are correlated if they share a common (set of) PEP(s) capable of applying a countermeasure on the flow corresponding to the alert.

Definition 6 ***Enforcement-based Alert Correlation :*** *Given a set of alerts and a set of deployed PEP, the Enforcement-based Alert Correlation groups alerts that can be processed by a common PEPs.*

In Equation 5.1, we write the basic correlation inference used in our Enforcement-based Alert Correlation approach.

$$\begin{aligned} A_1 \in RD(PEP_1) \wedge A_2 \in RD(PEP_1) \wedge \ldots \\ \Longrightarrow A^{ec} = \langle (A_1,\ A_2,\ \ldots), PEP_1 \rangle \end{aligned} \tag{5.1}$$

A_1 and A_2 are two different alerts. $RD(PEP_1)$ is the Responsibility Domain of the PEP_1.
A^{ec} represents the Enforcement-based Correlated Alert. It is composed of two components. The first component includes the set of correlated alerts. The second component includes the PEP(s) that is (are) capable to process the correlated alerts.
A^{ec} is intended to group one or more generated alerts together, to say "these alerts can be processed by the same PEP(s)".

5.1.2 Framework

The Policy Enforcement-based Alert Correlation Engine (PEACE) represented in Figure 5.1 and Figure 5.2 is the core of our framework. It is responsible of checking which alerts can be processed by a same set of PEP(s).
The configurations of deployed PEPs are retrieved and sent to the *RD-Approximator*. This latter performs the proposed approximations on collected configurations. Approximations are stored in a centralized database which is RD_DB. PEACE is then able to process the received alerts $\{A_1, A_2, \ldots, A_{n_t}\}$. n_t is the number of alerts generated during a period of time t.
The PEACE framework provides the security administrator with a set of correlated alerts $\{A^{ec}_1, A^{ec}_2, \ldots, A^{ec}_{n_{a'}}\}$, where $n_{a'} \leq n_a$.

5.2 Alignment Functions

The membership verification, $A \in RD(PEP)$, mentioned in Equation 5.1, requires that both A and $RD(PEP)$ are represented following the same multidimensional domain. However, on one hand, alerts are expressed in IDMEF. On the other hand, deployed PEPs are represented according to their PEP class domain as defined in Definition 1. These discrepancies between the IDMEF standard and PEP class expressions leads to semantic and syntactic differences which prevent us from easily computing the membership function defined in 5.3. Tables 5.1 and 5.2 and 5.3 allows to align the IDMEF attributes and PEP selectors. In IDMEF standard, *Source* and *Target* classes contain respectively information about the possible source(s) and target(s) of the event(s) that generated an alert. Since *Source* and *Target* have almost the same aggregation classes and attributes, we restrict our comparison in Tables 5.1, 5.2 and 5.3 on the *Source* class.

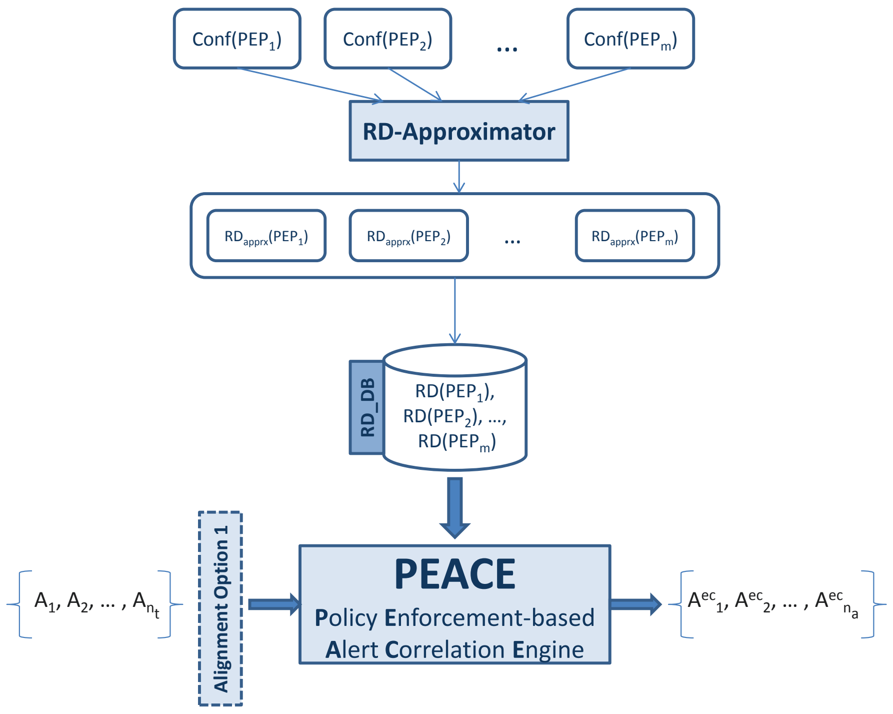

Conf(PEP$_1$)
Conf(PEP$_2$)
...
Conf(PEP$_m$)
RD-Approximator
RD$_{apprx}$(PEP$_1$)
RD$_{apprx}$(PEP$_2$)
...
RD$_{apprx}$(PEP$_m$)
RD_DB
RD(PEP$_1$), RD(PEP$_2$), ..., RD(PEP$_m$)
$A_1, A_2, \dots, A_{n_t}$
Alignment Option 1
PEACE
Policy Enforcement-based Alert Correlation Engine
$A^{ec}_1, A^{ec}_2, \dots, A^{ec}_{n_a}$

FIGURE 5.1 - PEACE Framework with alignment function option 1.

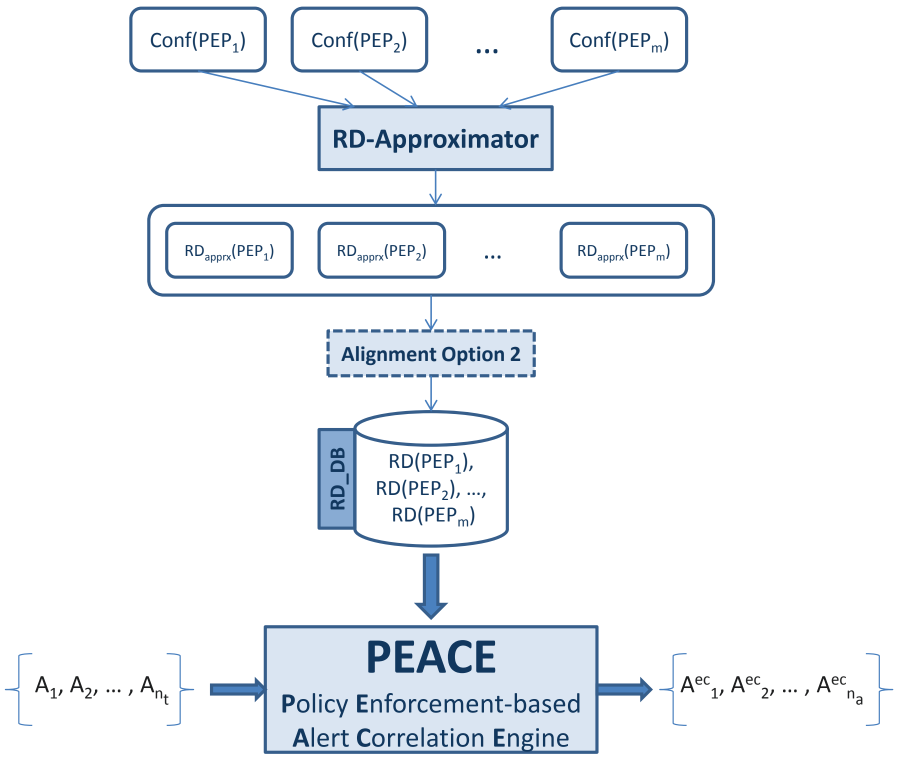


FIGURE 5.2 - PEACE Framework with alignment function option 2.

TABLE 5.1 - Analogy between IDMEF attributes and selectors of three PEPs (Iptable, ModSecurity, LDAP)

IDMEF (Source class) / **PEP**			**iptable**		**ModSecurity**		**LDAP**	
	Attribute	Type	Selector	Type	Selector	Type	Selector	Type
	Node							
	Category	enum	ipv4-addr, ipv6-addr (ip6iptable package)	-	ipv4-addr, ipv6-addr	-	ipv4-addr, ipv6-addr	-
	Location	string	(GeoIP db)	string	(geolookup operator, GeoIP db)	string	location	string
	Name	string	–src	string	hostname (hostname lookup)	string	-	-
Address	category	enum	-	-	-	-	-	-
	address	string	-s-add / –mac-source (mac module)	string / hexa	Remote-Address / Host	string	peername.ip = **ipv4** [%netmask] [port], peer-name.ipv6	string
	netmask	string	**-s-add**	integer	**Remote-Address**	integer	peername.ip = ipv4 [**%netmask**] [port]	LDAP String

Thus, PEACE requires to align alert domains with the PEPs' Responsibility Domains. To meet this requirement, we propose some *Alignment Functions*. Two

Table 5.2 - Analogy between IDMEF attributes and selectors of three PEPs (Iptable, ModSecurity, LDAP)

PEP / IDMEF (Source class)			iptable		ModSecurity		LDAP	
Attribute		Type	Selector	Type	Selector	Type	Selector	Type
User								
Category		enum	-	-	(Request-Body)	string	-	-
UserId	type	enum	-	-	-	-	-	-
UserId	name	string	–uid-owner	string	Remote-User	string	username, cn, user, givenName	string
UserId	number	integer	-	-	-	-	-	-

Table 5.3 - Analogy between IDMEF attributes and selectors of three PEPs (Iptable, ModSecurity, LDAP)

PEP / IDMEF (Source class)		iptable		ModSecurity		LDAP	
Attribute	Type	Selector	Type	Selector	Type	Selector	Type
Service							
ip_version	integer	-	-	-	-	-	-
iana_protocol_number	integer	-	-	-	-	-	-
iana_protocol_name	string	-	-	-	-	-	-
port	integer	-sport	integer	Remote-Port	integer	peername.ip = ipv4 [%net-mask] [**port**]	integer
portlist	string	-	-	-	-	-	-
protocol	string	-p	string	-	-	-	-

options are possible for Alignment Functions.

5.2.1 1^{st} option: From IDMEF to PEP class

This option consists of aligning the received IDMEF alert with the PEP class. In this case, Alignment has to be performed on each received alert. In Equation 5.2, we give the generic expression of Alignment Functions of the 1st option. We denote it by $align_{PEP}$.

$$\begin{array}{rl} align_{PEP}: & Dom(IDMEF) \longrightarrow Dom(PEP) \\ & \qquad A \quad \longmapsto A' = align_{PEP}(A) \\ & \\ where & A =< a_1, a_2, \ldots, a_{n_{idmef}} > \\ & and\ A' =< a'_1, a'_2, \ldots, a'_{n_{pep}} > \end{array} \tag{5.2}$$

$Dom(IDMEF)$ is a multidimensional domain including the different attributes defined in the IDMEF standard. n_{idmef} is the number of IDMEF attributes present in an alert A.
Each attribute a'_i, $i \in [1 \ldots n_{pep}]$, follows the semantics of selector S_i of the corresponding class of PEP. If the alignment function does not find in the IDMEF Alert A the necessary information to fill attributes a', they will be set as *empty*.
Alig hown
in F n the

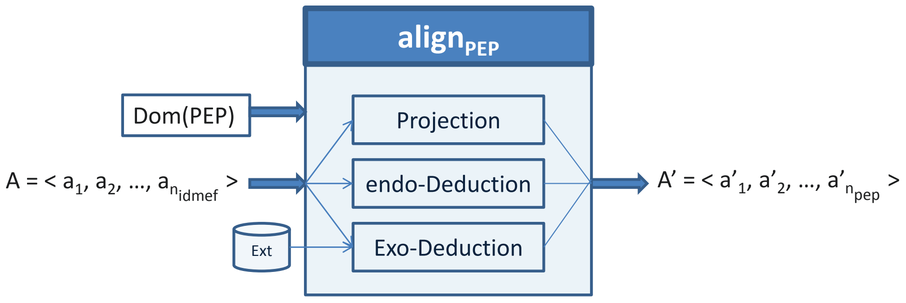


FIGURE 5.3 - Alignment Function: 1st option.

Alignment Function:

- **Projection**: this operation extracts the selector S_i directly from the IDMEF alert. For instance, based on Tables 5.1 and 5.2 and 5.3, projection function are performed on elements that are not written in bold or underlined. No additional operations are performed on the corresponding attribute. In this case, the attribute a_j, $j \in [1 \ldots n_{idmef}]$ has the semantics of selector S_i.
- **endo-Deduction**: this function performs a first order inference in order to instantiate the appropriate attribute a'_j in the alert A' as described in Equation 5.3. a'_j is of the same semantic of Selector S_j, $j \in [1 \ldots n_{pep}]$.

$$\begin{array}{ll} if & (a_i = val_1 \wedge a_{i'} = val_2 \wedge \ldots) \\ \Longrightarrow & a'_j = val_3 \end{array} \tag{5.3}$$

val_1 and val_2 are respectively the information filled in attributes a_i and $a_{i'}$. val_3 is the result of the inference which is the information of attribute a'_j.

From Tables 5.1, 5.2 and 5.3, elements that are in bold are to be used in endo-Deduction alignment function.

- **exo-Deduction**: using the information of the alert and referring to an external knowledge (represented by the "Ext" component in Figure 5.3 such as the Geographical Location database, exo-Deduction operation infer the value of a corresponding attribute a'_j in the alert A'. Like endo-Deduction, exo-Deduction is a set of first order inferences. This alignment function considers underlined elements in Table 5.1 such the GeoIP database.

5.2.2 2^{nd} option: from PEP class to IDMEF

Contrary to the 1^{st} option, *Alignment Functions*, in the 2^{nd} option, performs alignment operations in order to align the Responsibility Domains of PEPs with the IDME ın of alignm

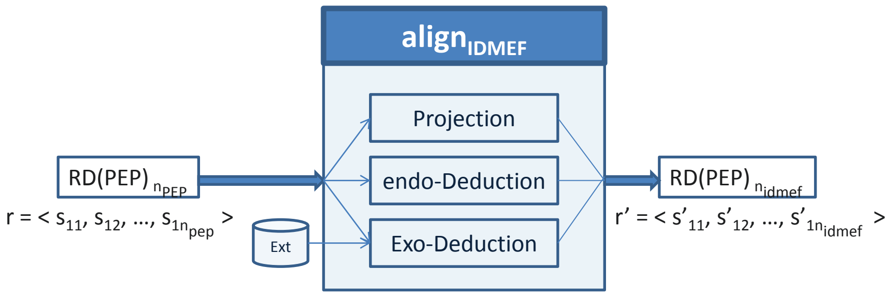

align$_{IDMEF}$
Projection
endo-Deduction
Exo-Deduction
RD(PEP) n_{PEP}
$r = < s_{11}, s_{12}, \ldots, s_{1n_{pep}} >$
Ext
RD(PEP) n_{idmef}
$r' = < s'_{11}, s'_{12}, \ldots, s'_{1n_{idmef}} >$

FIGURE 5.4 - Alignment Function: 2^{nd} option.

$$\begin{array}{lll} align_{IDMEF}: & Dom(PEP) \longrightarrow Dom(IDMEF) & \\ & RD(PEP) \longmapsto RD_{idmef}(PEP) & \\ & & (5.4) \\ where & Dim(RD_{idmef}(PEP)) = n_{idmef} & \end{array}$$

As discussed in Section 2.5, IDMEF has been conceived in order to ensure both of syntactic and semantic interoperability. Therefore, the application of previously described alignment operations becomes straightforward in the second option of alignment function.

* $n_{c_{pep}}$ is the number of deployed class of PEPs.

5.2.3 Comparison between alignment options

Alignment is performed during the pre-correlation phase. In Table 5.4, we compare the two options while considering three important criteria: the execution of the

TABLE 5.4 - Comparison between alignment options.

Criteria	**Option 1**	**Option 2**
Process	Online	Offline
Complexity	$n_{c_{pep}} \times n_a$	$n_{c_{pep}}$
Time consuming	Yes	No

alignment process, time consumption and complexity.
The second option aligns Responsibility Domains of deployed classes of PEP. It is performed in an offline mode relative to the correlation process. The first option aligns each alert once received with each deployed class of PEP. Such online performance increases the delay of correlation execution. Based on this comparative table, we can easily figure out that the second alignment option presents more interesting advantages for correlation process. In fact, it presents the advantage of being less time consuming because it has less complexity and can even be performed offline and not during correlation.

In the rest of this work, we assume that both alerts and Responsibility Domains of deployed PEPs are presented in the same multi-dimensional domain. We will use the notation defined in Equation 5.5 to represent both alert and PEP responsibility domains.

$$\begin{array}{ll} A & =< a_1, a_2, \dots, a_n > \\ RD(PEP) & = \bigcup\limits_{j \in [1 \dots n_r]} < s_{j1}, s_{j2}, \dots, s_{jn} > \\ & = \bigcup\limits_{j \in [1 \dots n_r]} v_j \\ where & n_r \text{ is the number of RD(PEP) rule vectors } v_j. \end{array} \tag{5.5}$$

5.3 Membership function: Alert ∈ RD(PEP)

The membership function in Equation 5.1 is a fundamental step for our Enforcement-based Alert Correlation approach. Usually, a vector $e =< e_1, e_2, \dots, e_n >$ is included in an n-dimensional domain D if $\exists\ d \in D$ such that $\forall i \in [1 \dots n], e_i = d_i$. We call this property the *membership* of e *in* D.
In our context, exact membership means that the deployed PEP is able to implement a countermeasure rule which is capable to process the alert. Therefore, it is necessary to verify the membership function of the alert A within the $RD(PEP)$. We denote it $\gamma(A, RD(PEP))$.
Since the $RD(PEP)$ is a set of n_r vector rules, we first define the membership of the alert within each Responsibility Domain of a rule vector v_j ($j \in [1 \dots n_r]$) denoted by $\gamma_j(A, RD(v_j))$. In Equation 5.6, we define $\gamma_j(A, RD(v_j))$:

$$\begin{array}{ll} \gamma_j(A, RD(v_j)) & = \underset{i\in[1\ldots n]}{\times} verif(a_i, v_{ji}) \\ \text{where} & \times \text{is the multiplication operator of real set } \mathcal{R}. \\ \text{and} & verif(a_i, v_{ji}) = \begin{cases} 1, & \text{if } a_i \subseteq v_{ji}. \\ 0, & \text{otherwise.} \end{cases} \end{array} \tag{5.6}$$

The membership of the Alert A within the $RD(PEP)$ means that there exist **at least** a rule vector v_j whose the $RD(v_j)$ covers all the selectors of the Alert. We define $\gamma(A, RD(PEP))$ in Equation 5.7.

$$\begin{array}{ll} \gamma(A, RD(PEP)) & = \underset{j\in[1\ldots n_r]}{\dot{+}} (\gamma_j(A, RD(v_j))) \\ \text{where} & \dot{+} \text{ is the logical OR operator} \end{array} \tag{5.7}$$

Note that:

$\gamma(A, RD(PEP)) \in \{0, 1\}$ and we have:

$$\begin{cases} \text{if}\gamma(A, RD(PEP)) = 1, \\ \Rightarrow \text{membership of the alert } A \text{ in the } RD(PEP). \\ \text{if}\gamma_j(A, RD(PEP)) = 0, \\ \Rightarrow \text{non-membership of the alert } A \text{ in the } RD(PEP). \end{cases}$$

5.4 Enforcement-based alert correlation workflow

In this section, we detail in Figure 5.5 the enforcement-based alert correlation workflow. It exposes the different steps that should be executed in order to decide whether the PEP is able to process an alert A or not.

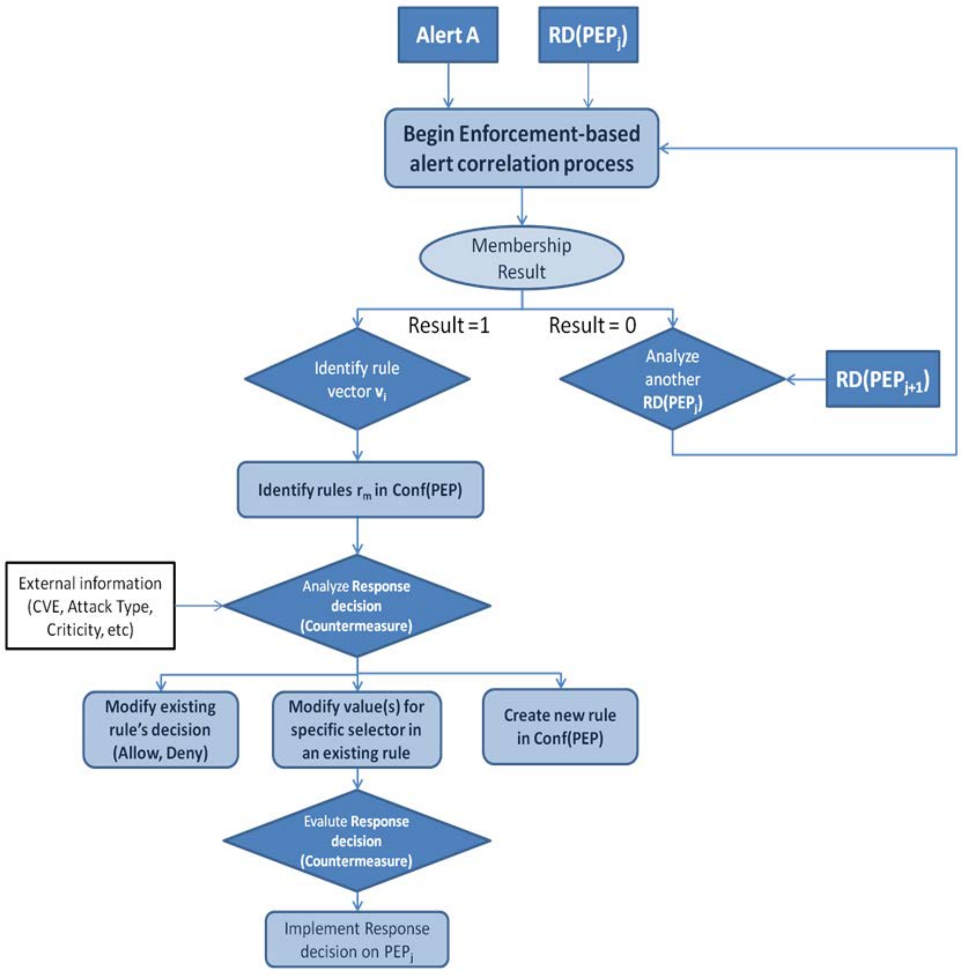

Alert A
RD(PEP$_j$)
Begin Enforcement-based alert correlation process
Membership Result
Result =1
Result = 0
Identify rule vector v_i
Analyze another RD(PEP$_j$)
RD(PEP$_{j+1}$)
Identify rules r_m in Conf(PEP)
External information (CVE, Attack Type, Criticity, etc)
Analyze Response decision (Countermeasure)
Modify existing rule's decision (Allow, Deny)
Modify value(s) for specific selector in an existing rule
Create new rule in Conf(PEP)
Evalute Response decision (Countermeasure)
Implement Response decision on PEP$_j$

FIGURE 5.5 - Enforcement-based alert correlation workflow with one alert A.

The membership verification is the first step in this workflow. If the alert A does not fall in the responsibility domain of a PEP, then another PEP will be analyzed. If the alert A falls in the responsibility domain of a PEP, then the process "identify rules in $Conf(PEP)$" is executed. The objective of this process is to identify the potential rules that may be affected to implement the countermeasure. After identifying the potential configured rules that can process the alert, we analyze the possibilities of countermeasure implementation. Hereafter, the possibilities are described:

- Modify existing rule's decision;
- Modify configured values for specific selectors in configured rules;
- Create new rule in $Conf(PEP)$.

We may need external information, such as the vulnerability and the criticality of the attack, about the generated alerts when analyzing the response decision.

5.5 Conclusion

This chapter introduces a novel alert correlation approach which based on the responsibility domain of a PEP. Since it is performed on mutli-dimensional elements, we first study the representation requirements of alerts and responsibility domain. For this aim, we propose alignment functions which allow the representation of alerts and responsibility domain vectors in the same multi-dimensional space. We then define a membership function which allows the identification of PEP that can process generated alert. Finally, the workflow of the proposed alert correlation approach is detailed.

Chapter 6

Applications and interpretations

Contents

6.1 Introduction

In this chapter, we focus on studying the application of our enforcement-based alert correlation. We first apply our approach on a single PEP environment. We demonstrate how our alert correlation approach allows the identification of potential rules in order to react to attacks. We then examine the application of our approach on a multi-PEP environment using a real case scenario. We explain how our enforcement-based alert correlation is able to support the security administrator in response decision and countermeasure implementation.

6.2 Application of PEACE on a single PEP use case

In this section, we detail the application of our proposed alert correlation approach on a single PEP use case. Respecting Equation 5.5, we remind that both alerts and Responsibility Domains of deployed PEPs are presented in the same multi-dimensional domain. Hence, the membership function is performed on vectors with same dimension.
The application on a single PEP use case assists the security administrator in identifying configured rules which will be modified in order to process correlated alerts. Hereafter, we study the application of the proposed responsibility domains of a PEP on our alert correlation.

6.2.1 Alert Correlation based on the $RD_{inf}(PEP)$

As defined in Section 4.7, $RD_{inf}(PEP)$ is composed of all the rule vectors configured in the deployed PEP. Verifying the membership of an alert within $RD_{inf}(PEP)$ implies the identification (as shown in Equation 6.1) of the appropriate rule vector in the $Conf_{selectors}(PEP)$, which is capable to implement the reaction decision. In such case, the reaction decision is generally a modification of the enforcement decision applied by the rule.

$$A \in RD_{inf}(PEP) \Rightarrow \ \exists\, i \in [1 \ldots n_r] \setminus \forall j \in [1 \ldots n],\ a_j \subseteq s_{ij} \tag{6.1}$$

Two alerts, $A_1 =< a_{11}, a_{12}, \ldots, a_{1n} >$ and $A_2 =< a_{21}, a_{22}, \ldots, a_{2n} >$, are correlated based on $RD_{inf}(PEP)$ when $\exists\, j_1, j_2 \in [1 \ldots n] \setminus a_{1j} \subseteq s_{ij_1}$and $a_{2j} \subseteq s_{ij_2}$. There are two possible cases:

- Case 1: $j_1 = j_2$. Then, both of the alerts are processed by the same rule r_{j_1}. In this case, we define the Enforcement-based correlated alert as follow:

$$A^{ec} = \langle (A_1,\ A_2), (r_{j_1}, RD_{inf}(PEP)) \rangle$$

  This correlated alert specifies the rule vector that will processes the correlated alerts A_1 and A_2.
- Case 2: $j_1 \neq j_2$. Then, the two alerts can be processed by the same PEP but not by the same rule.

Correlating alerts using $RD_{inf}(PEP)$ provides more than the verification of the capability of the PEP to process the correlated alerts, it also identifies the capability of each rule vector of $RD_{inf}(PEP)$ in implementing a response decision. For instance, in the case of a firewall, in order to process an alert, it is possible to modify an *accept* to *deny* enforcement decision.

6.2.2 Alert Correlation based on Rule-based first approximation

As detailed in Chapter 4 , rule vectors of $RD_{rb\_apprx_1}(PEP)$ are obtained by applying $rb()$ inference functions on $RD_{inf}(PEP)$. For instance, as shown in Equation 6.2, when two rule vectors $RD(r_1)$ and $RD(r_2)$ in $RD_{inf}(PEP)$ are *Partially Matched* and combinable, we merge them in a single rule $RD(r_{1,2})$ vector in $RD_{rb\_apprx_1}(PEP)$.

$$\begin{aligned} &RD(r_1) &&=< 140.192.37.*, *_{port}, *_{ip}, 80, tcp > \\ &RD(r_2) &&=< 140.192.37.30, *_{port}, *_{ip}, 21, tcp > \\ &\underset{rb()}{\Longrightarrow} RD(r_{1,2}) &&=< 140.192.37.*, *_{port}, *_{ip}, \{21, 80\}, tcp > \end{aligned} \tag{6.2}$$

This inference instantiates a generalized rule vector $RD(r_{1,2})$ which includes both of $RD(r_1)$ and $RD(r_2)$. $RD_{rb\_apprx_1}(PEP)$ correlates alerts that partially share a common (sub)set of attributes. Thus, applying $RD_{rb\_apprx_1}(PEP)$ in our alert correlation approach allows to correlate alerts which can be processed by modifying a common set of rules configured in the deployed PEP or defining new rules. For instance, the following alerts detailed in Equation 6.3 are correlated with $RD(r_{1,2})$. Since this latter is the result of merging $RD(r_1)$ and $RD(r_2)$, we conclude that both alerts A_1 and A_2 can be processed by rules r_1 and r_2.

$$\begin{aligned} A_1 &=< 140.192.37.140, 51000, 10.0.0.3, 80, \{tcp\} > \\ A_2 &=< 140.192.37.140, 4000, 10.0.0.5, 21, \{tcp\} > \end{aligned} \tag{6.3}$$

6.2.3 Alert Correlation based on Selector-based first approximation

As detailed in Chapter 4, rule vectors of $RD_{rb\_apprx_1}(PEP)$ are obtained by applying $sb()$ function on $RD_{inf}(PEP)$. The application of $sb()$ function on $RD_{inf}(PEP)$ consists on generating all the acceptable combinations between instantiated selector values. For instance, as shown in Equation 6.4, the application of $sb()$ function on RD_{r_1}, RD_{r_2} and RD_{r_3} results in $RD(r_{1,2,3})$ which is built of all the combinations between selector values defined in RD_{r_1}, RD_{r_2} and RD_{r_3}.

$$\begin{aligned}
RD(r_1) &=< 140.192.37.*, *_{port}, *_{ip}, 80, tcp > \\
RD(r_2) &=< 140.192.37.30, *_{port}, *_{ip}, 21, tcp > \\
RD(r_3) &=< 140.192.37.20, *_{port}, *_{ip}, 53, udp > \\
\underset{sb()}{\Longrightarrow} RD(r_{1,2,3}) &= \{140.192.37.* \times *_{port} \times *_{ip} \\
&\qquad \times \{21, 53, 80\} \times \{tcp, udp\}\}
\end{aligned} \tag{6.4}$$

$RD(r_{1,2,3})$ includes additional rule vectors which are totally different from the input rule vectors. Thus, Enforcement-based Alert Correlation based on $RD_{sb\_apprx_1}(PEP)$ correlates alerts which will not be necessary processed by a common set of configured rules.

6.2.4 Alert Correlation based on the generalized approximations

The $gen()$ function is performed on both of rule-based and selector-based first approximations respectively denoted by $RD_{rb\_apprx_1}(PEP)$ and $RD_{sb\_apprx_1}(PEP)$. $gen()$ consists in expanding the approximated Responsibility Domain of the PEP in order to have larger coverage of network flow vectors that may pass through the PEP. This fact leads to a higher number of alerts that fall in the responsibility domain of the considered PEP. For instance, applying $gen()$ function on $RD(r_{1,2})$ listed in example 6.2 results in $RD_{gen}(r_{1,2})$ defined in Equation 6.5:

$$RD_{gen}(r_{1,2}) \;=< 140.192.37.*, *_{port}, *_{ip}, [1 \dots 1024], \{tcp\} > \tag{6.5}$$

The evaluation of $RD_{gen}(r_{1,2})$coverage shows that $gen()$ function increases the coverage of $RD(r_{1,2})$ by $(2^{10} - 3)$ times. On the one hand, this Coverage Increase ($\mathcal{CI}$) explains how generalized approximations $RD_{rb\_apprx_2}(PEP)$ or $RD_{sb\_apprx_2}(PEP)$ allow us to correlated more alerts relative to the same PEP. On the other hand, correlated alerts will be processed by different countermeasure implementations. In this case, countermeasure implementations should be either modification of configured rule vectors or insertion of a new rule vector in the configuration of the PEP. Insertion of a new rule vector in a configuration of a PEP require a deep analysis and study in terms of conflict detection and policy validation.

TABLE 6.1 - Application of Responsibility Domain approximations in PEACE: a comparative table.

Approximated RD	**Approximation function**	**impact on Alert Correlation**
$RD_{inf}(PEP)$	–	Alerts are correlated using existing configured rules. + Countermeasure implementation are immediately identified – Restricted view on the PEP responsibility domain – Alerts that can be processed by injecting new rules are ignored
$RD_{rb\_apprx_1}(PEP)$	$rb(RD_{inf}(PEP))$	Alerts are correlated by a set of combinations between configured rules + Countermeasure implementation implicates at least one configured rule modification – Alerts that can be processed by injecting new rules are ignored
$RD_{sb\_apprx_1}(PEP)$	$sb(RD_{inf}(PEP))$	Alerts are correlated regarding all possible combination between configured values of selectors – Countermeasure implementation require deep analysis of involved rules in processing correlated alerts – Alerts that can be processed by injecting new rules with different selectors values are ignored
$RD_{rb\_apprx_2}(PEP)$	$gen(RD_{rb\_apprx_1}(PEP))$	Alerts are correlated regarding generalized configured rules – The amount of correlated alerts depends on the distribution over partitions of configured selectors values
$RD_{sb\_apprx_2}(PEP)$	$gen(RD_{sb\_apprx_1}(PEP))$	Alerts are correlated regarding generalized configured selectors values – The amount of correlated alerts depends on the distribution over partitions of configured selectors values

6.2.5 Interpretations and Analysis

We further compare the application of the different Responsibility Domain approximations on Alert Correlation. Since our proposed PEP responsibility domain model is build following a bottom-up approach and in blind manner, there must be a trade-off between correlation and the PEP responsibility domain approximation coverage. Table 6.1 depicts the advantages and drawbacks of each responsibility domain approximations in Enforcement-based Alert Correlation.
The ability to correlated alerts increases with the Coverage Increase since the more $RD_{apprx}(PEP)$ is wider than $RD_{inf}(PEP)$, the more we correlate alerts relative to the PEP.

Definition 7 *We define the* ***Coverage Increase*** *($\mathcal{CI}$) of the RD approximations ($RD_{apprx}(PEP)$), denoted by $\mathcal{CI}(RD_{apprx})$, as the additional rule vectors generated by the approximation.*

In Equation 6.6, we detail how $\mathcal{CI}$ is evaluated:

$$\mathcal{CI}(RD_{apprx}(PEP)) = \frac{|RD_{apprx}(PEP)|}{|RD_{inf}(PEP)|} \tag{6.6}$$

This metric is evaluated for the example described in 4.4. Table 6.2 includes the evaluations of this metric for the different approximations. In the second row of the Table 6.2, we give an evaluation of the cardinality of each approximations which is denoted by $\mid (RD_{apprx}(PEP) \mid$.
In the context of the running example, $RD_{sb\_apprx_1}$ and $RD_{sb\_apprx_2}$ have the same set of vectors. In fact, the configured values for selectors throughout the rules totally cover different partitions. Thus, in this case, the generalized function does not result in an additional number of elements as we assert from the second row of Table 6.2. Based on the evaluation of the coverage increase shown in Table 6.2, the generalization function performed on responsibility domain approximations has an important effect on the coverage increase. In our running example, it multiplies the coverage increase by 10^5 comparing it to the first rule-based approximation. This fact increases the ability to correlate alerts.

TABLE 6.2 - Evaluation of Coverage Increase Metric of the running example

	RD_{inf}	$RD_{rb\_apprx_1}$	$RD_{rb\_apprx_2}$	$RD_{sb\_apprx_1}$	$RD_{sb\_apprx_2}$
$\mid (RD_{apprx}(PEP) \mid$	$\backsim 2,16 * 10^{17}$	$13,5 * 10^{19}$	$7 * 10^{24}$	$21 * 10^{24}$	$21 * 10^{24}$
$CI(RD_{apprx}(PEP))$	1	$6,25 * 10^2$	$3,25 * 10^7$	$9,75 * 10^7$	$9,75 * 10^7$

6.3 Role of enforcement-based alert correlation in countermeasure implementations in a single PEP environment

In this section, we demonstrate how our correlation approach helps the security administrator in implementing countermeasures. We explain how our alert correlation approach enables us identifying PEPs that are capable to process a group of correlated alerts. The question is: "Once the PEP is identified, is it possible to correlate alerts based on the configured rules in order to optimize the countermeasure implementations?"
Since the responsibility domains approximations are build following a bottom up approach as described in Chapter 4, it is possible to identify configured rules that are to be modified in order to implement countermeasures. In our context, countermeasures are the modifications brought to the existing configuration of the PEP in order to mitigate detected attacks. These modifications includes for example expanding the value defined in a configured rule for a selector or a subset of selectors, merging two rules, etc.
Once alerts are correlated by a specific responsibility domain of a PEP, it is essential to identify subsets of alerts which can be processed by common set of configured rules. Figure 6.1, shows an example of the application of our enforcement-based alert correlation in countermeasure implementations.
We deal with a simple case of a PEP with three configured rules denoted by r_1, r_2 and r_3. These rules are represented by the circles having the orange colour in Figure 6.1. The responsibility domain approximation results on an RD(PEP) with two rule vectors: v_1 and v_2. These rules are represented by the red circles. The set $\{A_1, A_2, A_3, A_4, A_5\}$ represents the correlated alerts. Alerts are represented by green circles.
We note that alerts A_1, A_2 and A_3 were correlated regarding rule vector v_1. Meanwhile, only A_1 and A_2 can be processed by rule r_1. Alert A_3 can be processed by r_2.

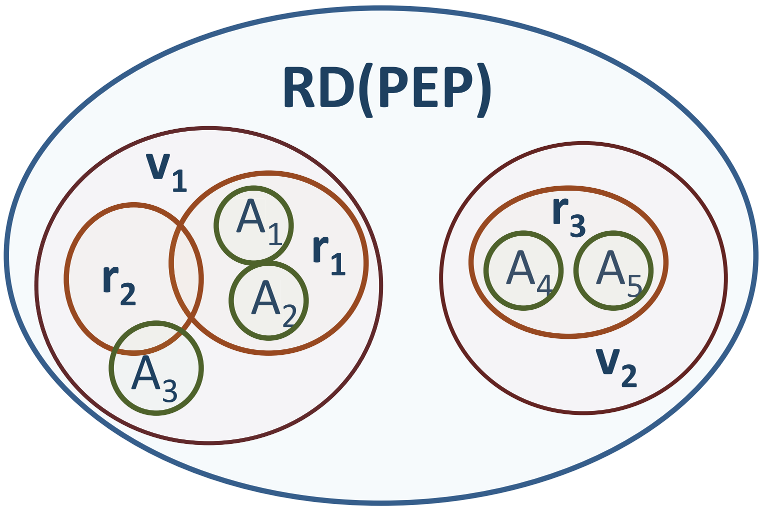


FIGURE 6.1 - Enforcement-based alert correlation and countermeasure implementation.

This distribution result of correlated alerts regarding configured rules can be demonstrated by the coverage metric ($\mathcal{CM}$) for a given alert A and a configured rule r. This metric, denoted by $\mathcal{CM}(A,r)$, is evaluated using the function $\delta(A,r)$ as defined in Equation 6.7. In the Equation 6.7, A_k is a correlated alert with the $RD(PEP)$ and r_p is a configured rule in $Conf(PEP)$.

$$\begin{aligned}\delta(A_k,r_p) &=< verif(a_{k1},r_{p1}), verif(a_{k2},r_{p2}),\ldots,verif(a_{kn},r_{pn}) > \\ &=< verif(a_{ki},r_{pi}) >_{i\in[1,\ldots,n]} \\ & verif(a_{ki},r_{pi}) \text{ is the binary function defined in Equation 5.6}\end{aligned} \tag{6.7}$$

We define $\mathcal{CM}(A,r)$ in Equation 6.8.

$$\mathcal{CM}(A_k,r_p) = \frac{\sum_{i\in[1\ldots n]} \delta_i(A_k,r_p)}{n} \tag{6.8}$$

$\mathcal{CM}(A_k,r_p)$ is a binary variable.
In the rest of this section, we consider the example of two configured rules r_1 and r_2 whose their responsibility domains is defined as follow:

$$\begin{aligned} RD(r_1) &=< 140.192.37.*, *_{port}, *_{ip}, 80, tcp > \\ RD(r_2) &=< 140.192.37.30, *_{port}, *_{ip}, 21, tcp > \end{aligned}$$

The corresponding rule vector $v_1 \in RD(PEP)$ for these two rules (referring to the generalization approximation algorithms 2 and 4 described in Sections 4.7.3.2 and 4.7.4.2) is:

$$v_1 = RD_{gen}(r_{1,2}) \quad =< 140.192.37.*, *_{port}, *_{ip}, [1\ldots 1024], \{tcp\} >$$

We consider two alerts defined as follow:

$$\begin{aligned} A_1 &=< 140.192.37.140, 51000, 10.0.0.3, 80, \{tcp\} > \\ A_2 &=< 140.192.37.30, 4000, 10.0.0.3, 53, \{tcp\} > \end{aligned}$$

Both A_1 and A_2 are in $RD(v_1)$.

6.3.1 Discussion for a single alert

When a single alert A falls in a responsibility domain of a PEP, it is important to identify the appropriate configured rules which will be responsible to process this alert. The function $\mathcal{CM}(A_k, r_p)$ defined in Equation 6.8 identifies common attributes between the alert and the configured rules. But, in order to implement the countermeasure, it is necessary to evaluate the modifications that should be made on the rule.
Therefore, we define the modification rate denoted by $\mathcal{M}_{r_p/A_k}$ in Equation 6.9.

$$\begin{aligned} \mathcal{M}_{r_p/A_k} = &\ 1 - \mathcal{CM}(A_k, r_p) \\ &= 1 \ - \ \frac{\sum\limits_{i \in [1 \dots n]} \delta_i}{n} \end{aligned} \tag{6.9}$$

$$\mathcal{CM}(A_k, r_p) \in \{0,1\} \Rightarrow \mathcal{M}_{r_p/A_k} \in \{0,1\}$$

For example, we evaluate this metric using alert A_2 and rule r_1.

$$\begin{aligned} \mathcal{M}_{r_1/A_2} = &\ 1 - \mathcal{CM}(A_2, r_1) \\ &= 1 - \tfrac{4}{5} = \tfrac{1}{5} \end{aligned}$$

The evaluation of $\mathcal{M}_{r_1/A_2}$ shows that a modification of the single attribute (the *destination port*) in the configured rule r_1 allows to handle the alert A_2.
In practice, it is important to set a modification rate threshold which we denote by Th_M. Th_M is the acceptance criteria of the modification rate $\mathcal{M}_{r_1/A_2}$.
If $\mathcal{M}_{r_1/A_2} < Th_M$, the identified configured rule could be considered in order to process the alert. Otherwise, if $\mathcal{M}_{r_1/A_2} > Th_M$, the modification rate is unacceptable.

6.3.2 Discussion for two alerts

One of the advantages of our enforcement-based alert correlation is the possibility to identify if countermeasure implementations can be aggregated or not. When dealing with two alerts or more, aggregation of countermeasure implementation is important.

Therefore, we analyze the possibility of aggregating countermeasure implementation by evaluating the differences between the coverage metric for two alerts A_1 and A_2 referring to a single implemented rule r_p. We propose the aggregation metric denoted by $\mathcal{AM}((A_1, A_2), r_p)$. In Equation 6.10, we define how we can compute this aggregation metric:

$$\mathcal{AM}((A_1, A_2), r_p) = \frac{\sum\limits_{i \in [1 \dots n]} \delta(a_{1i}, r_{pi}) \times \delta(a_{2i}, r_{pi})}{n} \tag{6.10}$$

$\times$ in Equation 6.10 is the multiplication operator of the real set $\mathcal{R}$.
Therefore, if alerts can be both entirely processed by the rule r_p (which means that $\delta(a_{1i}, r_{pi}) = \delta(a_{2i}, r_{pi}) = 1, \forall i \in [1 \dots n]$), both of the alerts can be aggregated by the rule r_p. Otherwise, it is important to set a tolerance threshold which we denote by Th_{AM}. This threshold point out the tolerance of the security administrator in modifying configured rules.
If $\mathcal{AM}((A_1, A_2), r_p) > Th_{AM}$, the identified configured rule could be considered in order to aggregated countermeasure for both og alerts A_1 and A_2. Otherwise, if $\mathcal{AM}((A_1, A_2), r_p) < Th_M$, the aggregation of countermeasure is not desirable.
Figure 6.2 represents the workflow described above. When evaluating the aggregation metric of rules with which alerts were correlated, it is possible to have multiple rules that satisfy the condition of Th_{AM}. In this case, the administrator should study the optimal response decision that will be implemented. Such This matter is out of the scope of our work and may require external knowledge.

Interpretation:

Our objective is to assist the security administrator in selecting which rules are potentially able to process a number of correlated alerts. As a consequence, we first provide the security administrator with a distribution of generated alerts based on the configured rules in a PEP. Then, we provide him with an evaluation of the modification rate (countermeasure implementation).

6.4 Application of PEACE in multi-PEP scenario

In this section, we analyze the application of our alert correlation approach in a real life use case where multiple PEPs are deployed. We accordingly deal with the Olympic Games security architecture which is composed of several commonly deployed security patterns. We first introduce the Olympic Games use case. We detail its security architecture and the different deployed PEPs and security patterns. Then, we discuss the application of our alert correlation approach on some of these patterns.

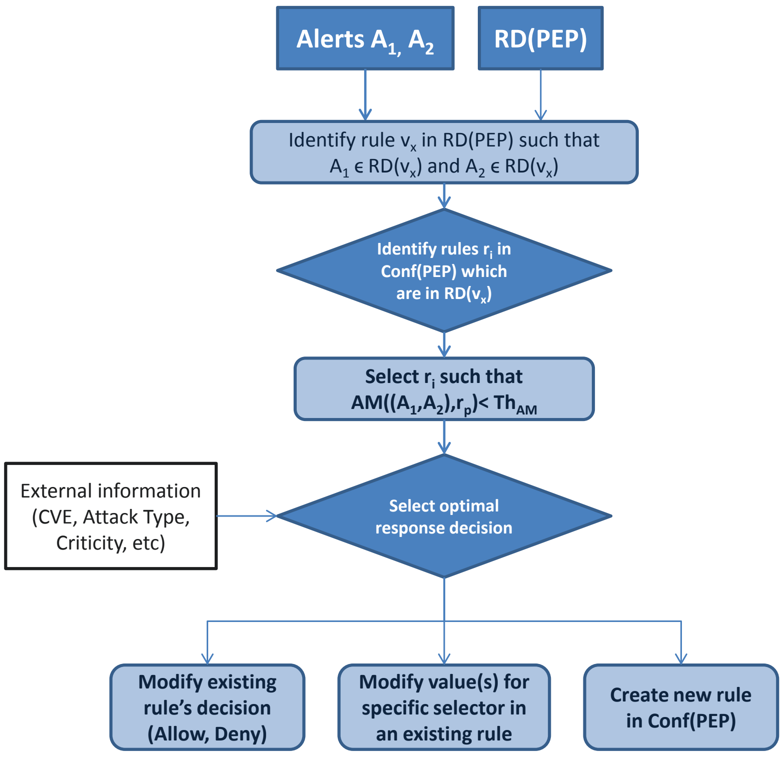

Alerts A_1, A_2
RD(PEP)
Identify rule v_x in RD(PEP) such that $A_1 \in RD(v_x)$ and $A_2 \in RD(v_x)$
Identify rules r_i in Conf(PEP) which are in RD(v_x)
Select r_i such that $AM((A_1,A_2),r_p) < Th_{AM}$
External information (CVE, Attack Type, Criticity, etc)
Select optimal response decision
Modify existing rule's decision (Allow, Deny)
Modify value(s) for specific selector in an existing rule
Create new rule in Conf(PEP)

FIGURE 6.2 - Enforcement-based alert correlation workflow with two alerts

6.4.1 Case Study: Olympic Games

Olympic Games IT System is a complex and massive IT architecture deployed by large team of people. Its mission is to provide services and real time information for competitions of around 20 disciplines that span more than 60 competition and non-competition venues, involving more than 10.000 athletes, 20.000 members of the media and 70.000 volunteers. In general, the Olympic Games IT system requires about 10,000 computers, 1,000 servers, 20,000 desktop phones and 2,500 Intranet terminals.
Olympic Games case study responds to the needs of improving the IT security of such critical system. This use case discusses the application of PEACE in an environment with multiple deployed PEP belonging to different classes and analyze all possible relationships between $RD(PEP)$ and their impact on the correlation.
In Figure 6.3, we give a simplified overview of the Olympic Games system architecture. It is mainly composed of 3 core systems which are:

- **Core Games System (CGS)**: a set of applications for assisting in the capture and management of data about people who will be attending the Games events and the staff supporting them (such as Volunteers, Electronic Staff Information

(ESI) system, etc).

- **Information Diffusion System (INFO)**: a set of applications that retrieve and distribute information related to, and supporting, of the Games. The information is provided by different sources e.g. Results system, interfaces with CGS, Weather provider etc. The information is processed and distributed to internal clients e.g. broadcasters, journalists, etc. It is also sent to external clients e.g. World News Press Agencies (WNPA), sports federations and governing bodies, and Internet Service Providers (ISPs).
- **Results System (RS)**: a set of applications for collecting real-time data during competitions (Timing & Scoring Systems) and distributing collected information to the INFO system (On Venue Results Systems).

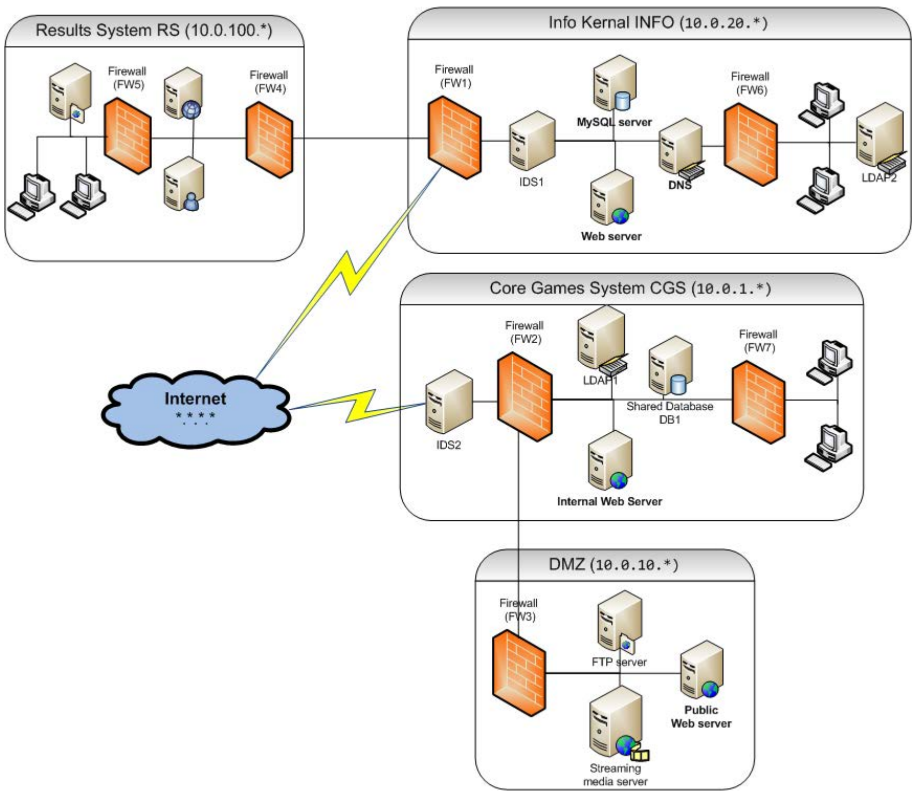


Figure 6.3 - Overview of Olympic Games System Architecture.

6.4.2 Analysis of relationships between multiple PEP Responsibility Domains

An intrusion response decision requires the identification of the PEP(s) that is (are) responsible of implementing appropriate countermeasures. In a complex environment such as the Olympic Games IT system where an important number of PEPs are deployed, it is challenging to select the proper PEP(s) that is (are) capable to enforce the intrusion response decision. Moreover, in a defense-in-depth strategy, the intrusion response decision results in the participation of different PEPs. Thus, it is important to study the relationships that exist between multiple PEP responsibility domains. Two cases can be distinguished: disjoint domains and joint responsibility domains. Figure 6.4 depicts these cases. It is important to remind that in our as-

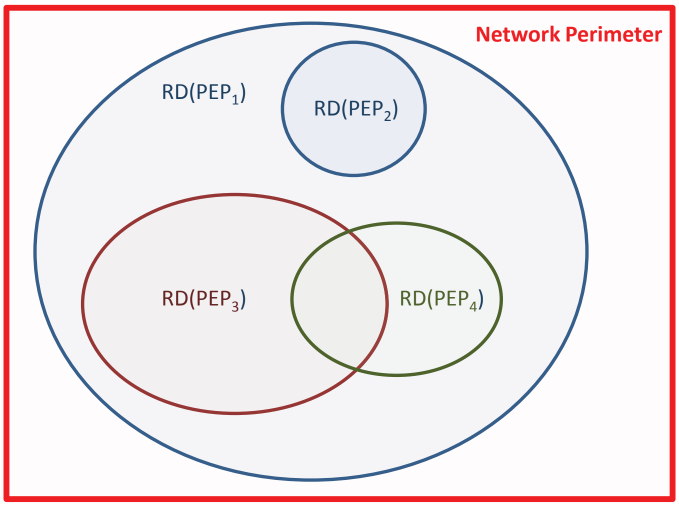


FIGURE 6.4 - Possible relationships between PEPs Responsibility Domains.

sumption, all the responsibility domains are aligned with the IDMEF format and have n dimensions. These high-dimensional domains are difficult to be geometrically represented. As a consequence, we restrain our representation to generic sets represented by circles 6.4.

Let's also remind that responsibility domains of deployed p PEPs are expressed as follow:
$RD(PEP_p) = \bigcup_{j\in[1\ldots n_r^p]} < d_{ji}^p >_{i\in[1\ldots n]}$. n_r^p is the number of vectors in $RD(PEP_p)$.
n is the IDMEF dimension. d_{ji}^p are the aligned dimension values of PEP_p.

- **Disjoint responsibility domains :** A responsibility domain is disjoint from another domain if their vectors are totally different.
  - $RD(PEP_2)$ and $RD(PEP_3)$ are ***Disjoint domains*** (we denote it $RD(PEP_2) \cap RD(PEP_3) = \varnothing$) , iff

$$\begin{gathered}\forall j' \in [1,\ldots,n_r^2], j'' \in [1,\ldots,n_r^3] \text{ and } \forall i \in [1,\ldots,n] \\ \text{such that } < d_{j'i}^2 > \neq < d_{j''i}^3 >\end{gathered} \tag{6.11}$$

- **Joint responsibility domains:**A responsibility domain is joint with another domain when it is partially (e.g. $RD(PEP_3)$ and $RD(PEP_4)$) or totally (e.g. $RD(PEP_1)$ and $RD(PEP_2)$) covered by this latter. We denote it $RD(PEP_3) \cap RD(PEP_4) = \mathcal{I}_{3,4}$.
  Let's $min(RD(PEP_3), RD(PEP_4))$ be the responsibility domain whose dimension is the minimum one. In our case, $min(RD(PEP_3), RD(PEP_4)) = RD(PEP_4)$
  - $RD(PEP_3)$ and $RD(PEP_4)$ are ***partially joint domains*** , iff

$$\begin{array}{c} \exists j' \in [1, \ldots, n_r^3] \text{ and } j'' \in [1, \ldots, n_r^4] \\ \text{such that } < d^2_{j'i} > \subseteq < d^1_{j''i} >, \ \forall i \in [1 \ldots n] \end{array} \tag{6.12}$$

In this case, $dim(\mathcal{I}_{3,4}) < dim(min(RD(PEP_3), RD(PEP_4)))$.

  - $RD(PEP_2)$ and $RD(PEP_1)$ are ***totally joint domains*** , iff

$$\begin{array}{c} \forall j'' \in [1, \ldots, n_r^2] \text{ and } j'' \in [1, \ldots, n_r^1] \text{ and } n_r^2 < n_r^1 \\ \text{such that } < d^2_{j''i} > \subseteq < d^1_{j'i} >, \ \forall i \in [1 \ldots n] \end{array} \tag{6.13}$$

In this case, $dim(\mathcal{I}_{1,2}) = dim(min(RD(PEP_1), RD(PEP_2)))$.

6.4.3 Common deployed security patterns description from Olympic Games use case

The Olympic Security Infrastructure is developed by **Atos** Company. Its main objective is to protect the IT infrastructure of the Olympic Games systems from any undesired and/or uncontrolled phenomena which can impact any parts of the result chain and associated services.
The SIEM infrastructure of Olympic Games IT system deal with a big challenge which is mainly due to the number of security event types (about 20,000), and the volume of generated events to be handled (around 11,000,000 alerts per day). However, the most critical aspect that a SIEM system faces in the Olympic Games is that those security events must be *processed* and *reacted* upon in real-time, [100].
The deployed security infrastructure of the Olympic Games IT system provides the administrators with defence-in-depth. In order to meet the requirements of such system, several PEPs are deployed such as firewalls, active directories, intrusion detection systems, etc.
From Figure 6.3, we highlight several security patterns that are commonly used in complex security architecture such as the Olympic Games IT system. We only discuss common patterns based on network firewall technology.
We distinguish three different security patterns that are detailed hereafter.

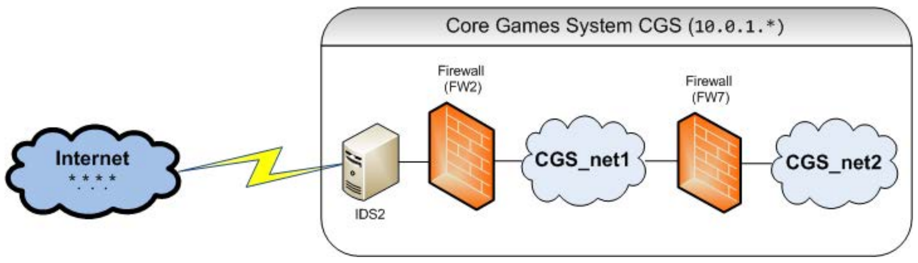


FIGURE 6.5 - First security architecture pattern: Pattern 1

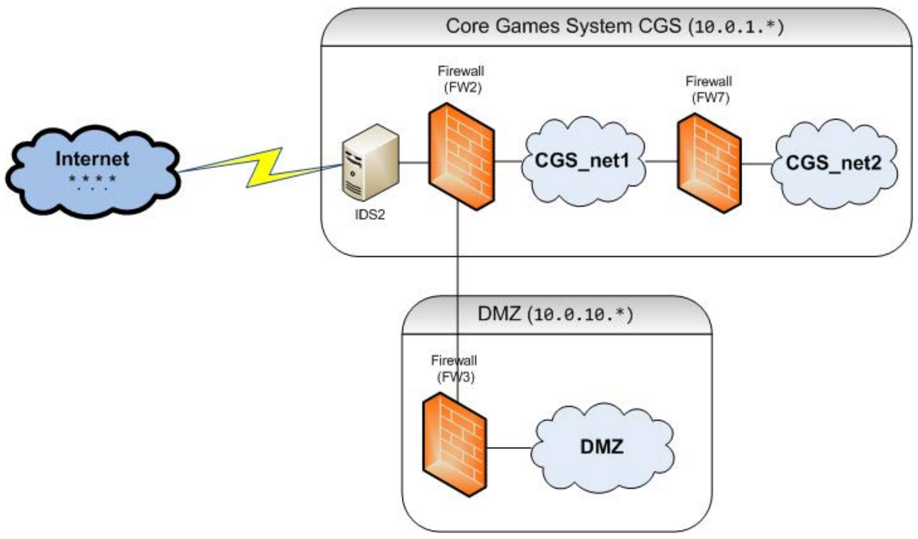


FIGURE 6.6 - Second security architecture pattern: Pattern 2.

Security Pattern 1

In this security pattern as described in Figure 6.5, the IDS (IDS2) is placed in front of the first firewall (FW2) which is controlling access to the entire CGS. Usually, such IDS placement in the architecture allows the security administrator to inspect all the traffic -unfiltered- passing through the network levels. The internal CGS network is divided in two sub-networks with different criticality. To meet the security requirements of CGS_net1 and CGS_net2, two firewalls, respectively FW2 and FW7 are deployed in serial. Such configuration enforces access control to CGS_net2 which is more critical.

Security Pattern 2

In this security pattern shown in Figure 6.6, the frontal firewall FW2 enforces the security of two different zones: the DMZ and the internal CGS network. Both

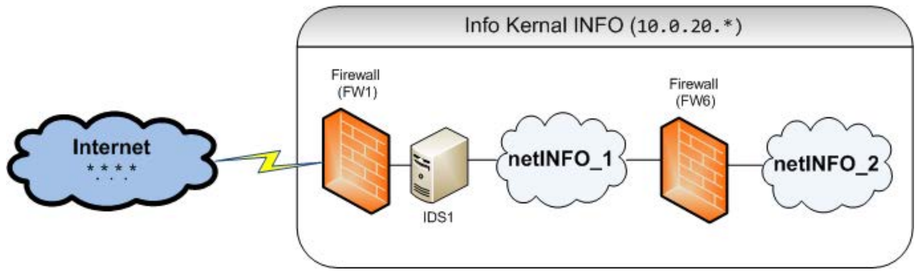


Figure 6.7 - Third security architecture pattern: Pattern 3.

of firewalls FW7 and FW3 protect these two zones which have different security requirements. Moreover, FW7 and FW3 basically inspect filtered traffic coming from FW2. In such security pattern, the firewall FW2 includes rules with larger responsibility domain than those of FW7 and FW3.

Security Pattern 3

We observe that in the security pattern shown in Figure 6.7, the IDS is placed after the frontal Firewall FW1. Therefore, the inspected traffic is the traffic which has been already filtered by firewall FW1. In this case, the generated alerts correspond to suspicious flow which has been already allowed by the firewall FW1.

6.4.4 Application of Enforcement-based Alert Correlation on multi-PEPs environment: Discussions and analysis

In the following, we represent the set of generated alerts by $\mathcal{A}$. $\mathcal{A}_1$ and $\mathcal{A}_2$ represent respectively the set of generated alerts by IDS_1 and IDS_2 respectively listed in pattern 3 and both of patterns 1 and 2.
We discuss in this part the impact of the relationships between the responsibility domains of multiple PEPs and the IDS placement on our alert correlation approach. Due to the non-disclosure of PEPs configurations and generated alerts, we maintain representing the responsibility domains of PEP and the set of generated alerts by abstract circles. As explained before, these circles are an abstract representation of multi-dimensional domains.

6.4.4.1 First pattern: discussion and analysis

As we explained before, in the first pattern, the IDS_2 is placed before the front-end firewall FW_2. As a consequence, the set of generated alerts $\mathcal{A}_2$ corresponds

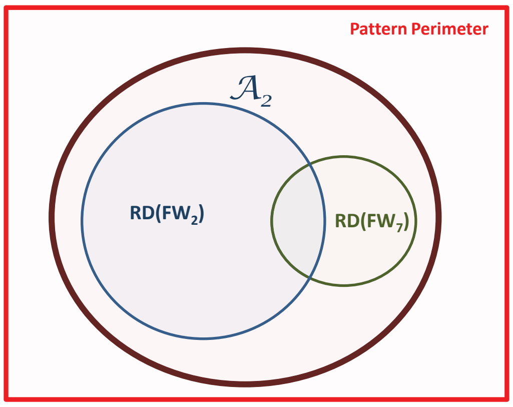


FIGURE 6.8 - Relationships between RD(PEP) deployed in pattern 1

to suspicious unfiltered traffic. Thus, $\mathcal{A}_2$ set includes in our case vectors of both $RD(FW_2)$ and $RD(FW_7)$. Since the front-end firewall FW_2 inspects global traffic, it is responsible of applying rules related to wide value range of selectors. This explains the dimension of $RD(FW_2)$ which covers an area almost equal to the network perimeter. The FW_7 inspects traffic related to specific traffic destined to a reduced zone in the supervised network. Thus, the implemented rules in FW_7 covers specific ranges of values for corresponding selectors. Therefore, the dimension of $RD(FW_7)$ is much less than the network perimeter and $RD(FW_2)$.

Enforcement-based Alert Correlation Advantages :
- Since the set of generated alerts $\mathcal{A}_2$ are correlated based on $RD(FW_2)$ and $RD(FW_7)$, it then becomes straightforward to process these alerts. In fact, we easily identify the firewall that will be able to process generated alerts and implement corresponding countermeasures.

- Based on the first pattern, our alert correlation approach allows setting a defense-in-depth strategy. The set $\mathcal{I}_{2,7}$ enable us correlating alerts that can be processed by both firewalls $RD(FW_2)$ and $RD(FW_7)$. In such case, the security administrator can decide to enforce the intrusion response decision using both of the deployed firewalls.

Enforcement-based Alert Correlation Disadvantages :
- Since our alert correlation approach closely depends on the approximation of the firewalls and then their configurations, it is possible that some generated alerts will not be correlated. This set of alerts requires more investigation. In fact, ignoring the default rule deployed in the firewall and the topology knowledge leads to a sub-estimation of the responsibility domain of the firewall. As demonstrated in Chapter 4, this sub-estimation depends on the approximation accuracy of the responsibility

domain.

6.4.4

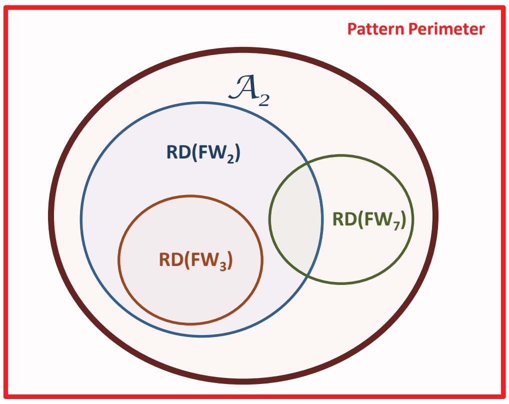


FIGURE 6.9 - Relationships between RD(PEP) deployed in pattern 2

As we explained before, the second pattern differs from the first pattern by the existing of the DMZ zone which is protected by a third firewall FW_3. In general, DMZ zone is composed of servers running critical publicly accessible services. As a consequence and in order to enforce the security of these services, the traffic destined to servers in DMZ zone is checked twice. Therefore, the $RD(FW_3)$ is totally joint to the $RD(FW_2)$. In such case, attacks against the DMZ zone servers can be mitigated either by the frontal firewall FW_2, or by the DMZ firewall FW_3, or when necessary by both of them.
In terms of enforcement-based alert correlation, the totally joint relations between $RD(FW_3)$ and $RD(FW_2)$ induces that all the alerts which are correlated based on $RD(FW_3)$ are immediately correlated based on $RD(FW_2)$. In fact, in this case, we have $\mathcal{I}_{2,3} = RD(FW_3)$.
Enforcement-based Alert Correlation advantages :
- Despite ignoring the network topology, we demonstrate the utility and applicability of our approach in the third pattern. In such case, our enforcement-based alert correlation supports the selection of proper PEP that are capable to process generated alerts.

Enforcement-based Alert Correlation disadvantages :
- Some of the correlated alerts based on the set $\mathcal{I}_{2,3}$ cannot be effectively processed by the DMZ firewall FW_3. In fact, it is possible that some of these alerts have to be only processed by the frontal firewall FW_2. An important recommendation in this case is to first mitigate the attacks at the level of the frontal firewall. Then,

investigations on countermeasures implementation have to be achieved in order to decide enforcing the policy at the DMZ firewall.

6.4.4.

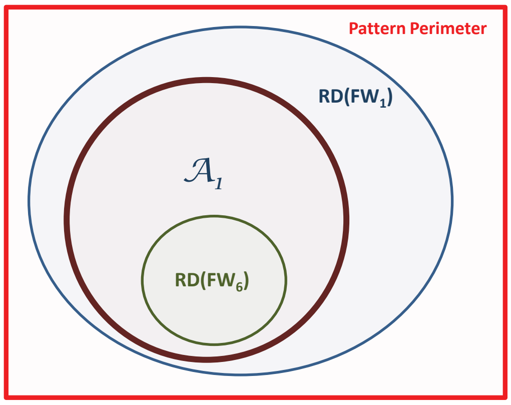


FIGURE 6.10 - Relationships between RD(PEP) deployed in pattern 3

As described before, in the third pattern, the IDS control filtered traffic by the firewall FW_1. As a consequence, the set of the generated alerts $\mathcal{A}_1$ is included in the responsibility domain of the frontal firewall FW_1. Obviously, all the generated alerts are correlated based on $RD(FW_1)$. Meanwhile, our enforcement-based alert correlation supports the administrator in identifying alerts that can be processed by the internal firewall FW_6.

6.4.5 Role of enforcement-based alert correlation in countermeasure implementations in a multi-PEP environment

Based on the analysis of the different pattern listed above, we conclude that the role of our enforcement-based alert correlation approach in countermeasure implementations in a multi-PEP environment is a *optimisation problem*. In fact, in a multi-PEP environment, alerts should be correlated while optimising the countermeasure implementation. In a real case scenario like the olympic games use case, we deal with multiple alerts and several deployed PEP that are responsible of enforcing the policy in order to mitigate detected attacks.
In such context, alert correlation should minimize the cost of implementing response decision. This cost covers the different modifications that should be brought to the security policy in order to process with generated alerts. Mainly, this cost represents the number of configuration modifications resulting of countermeasure implementations. For instance, modifying PEP's configuration is a challenging task for several

reasons: *i)* it is critical task to cope with because of related aspects such as conflict *ii)* it requires an in-depth analysis *iii)* it can be time consuming, etc. As a consequence, aggregate different countermeasures in order to process a number of alerts is a prominent solution. Hence, our enforcement-based alert correlation is able to support the security administrator in aggregating countermeasures in a multi-PEP environment.

Usually, we have:
$\mathcal{A} = \{A_1, A_2, \ldots, A_{n_a}\}$ represents the set of generated alerts
$\mathcal{PEP} = \{PEP_1, PEP_2, \ldots, PEP_{n_{pep}}\}$ represents the set of the deployed PEPs. We denote by $RD((PEP))$ the set of responsibility domains of PEPs in $\mathcal{PEP}$.
We note by $Corr(\mathcal{A}, RD((PEP)))$ the correlation function which correlated alerts in $\mathcal{A}$ with responsibility domains in $RD((PEP))$. This function results in a subsets of enforcement-based correlated alerts $\mathcal{A}^{ec} = \{A^{ec}1, A^{ec}1, \ldots, A^{ec}n_{a^{ec}}\}$ as shown in Equation 6.14.

$$Corr(\mathcal{A}, RD((PEP))) = \{A^{ec}_1, A^{ec}_2, \ldots, A^{ec}_{n_{a^{ec}}}\} = \mathcal{A}^{ec} \tag{6.14}$$

We model our maximisation problem as indicated in Equation 6.15:

$$\begin{aligned} &argmax(\mathcal{AM}(\mathcal{A}^{ec}, RD_{inf}(\mathcal{PEP})))) = \\ &\quad \{\forall\ A^{ec}_j |\ \ \exists\ PEP_k \in \mathcal{PEP} : \mathcal{AM}(A^{ec}_j, PEP_k) \approx Th_{AM}\} \end{aligned} \tag{6.15}$$

$\mathcal{AM}(\mathcal{A}^{ec}, RD_{inf}(\mathcal{PEP})))$ represents the overall aggregation metric of correlated alerts. It follows the definition of the function expressed in Equation 6.10. $RD_{inf}(\mathcal{PEP})$ is the set of the configurations of PEPs in $\mathcal{PEP}$.
Our optimisation problem aims at correlating alerts while ensuring an optimal rate of countermeasure modifications while processing simultaneously a common group of correlated alerts.

6.5 Discussions and conclusion

This chapter presents the application of the enforcement-based alert correlation in a single-PEP environment and multi-PEP environment. Our approach is not only efficient to determine and select the PEP that is capable to process a group of correlated alerts. But also, it allows the identification of the configured rules that should be modified in order to implement a countermeasure. It allows the aggregation of response decision implementation. As detailed in Chapter 2, most of the existing alert correlation techniques are able to support the administrator in response decision by making the number of generated alerts manageable. These approaches are based on the alert's content independent from the environment. Model-based approach developed in [78, 80] is the only correlation method dealing with information about the supervised information system. However, it does not make easier the response decision process. The advantage of our approach, comparing it to existing approaches, is the identification of PEP capable to process correlated alerts. Moreover, it allows the selection of appropriate configured rules that should be modified in order to implement countermeasures.

We provided an in-depth analysis of our correlation's role in response decision implementation. However, since we do not take into account the access control decision (axiom 3 in Chapter 4) when building the PEP's approximations, our approach is efficient in the sense that it allows the administrator to identify rules to be modified. In order to decide about the modification of the identified rule's decision, external knowledge, such as the attack impact and the vulnerability, is required. In this dissertation, we suppose that such knowledge is not available. Therefore, we do not deeply examine the impact of the PEP's enforced access control decisions on the role of our enforcement-based alert correlation in response decision.

Chapter 7

Conclusions and Perspectives

The open mind never acts: when we have done our utmost to arrive at a reasonable conclusion, we still - must close our minds for the moment with a snap, and act dogmatically on our conclusions.

George Bernard Shaw - 1856-1950

Contents

In conclusion, we summarize how each of the research topics presented in the first chapter has been pursued, and the contributions which have resulted. Next, we reflect on how we can improve our contributions and provide new research directions.
Throughout this thesis, our main objective was to propose novel alert correlation techniques to support the security administrator dealing with the tremendous number of alerts. Two correlation approaches are presented:

- Honeypot-based alert correlation (Chapter 3): This approach leverages the honeypot datasets in order to enhance alert correlation with knowledge about 'attackers';
- Enforcement-based alert correlation (Chapter 5): This approach deploys the knowledge about policy enforcement capabilities of the supervised network in order to enhance alert correlation with knowledge about the 'defender'.

7.1 Contributions

7.1.1 Honeypot-based alert correlation

The first proposed approach is related to the use of honeypot datasets in alert correlation as shown in Figure 7.1. In response to Objective 1 we have shown the usabilit

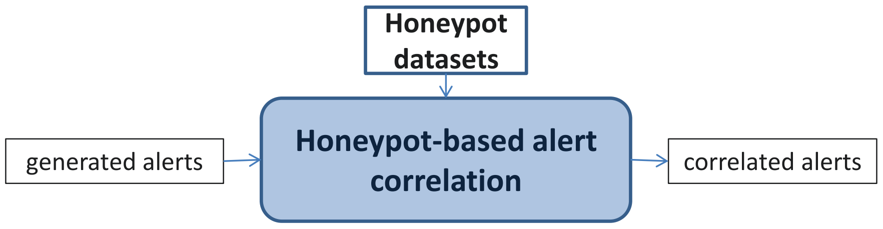


FIGURE 7.1 - Honeypot-based alert correlation.

of information gathered through honeypots in improving alert's related information. This information includes more details about the attacker such as his behavior, his propagation vector, his location, etc. As shown in Figure 7.1, the honeypot-based alert correlation explores the honeypot datasets in order to correlate the generated alerts.
We propose an alert enrichment process that allows the collection of appropriate information related to the locally generated alerts. This enrichment process is mainly based on enrichment selectors such us the ip source of the alert and its classification. We propose three filtering processes in order to increase the accuracy of alert's related knowledge. The designed filters are: temporal, semantic and configuration filters.
Our experiments were conducted on four honeypot-datasets. Experimental results allow us identifying honeypots' limitations. These limitations avoid improving alert's related information and enhancing alert correlation. We identified four major limitations that are:

- coverage limitation: It prevents alert correlation from efficiently study the causality link between alerts and then correlate alerts;

- unfilled attributes and references limitations: It impacts the overall quality of our honeypot-based alert correlation by making challenging the collection of additional information;
- lack of standard representation of data: It avoids the design of inference rules in order to facilitate the causality analysis between local and global events.;
- cross-references limitation: It avoids cooperation between different honeypot datasets.

These limitations were an impediment for our honeypot-based alert correlaion. It would be preferable if the honeypot datasets include better quality of data.
However, honeypots are still evolutionary and are being ameliorated to cover additional security application. But, in the context of our honeypot-based alert correlation, concise and complete information is required in order to enhance our knowledge about the information related to the alert.

7.1.2 Enforcement-based alert correlation

The second proposed approach is related to the use of policy enforcement capabilities in alert correlation as shown in Figure 7.2, in response to *Objective 2 and Objective*

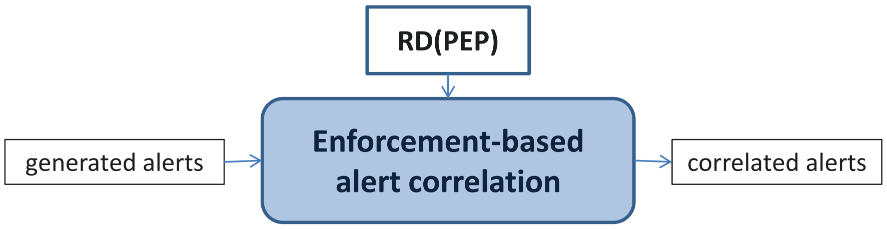


FIGURE 7.2 - Enforcement-based alert correlation.

responsibility domain of deployed PEP in order to correlate generated alerts. In this context, we first propose to model these capabilities referring to a proposed PEP model detailed in Chapter 4. We design a PEP model which illustrates the PEP's responsibility domain. Since the responsibility domain of the PEP is an abstraction of its capabilities in enforcing the security policy, we propose to approximate it by setting up to main approximations which are the rule-based approximations and the selector-based approximation. We study the coverage relationships between the different proposed approximations. The advantage of our methodology to approximate $RD(PEP)$ is the performance in a 'blind manner'. Also, the consideration of the PEP configuration makes the approximations more useful for response decision. However, it would be possible if we consider the external knowledge in order to build the approximations following a top down approach.

We investigate the application of the proposed PEP model in correlating alerts. Thus, we detail the correlation approach based on this model in Chapter 5. Two applications has been analyzed: a single-PEP and a multi-PEP environment in Chapter 6. We propose the role of this approach in response decision support. We provided an in-depth analysis of our correlation's role in response decision implementation. However, since we do not take into account the access control decision when building the PEP's approximations, our approach is efficient in the sense that it allows the administrator to identify rules to be modified.

7.2 Perspectives

The final section of this dissertation opens new areas of research showing problems that remain unsolved and which can be addressed in future works:

7.2.1 Application on other PEP classes

Expand the application of the enforcement-based alert correlation to include other PEP classes: We conducted our analysis based on a single PEP class which is the network firewalls. Although this class is commonly used in networks, there exists other PEP that should be analyzed in order to apply our responsibility domain approximations. The integration of other PEP class allows the application of our alert correlation on heterogeneous security systems.
In Tables 5.1, 5.2 and 5.3, we briefly introduce other PEP classes mainly ModSecurity and LDAP. Similar to network firewall such as iptable, ModSecurity PEP has five principle selectors that are: Remote-Address, Remote-port, Remote-user, hostname, Request-Body. LDAP's selectors are in general represented by string variables such as the common name (cn), the username, etc. In this case, it is possible to define a domain decomposition based on keywords or a hierarchy of values. For instance, LDAP attributes can be defined following a tree decomposition as shown in figure 7.3.

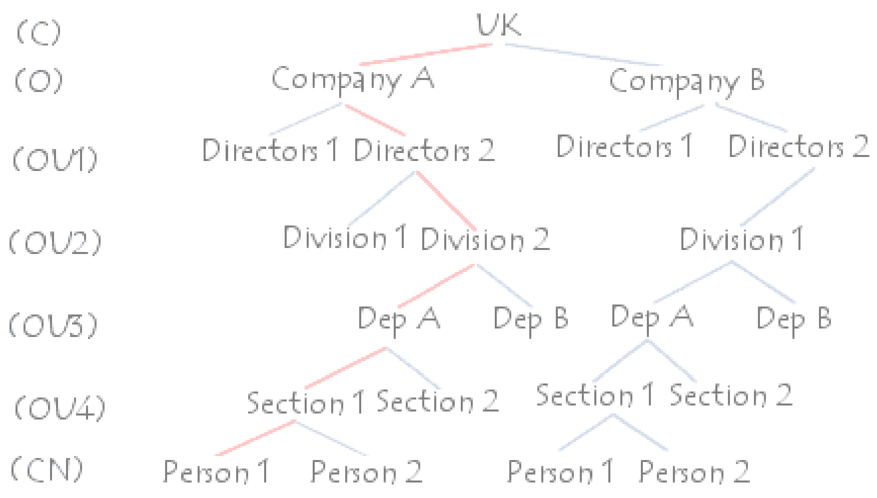


FIGURE 7.3 - LDAP tree decomposition.

The generalization function defined in this case could be the choice of a higher level for a specific selector.

7.2.2 Select appropriate responsibility domain approximation

Define criteria and metrics which allow the identification of the appropriate approximation function to be applied. The proposed coverage increases metric allows evaluating the differences between approximations independent of the context. Since our proposed Responsibility Domain approximations are built following a bottom up approach, it is possible that the security administrator prefer one approximation rather than another based on the configuration and its coverage. However, expert knowledge is required in order to decide about the appropriate approximation. Thus including external knowledge such as network topology and routing information and analyze its impact on the proposed enforcement-based alert correlation approach. Using such external knowledge, the responsibility domain approximations can be built following a top down approach.

7.2.3 Identification of countermeasure implementations

Develop a methodology which automatically identifies the countermeasure to be implemented in order to process correlated alerts. We studied the role of the enforcement-based alert correlation in response decision support. We point out that it supports the security administrator in identifying configured rules that should be modified or should be configured. We also demonstrate how our proposed alert correlation can be considered as an optimization problem as defined in Section 6.4.5:

$$\begin{aligned}&argmax(\mathcal{AM}(\mathcal{A}^{ec}, RD_{inf}(\mathcal{PEP})))) = \\ &\qquad \{\forall\ A_j^{ec}|\ \ \exists\ PEP_k \in \mathcal{PEP} : \mathcal{AM}(A_j^{ec}, PEP_k) \approx Th_{AM}\}\end{aligned}$$

This optimization system allow us identifying the PEPs with a minimum modifications that should be brought to the configuration in order to process correlated alerts. However, since we ignore the access control decision applied by the PEP when enforcing security policy, our approach does not allow the identification of the concrete countermeasure implementation.

List of Figures

List of Tables

Glossary of Acronyms

FW	Firewall
IDS	Intrusion Detection System
PEP	Policy Enforcement Point
RD	Responsibility Domain
SIEM	Security Information and Event Management

Glossary of Terms

Attack	An attempt to gain unauthorized access to system services, resources, or information, or an attempt to compromise system integrity.
Botnet	A number of Internet computers that, although their owners are most time unaware of it, have been set up to forward transmissions (including spam or viruses) to other computers on the Internet.
Countermeasure	Actions, devices, procedures, or techniques that meet or oppose (i.e., counters) a threat, a vulnerability, or an attack by eliminating or preventing it, by minimizing the harm it can cause, or by discovering and reporting it so that corrective action can be taken.
Denial of Service	An attempt against the availability of a particular system. The main objective is to disrupt service and network availability by attempting to reduce a legitimate user's bandwidth, or preventing access to service or system.
Honey-pot	Systems or accounts used to lure the intruder into pursuing a decoy controlled environment directly of his own volition. Honey-pots are placed near assets requiring protection, are made attractive, and are fully instrumented for intrusion detection and back tracking.
Host-based intrusion detection system (HIDS)	IDSs which operate on information collected from within an individual computer system. This vantage point allows host-based IDSs to determine exactly which processes and user accounts are involved in a particular attack on the Operating System.

Infectious threats	It corresponds to viruses, worms, trojan horses and other similar threats.
Intrusion detection system (IDS)	Hardware or software product that gathers and analyzes information from various areas within a computer or a network to identify possible security breaches, which include both intrusions (attacks from outside the organizations) and misuse (attacks from within the organizations.)
Intrusion prevention systemv(IPS)	Systems which can detect an intrusive activity and can also attempt to stop the activity, ideally before it reaches its targets.
Network-based intrusion detection systems (NIDS)	IDSs which detect attacks by capturing and analyzing network packets.
Policy Enforcement Point	Logical entity or place on a server that enforces policies for admission control and decisions in response to a request from a user wanting to access a resource on a computer or network server.
Risk	The level of impact on organizational operations (e.g., functions, image, or reputation), organizational assets, or individuals resulting from the operation of an information system given the potential impact of a threat and the likelihood of such threat occurring.
Security Policy	A set of criteria for the provision of security services. It defines and constrains the activities of a data processing facility in order to maintain a condition of security for systems and data.

Security Information and Event Management (SIEM)	Integrated information security oriented platform that offer the following services: Log management (log collection, storage, organization and retrieval); IT regulatory compliance (audit, validation or violation identification); Event correlation (normalization, fusion, verification, analysis); Active response (decision analysis, counter-measure response, prioritization); and Endpoint security (monitoring, updating, configuration).
Threat	A circumstance or event with the potential to adversely impact organizational operations (e.g., mission, functions, reputation), as well as organizational assets, individuals, other organizations, or even the Nation through unauthorized access, destruction, disclosure, modification of information, and/or denial of service.

Bibliography

[1] M. DCU and NUS. (2014) The Link between Pirated Software and Cybersecurity Beaches. [Online]. Available: http://play-it-safe.net/

[2] C. systems, "Cisco Security Information and Event Management Deployment Guide," http://www.cisco.com/c/dam/en/us/solutions, 2010.

[3] J. Tarala, "Implementing the 20 Critical Controls with Security Information and Event Management (SIEM) systems," https://www.sans.org/reading-room/analysts-program/siem-systems-arcsight, 2011.

[4] R. Kissel, *Glossary of Key Information Security Terms.* National Institute of Standards and Technology. U.S. Department of Commerce, 2013.

[5] "Information processing systems- ISO reference model- part 2: Security architecture. Standard, (7498-2)," 1989.

[6] J. R. McCumber, "Information Systems Security: A Comprehensive Model," in *Proceeding of the 14th National Computer Security Conference, NIST, Baltimore*, 1991.

[7] "Information technology, Security techniques, Information security management systems, Requirements," www.iso.org, 2013.

[8] S. Glass, T. Hiller, S. Jacobs, and C. Perkins, "RFC 2977: Mobile IP Authentication, Authorization, and Accounting Requirements," 2000.

[9] S. Tzu, *The Art of War.* Shambhala, 2005.

[10] T. Grance, J. Hash, M. Stevens, K. O'Neal, and N. Bartol, "Guide to Information Technology Security Services, NIST-800-35," 2003.

[11] "eXtensible Access Control Markup Language (XACML)," https://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf, February 2003.

[12] V. Zaborovsky, V. Mulukha, and E. Silinenko, “Access Control Model and Algebra of Firewall Rules,” 2011.

[13] R. Boutaba and A. Polyrakis, “Towards extensible policy enforcement points,” in *Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, ser. POLICY ’01. Springer-Verlag, London, UK, 2001, pp. 247–262. [Online]. Available: http://dl.acm.org/citation.cfm?id=646962.712111

[14] H. Hamed and E. Al-Shaer, “Taxonomy of conflicts in network security policies,” *Communications Magazine, IEEE*, vol. 44, no. 3, pp. 134–141, 2006.

[15] G. Betarte, A. Gatto, R. Martinez, and F. Zipitria, “ACTkit: A Framework for the Definition and Enforcement of Role, Content and Context-based Access Control Policies,” *Latin America Transactions, IEEE*, 2012.

[16] J. Anderson, “Computer Security Threat Monitoring and Surveillance,” 1980.

[17] D. Denning, “An Intrusion-Detection Model,” *IEEE Transactions on Software Engineering*, 1987.

[18] R. Bace and P. Mell, “Intrusion Detection Systems, NIST-800-31,” 2003.

[19] International Standard Organization (ISO) and International Electrotechnical Commission (IEC), “Information technology - Security techniques - IT intrusion detection framework (ISO/IEC TR 15974-2002,” 2002.

[20] T. F. Lunt, “A survey of intrusion detection techniques,” *Computers & Security*, vol. 12, no. 4, pp. 405 – 418, 1993.

[21] B. Mukherjee, L. Heberlein, and K. Levitt, “Network intrusion detection,” *Network, IEEE*, vol. 8, no. 3, pp. 26–41, 1994.

[22] H. Debar, M. Dacier, and A. Wespi, “Towards a taxonomy of intrusion-detection systems,” *Comput. Netw.*, vol. 31, no. 9, pp. 805–822, 1999.

[23] C. Krügel and T. Toth, “A survey on intrusion detection systems,” *TU VIENNA , AUSTRIA*, pp. 22–33, 2000.

[24] H. Debar, M. Dacier, and A. Wespi, “A revised taxonomy for intrusion detection systems,” *Annals of Telecommunications*, vol. 55, June 2000.

[25] S. Mukkamala and A. Sung, “A comparative study of techniques for intrusion detection,” in *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on*, 2003, pp. 570–577.

[26] J. M. Estevez-Tapiador, P. Garcia-Teodoro, and J. E. Diaz-Verdejo, “Anomaly detection methods in wired networks: a survey and taxonomy,” *Computer Communications*, vol. 27, no. 16, pp. 1569 – 1584, 2004.

[27] A. Mishra, K. Nadkarni, and A. Patcha, “Intrusion detection in wireless ad hoc networks,” *Wireless Communications, IEEE*, vol. 11, no. 1, pp. 48–60, 2004.

[28] N. Delgado, A. Gates, and S. Roach, “A taxonomy and catalog of runtime software-fault monitoring tools,” *Software Engineering, IEEE Transactions on*, vol. 30, no. 12, pp. 859–872, 2004.

[29] P. Kabiri and A. A. Ghorbani, “Research on intrusion detection and response: A survey,” *International Journal of Network Security*, vol. 1, pp. 84–102, 2005.

[30] A. Patcha and J.-M. Park, “An overview of anomaly detection techniques: Existing solutions and latest technological trends,” *Computer Networks*, vol. 51, no. 12, pp. 3448 – 3470, 2007.

[31] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez, “Anomaly-based network intrusion detection: Techniques, systems and challenges,” *Computers & Security*, vol. 28, no. 1-2, pp. 18 – 28, 2009.

[32] M. Xie, S. Han, B. Tian, and S. Parvin, “Anomaly detection in wireless sensor networks: A survey,” *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1302 – 1325, 2011.

[33] A. Ghorbani, W. Lu, and M. Tavallaee, *Network Intrusion Detection and Prevention*, ser. Advances in Informatio Security. Springer, 2010, vol. 47.

[34] P. Kazienko and P. Dorosz, “Intrusion Detection Systems (IDS) - Part 2: Classification, methods, techniques,” July 2004.

[35] A. Sundaram, “An introduction to intrusion detection,” 1996.

[36] J. McHugh, “Intrusion and intrusion detection,” *International Journal of Information Security*, vol. 1, pp. 14–35, 2001.

[37] A. Jones and R. Sielken, “Computer system intrusion detection: A survey,,” September Charlottesville, VA, 2000.

[38] P. M. Mell, “An Overview of Issues in Testing Intrusion Detection Systems, NISTIR-7007,” 2003.

[39] M. Erlinger, B. Feinstein, G. Matthews, S. Staniford, and A. Walther, “Experiences implementing a common format for ids alerts,” in *Computer Security Applications Conference, 2001. ACSAC 2001. Proceedings 17th Annual*, Dec 2001, pp. 113–113.

[40] D. Kim, H. Bang, and J.-C. Na, "Intrusion alert normalization method using AWK scripts and attack name database," in *Advanced Communication Technology, 2005, ICACT 2005. The 7th International Conference on*, vol. 1, Feb 2005, pp. 608–611.

[41] B. Bosch, "Intrusion detection parameterization exchange data model," in *MIPRO, 2012 Proceedings of the 35th International Convention*, May 2012, pp. 1490–1495.

[42] S.-K. Park, K.-Y. Kim, J.-S. Jang, and B.-N. Noh, "Supporting interoperability to heterogeneous ids in secure networking framework," in *Communications, 2003. APCC 2003. The 9th Asia-Pacific Conference on*, vol. 2, Sept 2003, pp. 844–848 Vol.2.

[43] C.-Y. Ho, Y.-C. Lai, I.-W. Chen, F.-Y. Wang, and W.-H. Tai, "Statistical analysis of false positives and false negatives from real traffic with intrusion detection/prevention systems," *Communications Magazine, IEEE*, vol. 50, no. 3, pp. 146–154, March 2012.

[44] O. Abouabdalla, H. El-Taj, A. Manasrah, and S. Ramadass, "False positive reduction in intrusion detection system: A survey," in *Broadband Network Multimedia Technology, 2009. IC-BNMT '09. 2nd IEEE International Conference on*, Oct 2009, pp. 463–466.

[45] O. B. Remi-Omosowon, "Statistical analysis of snort alerts," Plymouth, UK, 2009.

[46] Georgios P. Spathoulas and Sokratis K. Katsikas, "Reducing false positives in intrusion detection systems," *computer & Security 29*, pp. 35–44, 2010.

[47] N. Bakar, B. Belaton, and A. Samsudin, "False positives reduction via intrusion alert quality framework," in *Networks, 2005. Jointly held with the 2005 IEEE 7th Malaysia International Conference on Communication., 2005 13th IEEE International Conference on*, vol. 1, Nov 2005.

[48] H. Debar and A. Wespi, "Aggregation and correlation of intrusion-detection alerts," in *Recent Advances in Intrusion Detection*, ser. Lecture Notes in Computer Science, W. Lee, L. Mé, and A. Wespi, Eds. Springer Berlin Heidelberg, 2001, vol. 2212, pp. 85–103.

[49] A. Hofmann and B. Sick, "Online intrusion alert aggregation with generative data stream modeling," *Dependable and Secure Computing, IEEE Transactions on*, vol. 8, no. 2, pp. 282–294, March 2011.

[50] H. Ren, N. Stakhanova, and A. A. Ghorbani, "An online adaptive approach to alert correlation," *Springer-Verlag Berlin Heidelberg*, 2010.

[51] N. Bakar and B. Belaton, "Towards implementing intrusion alert quality framework," in *Distributed Frameworks for Multimedia Applications, 2005. DFMA '05. First International Conference on*, Feb 2005, pp. 198–205.

[52] L. Sptizner, *Honeypots: Tracking Hackers.* Addison-Wesley, 2001.

[53] D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levine, and H. Owen, "HoneyStat: Local Worm Detection Using Honeypots," in *Recent Advances in Intrusion Detection, RAID*, 2004.

[54] F. Pouget and M.Dacier, "Honeypot-based Forensics," in *AusCERT Asia Pacific Information technology Security Conference*, 2004.

[55] S. Krasser, J. B. Grizzard, H. L. Owen, and J. G. Levine, "The Use of Honeynets to Increase Computer Network Security and User Awareness," *Journal of Security Education*, vol. 1, pp. 23 – 37, 2005.

[56] I. Mokube and M. Adams, "Honeypots: Concepts, Approaches, and Challenges," *ACMSE 2007,Winston-Salem, North Carolina, USA*, March 2007.

[57] C. Seifert, I. Welch, and P. Komisarczuk, *HoneyC - The low-interaction client honeypot.* Addison-Wesley, 2006.

[58] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling, "The Nepenthes Platform: An Efficient Approach to Collect Malware," in *Recent Advances in Intrusion Detection*, ser. Lecture Notes in Computer Science, D. Zamboni and C. Kruegel, Eds. Springer Berlin Heidelberg, 2006, vol. 4219, pp. 165–184.

[59] C. Leita and M. Dacier, "SGNET: a worldwide deployable framework to support the analysis of malware threat models," in *7th European Dependable Computing Conference (EDCC)*, May 2008.

[60] C. Leita, M. Dacier, and F. Massicotte, "Automatic handling of protocol dependencies and reaction to 0-day attacks with ScriptGen based honeypots," in *Recent Advances in Intrusion Detection, RAID*, 2006.

[61] G. Portokalidis, A. Slowinska, and H. Bos, "Argos: an emulator for fingerprinting zero-day attacks," in *ACM Sigops EuroSys*, 2006.

[62] Caputre-HPC. [Online]. Available: www.nz-honeynet.org/capture.html

[63] J. R. Crandall and F. T. Chong, "A Security Assessment of the Minos Architecture," *SIGARCH Comput. Archit. News*, vol. 33, pp. 48–57, 2005.

[64] C. Leita, U. Bayer, and E. Kirda, "Exploiting diverse observation perspectives to get insights on the malware landscape," in *DSN 2010, 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, June 2010.

[65] L. Spitzner, "Specter: A Commercial Honeypot Solution for Windows," http://www.symantec.com/connect/articles/specter-commercial-honeypot-solution-windows, April 2003.

[66] C. Hoepers, "Honeynets and Honeypots: Companion Technology for Detection and Response," *AusCERT2004 Conference, Technical Stream*, May 2004.

[67] "Mwcollect," http://directory.fsf.org/wiki/Mwcollect, 2013.

[68] "honeytrap - a dynamic meta-honeypot daemon," http://honeytrap.carnivore.it/, 2013.

[69] H. Debar, D. Curry, and B. Feinstein, "The Intrusion Detection Message Exchange Format (IDMEF)," http://tools.ietf.org/pdf/rfc4765.pdf, 2007.

[70] M. Wood and M. Erlinger, "Intrusion Detection Message Exchange Requirements," http://tools.ietf.org/pdf/rfc4766.pdf, 2007.

[71] P. Chifflier and S. Tricaud, "Intrusion detection systems correlation: a weapon of mass investigation," *CanSecWest'08*, Paris, France, 2008.

[72] Y. B. Mustapha, H. Debar, and G. Jacob, "Limitations of honeypot/honeynet databases to enhance alert correlation," *MMM-ACNS*, St. Petersburg, Russia, 2012.

[73] A. Valdes and K. Skinner, "Probabilistic Alert Correlation," in *Recent Advances in Intrusion Detection*, ser. Lecture Notes in Computer Science, W. Lee, L. Mé, and A. Wespi, Eds., vol. 2212. Springer Berlin Heidelberg, 2001, pp. 54–68.

[74] S. J. Templeton and K. Levitt, "A requires/provides model for computer attacks," in *Proceedings of the 2000 Workshop on New Security Paradigms*, ser. NSPW '00. ACM, Ireland, 2000, pp. 31–38.

[75] F. Cuppens, "Managing alerts in a multi-intrusion detection environment," in *Proceedings of the 17th Annual Computer Security Applications Conference*, ser. ACSAC '01. IEEE Computer Society, December Washington, DC, USA, 2001.

[76] F. Cuppens and R. Ortalo, "LAMBDA: A Language to Model a Database for Detection of Attacks ," *Recent Advances in Intrusion Detection*, vol. 1907, pp. 197–216, 2000.

[77] P. Kabiri and A. A. Ghorbani, "A Rule-Based Temporal Alert Correlation System," pp. 66 –72, 2005.

[78] B. Morin, L. Mé, H. Debar, and M. Ducassé, "M2D2: A formal data model for ids alert correlation," in *In Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002)*, 2002, pp. 115–137.

[79] B. Morin and H. Debar, "Correlation of intrusion symptoms: An application of chronicles." in *Recent Advances in Intrusion Detection, RAID*, ser. Lecture Notes in Computer Science, G. Vigna, E. Jonsson, and C. Kruegel, Eds., vol. 2820. Springer, 2003, pp. 94–112.

[80] Benjamin Morin and Ludovic Mé and Hervé Debar and Mireille Duccassé, "M4D4: a Logical Framework to Support Alert Correlation in Intrusion Detection," *Information Fusion*, vol. 10, pp. 285–299, 2009.

[81] F. J. Mora-Gimeno, F. Maciá-Pérez, I. Lorenzo-Fonseca, J. A. Gil-Martínez-Abarca, D. Marcos-Jorquera, and V. Gilart-Iglesias, "Security Alert Correlation Using Growing Neural Gas," in *Proceedings of the 4th International Conference on Computational Intelligence in Security for Information Systems*, ser. CISIS'11. Springer-Verlag, Spain, 2011, pp. 76–83.

[82] S. Cheung, U. Lindqvist, and M. W. Fong, "Modeling multistep cyber attacks for scenario recognition." in *DISCEX (1).* IEEE Computer Society, 2003.

[83] P. M. Comparetti and F. Maggi, "Using WOMBAT APIs on Real-World Tasks," *The second WOMBAT Workshop*, pp. 67–81, September 2009.

[84] L. Li, H. Sun, and Z. Zhang, "The research and design of honeypot system applied in the lan security," *2nd International Conference on Software Engineering and Service Science (ICSESS), 2011 IEEE*, 2011.

[85] C. Leita and M. Cova, "HARMUR: Storing and analyzing historic data on malicious domains," *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, pp. 46–53, Salzburg, Austria, 2011.

[86] S. Zanero and P. M. Comparetti, "The WOMBAT API: querying a global network of advanced honeypots," in *BlackHat DC*, 2010.

[87] P. A. Porras, M. W. Fong, and A. Valdes, "A mission-impact-based approach to infosec alarm correlation," in *In Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID)*, ser. Lecture Notes in Computer Science, A. Wespi, G. Vigna, and L. Deri, Eds., vol. 2516. Springer Berlin Heidelberg, 2002, pp. 95–114.

[88] M. Shin and K. Jeong, "Alert Correlation Analysis in Intrusion Detection," in *Advanced Data Mining and Applications*, ser. Lecture Notes in Computer Science, X. Li, O. Zaïane, and Z.-h. Li, Eds. Springer Berlin Heidelberg, 2006, vol. 4093, pp. 1049–1056.

[89] F. Cuppens and A. Miège, "Alert Correlation in a Cooperative Intrusion Detection Framework," in *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, ser. SP '02. IEEE Computer Society, Washington, DC, USA, 2002, pp. 202–215.

[90] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence, "RFC 2904: AAA Authorization Framework," http://tools.ietf.org/pdf/rfc2904.pdf, 2000.

[91] A. Abou El Kalam, R. El Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarteand, A. Miege, C. Saure, and G. Trouessin, "Organization Based Access Control," in *4th International Workshop on Policies for Distributed Systems and Networks (Policy 2003)*. IEEE, 2003, pp. 120–131.

[92] P. Samarati and S. Vimercati, "Access Control: Policies, Models, and Mechanisms," in *Revised versions of lectures given during the IFIP WG 1.7 International School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design: Tutorial Lectures*, ser. FOSAD '00. Springer-Verlag, London, UK, 2001, pp. 137–196.

[93] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models," *Computer*, vol. 29, no. 2, pp. 38–47, Los Alamitos, CA, USA, 1996.

[94] J. Garcia Alfaro, F. Cuppens, N. Cuppens-Boulahia, and S. Preda, "MIRAGE: a management tool for the analysis and deployment of network security policies," in *3rd SETOP International Workshop on Autonomous and Spontaneous Security (Co-located with ESORICS 2010)*, Springer, Ed., vol. 6514, 2010, pp. 203 – 215.

[95] M. Cotton, L. Eggert, J. Touch, M. Westerlund, and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry," http://tools.ietf.org/pdf/rfc6335.pdf, 2011.

[96] J. Garcia Alfaro, F. Cuppens, and N. Cuppens-Boulahia, "Complete analysis of configuration rules to guarantee reliable network security policies," *International Journal of Information Security*, vol. 7, pp. 103–122, 2008.

[97] R. Braden, "Requirements for Internet Hosts – Communication Layers," http://tools.ietf.org/pdf/rfc1122.pdf, 1989.

[98] E. S. Al-shaer and H. H. Hamed, "Discovery of policy anomalies in distributed firewalls," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 4, March 2004, pp. 2605–2616.

[99] Z. Pawlak, "Rough Sets," *International Journal of Information and Computer Sciences*, vol. 11, pp. 341–356, 1982.

[100] R. Rieke, L. Coppolino, A. Hutchison, E. Prieto, and C. Gaber, "Security and Reliability Requirements for Advanced Security Event Management," ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012.

[101] G. G. Granadillo, Y. B. Mustapha, N. Hachem, and H. Debar, "An ontology-driven approach to model SIEM information and operations using the SWRL formalism," *Int. J. Electron. Secur. Digit. Forensic*, Geneva, SWITZERLAND, 2012.

[102] Y. B. Mustapha, H. Debar, and G. Blanc, "Policy Enforcement Point Model," in *10th International Conference on Security and Privacy in Communication Networks*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, Beijing, China, 2014.

[103] Y. Ben Mustapha and H. Debar, "Service Dependencies-Aware Policy Enforcement Framework Based on Hierarchical Colored Petri Net," *Internation Syposium on Communiction and Security*, vol. 377, Mysore, India, 2013.

[104] G. Gonzalez Granadillo, Y. Ben Mustapha, N. Hachem, and H. Debar, "An ontology-based model for siem environments," in *Global Security, Safety and Sustainability and e-Democracy*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, C. Georgiadis, H. Jahankhani, E. Pimenidis, R. Bashroush, and A. Al-Nemrat, Eds. Springer Berlin Heidelberg, 2012.

[105] N. Hachem, Y. B. Mustapha, G. G. Granadillo, and H. Débar, "Botnets: Lifecycle and Taxonomy," in *6th Conference on Network Architectures and Information Systems Security (SAR-SSI)*, France, 2011, pp. 1–8.

[106] G. G. Granadillo, H. Débar, G. Jacob, C. Gaber, and M. Achemlal., "Individual Countermeasure Selection based on the Return On Response Investment Index." *International Conference Mathematical Methods, Models and Architectures for Computer Network Security*, pp. 156–170, 2012.

[107] G. G. Granadillo, H. Débar, G. Jacob, and L. Coppolino., "Combination Approach to Select Optimal Countermeasures based on the RORI Index." *Second International Conference on the Innovative Computing Technology*, pp. 38–45, 2012.

Author's publications

Publications in international peer-reviewed journals

1 G. Gonzalez Granadillo and Y. Ben Mustapha and N. Hachem and H. Debar. An Ontology-driven Approach to Model SIEM Information and Operations Using the SWRL Formalism *International Journal of Electronic Security and Digital Forensics*, vol. 4, no. 2/3, pp. 104–123. Inderscience, 2012, [101].

Publications in international peer-reviewed conferences

1 Yosra Ben Mustapha, Hervé Debar and Gregory Blanc, *Policy Enforcement Point Model*, 10th International Conference on Security and Privacy in Communication Networks, August 2014, [102].

2 Yosra Ben Mustapha, Hervé Debar, *Service Dependencies-Aware Policy Enforcement Framework based on Hierarchical Colored Petri Net*, International Symposium on Security in Computing and Communications, Mysore, India, August 2013, [103].

3 Yosra Ben Mustapha, Hervé Debar, Grégoire Jacob, *Limitation of Honeypot / Honeynet Databases to Enhance Alert Correlation*, 6th International Conference on Communications and Networking, St. Petersburg, Russia, October 2012, [72].

4 G. Gonzalez Granadillo and Y. Ben Mustapha and N. Hachem and H. Debar. An Ontology-based Model for SIEM Environments *Proceedings of the 7th International Conference in Global Security, Safety and Sustainability*, ICGS3 vol. 99, pp. 148–155. Lecture Notes in Computer Science, Springer, 2011, [104]

5 N. Hachem and Y. Ben Mustapha and G. Gonzalez Granadillo and H. Debar. Botnets: Lifecycle and Taxonomy *Proceedings of the 6th Conference on Network Architectures and Information Systems Security*, SAR-SSI IEEE, 2011, [105].

Contributions to European projects

1 VIS-SENSE. WP3 (D3.3) Attack Attribution Module. Technical report, VIS-SENSE European Project from the 7th Framework Program, as part of the ICT VIS-SENSE project (grant no. 257495).

2 VIS-SENSE. WP3 (D3.1) Sepecifications of the Network Analytics Algorithms. Technical report, VIS-SENSE European Project from the 7th Framework Program, as part of the ICT VIS-SENSE project (grant no. 257495).

3 VIS-SENSE. WP2 (D2.1) Network Iformation Sources. Technical report, VIS-SENSE European Project from the 7th Framework Program, as part of the ICT VIS-SENSE project (grant no. 257495).

Appendix A

Résumé en français

Corrélation d'alertes:
un outil efficace d'aide à la décision
pour répondre aux intrusions

Les failles de sécurité sont de plus en plus critiques pour les entreprises. Ainsi, les mesures de sécurité les plus appropriées sont indispensables pour assurer la sécurité des entreprises et de l'utilisateur. Des mécanismes de sécurité sont déployés dans les réseaux afin de prévenir les attaques et réagir contre elles lorsqu'elles sont détectées. Actuellement, les systèmes pour la Sécurité de l'Information et la Gestion des Evénements (connus en anglais par SIEM: Security Information and Event Management) sont au cœur des centres opérationnels de sécurité. Les SIEMs ont été développés pour assister le responsable de la sécurité des systèmes d'information (RSSI) dans la réalisation de ses responsabilités. Ils combinent deux aspects différents: le système de gestion de sécurité des données et le système de gestion des événements [2]. Les SIEMs se concentrent sur:

- la collection des journaux dévénements et leurs normalisations
- l'analyse et la corrélation des événements en provenance de différents capteurs (anti-virus, pare-feux, systèmes de détection d'intrusion, etc)
- l'aggrégation des événements et la création de rapports d'activité.

Les travaux de recherches sur les SIEMs se sont toujours concentrés sur la création de rapports d'attaques pour les entreprises dans le but de maintenir leur politique de sécurité à jour. La corrélation d'alertes représente le coeur des SIEMs. Toutefois, la plupart des travaux traitant ce sujet ne prennent en compte que les alertes générées par les sondes déployées dans le réseau. Ces techniques de corrélation ne considérent pas d'autres données telles que le comportement de l'attaquant, la capacité de réponses aux attaques, etc. Cependant, des recherches récentes sur la corrélation d'alertes (comme l'étude présentée [78, 80]) ont montré l'importance de ces données.

Dans cette thèse, nous proposons deux approches différentes de corrélation d'alertes permettant d'avoir un diagnostic plus précis des alertes générées par les SIEMs.
La première approche est basée sur les pots de miel. Son objectif est d'améliorer le contenu des alertes en exploitant la connaissance collectée par les pots de miels sur les attaquants.
La deuxième approche est basée sur l'application de la politique de sécurité. Son objectif est de prendre en compte les moyens de mise en application de la politique de sécurité implémentés dans le réseau en corrélant les alertes.
Énoncé de la problématique :

Les systèmes pour la Sécurité de l'Information et la Gestion des Evénements (SIEM) doivent traiter un nombre très important d'évènements de sécurité sans avoir la connaissance suffisante sur l'attaquant ni sur les moyens de défense.

Énoncé de la thèse:

L'évaluation de l'impact d'une attaque ainsi que l'aide à la décision prenant en compte la contremesure exigent une meilleure connaissance de l'attaquant ainsi que les moyens de défense déployés au sein du réseau. Améliorer notre connaissance sur l'attaquant nous permet de mieux comprendre les alertes. Identifier les moyens d'application de la politique de sécurité contribue à une décision de réponse plus efficace aux attaques.

Les objectifs de cette thèse sont:

- *Objectif 1*: la conception et l'expérimentation d'une nouvelle méthodologie de corrélation d'alertes qui bénéficie d'informations collectées par les pots de miel;

- *Objectif 2*: la conception et l'étude d'une méthodologie de corrélation d'alertes basée sur le modèle de domaine de responsabilité d'un PEP proposé;

- *Objectif 3*: l'application des approches de corrélation proposées sur un cas d'étude de cas pratique.

Contributions: Les approches de corrélation d'alertes développées dans cette thèse s'appuyent sur les données fournies par les pots de miel et les configurations des PEPs déployés au sein du réseau.
Dans notre approche "corrélation d'alertes basée sur les pots de miel", nous étudions les informations concernant les attaquants fournies par les pots de miel dans l'objectif de corréler les alertes.
En plus, puisque les données sur les attaquants ne sont généralement pas suffisantes pour améliorer la prise de décision pour les intrusions, nous proposons une "approche de corrélation d'alertes basée sur l'application de la politique de sécurité"

qui s'appuie sur les PEPs déployés dans le réseau.
Nous détaillons les contributions de cette thèse comme suit:

- Identification des données appropriées fournies par les pots de miel afin d'enrichir nos connaissances sur les alertes générées *(Objectif 1)*;
- Conception d'un processus d'enrichissement d'alertes qui améliore, de façon adéquate, l'information relevée dans les alertes générées *(Objectif 1)*;
- Identification des limitations des pots de miel quant à la corrélation d'alertes *(Objectif 1)*;
- Conception d'un modèle de PEP illustrant le rôle d'un PEP dans un processus de décision prenant en compte la contremesure. Notre modèle est basé sur la notion de domaine de responsabilité d'un PEP *(Objectif 2)*;
- Développement d'une approche de corrélation d'alertes basée sur l'application de la notion de domaine de responsabilité d'un PEP *(Objectif 2)* et l'analyse de son application sur les architectures de sécurité couramment déployées dans un réseau. *(Objectif 3).*

A.1 État de l'art

Dans cette partie, nous passons en revue les travaux de recherche sur la sécurité des réseaux qui sont liés à nos approches développées dans cette thèse. Nous introduisons d'abords la notion de l'application de la politique de sécurité dans la sectionA.1.1. Puis, nous exposons les différentes techniques de détection d'intrusions dans la section A.1.2 et les techniques des pots de miel dans la section A.1.3. Ces techniques de détection d'intrusions génèrent de nombreuses alertes difficiles à gérer. C'est pour cette raison, la notion de corrélation d'alertes a été développée. Dans la section A.1.3, nous expliquons cette notion qui sera le cœur de cette thèse.

A.1.1 Application de la politique de sécurité

L'application de la politique de sécurité est la mise en place et configuration des mécanismes de contrôle d'accès à travers la mise en place et la mise à jour des configurations des PEPs. Ces éléments de sécurité sont généralement déployés à des endroits critiques du réseau. Le terme PEP a été introduit au départ (cf. XACML [11]) comme une entité capable d'effectuer un contrôle d'accès en appliquant des décisions sur les requêtes. En se référant à la dernière version du [11], un PEP est présenté sous plusieurs formes. Il peut être une passerelle d'accès à distance, un serveur web ou un agent de boîtes à courrier électronique, etc.
Plusieurs définitions du PEP ont été données [12–15]. Nous pouvons conclure qu'un PEP peut être présenté sous la forme d'un produit dédié (comme par exemple les pare feux, les systèmes de détection d'intrusions, etc) ou d'un module intégré (par

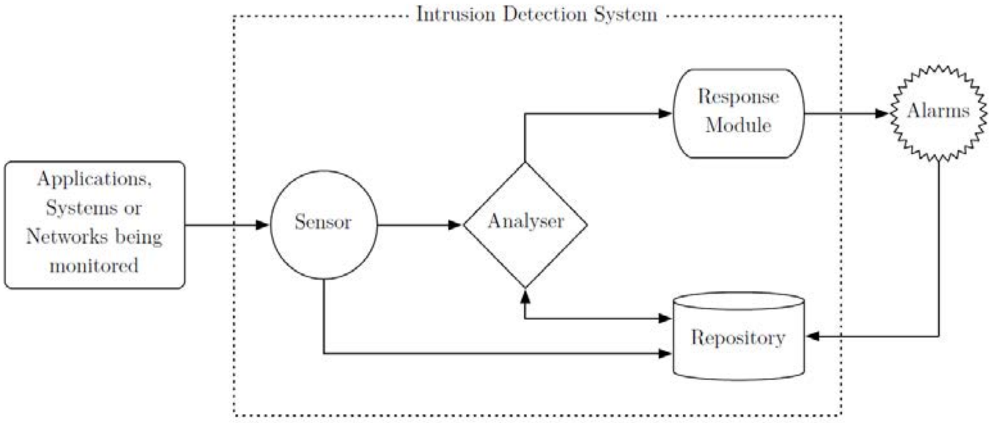

Intrusion Detection System
Applications, Systems or Networks being monitored
Sensor
Analyser
Response Module
Alarms
Repository

FIGURE A.1 - Architecture basique d'un système de détection d'intrusions selon [19].

exemple les modules de mise à jour, etc.).

A.1.2 Les systèmes de détections d'intrusions IDS

La figure A.1 représente l'architecture générique d'un système de détection d'intrusions. Ce système est composé d'une sonde, d'un analyseur, d'un module de réponse et d'un répertoire [19]. Généralement, les IDS génèrent des alertes lorsqu'ils détectent une attaque. Dans la littérature, nous pouvons trouver plusieurs approches de systèmes de détection d'intrusions. Plusieurs taxonomies ont été proposées [17, 20–32]. La classification des IDS est conditionnée par le mécanisme de détection utilisé, le comportement lors de la détection, la nature de l'information analysée, etc.
Malgré les différents avantages que les IDS ont, ces derniers souffrent de quelques limitations à savoir le format des alertes générées, le taux de faux positifs, les alertes dupliquées et l'aspect incomplet des informations contenues dans les alertes. De nos jours, avec l'expansion des réseaux, une des limitations qui s'avère critique est l'augmentation du volume d'alertes générées.

A.1.3 Les pots de miel

Un pot de miel est un environnement où des vulnérabilités ont été délibéremment introduites pour attirer les attaquants et par la suite étudier leurs comportements [53, 54]. Ces environnements intéragissent avec l'attaquant de différentes manières. Il existe des pots de miels avec une interaction minimale. Ces pots de miel imitent un service de base. Il existe aussi des pots de miel avec une interaction moyenne. Ils sont plus sophistiqués que la première catégorie. Ces pots de miel sont capable d'imiter un ensemble de services et d'effectuer des opérations plus complexes. La dernière catégorie des pots de miel est dite à interaction haute et sont capable d'imiter un système de communication réel. Cette dernière catégorie est capable d'intéragir d'une manière plus complète avec l'attaquant.

A.1.4 Corrélation d'alertes

La corrélation d'alertes a été conçue pour pallier les limitations des IDSs. Principalement, elle permet de réduire le nombre d'alertes générées par ces derniers. Dans la littérature, on trouve plusieurs approches de corrélation d'alertes que nous catégorisons de la manière suivante:

- Corrélation d'alertes basée sur la similarité: cette approche regroupe les alertes similaires selon un critère de corrélation.
- Corrélation d'alertes basée sur la connaissance: cette approche est basée sur une connaissance des activités malicieuses et des scénarios d'attaques. Elle requiert la présence de règles expertes et d'heuristiques pour corréler les alertes. On distingue deux sous catégories:
    - Corrélation d'alertes basée sur les règles: connue sous le nom d'approche basée sur les causes et conséquences d'une attaque. Ces approches ne requièrent aucune connaissance sur les scénarios d'attaque.
    - Corrélation d'alertes basée sur les scènarios: l'attaquant exécute souvent plusieurs actions malicieuses pour atteindre son objectif d'intrusion. Ces actions élémentaires constituent les étapes d'un scénario. La corrélation d'alertes basée sur les scénarios regroupe les alertes qui correspondent à un scénario bien défini.
- Corrélation d'alertes basée sur les modèles: Cette approche qui a été présentée dans le travail de M2D2 [78] et M4D4 [80] vise à considérer le contexte dans lequel l'alerte a été générée en corrélant les alertes. Elle inclut les informations contextuelles et topologiques liés à l'alerte.

A.2 Approche de corrélation d'alertes basée sur les pots de miel

Les techniques de corrélation d'alertes existantes [50,71,73–75,78–80,82] sont basées sur les bases de données des alertes générées en local par les IDSs. La vue de ces derniers est limitée par les frontières fonctionnelles (liés à la configuration de l'IDS) et structurelles (liés à la couverture topologique du réseau) du système surveillé. Les pots de miel fournissent une information qui enrichit et améliore notre connaissance sur l'attaquant et par conséquent, l'information présente dans les alertes.

A.2.1 Définition de l'approche

La corrélation d'alertes basée sur les pots de miel que nous proposons consiste à exploiter l'information sur les attaquants collectée par les pots de miel pour améliorer la corrélation des alertes générées en local (par les IDSs déployés dans le réseau supervisé).
Cette approche est capable d'analyser le lien de causalité entre les alertes détectées localement au niveau du système supervisé et les phénomènes d'attaque observés globalement au niveau des pots de miel. Elle est bénéfique pour ré-évaluer la gravité et l'impact de l'attaque.
La figure A.2 présente le processus de corrélation d'alertes basée sur les pots de miel. Ce processus est principalement composé de trois étapes qui sont:

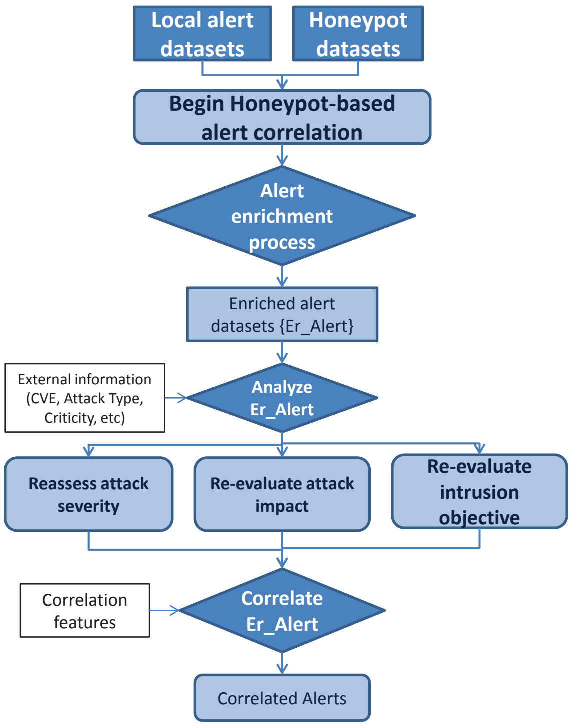

Local alert datasets
Honeypot datasets
Begin Honeypot-based alert correlation
Alert enrichment process
Enriched alert datasets {Er_Alert}
External information (CVE, Attack Type, Criticity, etc)
Analyze Er_Alert
Reassess attack severity
Re-evaluate attack impact
Re-evaluate intrusion objective
Correlation features
Correlate Er_Alert
Correlated Alerts

FIGURE A.2 - Processus de corrélation d'alertes basée sur les pots de miel.

- Processus d'enrichissement d'alerte: L'objectif de cette étape est l'enrichissement du contenu des alertes générées localement par l'information sur l'attaquant. Le résultat de cette étape est une alerte enrichie notée *Er_ alert*. Cette étape sera détaillée dans la section A.2.3;
- Analyse de l'alerte enrichie (Er_Alert): À cette étape, le contenu de l'alerte enrichie Er_alert est analysé dans le but de ré-évaluer la sévérité de l'attaque, son impact et l'objectif de l'intrusion.
- Corrélation des alertes enrichies: Cette étape a pour objectif la corrélation des

alertes enrichies en appliquant des approches de corrélation existantes. Il est possible de combiner différentes techniques de corrélation d'alertes. Le résultat de cette étape est l'obtention d'alertes corrélées selon la technique de corrélation appliquée.

A.2.2 Sources d'information

L'approche de corrélation d'alerte basée sur les pots de miel exploite des informations hétérogènes collectées par différentes vues: une vue locale et une vue globale. L'information locale est collectée à travers des sondes comme les systèmes de détéction d'intrusion, pare feux, etc. Ces sondes sont déployés pour signaler des traces d'activités malveillantes affectant le réseau local supervisé.
L'information globale est récupérée des bases de données de pots de miel. Comme décrit dans le paragraphe A.1.3, l'objectif primaire des pots de miel est l'étude du comportement de l'attaquant. Les données collectées par les pots de miel contiennent des informations sur le comportement des logiciels malveillants, leurs vecteurs de propagation, la stratégie de propagation, les relations entres les exploits, la localisation des attaquants, la caractérisation de leurs sources et destinations, etc [56, 59].

A.2.3 Processus d'enrichissment des alertes

Dans le contexte de l'approche de corrélation d'alertes basée sur les pots de miel, nous enrichissons notre connaissance locale sur les alertes avec l'information globale sur la menace. Par conséquent, il est essentiel de définir une stratégie d'enrichissement d'alerte qui améliore les connaissances liées à l'alerte avec des informations externes appropriées. La figure A.3 décrit une vue d'ensemble du processus d'enrichissement d'alerte. Comme le montre la figure A.3, nous collectons d'abord des données correspondantes à des critères d'enrichissement. Ces critères se basent principalement sur les attributs des alertes générées respectant le format IDMEF (Intrusion Detection Message Exchange Format) [69]. Selon le standard IDMEF, l'alerte est composée de plusieurs classes agrégées. Les classes suivantes sont considérées comme sélecteurs pour interroger l'ensemble des données des pots de miel:

Source: cette classe comprend l'adresse IP source de la menace détectée.

DetectTime: représente le moment où l'attaque a été détectée par l'analyseur en local. Dans les bases de données des pots de miel, les activités capturées sont généralement horodatées. Ceci nous permet d'analyser l'évolution de l'état de sécurité de l'alerte pendant une fenêtre de temps spécifique.

Classification: représente la sémantique de l'alerte. Par exemple, les systèmes de détection d'intrusion classent les alertes selon le type de l'attaque. Ceci est lié au type de la menace détectée par des pots de miel.

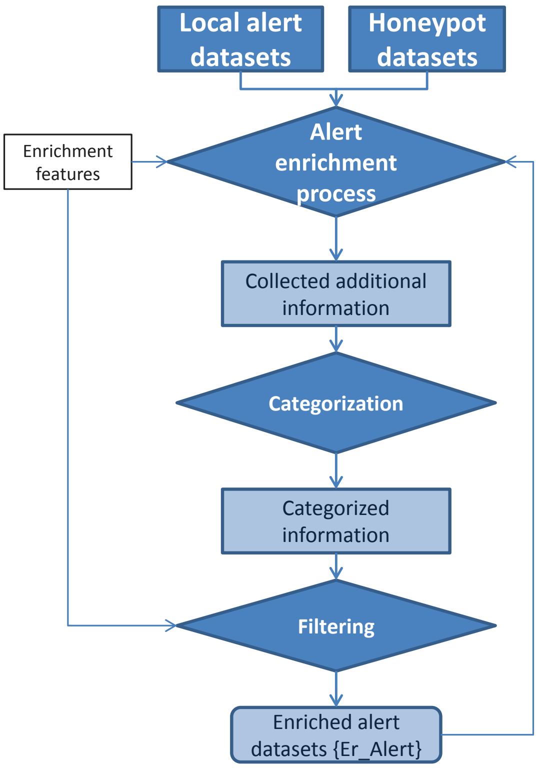


FIGURE A.3 - Processus d'enrichissement d'alertes.

Par exemple, selon la source de l'alerte générée, nous recueillons les connaissances globales sur le serveur analysé par les pots de miel. Ainsi, nous pouvons évaluer son profil de sécurité et son évolution au fil du temps. Une fois que cette collecte d'information est effectuée, nous catégorisons les données collectées selon une catégorisation bien définie dans cette thèse (information temporel, information sémantique, information contextuelle). Nous proposons aussi un processus de filtrage qui est composé par trois types de filtres: temporel, sémantique et filtre de configuration. Ces filtres permettent d'éliminer les données qui sont moins susceptibles de pouvoir lier l'alerte détectée en local avec les phénomènes de menaces globalement détectées.

A.2.4 Expérimentations et évaluations

Les expériences menées dans cette partie sont basées sur quatre bases de données de pots de miel qui sont: SGNET v1 [59], SGNET v2, HARMUR v1 [85] et HARMUR v2. Les alertes en local ont été collectées par une sonde Snort v2.8 qui a été mise en place dans notre réseau d'université.

Source d'information locale

Durant quatre mois de déploiement, notre IDS a généré 183170 alertes qui correspondent à 2499 adresses IP sources uniques. Nous résumons ces alertes dans le tableau A.1, triées par leur classement (détaillé dans la première colonne du tableau). Dans la deuxième colonne du tableau, nous donnons le nombre d'alertes générées au cours de ces quatre mois pour chaque classification. Dans la troisième colonne, nous identifions le nombre d'adresses IP source unique. Les deux dernières colonnes montrent le nombre d'alertes filtrées et leur adresse IP source unique.

Table A.1 - Classification des alertes générées par la sonde IDS déployée en local. Dans ce tableau, nous identifions le nombre total des adresses IP sources uniques pour toutes les alertes.

Classification	**Nombre d'alertes**	**Nombre total d'adresses IP source uniques**	**Nombre d'alertes filtrées**	**Nombre d'adresses IP source uniques filtrés**
attempted-recon	156338	2198	870 (0,5%)	132 (6%)
attempted-dos	1	1	1 (100%)	1 (100%)
attempted-user	1540	28	1540 (100%)	28 (100%)
misc-activity	839	174	23 (3%)	21 (12%)
trojan-activity	3	3	3 (100%)	3 (100%)
bad-unknown	22573	34	51 (0,2%)	9 (26,5%)
unclassified	1876	61	1876 (100%)	61 (100%)
Global	183170	2499	4364 (2,5%)	255 (10%)

Comme le montre le Tableau A.1, différentes alertes ont été filtrées pour améliorer les performances du processus d'enrichissement. En effet, le système de détections d'intrusions Snort est connu par son taux élevé de faux positifs.

Source d'information globale

Les bases de données de pots de miel que nous avons utilisées dans nos expériences ont été fournies par le projet européen Vis-Sense. Elles comprennent des données caractérisant les menaces et le profil de sécurité des serveurs web suspicieux.
La base de données HARMUR [85] contient des informations portant sur l'évolution de la sécurité et de l'information contextuelle associés aux menaces des serveurs web.
La base de données SGNET [59] contient des informations sur les stratégies de propagation des logiciels malveillants. En effet, SGNET est un déploiement de pots

de miel distribués implémentant différents outils, à savoir ScriptGen [60], Argos [61] et Nepenthes [58].

Résultats expérimentaux

Dans nos expériences, nous avons commencé par la première étape importante qui est le processus d'enrichissement des alertes. Nous avons considéré l'adresse IP source comme un critère d'enrichissement. Les expériences menées ont montré que le nombre des alertes qui peuvent être enrichies est très réduit comme le montre le Tableau A.2. Nous concluons qu'un grand nombre d'adresses IP source associés aux

Table A.2 - Résultats expérimentaux utilisant les adresses IP source des alertes. Les résultats montrés dans ce tableau sont basés sur les adresses IP source des alertes filtrées comme détaillé dans le Tableau A.1.

Classification	**Number of filtered IP source address**	**SGNET** v1	**SGNET** v2	**HARMUR** v1	**HARMUR** v2
attempted-recon	132	–	–	**1** (SNMP AgentX/TCP)	–
attempted-dos	1	–	–	–	–
attempted-user	28	–	–	–	–
misc-activity	21	–	–	–	–
trojan-activity	3	–	–	–	–
bad-unknown	9	**1** (DNS SPOOF)	–	–	**1** (DNS SPOOF)
unclassified	61	–	–	**1** (tcp portscan) & **1** (tcp portscan,tcp portsweep)	**1** (tcp portscan)

alertes n'ont pas été analysées par des pots de miel. Par conséquent, de nombreuses alertes ne seront pas enrichies systématiquement. En plus, les données obtenues des bases de données de pots de miel et liées aux alertes filtrées sont difficile à explorer dans le contexte de notre approche de corrélation. Nous spécifions dans le paragraphe suivant les différentes limitations des pots de miels dans le cadre de notre approche de corrélation.

A.2.5 Limitations des pots de miel dans l'amélioration de la corrélation d'alertes

Pendant nos expériences, nous avons identifié quatres limitations majeurs de l'utilisation des pots de miel dans l'amélioration de la corrélation d'alertes.

A.2.5.1 Limitation de couverture

Environ 95 % des alertes n'avaient pas d'objet dans les bases de données de pots de miel et correspondant à l'adresse IP source des alertes. Ceci est principalement dû au niveau d'interaction des pots de miel. Même si les pots de miel sont capable d'intéragir avec les attaquants selon différentes techniques, ils ne sont pas capables de couvrir une large variétée d'activités comme l'interaction réelle entre l'attaquant et le système.

A.2.5.2 Limitation des attributs et références manquants

Dans nos expériences, nous avons remarqué un important nombre d'objects dans les bases de données expérimentales de pots de miel qui ont des attributs non remplis. Par conséquent, nous avons mené une étude statistique sur ces bases de données sur un type d'objet qui est l'objet "menace". Nous observons que 85 % des objets étudiés avaient des références manquantes. Ceci nous empêche essentiellement d'exécuter les filtres qui ont été définis dans notre approche.

A.2.5.3 Manque d'un standard pour représenter les données de pots de miel

Dans nos expériences, nous avons observé que plusieurs attributs ne respectaient pas une nomenclature standard. Généralement, les attributs sont remplis suivant une nomenclature générique qui n'apporte pas l'information essentielle pour l'évaluation qualitative des menaces. Ceci nous empêche d'analyser le lien de causalité qui peut exister entre les événements détectés à l'échelle global et les alertes générées en local.

A.2.5.4 Limitation de croisement de références

Nous avons remarqué une absence de coopération entre les bases de données expérimentales de pots de miel. Il était difficile de retrouver plus d'information par

rapport à un objet spécifique dans d'autres base de données. Dans le cadre de notre approche de corrélation et d'enrichissement d'alertes basée sur les pots de miel, il est nécessaire que nous profitions d'une vue complète et globale des phénomènes d'attaques telles que les serveurs infectés, l'évolution de leur sécurité au cours du temps, la caractérisation et la spécificité des exploits.
Ces limitations présentent un blocage pour mieux explorer l'usage des pots de miels dans la corrélation d'alertes. Ainsi, nous développons dans la suite une nouvelle approche de corrélation d'alertes qui est basée sur l'application de la politique de sécurité.

A.3 Modèle de point d'application de politique (PEP)

Les approches de corrélation d'alertes existantes telles que [50, 71, 73–75, 78–80, 82] fonctionnent indépendamment des capacités d'application de politique de sécurité déployées dans le réseau. Cela est dû à l'absence d'un modèle unifié de points d'application de politique (PEP).
Les PEPs, une fois déployés et configurés, sont responsables d'appliquer des règles sur des flux réseau spécifiques. Comme les décisions de réponses aux intrusions et les contre-mesures sont généralement des modifications qui doivent être apportées à la configuration de ces PEPs, il est important d'identifier facilement le PEP approprié. Si les alertes sont corrélées en se basant sur les capacités du PEP, leur traitement serait plus efficace. Dans cette section, nous détaillons le modèle du PEP que nous proposons. Ce modèle nous permet non seulement d'avoir une bonne compréhension des capacités d'application de politique déployé dans le réseau mais également d'apporter d'autres solutions aux problèmes de gestion de politique de sécurité et de détection d'intrusion.

A.3.1 Définitions et axiomes

Le PEP est une entité de sécurité capable d'appliquer sur les requêtes les décisions d'application de politique représentées par $\{d_1, d_2, d_3, \ldots, d_p\}$ (p est le nombre total de toutes les décisions qui peuvent être appliquées par le PEP). Le PEP est un ensemble de règles désignées par $r_1, r_2, \ldots, r_m$ (m est le nombre total de règles appliquées par le PEP). Ci-dessous nous donnons une représentation algébrique caractérisant un PEP:

$$PEP = \{r_j\}_{j \in [1 \ldots m]}$$

Comme définie dans les modèles de contrôle d'accès [12, 15, 91–94], une règle de contrôle d'accès r_j s'écrit généralement sous la forme suivante:

$$\begin{aligned} &r_j : \{\mathit{conditions}\}_j \longrightarrow \{\mathit{decisions}\}_j \quad \forall j \in [1, \ldots, m] \\ &r_j : C_j \longrightarrow D_j \quad \forall j \in [1, \ldots, m] \end{aligned}$$

En général, C_j est un ensemble de conditions définies sur des attributs que nous appelons *Sélecteurs* et noté par $\{S_i\}_{i\in[1...n]}$. Par conséquent, une règle r_j est représentée par:

$$r_j : \underset{i\in[1...n]}{\times s_i} \longrightarrow \{d_k\}_{k\in[1...p]} \quad \forall j \in [1, \dots, m]$$

s_i $(i \in [1 \dots n])$ sont des instantiations de sélecteurs.
Avant de détailler le modèle proposé, il est important de détailler nos hypothèses de modélisation:

- **Axiome 1** La vérification des faux positifs est en dehors du champ de notre travail. En plus, nous supposons que toutes les alertes sont écrites selon le standard IDMEF (Intrusion Detection and Message Exchange Format) décrit dans [69]
- **Axiome 2** Tous les PEP ont un ensemble fini de règles;
- **Axiome 3** Nous ignorons la règle par défaut appliquée par le PEP;
- **Axiome 4** Les domaines de tous les sélecteurs sont finis;
- **Axiome 5** Nous ignorons la décision qui peut être appliquée par le PEP;
- **Axiome 6** La définition du domaine de responsabilité d'un PEP doit dépendre seulement des caractéristiques intrinsèques au PEP.

Nous détaillons ci-dessous la définition du domaine de responsabilité d'un PEP:

Définition 1 ***Domaine de responsabilité d'un PEP:*** *Chaque PEP, une fois mis en place dans un réseau, a un domaine d'applicabilité unique que nous appelons "Domaine de responsabilité". Nous le notons $RD(PEP)$. Ce domaine est une abstraction de l'implémentation et de la configuration d'un PEP et de ses capacités intrinsèques d'application de politique.*

$RD(PEP)$ est un domaine multidimensionnel borné. Dans la définition 2, nous détaillons ces bornes.

Définition 2 *$RD_{inf}(PEP)$ est l'union de tout les domaines de responsabilité de règles configurées dans un PEP mis en place.*
$RD_{sup}(PEP)$ considère les contraintes environnementales du PEP mis en place. L'identification de cette borne requiert une connaissance externe de la visiblité topologique du PEP. Suivant l'axiome 4, cette borne ne sera bas identifiée dans ce travail.

A.3.2 Domaine de responsabilité d'un PEP

Notre objectif est d'analyser les différentes possibilités d'une approximation appropriée d'un $RD(PEP)$ sans perdre les spécifités du PEP mis en place. Dans notre approche, nous considérons que la configuration du PEP déployé dans le réseau est le seul point de départ pour toutes les approximations. La configuration du PEP constitue $RD_{inf}(PEP)$. En partant de $RD_{inf}(PEP)$, nous définissons des opérations afin de construire les différentes approximations. Ces opérations prennent en compte les relations qui existent entre les règles, les caractéristiques des sélecteurs et leurs propriétés de combinaison.

I

g

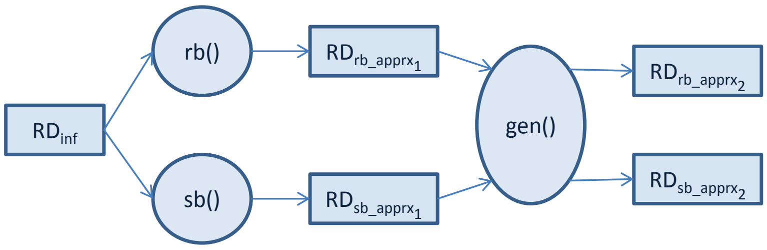


FIGURE A.4 - Vue d'ensemble des approximations du domaine de responsabilité.

La figure A.4 représente une vue d'ensemble des différentes étapes et fonctions d'approximations.

$$rb() : \quad \mathcal{U} \longrightarrow \mathcal{U}$$
$$RD_{inf}(PEP) \longmapsto RD_{rb\_apprx1(PEP)}$$

La première approximation basée sur les règles $RD_{rb\_apprx_1}(PEP)$ est le résultat de l'application de la fonction $rb(RD_{inf})$ sur tout les vecteurs du domaine $RD_{inf}(PEP)$.

$$sb() : \quad \mathcal{U} \longrightarrow \mathcal{U}$$
$$RD_{inf}(PEP) \longmapsto RD_{sb\_apprx_1}(PEP)$$

La première approximation basée sur les sélecteurs $RD_{sb\_apprx_1}(PEP)$ est le résultat de l'application de la fonction $sb(RD_{inf})$ sur tout les vecteurs du domaine $RD_{inf}(PEP)$.

$$gen() : \quad \mathcal{U} \longrightarrow \mathcal{U}$$
$$< s_j, j \in [1 \dots n] > \longmapsto < \delta_{k_j}, j \in [1 \dots n] >$$

La fonction gen() fait référence à un processus de généralisation des valeurs de sélecteurs suivant une décomposition de leurs domaines. Pour chaque valeur d'un sélecteur s_j, gen() identifies une plage plus générique de la valeur s_j.

A.4 Approche de corrélation d'alertes basée sur l'application de politique

A.4.1 Définition et objectif

Ci-dessous, nous donnons la définition de notre approche de corrélation d'alertes basée sur le renforcement de politique.

Définition 3 ***Corrélation d'alertes basée sur l'application de politique :*** *Étant donné un ensemble d'alertes et un ensemble de PEPs déployés, la corrélation d'alertes basée sur l'application de politique groupe les alertes qui peuvent être traitées par un ensemble commun de PEPs.*

L'inférence basique utilisée dans cette approche est:

$$\begin{gathered} A_1 \in RD(PEP_1) \wedge A_2 \in RD(PEP_1) \wedge \ldots \\ \implies A^{ec} = \langle (A_1,\ A_2,\ \ldots), PEP_1 \rangle \end{gathered}$$

Soient deux alertes , A_1 et A_2. RD_{PEP_1} est le domaine de reponsabilité du PEP_1. A^{ec} représente le résultat de la corrélation. Elle contient deux composantes. La première composante inclut l'ensemble des alertes qui sont corrélées. La deuxième composante inclut le/les PEPs qui sont capable de traiter les alertes corrélées.

A.4.2 Processus de l'approche de corrélation d'alertes basée sur l'application de politique

Le moteur de corrélation d'alertes basée sur l'application de politique (PEACE) est représenté dans la figure A.5 et figure A.6. Il est responsable de vérifier quelles alertes pourraient être traitées par un même ensemble de PEPs.
Les configuration des PEPs mis en place sont récupérées et envoyées au *RD-Approximator*. Ce dernier calcul les approximations des domaines de responsabilité des PEP. Ainsi, PEACE est capable de traiter les alertes reçues $\{A_1, A_2, \ldots, A_{n_t}\}$ où n_t est le nombre d'alertes générées pendant une durée de temps t.

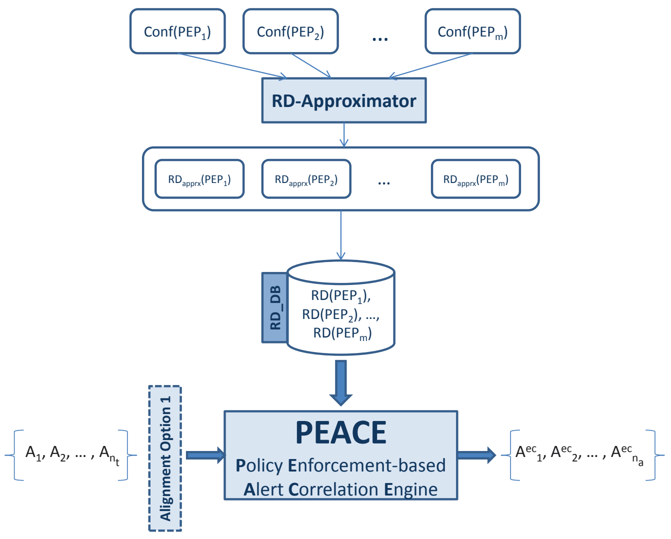

Conf(PEP$_1$)
Conf(PEP$_2$)
...
Conf(PEP$_m$)
RD-Approximator
RD$_{apprx}$(PEP$_1$)
RD$_{apprx}$(PEP$_2$)
...
RD$_{apprx}$(PEP$_m$)
RD_DB
RD(PEP$_1$), RD(PEP$_2$), ..., RD(PEP$_m$)
Alignment Option 1
A$_1$, A$_2$, ... , A$_{n_t}$
PEACE
Policy Enforcement-based Alert Correlation Engine
A$^{ec}_1$, A$^{ec}_2$, ... , A$^{ec}_{n_a}$

FIGURE A.5 - Structure de PEACE avec fonction d'alignement option 1.

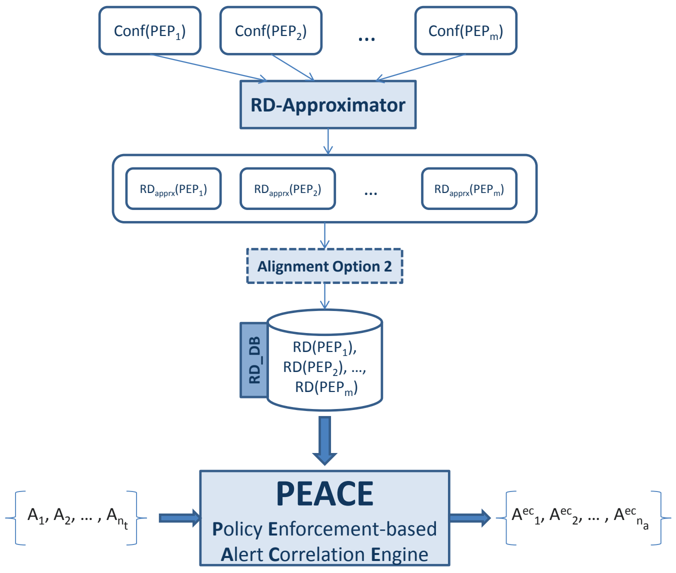

Conf(PEP$_1$)
Conf(PEP$_2$)
...
Conf(PEP$_m$)
RD-Approximator
RD$_{apprx}$(PEP$_1$)
RD$_{apprx}$(PEP$_2$)
...
RD$_{apprx}$(PEP$_m$)
Alignment Option 2
RD_DB
RD(PEP$_1$), RD(PEP$_2$), ..., RD(PEP$_m$)
A$_1$, A$_2$, ... , A$_{n_t}$
PEACE
Policy Enforcement-based Alert Correlation Engine
A$^{ec}_1$, A$^{ec}_2$, ... , A$^{ec}_{n_a}$

FIGURE A.6 - Structure de PEACE avec fonction d'alignement option 2.

Les fonctions d'alignement montrées dans les deux figures A.5 et A.6 sont responsable d'alignement de format des alertes avec les domaines de responsabilité des PEPs. Dans l'option 1, ces fonctions d'alignement procèdent à l'alignement des alertes (écrites en IDMEF) avec chaque classe de PEP. Dans l'option 1, ces fonctions d'alignement procèdent à l'alignement de tous les domaines de responsabilité avec le même format des alertes qui est IDMEF.

A.4.3 Applications et intérprétations

Nous avons étudié l'application des approximations de domaine de responsabilité dans la corrélation d'alertes basée sur l'application de politique dans le cas où un seul PEP est mis en place et dans le cas où plusieurs PEPs sont déployés. Cette dernière étude a été menée sur des motifs d'architecture de sécurité. En ce qui concerne l'étude de cas d'un seul PEP, nous avons pu élaborer le tableau comparatif A.3 entre les différentes approximations et leurs impact sur la corrélation.

TABLE A.3 - Application des approximations de domaine de responsabilité dans PEACE: un tableau comparatif.

RD approximé	**Fonction d'approximation**	**impact sur la corrélation d'alertes**
$RD_{inf}(PEP)$	–	Les alertes sont corrélées selon les règles configurées. + L'implémentation des contre mesures est immédiatement identifiée − Une vue restreinte du domaine de responsabilité du PEP − Les alertes qui pourraient être traitées par l'introduction de nouvelles règles sont ignorées
$RD_{rb\_apprx_1}(PEP)$	$rb(RD_{inf}(PEP))$	Les alertes sont corrélées par une combinaison de règles configurées + L'implémentation des contre mesures implique au moins une modification d'une règle configurée − Les alertes qui pourraient être traitées par l'introduction de nouvelles règles sont ignorées
$RD_{sb\_apprx_1}(PEP)$	$sb(RD_{inf}(PEP))$	Les alertes sont corrélées selon toutes les combinaisons possible entre les valeurs de sélecteurs configurées − L'implémentation des contre mesures requiert une analyse profonde des règles impliquées dans la corrélation d'alertes − Les alertes qui pourraient être traitées par l'introduction de nouvelles règles avec des valeurs différentes de sélecteurs sont ignorées
$RD_{rb\_apprx_2}(PEP)$	$gen(RD_{rb\_apprx_1}(PEP))$	Les alertes sont corrélées selon les règles configurées généralisées − L'ensemble des alertes corrélées dépend de la partition des valeurs configurées des sélecteurs
$RD_{sb\_apprx_2}(PEP)$	$gen(RD_{sb\_apprx_1}(PEP))$	Les alertes sont corrélées selon les valeurs des sélecteurs généralisées − L'ensemble des alertes corrélées dépend de la partition des valeurs configurées des sélecteurs

Dans le cas de plusieurs PEPs mis en place, nous avons conclus que dans un tel environnement, la corrélation est considérée comme un problème d'optimisation. En effet, les alertes doivent être corrélées tout en optimisant l'implémentation des contremesures. Dans ce contexte, la corrélation d'alertes doit minimiser le coût d'implémentation de la décision de réponse aux intrusions. Ce coût couvre les différentes modifications qui doivent être apportées à la politique de sécurité. Dans ce contexte, des travaux de sélections de contremesures ont été développés comme [106,107]

A.5 Conclusions et travaux futurs

L'objectif de cette thèse est de développer de nouvelles techniques de corrélation d'alertes pour aider le responsable de la sécurité des systèmes d'information (RSSI) dans le traitement du nombre très important des alertes générées.
Deux approches de corrélation ont été développées:

- Corrélation d'alertes basée sur les pots de miel (Chapitre 3): Cette approche exploite les bases de données des pots de miel pour améliorer la corrélation d'alertes par la connaissance sur l'attaquant. Nos expériences ont été réalisées en se basant sur quatre bases de données de pots de miel. Elles nous ont permis d'identifier quatre limitations majeures pour améliorer la corrélation d'alertes;
- Corrélation d'alertes basée sur l'application de politique (Chapitre 5): Cette approche exploite la connaissance sur les capacités de l'application de politique pour améliorer la corrélation d'alertes avec la connaissance sur la contremesure. Nous avons étudié l'application de cette approche sur des environnements à un seul PEP et à plusieurs PEPs. Cette étude a montré que cette approche joue un rôle dans l'implémentation de contremesures.

La première approche a contribué à l'identification des limites des pots de miel existants dans l'amélioration de la corrélation d'alertes.
La deuxième approche a contribué à la démonstration de l'applicabilité des capacités de l'application de politique de sécurité mise en place dans le réseau dans la corrélation des alertes. Nous avons étudié l'application de l'approche sur des motifs d'architecture de sécurité qui sont communs à tout les réseaux étendus.

Les travaux futurs se concentreront à considérer d'autres types de PEPs, sélectionner l'approximation du domaine de responsabilité le plus approprié et identifier l'implémentation des contremesures.