Titre: Title:	Intrusion Detection from Heterogenous Sensors
Auteur: Author:	Alireza Sadighian
Date:	2015
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Sadighian, A. (2015). Intrusion Detection from Heterogenous Sensors [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1702/

UNIVERSITÉ DE MONTRÉAL

INTRUSION DETECTION FROM HETEROGENOUS SENSORS

ALIREZA SADIGHIAN
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(GÉNIE INFORMATIQUE)
MARS 2015

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

INTRUSION DETECTION FROM HETEROGENOUS SENSORS

présentée par : SADIGHIAN Alireza
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. QUINTERO Alejandro, Doctorat, président
M. FERNANDEZ José M, Ph. D., membre et directeur de recherche
M. GAGNON Michel, Ph. D., membre
M. TRAORÉ Issa, Ph. D., membre externe

DEDICATION

I am proud to dedicate this thesis to my wife,
Roxanne
whose love and confidence is a constant source of inspiration and encouragement....

ACKNOWLEDGEMENTS

First of all, I would like to thank Almighty God. Without his wish, nothing is possible.

The completion of this dissertation was only possible with the supports and contributions of many people.

I would like to take this opportunity to thank my supervisor, Professor José M Fernandez, for all his substantial advices and inspiration throughout this research. I mainly learned from him the skills of conducting a research, collaborating with other colleagues, presenting ideas, etc.

I would like to thank members of the jury Dr. Alejandro Quintero, Dr. Michel Gagnon and Dr. Issa Traoré for their valuable comments on my thesis.

I would like to thank Dr. Saman Taghavi Zargar and Dr. Antoine Lemay for their productive feedbacks and comments during this research.

I am very thankful to all my friends and colleagues at the SecSI Lab, specially Pier-Luc St-Onge, Carlton Davis, Joan Calvet, Fanny Lalonde, Abdelali Zerhouani and François Labrèche for their valuable feedback and constant comradeship.

I would also like to thank Dr. Ali Zand, Dr. Gianluca Stringhini, Dr. Christopher Kruegel, Dr. Giovanni Vigna and all the colleagues at the Computer Security Lab of University of California Santa Barbara (UCSB) for their great advices that significantly improved my research skills.

Thanks to NSERC ISSNet for funding this research and Groupe Access Company for supporting this research.

Special thanks to Mr. Yves Lépine, Mr. Douglas Elie, Mr. Habib Malik, Mr. Vivien Tognisse, Mr. Saeed Sarenche and all the friends and colleagues at Groupe Access Company for their substantial supports.

Finally, I am deeply grateful for the supports of my family : my father Abbas, my mother Roqayyeh, my sister Nahid and her husband Habib, mother-in-law Tahereh, brother-in-law Sajjad and sister-in-law Maryam.

RÉSUMÉ

De nos jours, la protection des systèmes et réseaux informatiques contre différentes attaques avancées et distribuées constitue un défi vital pour leurs propriétaires. L'une des menaces critiques à la sécurité de ces infrastructures informatiques sont les attaques réalisées par des individus dont les intentions sont malveillantes, qu'ils soient situés à l'intérieur et à l'extérieur de l'environnement du système, afin d'abuser des services disponibles, ou de révéler des informations confidentielles. Par conséquent, la gestion et la surveillance des systèmes informatiques est un défi considérable considérant que de nouvelles menaces et attaques sont découvertes sur une base quotidienne.

Les systèmes de détection d'intrusion, Intrusion Detection Systems (IDS) en anglais, jouent un rôle clé dans la surveillance et le contrôle des infrastructures de réseau informatique. Ces systèmes inspectent les événements qui se produisent dans les systèmes et réseaux informatiques et en cas de détection d'activité malveillante, ces derniers génèrent des alertes afin de fournir les détails des attaques survenues. Cependant, ces systèmes présentent certaines limitations qui méritent d'être adressées si nous souhaitons les rendre suffisamment fiables pour répondre aux besoins réels. L'un des principaux défis qui caractérise les IDS est le grand nombre d'alertes redondantes et non pertinentes ainsi que le taux de faux-positif générés, faisant de leur analyse une tâche difficile pour les administrateurs de sécurité qui tentent de déterminer et d'identifier les alertes qui sont réellement importantes. Une partie du problème réside dans le fait que la plupart des IDS ne prennent pas compte les informations contextuelles (type de systèmes, applications, utilisateurs, réseaux, etc.) reliées à l'attaque. Ainsi, une grande partie des alertes générées par les IDS sont non pertinentes en ce sens qu'elles ne permettent de comprendre l'attaque dans son contexte et ce, malgré le fait que le système ait réussi à correctement détecter une intrusion. De plus, plusieurs IDS limitent leur détection à un seul type de capteur, ce qui les rend inefficaces pour détecter de nouvelles attaques complexes. Or, ceci est particulièrement important dans le cas des attaques ciblées qui tentent d'éviter la détection par IDS conventionnels et par d'autres produits de sécurité. Bien que de nombreux administrateurs système incorporent avec succès des informations de contexte ainsi que différents types de capteurs et journaux dans leurs analyses, un problème important avec cette approche reste le manque d'automatisation, tant au niveau du stockage que de l'analyse.

Afin de résoudre ces problèmes d'applicabilité, divers types d'IDS ont été proposés dans les dernières années, dont les IDS de type composant pris sur étagère, commercial off-the-shelf

(COTS) en anglais, qui sont maintenant largement utilisés dans les centres d'opérations de sécurité, Security Operations Center (SOC) en anglais, de plusieurs grandes organisations. D'un point de vue plus général, les différentes approches proposées peuvent être classées en différentes catégories : les méthodes basées sur l'apprentissage machine, tel que les réseaux bayésiens, les méthodes d'extraction de données, les arbres de décision, les réseaux de neurones, etc., les méthodes impliquant la corrélation d'alertes et les approches fondées sur la fusion d'alertes, les systèmes de détection d'intrusion sensibles au contexte, les IDS dit distribués et les IDS qui reposent sur la notion d'ontologie de base. Étant donné que ces différentes approches se concentrent uniquement sur un ou quelques-uns des défis courants reliés aux IDS, au meilleure de notre connaissance, le problème dans son ensemble n'a pas été résolu. Par conséquent, il n'existe aucune approche permettant de couvrir tous les défis des IDS modernes précédemment mentionnés. Par exemple, les systèmes qui reposent sur des méthodes d'apprentissage machine classent les événements sur la base de certaines caractéristiques en fonction du comportement observé pour un type d'événements, mais ils ne prennent pas en compte les informations reliées au contexte et les relations pouvant exister entre plusieurs événements. La plupart des techniques de corrélation d'alerte proposées ne considèrent que la corrélation entre plusieurs capteurs du même type ayant un événement commun et une sémantique d'alerte similaire (corrélation homogène), laissant aux administrateurs de sécurité la tâche d'effectuer la corrélation entre les différents types de capteurs hétérogènes. Pour leur part, les approches sensibles au contexte n'emploient que des aspects limités du contexte sous-jacent. Une autre limitation majeure des différentes approches proposées est l'absence d'évaluation précise basée sur des ensembles de données qui contiennent des scénarios d'attaque complexes et modernes.

À cet effet, l'objectif de cette thèse est de concevoir un système de corrélation d'événements qui peut prendre en considération plusieurs types hétérogènes de capteurs ainsi que les journaux de plusieurs applications (par exemple, IDS/IPS, pare-feu, base de données, système d'exploitation, antivirus, proxy web, routeurs, etc.). Cette méthode permettra de détecter des attaques complexes qui laissent des traces dans les différents systèmes, et d'incorporer les informations de contexte dans l'analyse afin de réduire les faux-positifs. Nos contributions peuvent être divisées en quatre parties principales : 1) Nous proposons la *Pasargadae*, une solution complète sensible au contexte et reposant sur une ontologie de corrélation des événements, laquelle effectue automatiquement la corrélation des événements par l'analyse des informations recueillies auprès de diverses sources. *Pasargadae* utilise le concept d'ontologie pour représenter et stocker des informations sur les événements, le contexte et les vulnérabilités, les scénarios d'attaques, et utilise des règles d'ontologie de logique simple écrites en Semantic Query-Enhance Web Rule Language (SQWRL) afin de corréler diverse informations et de

filtrer les alertes non pertinentes, en double, et les faux-positifs. 2) Nous proposons une approche basée sur, méta-événement , tri topologique et l'approche corrélation d'événement basée sur sémantique qui emploie *Pasargadae* pour effectuer la corrélation d'événements à travers les événements collectés de plusieurs capteurs répartis dans un réseau informatique. 3) Nous proposons une approche alerte de fusion basée sur sémantique, contexte sensible, qui s'appuie sur certains des sous-composantes de *Pasargadae* pour effectuer une alerte fusion hétérogène recueillies auprès IDS hétérogènes. 4) Dans le but de montrer le niveau de flexibilité de *Pasargadae*, nous l'utilisons pour mettre en œuvre d'autres approches proposées d'alertes et de corrélation d'événements. La somme de ces contributions représente une amélioration significative de l'applicabilité et la fiabilité des IDS dans des situations du monde réel.

Afin de tester la performance et la flexibilité de l'approche de corrélation d'événements proposés, nous devons aborder le manque d'infrastructures expérimental adéquat pour la sécurité du réseau. Une étude de littérature montre que les approches expérimentales actuelles ne sont pas adaptées pour générer des données de réseau de grande fidélité. Par conséquent, afin d'accomplir une évaluation complète, d'abord, nous menons nos expériences sur deux scénarios d'étude d'analyse de cas distincts, inspirés des ensembles de données d'évaluation DARPA 2000 et UNB ISCX IDS. Ensuite, comme une étude déposée complète, nous employons *Pasargadae* dans un vrai réseau informatique pour une période de deux semaines pour inspecter ses capacités de détection sur un vrai terrain trafic de réseau. Les résultats obtenus montrent que, par rapport à d'autres améliorations IDS existants, les contributions proposées améliorent considérablement les performances IDS (taux de détection) tout en réduisant les faux positifs, non pertinents et alertes en double.

ABSTRACT

Nowadays, protecting computer systems and networks against various distributed and multi-steps attack has been a vital challenge for their owners. One of the essential threats to the security of such computer infrastructures is attacks by malicious individuals from inside and outside of the system environment to abuse available services, or reveal their confidential information. Consequently, managing and supervising computer systems is a considerable challenge, as new threats and attacks are discovered on a daily basis.

Intrusion Detection Systems (IDSs) play a key role in the surveillance and monitoring of computer network infrastructures. These systems inspect events occurred in computer systems and networks and in case of any malicious behavior they generate appropriate alerts describing the attacks' details. However, there are a number of shortcomings that need to be addressed to make them reliable enough in the real-world situations. One of the fundamental challenges in real-world IDS is the large number of redundant, non-relevant, and false positive alerts that they generate, making it a difficult task for security administrators to determine and identify real and important alerts. Part of the problem is that most of the IDS do not take into account contextual information (type of systems, applications, users, networks, etc.), and therefore a large portion of the alerts are non-relevant in that even though they correctly recognize an intrusion, the intrusion fails to reach its objectives. Additionally, to detect newer and complicated attacks, relying on only one detection sensor type is not adequate, and as a result many of the current IDS are unable to detect them. This is especially important with respect to targeted attacks that try to avoid detection by conventional IDS and by other security products. While many system administrators are known to successfully incorporate context information and many different types of sensors and logs into their analysis, an important problem with this approach is the lack of automation in both storage and analysis.

In order to address these problems in IDS applicability, various IDS types have been proposed in the recent years and commercial off-the-shelf (COTS) IDS products have found their way into Security Operations Centres (SOC) of many large organisations. From a general perspective, these works can be categorized into: machine learning based approaches including Bayesian networks, data mining methods, decision trees, neural networks, etc., alert correlation and alert fusion based approaches, context-aware intrusion detection systems, distributed intrusion detection systems, and ontology based intrusion detection systems. To the best of our knowledge, since these works only focus on one or few of the IDS challenges, the problem as a whole has not been resolved. Hence, there is no comprehensive work addressing all

the mentioned challenges of modern intrusion detection systems. For example, works that utilize machine learning approaches only classify events based on some features depending on behaviour observed with one type of events, and they do not take into account contextual information and event interrelationships. Most of the proposed alert correlation techniques consider correlation only across multiple sensors of the same type having a common event and alert semantics (homogeneous correlation), leaving it to security administrators to perform correlation across heterogeneous types of sensors. Context-aware approaches only employ limited aspects of the underlying context. The lack of accurate evaluation based on the data sets that encompass modern complex attack scenarios is another major shortcomings of most of the proposed approaches.

The goal of this thesis is to design an event correlation system that can correlate across several heterogeneous types of sensors and logs (e.g. IDS/IPS, firewall, database, operating system, anti virus, web proxy, routers, etc.) in order to hope to detect complex attacks that leave traces in various systems, and incorporate context information into the analysis, in order to reduce false positives. To this end, our contributions can be split into 4 main parts: 1) we propose the *Pasargadae* comprehensive context-aware and ontology-based event correlation framework that automatically performs event correlation by reasoning on the information collected from various information resources. *Pasargadae* uses ontologies to represent and store information on events, context and vulnerability information, and attack scenarios, and uses simple ontology logic rules written in Semantic Query-Enhance Web Rule Language (SQWRL) to correlate various information and filter out non-relevant alerts and duplicate alerts, and false positives. 2) We propose a meta-event based, topological sort based and semantic-based event correlation approach that employs *Pasargadae* to perform event correlation across events collected form several sensors distributed in a computer network. 3) We propose a semantic-based context-aware alert fusion approach that relies on some of the subcomponents of *Pasargadae* to perform heterogenous alert fusion collected from heterogenous IDS. 4) In order to show the level of flexibility of *Pasargadae*, we use it to implement some other proposed alert and event correlation approaches. The sum of these contributions represent a significant improvement in the applicability and reliability of IDS in real-world situations.

In order to test the performance and flexibility of the proposed event correlation approach, we need to address the lack of experimental infrastructure suitable for network security. A study of the literature shows that current experimental approaches are not appropriate to generate high fidelity network data. Consequently, in order to accomplish a comprehensive evaluation, first, we conduct our experiments on two separate analysis case study scenarios, inspired from the DARPA 2000 and UNB ISCX IDS evaluation data sets. Next, as a complete

field study, we employ *Pasargadae* in a real computer network for a two weeks period to inspect its detection capabilities on a ground truth network traffic. The results obtained show that compared to other existing IDS improvements, the proposed contributions significantly improve IDS performance (detection rate) while reducing false positives, non-relevant and duplicate alerts.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SIGNS AND ABBREVIATIONS

AAFID	Autonomous Agent for Intrusion Detection
ABAIS	Agent Based Artificial Immune System
AIS	Artificial Immune System
AM	Analyzing Module
ANN	Artificial Neural Network
CAC	Central Analyzer and Controller
C&C	Command and Control
CAPEC	Common Attack Pattern Enumeration and Classification
CEE	Common Event Expression
CEP	Complex Event Processing
CMS	Content Management System
COTS	Commercial Line Interface
CS	Context Sensor
CSM	Cooperating Security Management
CTF	Capture The Flag
CVE	Common Vulnerabilities and Exposures
DAG	Directed Acyclic Graph
DBMS	Data Base Management System
DBN	Dynamic Bayesian Network
DD	Data-dependent Decision
DDoS	Distributed Denial of Service
DIDS	Distributed Intrusion Detection System
DIPPS	Distributed Intrusion Prediction and Prevention System
DL	Description Logic
DoS	Denial of Service
D-SCIDS	Distributed Soft Computing-based Intrusion Detection System
FCM	Fuzzy C-Means
FN	False Negative
FP	False Positive
FSM	Finite State Machine
GA	Genetic Algorithm
GNN	Genetic Neural Network
HAN	Home Area Network

HIDE	Hierarchical Intrusion Detection
HIDS	Host Based Intrusion Detection System
HMM	Hidden Markov Model
IDMEF	Intrusion Detection Message Exchange Format
IDS	Intrusion Detection System
ID2S	Intrusion Detection and Diagnosis System
IEEE	Institute of Electrical and Electronics Engineer
IETF	Internet Engineering Task Force
IDS	Intrusion Detection System
IIS	Internet Information Services
IPS	Intrusion Prevention System
IoT	Internet of Things
IT	Information Technology
KDD	Knowledge Discovery and Data Mining
MAS	Multi-Agent System
NAN	Neighborhood Area Network
NIDS	Network Based Intrusion Detection System
NVD	National Vulnerability Database
OS	Operating System
OVM	Ontology for Vulnerability Management
OWL	Ontology Web Language
OWL-DL	Ontology Web Language Description Logic
PPV	Positive Prediction Value
ROC	Receiver Operating Characteristic
RST	Rough Set Theory
SAO	Security Attack Ontology
SASO	Security Algorithm-Standard Ontology
SAVO	Security Asset-Vulnerability Ontology
SDO	Security Defense Ontology
SFO	Security Function Ontology
SGDIDS	Smart Grid Distributed Intrusion Detection System
SOC	Security Operation Center
SQWRL	Semantic Query-Enhanced Web Rule Language
SVM	Support Vector Machine
SWN	Sensor Wireless Network
SWRL	Semantic Web Rule Language

TN	True Negative
TPE	Transfer Probability Estimation
UML	Unified Modeling Language
UNM	University of New Mexico
WAN	Wide Area Network
WIND	Workload-aware Intrusion Detection

CHAPTER 1 INTRODUCTION

Today, the rapid growth of virtualization and cloud computing and the emergence of the several heterogeneous infrastructures, has led to the evolution of various applications, services, and systems within a computer network. At the same time, there is an increasing trend of attacks on these assets to exploit their possible vulnerabilities by malicious entities. Hence, technologies and defensive systems aiding the efforts of Information Technology (IT) personnel to improve the responsiveness and reliability of their organizations IT assets (i.e. network infrastructure and computer systems) continue to be paramount.

Intrusion detection systems (IDS) are among the most popular of the front line tools to defend computation and communication infrastructures from intruders. IDS collect and analyze information from computers and network devices to identify possible security breaches against the systems or the network infrastructure. Various IDS types have been proposed in the past two decades and commercial off-the-shelf (COTS) IDS products have found their way into Security Operations Centres (SOC) of many large organisations. Nonetheless, the usefulness of single-source IDS has remained relatively limited due to two main factors: their inability to detect new types of attacks (for which new detection rules or training data are unavailable) and the often very high rate of false positives. Due to the increasing prevalence of complex multi-pronged attacks, the necessity for organizations to have access to and deploy reliable IDS that correlate across all available sensor types and other sources of security-related information (including system and application logs) is undeniable. This is especially important with respect to targeted attacks that try to avoid detection by conventional IDS and by other security products.

One of the approaches that have been suggested to address these problems is that of alert correlation, where the alert stream from several different IDS, or more generally various alert sensors, is jointly considered and analysed to provide a more accurate threat picture. However, these systems are only limited to IDS, and they do not employ other sensor types (e.g. firewalls, web servers, operating systems, antiviruses, databases, etc.) that generate valued event logs helping to identify malicious behaviours happening inside a computer network. In addition, these systems do not take into account contextual information in their analysis that can be considerably useful to reduce non-relevant alerts and false positives rate. Lack of automation is another important problem of current alert correlation systems.

The work presented in this thesis strives to address the problems described above, and provide a comprehensive solution to significantly improve IDS effectiveness. We propose a

comprehensive ontology-based (automated) event correlation system that 1) correlates event logs across several heterogenous types of sensors, 2) incorporates contextual information into its analysis to reduce false positives and non-relevant alerts.

To get to this point, we start by providing a brief overview of IDS types and their detection methods in section 1.1. In Section 1.2, we introduce some improvements proposed by researchers to make current IDS technology capable to mitigate the influences of sophisticated attacks. Using the analysis of the current situation as a starting point, we define our research problem in Section 1.3. Section 1.4 presents our research objectives in this thesis to address current IDS technology problems. Section 1.5 presents our contributions in this thesis. Finally, Section 1.6 details the organization of this thesis in which the efforts to achieve our research aim are summarized.

1.1 Intrusion Detection Systems

Intrusion Detection Systems play a key role in the surveillance and monitoring of computer network infrastructures. These systems inspect events that have occurred in computer systems and networks, and in case of any malicious behavior, they generate appropriate alerts describing the attacks' details.

In the recent years, several types of IDS have been proposed. These systems can be split into several categories based on various factors (Figure 1.1) [49]:

— **Detection method**: Intrusion detection systems in term of detection method can be categorized into three main categories:

1. Misuse-based IDS: A misuse-based IDS is based on an attack signature-based approach that compares every happening event with the attacks existing in its signature database [36, 79, 128]. Once it finds any match, a particular alert, according to the type of the event, will be reported. In these IDS, signature database shall be updated to contain more recent attacks. Although these IDS have higher detection rate, their main shortcomings are: 1) they are not able to detect zero-day attacks, 2) modeling complex signatures in these systems is a time consuming and difficult task.
2. Anomaly-based IDS: An anomaly-based IDS, based on the normal behavior of a system, models its normal profile. Any deviation from this normal profile is reported as a malicious behaviour. If the system models the normal profile accurately enough, its false positive rate will be significantly reduced. However, because the accurate modeling of the normal profile is a difficult task, the false

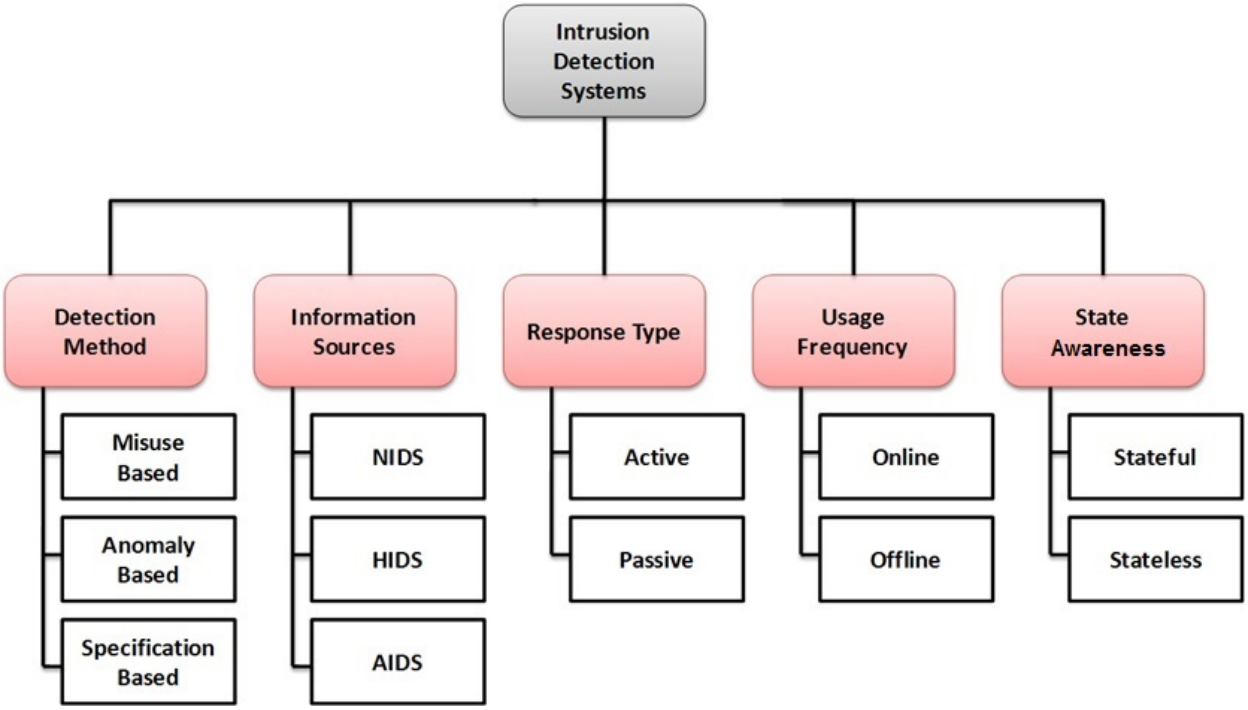

Figure 1.1 Intrusion detection systems categories

positive rates in these systems are higher than signature-based IDS.

3. Specification-based IDS: in these IDS, a number of specifications are defined for network protocols or software processes. Any deviation from these specifications is considered as a suspicious event.

— **Information sources**: Intrusion detection systems in term of information sources are categorized into three major categories [21]:

1. Host-based IDS (HIDS): basically, a HIDS monitors a single host to detect anomalous attempts. It analyzes a wide variety of activities including applications and the system logs, operating system audit trails, and contextual information. For example, it usually watches for suspicious login attempts, unauthorized or abnormal file access, etc. Thus, HIDS have detailed information in hand for their analysis.

2. Network-based IDS: NIDS monitor and analyze the ongoing traffic at several locations of a network. Its analysis can be in different levels of sophistication. NIDS might be distributed in the entire of a network, but mostly in the strategic places. In such cases their main components can be detection sensors, management server, database server, and consoles.

3. Application-based IDS: this type of IDS are a subset of HIDS, and they usually monitor one or a number of specific applications, such as database management systems (DBMS), content management systems (CMS), accounting systems, etc. Their main information sources are the log files of these applications.

— **Types of response**: in term of how to react to a particular suspicious behavior, intrusion detection systems can be categorized into two main categories:

1. Active IDS: Active systems automatically undertake further actions either to prevent an attack or to collect additional information. In case of suspicious events, these systems may look closer to the events to do more detailed analysis. These systems may have not capability to directly deny the activities of an intruder. However, they can change the system configuration such that proceeding by the attack becomes at least more difficult.
2. Passive IDS: Passive IDS inform to the system administrators when any suspicious event is happening. Pop-up messages and email notification are some of the standard tasks of passive IDS.

— **Usage frequency**: in term of usage frequency, intrusion detection systems can be categorized into two main categories:

1. Online IDS: An online IDS performs continuous and real-time analysis on network events immediately after they happen. The main advantage of this approach is that system activities can be analyzed timely. Thus, an appropriate response can be issued once an attack is detected. However, the system overhead in these cases is significantly high.
2. Offline IDS: Offline IDS analyze network traffic periodically, and looks for abnormal events [29]. They do not provide any protection between two consecutive analyses. Therefore, in case of any successful attack, they can be used only for postmortem analysis.

— **State awareness**: in term of state awareness IDS can be categorized into two major categories:

1. Stateless IDS: Stateless IDS analyze events independently from each other. In other words, these systems do not take into account the events relationships. Even though these IDS simplify the system design process, they are not able to detect complex attacks, such as distributed and multi-step attacks.
2. Stateful IDS: Stateful IDS maintain information related to the past events. When they are inspecting an event, they consider its relation to the past events (the

event stream approach). While these systems introduce additional complexity in the design process, they have significant advantages i.e. they have a potential to detect complex attacks.

In order to choose a specific IDS type for a particular application, we need to inspect IDS types based on a comprehensive list of criteria. For this purpose, a number of criteria have been defined by researchers [24, 49]. These criteria are as follows:

- **Accuracy**: An IDS should detect normal and abnormal events accurately. Inaccuracy occurs when: 1) an IDS considers an abnormal or malicious event as a normal or legitimate one (false negative), 2) an IDS identifies a normal or legitimate event as an abnormal or malicious one (false positive).
- **Performance**: performance indicates the processing rate of audit events. Typically, online IDSs have higher performance because they should react to the audit events in a real-time mode.
- **Completeness**: completeness indicates the ability to detect all attack types. Therefore, incompleteness occurs when an IDS is not able to detect and analyze a particular attack type. Measuring this criterion is more difficult than others because it needs detailed knowledge about attack types.
- **Adaptability and scalability**: it is necessary for an IDS to be configured to detect new attacks and operate under new circumstances. In other words, adaptability and scalability indicate on the ability of an IDS to adapt with new environments (topologies, systems, software, etc), new configurations, and even new scales.
- **Availability**: sometimes attackers try to target the availability of IDS using attacks like denial of service (DOS). An IDS has to be resistant against such attacks. This criterion is important particularly for those IDS protecting very sensitive environments, such as military and commercial environments.
- **Timeliness**: an appropriate IDS should analyze every event as quickly as possible to protect the underlying system from upcoming significant problems. This criterion is very important especially for online IDS that perform their tasks in a real-time mode.

1.2 Improvements on Intrusion Detection Systems

As the various IDS available (COTS or research prototypes) can only detect attacks for which rules have been written (rule-based detection) or that were present in data used to train them (anomaly-based detection), each IDS tends to perform differently on each class of attack [1]. Therefore, many researchers and industry practitioners have tried to increase overall detection efficacy by running several different IDS on the same data sources (i.e. live

feed of system and network events), and combining their outputs.

In order to facilitate this task, several researchers have attempted to introduce tools and processes to implement *alert correlation*, where alerts generated by distributed IDS sensors located at different locations are integrated, correlated and jointly analysed to produce compact reports on security status [177, 198]. When each of these IDS examine the same type of data, one can speak of *homogeneous IDS correlation*. In fact, the majority of research and real-world deployment of correlation approaches involves the analysis of alerts generated by different network IDS (NIDS), such as SNORT or Bro, examining network traffic streams at different network locations.

One notorious sub-case of homogeneous correlation is alert fusion [15, 31, 89, 137, 143, 152, 188], where all IDS are examining events from the exact same data source and where a decision as to which alert-generating events are most relevant. Hence, for each event we might have some IDS generating an alert while others do not. Alert fusion thus includes the decision process in which for each alert-generating event, we consider the output of all IDS and decide how important and relevant it is, and how to report in a concise form usable by a security analyst. While alert fusion is notionally a specific case of alert correlation of same-source IDS, it can also be used as an intermediate step in more generic heterogeneous correlation of IDS monitoring different types of event streams.

Nonetheless, most attacks, whether automated malware infections or manual network intrusions, do not leave traces only on network traffic captures but also on host-based IDS (HIDS) and other security products, and sometimes even on non security-related logs of commodity or corporate applications. This fact has been successfully exploited by security analysts worldwide to detect sophisticated attacks by visually or manually correlating these various information and alert sources. Because all of these sensors examine different types of events and raw data sources, one can speak in this case of *heterogeneous alert correlation*.

One of the important difficulties of heterogeneous correlation is the integration of data from various alert sources, each having potentially different formats and semantics. In order to be useful, the integrated information must capture the generic properties pertaining to all types of alerts in order to allow the analyst to consider the information as a whole. At the same time, sensor-specific attributes must also be retained in order to preserve the ability for the security analyst to drill down and refine his analysis, such as for finding root causes, determining attack type, objectives, etc. Having recognised the usefulness of alert correlation, whether homogeneous or heterogeneous, security researchers have attempted to create unified models for events and alerts, such as the notorious Intrusion Detection Message Exchange format (IDMEF) [50], which is now supported by many COTS NIDS and HIDS. However,

IDMEF does not solve all integration woes. It does not gracefully support non-standard attributes that might be needed for refined analysis (except through "user" fields) and is not suited for integrating other types of information that security analysts might want to correlate, such as application logs, configuration information, etc.

In fact, one very fundamental principle of event management is that security analysts must be able to understand and consider in which context an event originated. This is what allows to consider the relevance and relative importance of events. Contextual information that can be relevant to security analysis can include network topology and protocols, network, system or application configurations, user profiles and roles, etc. For example, an IDS alert that describes a malicious activity exploiting a PHP buffer overflow vulnerability (e.g. CVE-2014-4049), is considered as a real or relevant alert if and only if the PHP application is installed in the underlying context. Otherwise, it will be considered as a non-relevant alert even though it correctly recognizes the attack. Unfortunately, security analysts often need to manually gather such information from multiple systems to feed the correlation process in order to integrate and validate the alerts and identify the consequences of any intrusion. This is why certain researchers have proposed approaches to automatically include such contextual information into the event correlation process, an approach referred to as *context-aware* event correlation. The simple false positive-reducing idea applied here is simple and intuitive: events that are related to a certain type of attack are only relevant if the context in which they happen is indeed vulnerable to that type of attack. Thus, for context-aware event correlation to be useful it must also consider vulnerability information and, potentially, also attack models that describe how attacks require vulnerabilities and how they generate alert-triggering events.

Here again, the difficulty in implementing such approaches resides in integrating the information into a data model that is generic enough to allow a global view of the data, while retaining maximum data granularity for drill-down analysis. For example, assume that in one query we intend to list all the network-based events, and in another query we intend to list only proxy server events. As proxy server events are a specific subset of network events, in the first query we employ roll-up operations, and in the second query we employ drill-down operations in the objects hierarchy to obtain appropriate results. Furthermore, whether we are considering data representations for alerts, context, vulnerabilities or attacks, the ever-changing nature of threats and of our own IT infrastructures make it unattainable to try to design a unified one-size-fits-all data model. Flexibility and extensibility of the data model is thus a key requirement of any such approach. Lastly, the method by which security analysts extract information and intelligence from such data stores must itself also be flexible and extensible. It must support generic simple queries and detailed analysis, and

furthermore it must be relatively simple and quick for analyst to implement and run various correlation paradigms and algorithms.

1.3 Problem Statement

Intrusion detection systems are one of the key requirements of every organization to ensure confidentiality, integrity and availability of its IT resources. Unfortunately security professionals confirm that IDS are not technically advanced enough to detect, verify and assess many sophisticated attacks particularly in critical environments. At the other end of the technical spectrum, IDS often output a large amount of alerts having numerous redundant and
non-relevant alerts which make it a flustering task for security administrators to determine real alerts. Part of the problem is that most of the current IDS do not take to account contextual information (type of server, application, user, network location, etc.), and therefore a large portion of the alerts are non-relevant in that even though they correctly recognize an intrusion, the intrusion fails to reach its objectives. Additionally, to detect newer and complicated attacks, relying on only one detection sensor type is not adequate, and as a result many of the current IDS are unable to detect them. While many system administrators are known to successfully manually incorporate context information and many different types of sensors and logs into their analysis, an important problem with this approach is the lack of automation in both storage and analysis.

During recent years many research projects have proposed to address the mentioned problems in IDS applicability. However, as most of these works have focused on only one of these aspects, the problem as a whole has not been resolved. Most of the proposed alert correlation techniques consider correlation only across multiple sensors of the same type having a common event and alert semantics (homogeneous correlation), leaving it to security administrators to perform correlation across heterogeneous types of sensors. Another major shortcoming is the lack of accurate evaluation based on the data sets that encompass modern complex attack scenarios.

In summary, the most important shortcomings of current IDS technology are as follows:

- **Alert flooding**: The ubiquitous phenomenon of $alertflooding$ is exacerbated by the fact that most IDS generate large numbers of low-level alerts without any high level description, hence making it time consuming and difficult for security analysts to determine valid and important alerts and act upon them appropriately.
- **False positives and false negatives**: One of the major challenges in real-world IDS is the large amount of false positives that they generate, making it a difficult task for

security administrators to determine and identify real and important alerts.

— **Non-relevant alerts**: Most of the current IDS do not take into account contextual information (type of server, application, user, network location, etc.) to verify the correctness of generated alerts. Consequently, a large portion of the alerts are non-relevant in that even though they correctly recognize an intrusion, the intrusion fails to reach its objectives.

— **Continuous human interaction**: Human interaction is one of the fundamental requirement of current IDS. Usually, system administrators analyze (i.e. verifying, correlating, fusing, etc.) generated alerts, update signature databases, etc. This interaction significantly reduces detection speed and efficiency. Therefore, the lack of automation in both storage and analysis is considered as another main challenges of these systems.

— **Unable to detect zero-day attacks**: Ability to recognize new attacks when they are launched for the first time, is not principally possible for current IDS technology. This fact reduces the overall system performance and makes them unreliable in real environments.

— **Unable to detect multi-step attacks**: Detection of the recently emerged sophisticated multi-step attacks is a big challenge in current IDS research and technology. These attacks are mostly based on scenarios that employs methods such as social engineering, spam emails, Phishing, etc. to compromise legitimate users and machines, and through them accomplish malicious activities. In order to detect such complex attacks, relying on only one detection sensor type is not adequate, and as a result many of the current IDS are unable to detect them.

— **Limited scalability and flexibility**: Current intrusion detection and event correlation systems are not able to achieve the level of scalability required to effectively inspect high volume of audit data collected from distributed agents in large networks. Moreover, as they do not take into account contextual information, their easy reconfiguration to adapt in new environments or to employ various approaches in their detection and correlation engine is a problematic task.

— **Lack of proper and effective evaluation methods**: Effective evaluation of intrusion detection systems has been a considerable challenge for many years even though various evaluation approaches have been proposed. The most common approach has been to send attack traffic within some background traffic, and test the detection ability of systems. However, publicly available data sets do not include most of the recent complex attacks to efficiently evaluate intrusion detection systems.

1.4 Research Objectives

To address the challenges described in the previous section, the goal of this thesis is to design a comprehensive event correlation system that can provide: 1) context awareness, 2) system automation, 3) sensor heterogeneity, and 4) appropriate correlation through various event logs. Our proposed approach consists of a set of extensible ontologies to automatically perform event correlation by reasoning on the information collected from various information resources.

We summarize the detailed objectives of our research as the following items:

1. Develop a unified and comprehensive ontology-based and context-aware event correlation framework to seamlessly and automatically implement various alert and event
   correlation approaches on the same data model. The proposed framework will be flexible enough to be employed as a reference correlation framework to implement other works.
2. In order to allow IDS analysis automation, designing comprehensive and extensible ontologies, allowing correlation and reasoning with information collected from various resources including system context, vulnerabilities, attacks, and event logs. Additionally, In order to allow context-awareness, incorporating useful contextual information into the analysis from either explicit information in Configuration Management Systems (CMS) or from implicit information obtained by user and system profiling techniques.
3. Propose a semantic-based and context-aware event correlation approach performing heterogenous correlation from various sensor types (NIDS, HIDS, routers, firewalls, antiviruses, databases, operating systems, applications, etc.), and a semantic-based context-aware alert fusion approach fusing alerts received from various network-based intrusion detection systems.
4. Evaluate the flexibility and effectiveness of the proposed framework by applying to different deployment and analysis contexts, from simple to complex IT infrastructures, generic threat detection to complex attack forensics analysis.
5. A complete evaluation of the effectiveness and performance of the proposed event correlation and alert fusion approaches based on data sets including recent complicated attack scenarios and realistic laboratory experimentation.
6. Preparing a new data set in the lab environment to test and evaluate the proposed ontology-based and context-aware event correlation approach. The data set will need

to adhere to the protocol specifications, and will need to include multi-step attacks designed based on recently emerged attack scenarios.

1.5 Thesis Structure

This document presents a summary of our efforts to tackle the problems of generating a large amount of false positives and the lack of automation in intrusion detection systems. Various sections focus on the different efforts made to tackle our research objectives.

Chapter 2 presents a critical review of the current state of the art. In particular, in this chapter, we review machine learning-based intrusion detection systems, alert correlation systems, alert fusion systems, host-based intrusion detection systems, distributed intrusion detection systems, context-aware intrusion detection systems, and evaluation metrics and data sets.

Chapter 3 provides a background and basic knowledge required to follow the next sections. We bring an introduction about ontologies and their impact in computer security research. We also introduce Ontology Web Language Description Logic (OWL-DL) and Semantic Query-Enhanced Web Rule Language. A review on recent ontology-based intrusion detection systems is another part of this chapter.

Chapter 4 presents *Pasargadae*, the proposed context-aware and ontology-based event correlation framework. In this chapter, we describe the event correlation phases using *Pasargadae*, designed ontologies (i.e. context, event, vulnerability and attack), and the correlation engine. Next, we propose a novel semantic-based and context-aware event correlation approach that employs *Pasargadae* as its main framework to perform event correlation. We describe how *Pasargadae* can be employed to implement other alert and event correlation approaches. In the last section of this chapter, we describe our proposed semantic-based and context-aware alert fusion approach.

In Chapter 5, we describe how *Pasargadae* which is the main framework of all the proposed approaches, was implemented in our lab and field test environment. For this purpose, we explain various tools and methods that were employed to implement *Pasargadae*'s components.

Chapter 6 demonstrates the flexibility of our framework by applying it to analyze some different case studies. In this chapter, we evaluate the proposed correlation and fusion approaches from various perspective and based on different popular data sets.

In Chapter 7, as a complete field test, we concentrate on applying the proposed event correlation and alert fusion approaches in a real network environment to analyze ground truth network traffic. We mainly evaluate the performance and efficiency of the proposed

approaches to show how these approaches behave in real world usages.

Finally, Chapter 8 presents the general discussion of our results and contributions with respect to our initial research objectives. This chapter also proposes avenues for future research that have been opened by our contributions.

CHAPTER 2 PREVIOUS WORK IN INTRUSION DETECTION SYSTEMS

During recent years, many sophisticated attacks have emerged that target computer systems located in various organizations. These multi-step and coordinated attacks pose significant threats to these organizations and their intellectual property. In order to protect these valued resources against complex attacks, Intrusion Detection Systems (IDS) and alert correlation systems play a significant role to detect and report these threats. Consequently, a large number of intrusion detection and alert correlation approaches have been proposed in the literature over the recent years.

From a classification point of view, Cuppens and Miege [43] classify intrusion detection and alert correlation approaches into two main categories:

— **Explicit alarm correlation**, which relies on the capabilities of security administrators to express logical and temporal relationships between alerts in order to detect complex multi-step attacks. For instance, Morin and Debar [122], propose an explicit correlation scheme based on the formalism of *chronicles*. Other researchers have proposed imperative languages to express logical and temporal relationships in attacks in order to correlate sequences of the alerts [45, 170].

— **Implicit alarm correlation**, which is based on employing machine learning and data mining techniques to fuse, aggregate and cluster alerts for alert correlation and intrusion detection purposes. For instance, Chen and Aritsugi [34], employ Support Vector Machines (SVM) and co-occurrence matrices in order to propose a masquerade detection method. In [142], Raftopoulos performs log correlation using C4.5 decision tree classifiers after analysing the diagnosis of 200 infections that were detected within a large operational network. Almgren, Lindqvist and Jonsson [14] use Bayesian networks to correlate alerts generated from several audit sources to improve detection accuracy.

From another perspective, Yusof *et al.* [192] categorize intrusion detection and alert correlation techniques into four main categories: 1) Similarity-based techniques, 2) Pre-defined attack scenarios techniques, 3) pre-requisite and consequences of individual techniques, and 4) Statistical causal analysis techniques. The main goal of our research work is proposing a model having most of the advantages of these categories while reducing major disadvantages.

This chapter presents a review of the state of the art of intrusion detection and alert correlation approaches. It starts by presenting machine-learning and data-mining based intrusion detection approaches. In section 2.2 and 2.3, we explain the importance of alert

correlation and alert fusion in improving IDS performance and efficiency. Section 2.4 presents context-aware alert correlation and intrusion detection systems. In section 2.5, we describe distributed and agent-based intrusion detection systems. Section 2.6 presents host-based intrusion detection systems. Section 2.7 reviews intrusion prediction and prevention techniques. In section 2.8, we describe some evaluation metrics in order to compare the efficiency of intrusion detection and alert correlation approaches. Section 2.9 presents some of the popular data sets that are frequently employed to evaluate the performance and efficiency of intrusion detection and alert correlation techniques. Finally, section 2.10 inspects the advantages and disadvantages of the prior works, and describes our objectives in this work considering the prior works.

2.1 Intrusion Detection and Alert Correlation Using Machine Learning Techniques

In order to analyze network traffic and classify every event into normal or malicious classes, machine learning techniques can be an appropriate solution. Hence, during recent years various machine learning-based intrusion detection and anomaly detection systems have been proposed. Some of these approaches employ only a single learning technique, such as neural networks, genetic algorithms, decision trees, etc. On the other hand, some approaches are based on a combination of several learning techniques, such as hybrid or ensemble techniques.

Essentially machine learning techniques are split into two major categories: supervised and unsupervised learning techniques. Supervised techniques, first, in the training phase, learn classes using a labeled training data set. Then, they analyze and classify a test data set based on the knowledge learned in the training phase. In such techniques, the training data set plays an important role in improving total performance. On the other hand, in the unsupervised techniques, the training data set is not labeled. In these techniques, some different data sets are given to the machine to cluster them by their similar features. In the following, we describe how machine learning techniques have been employed by researchers to propose new intrusion detection approaches.

2.1.1 Bayesian Networks

Bayesian networks represent another approach to detect and prevent anomalous activities in computer networks. In this technique, there is a Directed Acyclic Graph (DAG) which represents causal relationship of events [174]. In this graph, the nodes represent events and the edges represent their causal relationship. The graph can be used for both inference and

prediction purposes. Bayesian networks are considered as a supervised machine learning technique. Both learning the graph and the probabilistic table are possible in these systems. During recent years many intrusion detection and prevention approaches based on Bayesian networks have been proposed [63, 100, 101, 124, 133, 148, 173, 174]. In the following we briefly describe some of these works.

Tuba and Bulatovic in 2009 [173], proposed a standalone IDS based on a large Bayesian network. Their proposed approach consists of two main steps: 1) developing a tool to design a number of small components representing the basis of a Bayesian network, 2) developing a tool interconnecting these components in a way that provides efficient control on network complexity. To this end, they define a small number of natural templates called idioms, that facilitate the design of Bayesian networks. The main feature of these idioms is their ability to represent a graphical structure without probabilistic tables. Consequently, these idioms speed-up the Bayesian network development, and improve their quality. In order to evaluate the proposed approach, the authors have used it to improve privacy of medical data, and they show that using Bayesian networks for intrusion detection purpose outcomes efficient and acceptable results. However, we believe that the proposed approach cannot eliminate non-relevant alerts because it does not take into account contextual information in the intrusion detection process.

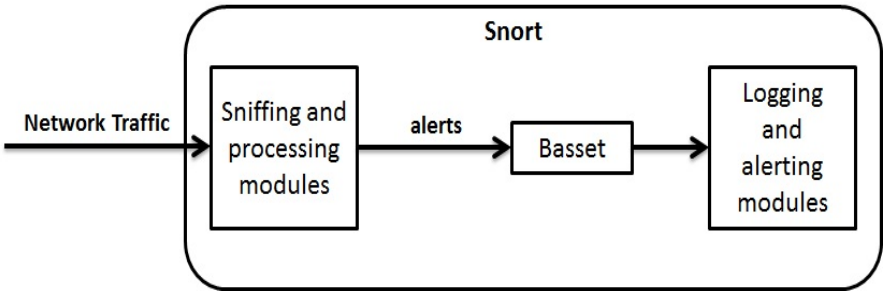

Figure 2.1 The integration of Basset and Snort [174]

Tylman in 2010 [174], presented a system called Basset that improves functionality of Snort IDS by incorporating a Bayesian network as an additional processing stage. Based on the authors claim, the flexible structure of Basset makes it applicable in both misuse-based and anomaly-based detection processes. They use Snort as the first stage of their detection process which has two main advantages: 1) Snort works as a reliable packet sniffer, 2) it provides valuable information for the next steps. The proposed approach intercepts Snort alerts before transferring into the output plug-in. First, the intercepted alerts are fed to the Basset for more analysis. Then, the results are fed to the Snort output plug-in. Figure 2.1 illustrates this process. The main disadvantage of this approach is the modifications that the

system enforces to Snort's core engine.

Dewan *et al.* in 2010 [63], proposed a learning algorithm for adaptive NIDS based on naive Bayesians and decision trees. The main goal of proposing this work was addressing some difficulties of data mining based approaches, such as reducing noise in training data, handling continuous attributes, and dealing with missing attribute values. The authors have evaluated the proposed work using KDD 99 data set, and based on their experiments, they believe that it significantly reduces the false positive rate. However, because KDD does not cover most of the recent complex attacks, we cannot consider it as a reliable evaluation factor.

2.1.2 Data Mining Methods

Essentially, during an intrusion detection process a large volume of host and network traffic is analyzed. Data mining methods can be considerably useful to perform such analyses. Generally, a data mining process has three main steps [162]: 1) initial exploration, 2) pattern identification, and 3) deployment. Various data mining approaches such as classification, clustering, association rule mining, and outlier detection approaches are frequently used during intrusion detection processes to determine and analyze captured traffics, and eventually, discover malicious behaviors [51, 58, 78, 102, 105, 106, 162, 190].

Xiao *et al.* in 2010 [190], proposed a novel unsupervised data mining based method to efficiently handle generated alerts by IDS. In this work, they use an outlier detection approach to determine true alerts and reduce false positives. They assign an FP score to each alert that indicates its probability of being a false positive. Their proposed approach mainly has three steps. First, using a mining algorithm, all existing frequent item sets in the data set are recognized. Then, based on the frequent patterns, the outlier score of each transaction is calculated. Finally, based on the outlier scores, all the transactions are sorted in ascending order, and the first $p\%$ is chosen as candidate outliers. In this work, a dynamic set of features is considered to describe normal behaviors, i.e. once a new alert is generated, a new normal behavior emerges. This fact based on the authors claim, is the main advantage of the proposed approach. In order to update the feature set, the corresponding frequent pattern of any new behavior should be constantly added to the feature set. However, the main shortcoming of this work is that it does not take into account the semantic information of the alerts.

Barbara *et al.* in 2001 [25], proposed the ADAM (Audit Data Analysis and Mining) anomaly detection system that combines association rules mining and classification to detect malicious events in a TCPdump audit trail. First, ADAM employs data mining to build normal behaviour profiles. Next, based on a sliding-window online algorithm it recognizes frequent

item sets in the last D connections and compares them with the normal profiles to detect malicious events. An important limitation of this work is that it needs a comprehensive list of normal profiles to perform an efficient anomaly detection that is very difficult task to prepare. Otherwise, it will produce a large number of false positives. It also does not consider the alerts semantic information in its analysis.

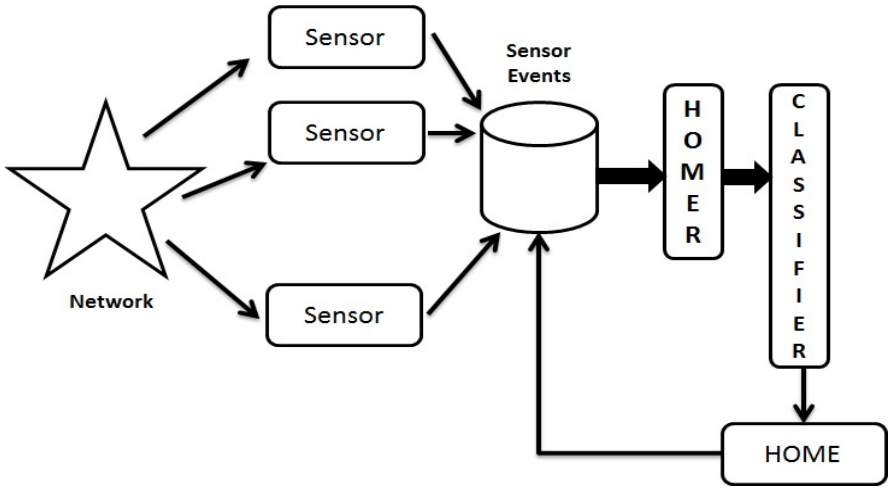

Figure 2.2 Srinivasulu's intrusion detection framework [162]

Srinivasulu *et al.* in 2009 [162], applied data mining classification techniques into an intrusion detection process to improve its efficiency. In this work, CART [114, 140], Naïve Bayesian and Artificial Neural Network (ANN) are applied into an intrusion detection process. Consequently, the results were compared together to show which classification method is more efficient. Figure 2.2 illustrates the proposed framework. In the proposed framework, in order to determine false positives, before transferring into the classifier, the alerts are filtered by HOMER. The results indicate that Induction tree and ANN are more efficient than Naïve Bayesian. However, time complexity of ANN is more than other classifiers. The main shortcoming of this approach is that the classifiers only classify events based on limited number of features depending on behaviours observed with limited number of events. Moreover, they do not take into account contextual information and event interrelationships.

2.1.3 Decision Tree

Decision trees are one of the most popular machine learning techniques. They provide a simple and efficient predictive model. Decision trees are useful to define and improve intrusion detection rules while reducing the manual analysis rate. During recent years many intrusion detection approaches based on decision trees have been proposed [9, 15, 62, 126, 127, 156,

163, 165, 189]. In most of these approaches, decision trees play an integrated classifier role to improve overall detection efficiency. In the following, we briefly describe some of these works.

Mulay *et al.* in 2010 [126], proposed a hybrid intrusion detection approach based on Support Vector Machines (SVM) and decision trees. In this work, their main goal from combining SVM and decision trees is to improve the training time, testing time and accuracy of IDS. Figure 2.3 illustrate the proposed intrusion detection approach. First, the proposed approach prepares five SVM models for five types of labeled data. Then, these five types of patterns are organized into a binary tree. The authors believe that combining these two techniques produces better results than using a single classifier. However, they have proved it by a comprehensive experiments. Another shortcoming of this work is that it cannot detect zero-day attacks because of using only supervised learning techniques.

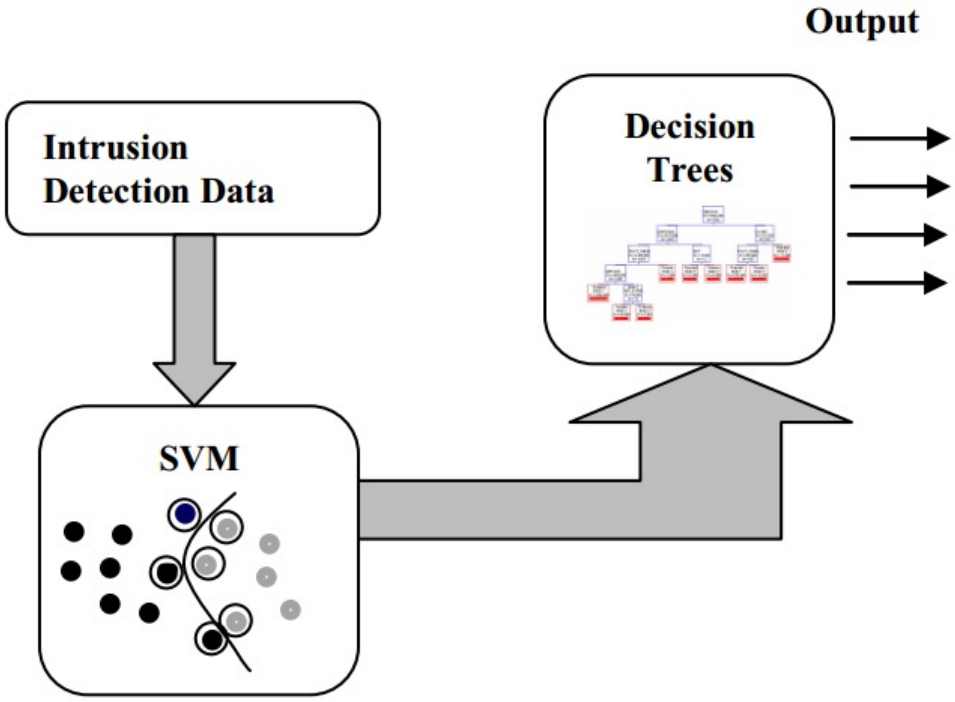

Figure 2.3 Mulay's proposed IDS [126]

Stein *et al.* in 2005 [163], proposed a novel misuse-based intrusion detection approach using a combination of Genetic Algorithms (GA) as a feature selector and decision trees as a classifier. Figure 2.4 illustrates the proposed framework. First, the initial population is randomly produced. Within this population, each individual consists of 41 genes where each one represents a particular aspect of a network connection. In this framework, for each individual a C4.5 program [140] is considered. Once, the fitness values of all the individuals

were computed, the GA generates the next generation that is an optimized feature set. Based on the authors claim, the proposed hybrid approach improves the system efficiency because it concentrates only on the related features. However, the proposed approach does not consider the individual relationships. Consequently, it can not detect multi-step and distributed attacks.

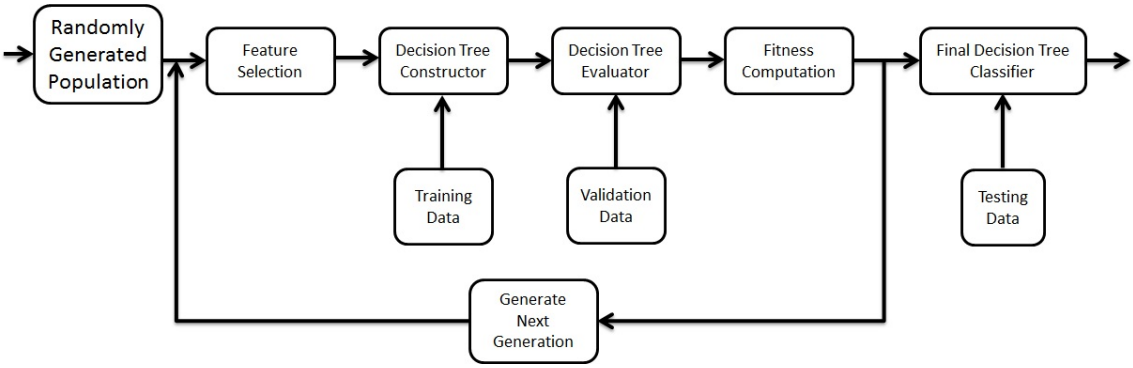

Figure 2.4 Stein's hybrid GA/decision tree intrusion detection approach [163]

Xiang *et al.* in 2008 [189], proposed a hybrid multi-level intrusion detection approach based on decision trees as a supervised classifier and Bayesian clustering techniques as an unsupervised classifier. In the proposed approach that has four classification steps, C4.5 has been used as a decision tree classifier. First, it categorizes network traffic into three subclasses: DOS, $Probe$ and $Others$. Then, the $Others$ class is categorized into two subclasses: $Normal$ and $Attack$. In the third step, the $Attack$ class is categorized into $U2R$ and $R2L$ subclasses. Finally, in the last step, the attacks are categorized into more specific subclasses. The main reason of using unsupervised classifier in the third step is the close similarity between $U2R$ and $R2L$ with the $Normal$ traffic. Therefore, after filtering out the $Normal$ connections, C4.5 categorizes $U2R$ (User-to-Root) and $R2L$ (Remote-to-Local) into more specific subclasses. Based on the authors claim, combining both supervised and unsupervised classifiers significantly improves the system efficiency while reducing the false positive rate. However, the main shortcoming of this work is that they do not take into account the contextual information to eliminate non-relevant alerts. Ignoring semantic information of the alerts is another limitation of this work.

2.1.4 Artificial Neural Networks

An Artificial Neural Networks (ANN) is a computational model based on biological neural networks. In this model, first, the system using a training data set, learns various patterns. Then, it can be applied to a test data set. Because of significant adaptability and learning

capability, ANN can be very useful for intrusion detection purposes. During recent years, many intrusion detection approaches based on ANN have been proposed [13, 27, 28, 40, 110, 125, 139, 181, 183]. In the following we briefly describe some of them.

Beqiri in 2009 [27], proposed an ANN based intrusion detection approach. In this work, novel concepts such as multi-layer perception mode, multi-level perceptron neural networks, and Hierarchical Intrusion Detection (HIDE) have been presented. In this paper the authors highlight major challenges of the intrusion detection research such as high false positive rate, continuous human interaction, etc. Based on the authors claim, unsupervised learning and fast network convergence are two significant capabilities of neural networks that can be integrated into intrusion detection processes. However, this work classifies events based on some features depending on behaviour observed with one type of events, and they do not take into account contextual information and event interrelationships.

Wang *et al.* in 2009 [183], using feed forward Backward Propagation (BP) neural networks proposed an intrusion detection approach based on workflow feature definition. In the proposed approach, workflow allows to define new attack sequences to assist BP neural networks in order to detect new attack types. First, the system analyzes network traffic to classify normal users' behaviors. Then, it detects possible existing attacks. During the training phase, in order to simulate real attacks, the authors have injected noisy data into the training data set. In this phase, they use several noisy data sets to simulate various attacks. The noise level has direct influence on the recognition percentage. The result will be acceptable only if it satisfies precise corrected data set with attack workflow feature and low noise level. The main difficulty of this work is creating a comprehensive training data set that can significantly reduce false positive rate.

Linda *et al.* in 2009 [110], proposed IDS-NNM (Intrusion Detection System using Neural Network based Modeling) wherein a specific window-based feature extraction technique is derived from the analysis of network traffic in a critical infrastructure. For this purpose, a combination of two neural network algorithms, the Error Back-Propagation and the Levenberg-Marquardt algorithm, is employed for training purpose. Based on their experiments, the authors believe that the proposed approach is able to detect long intrusion attacks as well as short intrusion attempts. However, the main shortcoming of this work is its inability to detect zero-day attacks. Preparing a comprehensive training data set is another difficulty of this works.

2.1.5 Fuzzy Logic

Fuzzy logic has several significant characteristics that make it useful in various research areas as well as intrusion detection systems. The three major reasons that researchers use fuzzy logic in the IDS research are: 1) security itself involves fuzziness, 2) fuzzy systems can easily integrate many data types generated from various sources, and 3) the generated alerts by intrusion detection sensors are mostly fuzzy because we cannot strictly emphasize on their intrusiveness. During recent years, a number of fuzzy logic based intrusion detection approaches have been proposed [56, 57, 83, 166, 166, 181]. In the following, we describe the works proposed by Tajbakhsh [166], Wang [181] and Sangeetha [147].

Tajbakhsh *et al.* in 2009, proposed an intrusion detection framework based on data mining. In this framework, they perform association-based classification using fuzzy association rules within the classification engine. During the classification process, every new sample is classified considering its compatibility with predefined classes. In order to induce desired set of association rules, a modified version of the standard Apriori algorithm [103] is employed. Essentially, the proposed framework has two main phases namely training and detection phases. A trapezoid membership, based on Fuzzy C-Means (FCM) clustering, is used as a fuzzy membership approach. Within the item reduction module, an association hyper-edge that is basically sets of items that are strongly predictive to each other, is used. The authors believe that the proposed approach has several advantages such as human comprehensible rules, handling symbolic attributes, and efficient classification on large data sets. However, the main difficulty of this work is preparing a comprehensive training set including various samples. Otherwise, it will generate a huge rate of false positives.

Wang *et al.* in 2010, proposed a hybrid ANN and fuzzy clustering based intrusion detection approach, called FC-ANN. In this work, first, they use a fuzzy clustering technique to generate some training data sets. Next, using these training data sets, a number of ANN modules are separately trained. Finally, the results are aggregated by a fuzzy aggregation modules. Figure 2.5 illustrates a graphical version of this process. The authors based on their experiments on the KDD CUP 1999 data set, believe that dividing the training data set into some subsets improves detection performance. However, this work does not take into account the semantic information and interrelationship of the alerts. Hence, it cannot detect distributed and multi-step attacks.

Sangeetha *et al.* in 2010, proposed an application layer intrusion detection approach called FASIDS based on the Fuzzy Rule-Base algorithm. The proposed approach combines a semantic-based IDS with a fuzzy-based one. In this approach, HTTP traffic headers and payloads on the application layer are analyzed for possible intrusions. Figure 2.6 illustrates

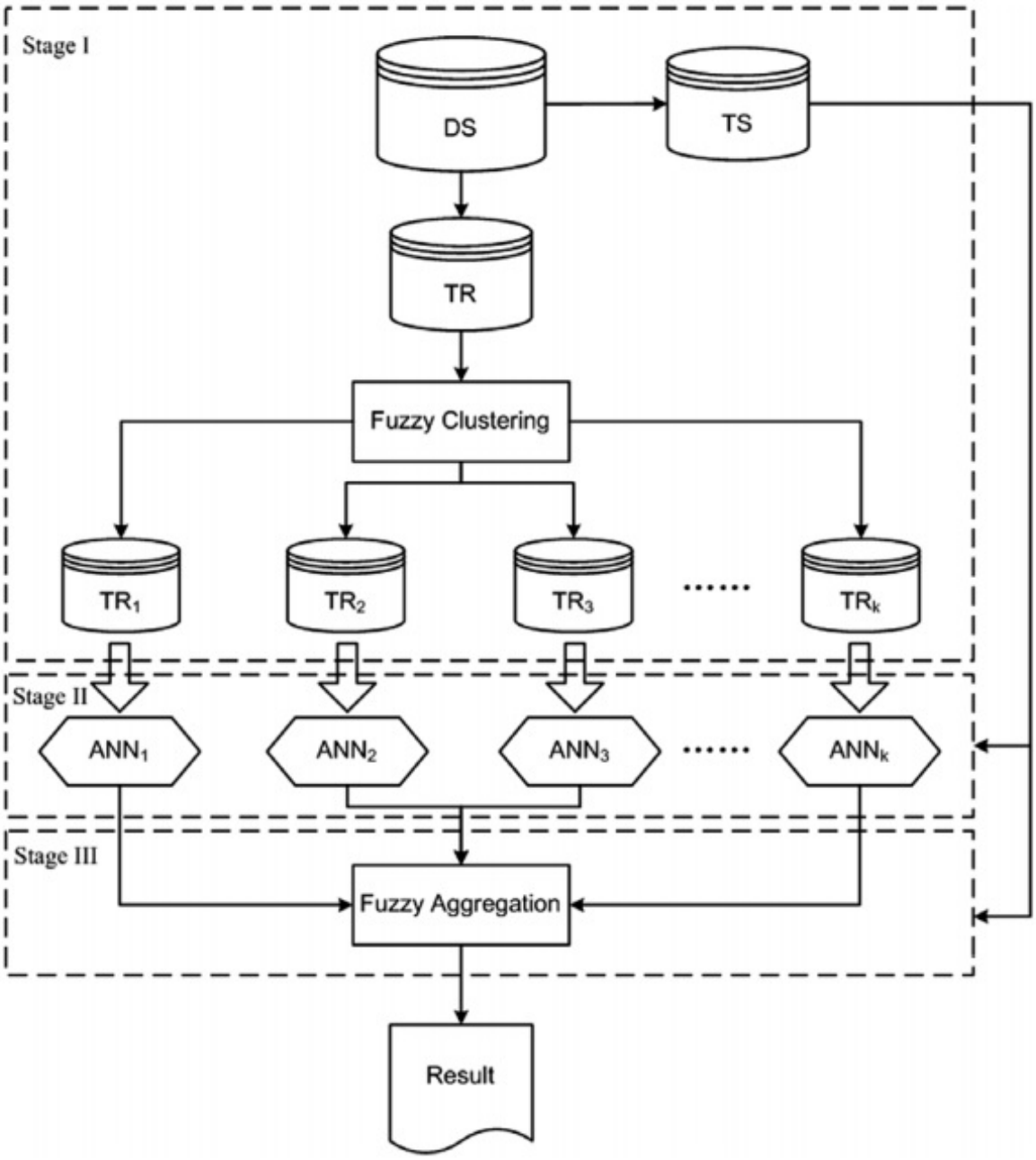

Figure 2.5 FC-ANN framework [181]

the FASIDS block diagram view. As the figure shows, first, the HTTP sniffer captures the application layer traffic. Session Dispatcher module differentiates the header and the payload of the captured data, and transfers each one to the corresponding modules. Header Analyzer module, lists objects existing within HTTP packets. IDS Interpreter module, in case of any match between its rules and the input patterns, reports an intrusion. Otherwise, it transfers the patterns into the FIDS module for more analysis. Based on the authors claim, the proposed approach combines both the advantages of semantic-based IDS and fuzzy-based IDS. Consequently it provides more efficiency compared to other approaches that use only one of these methods. However, one of the major shortcomings of this work is that it is limited to the application layer, and in particular, it only covers the HTTP traffic. Whereas, most of the current multi-step attacks leave traces of their prior activity at the several layers of the protocol stack, which can be leveraged in the intrusion detection process.

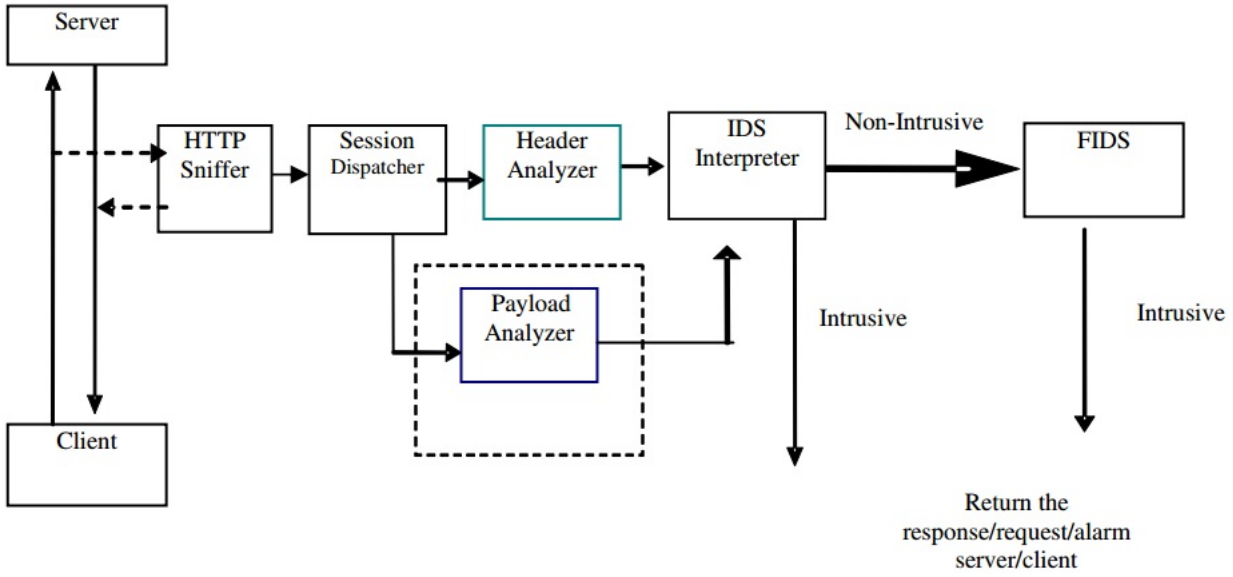

Figure 2.6 Block diagram view of integrated FASIDS [147]

2.1.6 Genetic Algorithm

Genetic Algorithms (GA) are basically a type of search algorithm that provides optimal solutions for various problems. Some of the major features of GA are adaptability with new environments, robustness to noise, self-learning capability, ability to produce initial rules without prior knowledge, and intrinsically parallelism. During recent years, many GA-based intrusion detection approaches have been proposed [12, 23, 74, 85, 95, 102, 108]. In the following we describe the works presented by Li, Bankovic and Jiang.

In 2004 Li [108], proposed a genetic algorithm-based IDS that represents how the information of network connections can be modeled as chromosomes. Figure 2.7 illustrates how GA has been applied to improve intrusion detection rule set's quality. DARPA data set has been used as the input into GA. The authors believe that this implementation of the genetic algorithm is unique because it takes into account both temporal and spatial information of network connections. However, using DARPA data set which covers only some old and basic attacks is not enough as the input of GA. We believe that in this phase they can use more comprehensive attack samples to outcome more optimized rules.

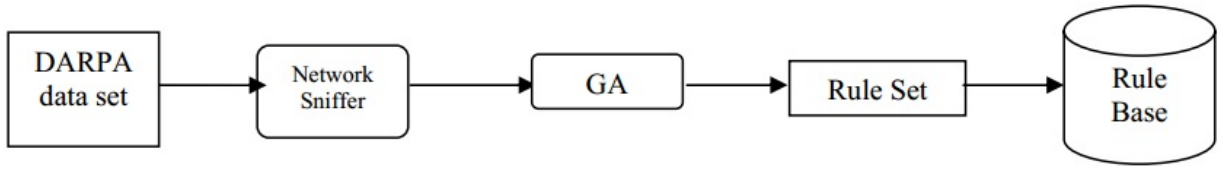

Figure 2.7 Architecture of applying GA into intrusion detection [108]

Bankovic *et al.* in 2009 [23], proposed a GA-based framework to classify network connections. The proposed framework composed of serial combination of two GA-based intrusion detection systems. The first system that is a single linear classifier, acts as an anomaly-based IDS. Therefore, both its detection rate and false positive rate are higher than the second one which is a rule-based classifier. The linear classifier splits connections into two classes: normal and potential attack. The main advantage of this classifier is its very low false negative rate. On the other hand, the rule-based classifier has the advantage of filtering and reducing false positive alerts. Consequently, there is a strong classifier upon a weak classifier to improve detection rate while reducing false positives. The authors believe that the proposed framework has some capabilities such as the high accuracy, the ability of dealing with rare classes, the inherent adaptability, and the feasibility of hardware implementation. We also believe that the proposed framework that combines an anomaly-based and a misuse-based detection system, is more effective than using only one classifier. However, this work does not consider semantic information of the alerts. It also does not use the interrelationship information of the alerts to detect multi-step and distributed attacks.

Jiang *et al.* in 2009 [95], proposed an anomaly-based intrusion detection approach using Genetic Neural Networks (GNN). In this approach they combine the advantages of both genetic algorithms (appropriate global searching) and BP neural networks (accurate local searching). Figure 2.8 illustrates the proposed approach. In this approach, first, network packets are collected and converted to a standard format analysable by GNN. Then, the GNN

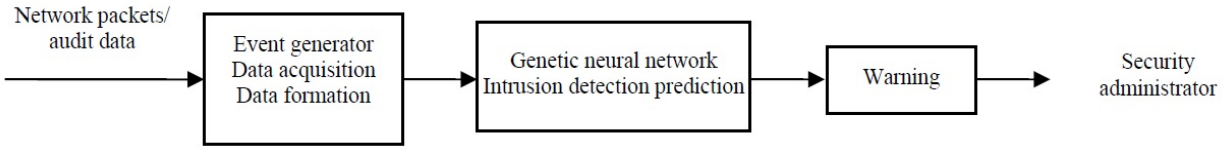

Figure 2.8 Jiang's GNN-based Intrusion Detection Model [95]

module analyzes the input data and in case of any malicious activity, generates appropriate warnings for security administrators. Based on the authors claim, the proposed approach has some significant advantages such as fast learning and higher accuracy. However, the main performance of this work depends on the initial data that GA uses to generated detection rules. If this data does not cover most of the common attacks, the system produces a huge rate of false positives.

2.1.7 Support Vector Machine

Support Vector Machines (SVM) are one of the popular machine learning techniques applied to various research areas such as pattern recognitions, nonlinear classifications, function estimations, density estimations, and intrusion detection systems. Essentially, SVM uses non-linear mapping to transfer an input data into a higher dimension. Therefore, an efficient mapping into a sufficient high dimension splits the input data into two main classes. This strategy can be useful to detect normal and malicious activities. During recent years SVM has been widely used in intrusion detection research [35, 36, 86, 109, 125, 164, 167]. In the following, we describe some of these works.

Teng *et al.* in 2010 [167], proposed a cooperative intrusion detection approach using fuzzy SVM. The proposed approach consists of three detection agents for TCP, UCP, and ICMP connections. Figure 2.9 illustrates the proposed framework. First, within the data preprocessing component, tasks such as filtering, cleaning, integrating, preprocessing data, attribute selection, and data conversion are accomplished. Then, the preprocessed data are transferred into its corresponding detection agent based on its protocol. In SVM all the input data should be in a same dimension. Therefore, all the input data are converted to a unified format. Next, the detection component analyzes the received data, and classifies it into intrusion or normal classes. Finally, the response unit produces appropriate responses based on the detected intrusions. The authors believe that classifying the network traffic based on network protocols improves the detection speed and efficiency. However, the main shortcoming of this work is that it cannot detect multi-step and distributed attacks because

it does not take to account alert interrelationships. Moreover, it only classify events based on some limited features depending on behaviour of observed traffic.

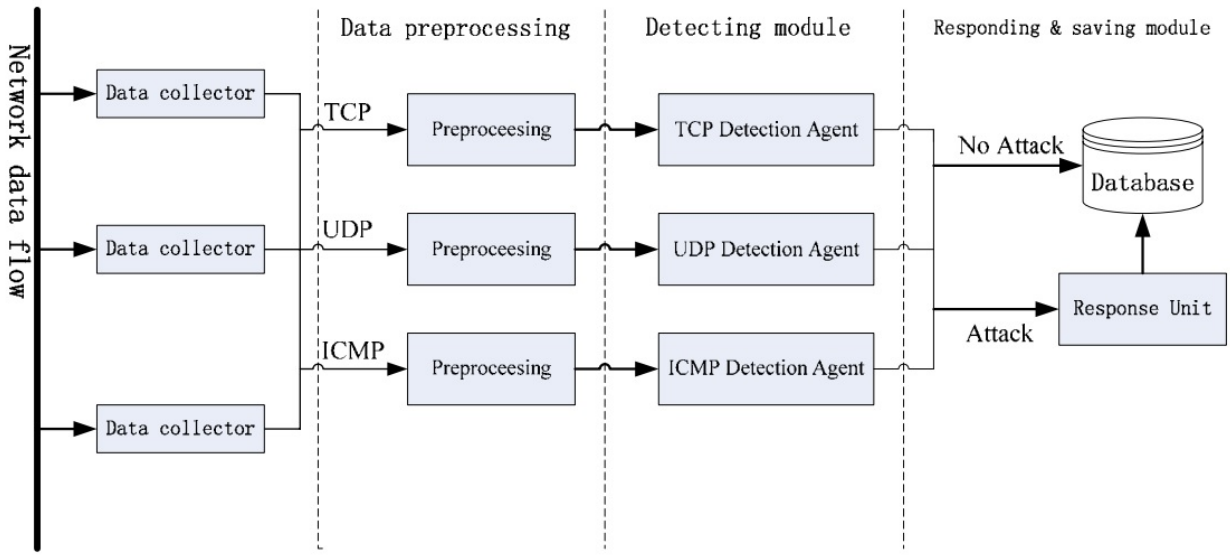

Figure 2.9 The architecture of the proposed IDS based on multi-FSVM [167]

Chen *et al.* in 2010 [36], proposed an intrusion detection approach using Rough Set Theory (RST) and Support Vector Machine (SVM). RST [32, 135] is one of the popular data mining techniques that reduces feature sets into an optimal subset. In the proposed approach, first, RST is used to preprocess the data and reduce its dimensionality. Then, the features selected by RST are transferred into the SVM module for both training and testing purposes. The SVM module splits the input data into two main classes: intrusion and normal. The authors believe that the RST-SVM achieves higher accuracy compared to the full feature or entropy methods. However, they have evaluated the proposed approach only against the MIT DARPA 1998 data set that only covers some basic attacks and not recently emerged complex attacks.

2.1.8 Hidden Markov Models

Hidden Markov Models (HMM) are probabilistic finite state machines to model stochastic sequences [67]. They are doubly stochastic processes consisting of an underlying Markov process that is not observable, and an observable process determined by the underlying Markov process [33]. In recent years, HMM has been applied to a variety of applications such as signal processing, pattern recognition, speech recognition, time series analysis, as well as intrusion detection systems [33, 37, 67, 82, 88, 98, 195]. In the following we review

some of the recently proposed intrusion detection approaches based on HMM.

Flores *et al.* in 2010 [67] proposed an anomaly-based intrusion detection model based on Hidden Markov Models (HMM). In this work, a genetic algorithm is employed to train the HMM. The number of states, connections and weights, and probability matrix of states distributions are specified by the GA. First, the IDS takes a time series as input data, and trains the HMM without any human interference. Then, the trained HMM model is used for anomaly detection purposes. After evolution of HMM using the GA, it can be used to determine which observation sequences are anomalous. It is worth pointing out that HMM focuses on statistic-based anomaly detection. First, the statistic-based normal profile is constructed. Then, it can be used to determine anomalous behaviors. In their experiments, the authors compared their method with the Baum-Welch algorithm [141]. The performance of this work mostly depends to the comprehensiveness of the training data. If the training data does not cover enough attack types, the system will produce a huge rate of false positives.

Zeng *et al.* in 2009 [195], proposed a host-based anomaly detection approach using HMM. In this paper, to overcome one of the common shortcomings of HMM, such as excessive training complexity, the authors combine Rough Set Reduction and HMM. Their method has two phases. First, In the training phase the system call data is transformed to HMM observation sequences. Then, based on the observations, the HMM is evolved. In the testing phase, based on the observed system calls, the HMM is used to calculate the probability of whether it is anomalous or not. In this paper in order to train HMM, the Baum-Welch algorithm is used, and to evaluate the system, the Forward-Backward algorithm is used. Figure 2.10 illustrates the detection model based on HMM and rough set reduction. According to their experiments, the authors believe that their proposed approach in practice is more efficient than others, and it improves the detection rate and reduces the false alarm rate constantly. However, it does not consider the semantic information of the alerts to detect multi-step and distributed attacks. Limitation to a specific feature set is another shortcoming of this work.

Ariu *et al.* in 2011 [16], proposed an HMM-based IDS, called HMMPayl, which analyzes HTTP payloads at the byte level to detect malicious events. As figure 2.11 illustrates, HMMPayl analyzes HTTP payloads in three steps:

1. The proposed feature extraction algorithm allows the HMM to produce an effective statistical model being sensitive to the details of attacks.
2. In order to have robust detection system, a comprehensive training data set is prepared.
3. Multiple classifiers are employed to improve both efficiency and the difficulty of evading the IDS.

The authors have tested HMMPayl on several data sets of legitimate traffic and attacks

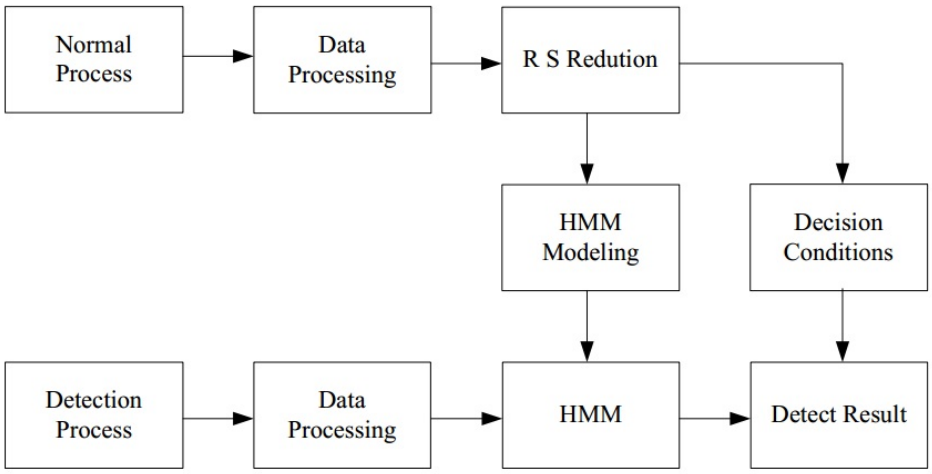

Figure 2.10 Zeng's Detection Model Based on HMM and Rough Set Reduction [195]

including generic attacks, Shell-code attacks, CLET attacks and XSS-SQL attacks. Based on their experiments, they believe that compared to other proposed works, HMMPayl provides more efficient and accurate detection results. However, this work is limited to only the HTTP traffic, and it does not cover other traffic types.

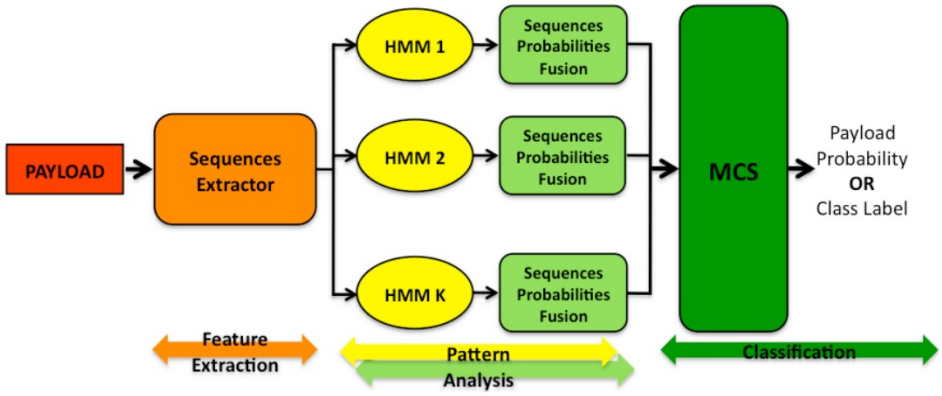

Figure 2.11 HMMPayl architecture [16]

2.2 Alert and Event Correlation

Generally, the concept of correlation is considered as two distinct process (Figure 2.12): Event Correlation and Alert Correlation. First, we differentiate alert and event correlation processes. Alert correlation is the process of gathering alerts from a number of IDSs (misuse-based or anomaly-based) and generating a high-level description of the happening

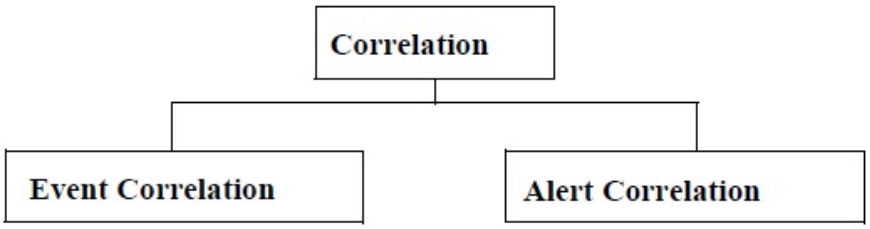

Figure 2.12 The subcategories of the correlation process

malicious behaviours. On the other hand, event correlation relates not only to IDSs, but also other event generators such as firewalls, routers, operating systems, antiviruses, web servers, etc. we believe that to detect recently emerged super complex and multi-step attacks, a well-defined event correlation approach can be more efficient than alert correlation approaches. Essentially, alert and event correlations are accomplished in multi-step processes, and they have a number of potential advantages, such as decreasing false positive rate and increasing detection accuracy. During recent years, researchers have proposed several alert and event correlation approaches [43, 48, 61, 123, 145, 177, 192]. In the following we describe some of these works.

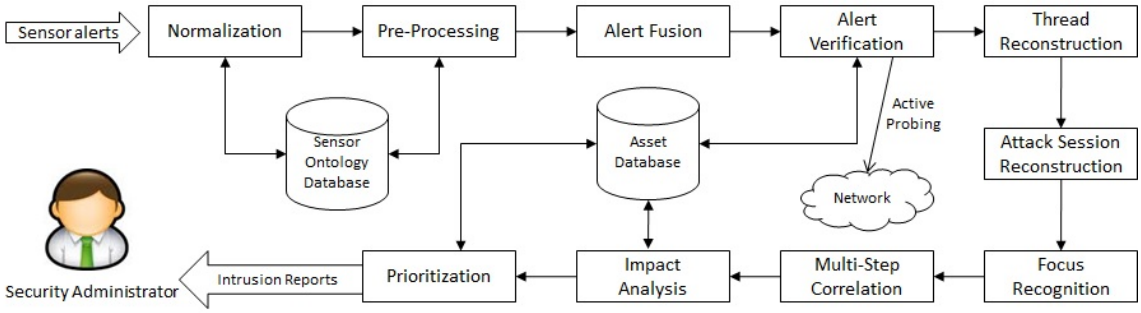

Figure 2.13 Valeur et al. alert correlation framework [177]

Valeur *et al.* in [177], proposed a correlation workflow having 10 steps including normalization, pre-processing, alert fusion, alert verification, thread reconstruction, attack session reconstruction, focus recognition, multi-step correlation, impact analysis and prioritization. As Figure 2.13 illustrates, this is perhaps the most comprehensive approach because other works concentrate on only a particular aspect or limited aspects of the correlation process, such as alert fusion or attack thread reconstruction. Based on their experiments, the authors believe that the effectiveness of each component is dependent on the data sets being analyzed; attack properties, network topology, available meta-data, etc. can influence the efficiency of the correlation process. They applied their model to seven different data sets, and inspected

the results. They believe that the proposed approach significantly reduces the number of false alerts. However, the main limitation of this work is that the authors only rely on alerts produced via intrusion detection sensors, but, not the other logging resources such as operating systems, databases, anti viruses, firewalls, etc. We believe that using such resources has direct impact on the efficiency of a correlation process. Moreover this work only employs a limited part of context information (i.e. target configuration). Whereas, using other types of context information can significantly improve the final performance.

Yusof *et al.* in 2008 [192], analyzed alert correlation techniques and listed their advantages and disadvantages in terms of being prone to alert flooding, contextual problem, false alerts, and scalability. In this paper, from the domain perspective, heterogeneous log resources have been categorized into network, wireless, host, application and sensor logs. Figure 2.14 illustrates this classification. Additionally, the authors in this paper categorize alert correlation techniques into four main categories (Figure 2.14):

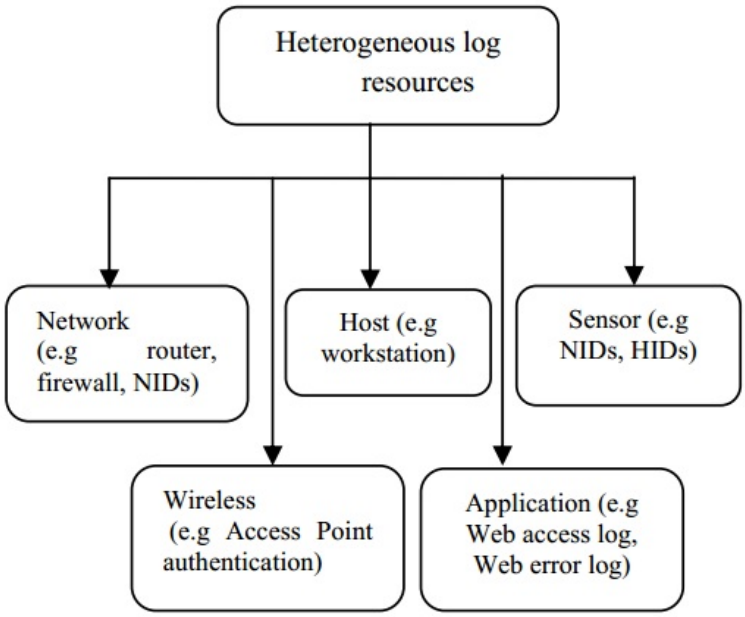

Figure 2.14 Yusof's Domain Perspective of Heterogeneous Log Resources [192]

— **Similarity-based techniques**: these techniques compare an alert with all the existing alert threads having similar attributes. Where a match is found, the alert is correlated with the existing thread. Otherwise, a new alert thread is created.
— **Pre-defined attack scenarios techniques**: these techniques believe that every attack requires a sequence of steps to happen in order to succeed. In this technique, every low-level alert is compared with the steps of a number of pre-defined attack

scenarios in order to correlate with the best fit. The main shortcoming of this technique is its restriction to known attack scenarios. Consequently, this technique is not necessarily able to detect zero-day attacks.

— **Pre-requisite and consequences of individual attack techniques**: these techniques are based on the fact that every attack has some pre-requisites that are absolutely necessary for the attack to proceed. Moreover, in these techniques, attack consequences are those events that take place after the attack succeeds. Unlike the previous techniques, these techniques are not restricted to known attack scenarios, and it is an appropriate technique to detect zero-day and multi-step attacks.

— **Statistical causal analysis techniques**: these techniques are based on the anomaly-based intrusion detection approaches. They employ statistical causality analysis to correlate alerts that are related to some specific attacks in order to reconstruct attack scenarios. As these techniques are based on pure statistical analysis, having pre-defined knowledge about attack scenarios is not required.

Morin *et al.* in 2002 [123], proposed M2D2 which is a data model for IDS alert correlation. M2D2 performs its analysis based on four information resources: the characteristics of the monitored information system, the vulnerabilities, the security tools used for the monitoring, and observed events. It integrates these concepts into a unified framework for more analysis. In this work, they formally define the M2D2 date model that ensures that processing of security information and more specifically alert correlation is anchored on a comprehensive model representing the information being processed. However, the main drawback of this work is the lack of evaluation based on reference data sets. Moreover, the described examples targets limited aspects of a comprehensive alert correlation process.

2.3 Alert Fusion

Employing distributed IDS in order to exploit their unique detection capabilities enhances organizations overall detection efficiency. However, efficient alert correlation and fusion (aggregation) approaches are required to manage the large number of alerts produced by distributed IDS. Alert fusion is a special case (sometimes a sub-process) of alert correlation that collects and analyzes alerts independently generated from the same potentially malicious event by different IDS, in order to make an appropriate final decision about the event [77].

Alert fusion improves detection accuracy, fault-tolerance, stability, and reliability of IDS and helps make appropriate decisions. During recent years, sensor fusion has been widely used in many types of applications such as defence, geological industry, speech recognition, pattern recognition, as well as intrusion detection systems. In intrusion detection systems, to cover

the limitations of using only one detection method, alert fusion-based methods have been widely used to obtain more accurate and reliable decisions [72, 73, 77, 154, 168, 191, 197]. Essentially, a fusion system includes two major phases [197]: selection of base detection sensors, and specifying the fusion mechanism. Most of the alert fusion approaches that have been proposed to date are mainly categorized into two categories:

— **Winner-take-all approaches**. The final decision over the received alerts from various IDS is made based on the decision of the IDS that has the highest measurement value. Some of the alert fusion approaches of this category include majority vote, weighted majority vote, behavior knowledge space, naive-Bayes combination, and Dempster-Shafer combination

— **Weight-based approaches.** Different weights are assigned to each IDS as its importance indicator on the final decision. The final decision is made based on the weighted sum of the measurement values of all the IDS. Some of the alert fusion approaches of this category include neural networks and weighted average.

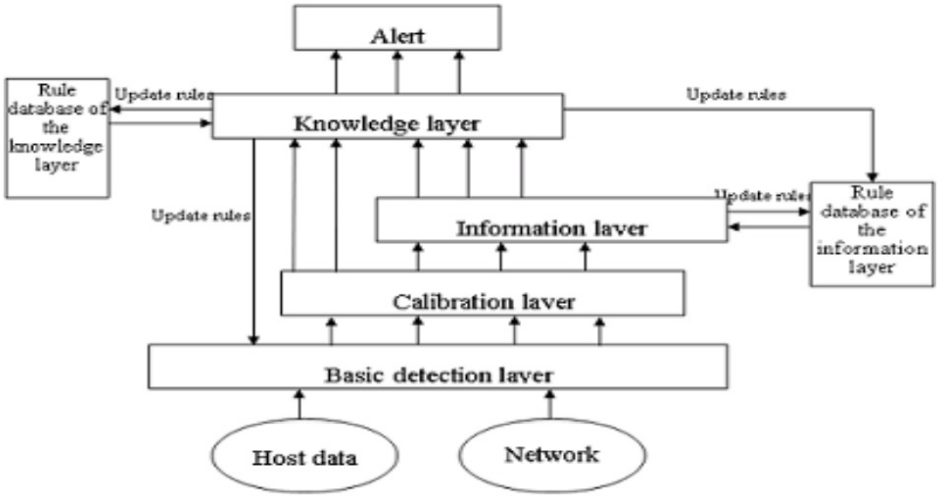

Figure 2.15 The Hierarchical Architecture of Zhao's Proposed System [197]

Zhao *et al.* in 2009 [197], proposed a robust and scalable intrusion detection system based on Data Fusion Theory. In the approach proposed, multiple detection measures were fused in a system such that its final false positive and false negative rate were dramatically decreased. The whole detection system has three layers: a basic detection layer, an information layer, and a knowledge layer. Figure 2.15 depicts the hierarchical architecture of the system. In the basic detection layer, various IDS agents having different detection approaches (misuse-based, anomaly-based and application based) have been arranged. In this layer, all the detection data are transferred into the data fuse module in the upper layer. Therefore, each detection

agent has a partial impact on the final decision. The calibration layer integrates the generated outputs from the basic detection agents. As, in the basic layer, a variety of detection methods having different output formats are employed, the calibration layer converts them to a unified format analyzable by the upper layers. Next, in the information layer, a reasonable and effective decision about the received alerts from various agents is made. Finally, the knowledge layer inspects the information layer's output and refines the output as appropriate as possible in order to decrease false alarms. In this layer, based on the revived decisions, a more specific and comprehensive perception about system's security state is acquired. Neural nets and Dempster–Shafer (DS) theory are two popular fusion methods. However, both have scalability and efficiency problem. The authors based their experiments on the KDDCup99 data set. They believe that the final detection rate of their proposed approach is higher than or close to the best detection rate of the basic detectors. It means that this approach has lower false negative and false positive rate as well as better scalability. However, there are some drawbacks in their proposed approach. First of all, relying on only fusion techniques is not enough to obtain acceptable detection results; we believe that these techniques should be combined with correlation techniques to result efficient detection result. Furthermore, KDDCup99 data set does not include most of the recent complex attack scenarios.

Thomas *et al.* in 2009 [168], proposed a Data-dependent Decision (DD) fusion method for multi-sensor intrusion detection systems. This work is an improvement of their previous DD fusion approach [169]. In this paper, they improve the efficiency of IDS by fusing alerts from multiple sensors. The Chebyshev inequality is employed to specify the fusion threshold. Each IDS should be parameterized with a threshold which has a direct impact on its performance. If the threshold becomes very large, some potentially attacks will not be detected. On the other hand, small threshold values result more false alarms. These thresholds should be defined optimally to obtain an appropriate system-wide performance. Therefore, for each sensor considering its decision importance, a particular threshold is considered. Figure 2.16 illustrates the architecture of a DD fusion. This architecture has three main phases: first, the sensors generate their alerts, then, the Neural Network Learner specifies a weight for each alert indicating on its importance, finally, the fusion unit performs fusion aggregation. The neural network in this data-dependent architecture is a supervised learning system that performs preprocessing tasks for the fusion unit. When an alert is correctly classified via detection sensors, the neural network will gain some knowledge, and consequently, it will be stabilized gradually. The authors have evaluated their method both theoretically and experimentally. For the experimental evaluation, they have used the DARPA 99 data set. Based on the authors claim, their experimental results validate the correctness of their theoretical analysis. However, as each detection sensor detects some particular attacks with

an acceptable correctness, one interesting suggestion is defining the thresholds dynamically based on the contextual information.

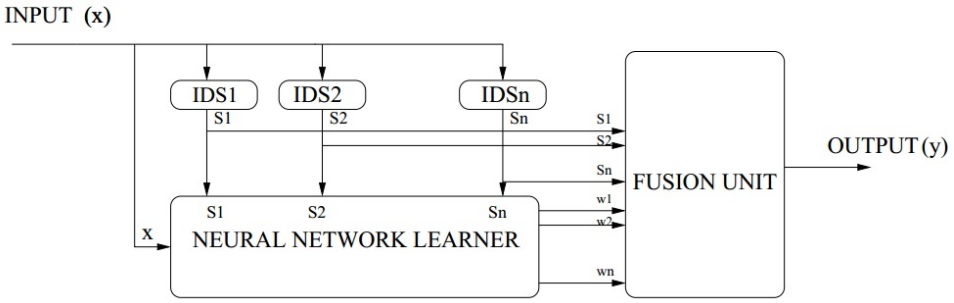

Figure 2.16 Data-dependent decision fusion architecture [168]

2.4 Context-Aware Intrusion Detection and Alert Correlation Systems

The term "Context" based on Dey's definition is *"Any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves"* [10]. One of the main characteristics of context-aware systems is their ability to adapt to underlying context without any explicit human interference. The Pervasive (ubiquitous) computing is one of the areas where contextual information has a principle role because the entities (users, objects, etc) have higher dynamicity [22]. In computer security and more specifically in malware detection, context can play a significant role in improving system efficiency. Since each attack requires particular context (network topology, protocols, system configuration, application configuration, etc) to proceed, using contextual information has undoubtedly a considerable role. More specifically, using contextual information can considerably reduce the false positives and non relevant alert rates. According to [70], contextual information can be categorized into four main classes:

— **Network-related context**: network related context includes information such as network topology, communication protocol, traffic rate, etc.

— **Target configuration**: target configuration typically includes information related to the operating system and applications running on the target system. Using these information and some vulnerability databases, it is possible to figure out whether an attack succeed or not.

— **Vulnerability assessment**: vulnerability assessment usually relies on tools that scan target system vulnerabilities to inspect whether the target is safe against security flaws used by an attack.

— **Attack side effects**: attack side effects are determined by inspecting target system behaviors to figure out whether an attack was successful. One solution is using IDS like Snort or Bro to analyze target reactions to some of the predefined attacks.

Contextual information has proved useful in better identifying specific alerts or in improving IDS efficiency. Gagnon, Massicotte and Esfandiari [70] have studied the use of target configuration as context information in order to identify non-critical alerts. In this paper they use target configuration (i.e., operating system and applications) as the context information. According to this paper, those alerts that are not related to a successful attack are called non-critical alerts. These alerts pose two types of problems: system administrators should spend a long time to investigate them, and they disrupt the normal activities of the underlying network. In order to have a fully automated evaluation process, the proposed approach relies on three information sources: 1) a well-documented attack data set, 2) alarms generated by Snort, 3) a vulnerability database. Figure 2.17 illustrates how this information is used during the evaluation process. First, complete configuration of the target machine is obtained based on the Snort alarms and data set information. Then, a list of products which are vulnerable to the underlying attacks are determined based on the Snort's references to Security Focus [3]. Next, criticality or non criticality of the alerts are investigated based on the evaluation algorithm. Finally, the outcome of the algorithm is verified using the data set documentation. According to their experiments, the authors believe that target configuration has valuable impact on the detection accuracy of IDS. Additionally, they believe that existing operating system discovery tools are not efficient enough to extract accurate context information.

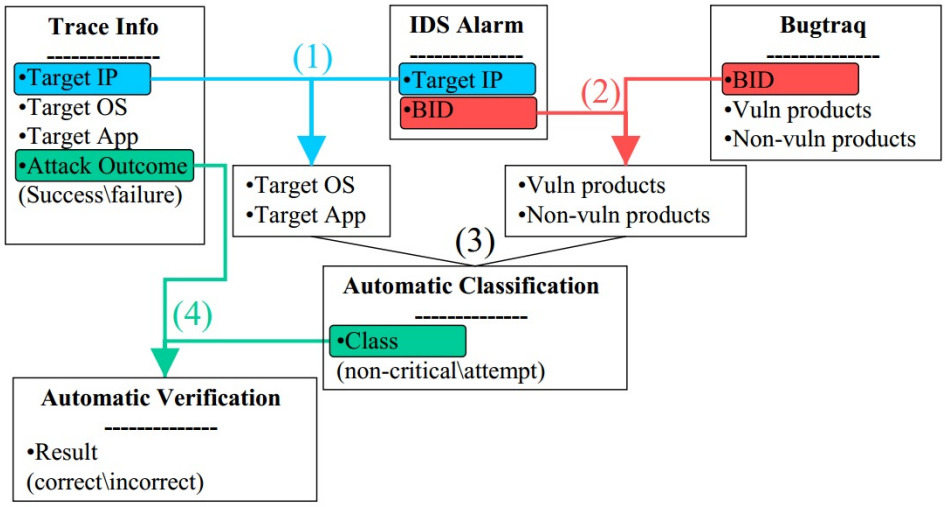

Figure 2.17 Gagnon's automatic evaluation process [70]

The Workload-aware Intrusion Detection (WIND) proposal by Sinha, Jahanian, and Patel [153] combines network workload information with Snort rules to improve its efficiency. To this end, they have added two distinct components over Snort: 1) a profiler that inspects the network traffic and the input rules to outcome a packet inspection strategy, 2) an evaluation engine that pre-processes rules based on the profiler, and evaluates incoming packets to determine the set of applicable signatures. According to their experiments, they believe that a workload-aware IDS outperforms Snort IDS by 1.6 times for all rules, and 2.7 times for web-based rules by consuming $10 - 20\%$ less memory.

Massicotte *et al.* in 2005 [117], analyzed the effectiveness of context information in the accuracy of Snort detection. To this end, they combined Snort signatures with context information collected using Nessus scripts [55], and the Bugtrac vulnerabilities database [3]. In this work they use a modified version of the M2D2 network model [123] which provides a network model for information correlation with network context information. In this paper, the authors employ Snort as an intrusion detection system, Nessus as a vulnerability detection system, and Bugtraq and CVE as the popular vulnerabilities databases in their experiments. These information sources can be used mainly for two purposes: to populate static information in the network model, and for specifying enhanced detection rules combined with the network context to reduce false positives. Since Nessus scripts are not in the same format as Snort rules or Bugtrag vulnerability entries, detection rules and the contextual information is combined manually. As future work, they intend to design some rule sets to collect hosts, OS, protocol discovery, service discovery, switch and router discovery, and host IP configuration information.

Unfortunately, these studies only consider partial contextual information, such as target configuration or network traffic, and do not allow for inclusion of other types of context concepts, such as users profiles, organizations profiles, attack side effects, full network information, etc. Lack of automation is another significant drawback of this work.

2.5 Distributed Intrusion Detection Systems

A Distributed Intrusion Detection System (DIDS) consists of a number of detection sensors or agents distributed over a large network. In these systems, all the agents communicate together or with central servers that control accuracy of the intrusion detection process. Every detection agent in a DIDS might employ a specific detection method to analyze ongoing traffic that makes it capable to detect a particular attack type. Consequently, the whole performance and efficiency of the system will be significantly improved. Some of the major advantages of DIDS are:

— Possibility of early detection of complex and coordinated attacks
— Improving the quality of network monitoring and incident analysis
— Having a better control on worms spread

During recent years many DIDS have been proposed by researchers. AAFID (Autonomous Agents for Intrusion Detection) [160], DIDS (Distributed Intrusion Detection System) [157], EMERALD (Event Monitoring Enabling Responses to Anomalous Live Disturbances) [138], and CSM (Cooperating Security Management) [185] are some of the proposed systems. However, designing a fully distributed DIDS without having shortcomings like single point of failure or generating many false alerts is still one of the main challenges of DIDS research. In the following we describe some of the recent works.

Zhang *et al.* in 2011 [196], proposed a hierarchical and distributed intrusion detection system for smart grids called the Smart Grid Distributed Intrusion Detection System (SGDIDS). SGDIDS using its Analysis Module (AM) that employs some classification techniques such as Support Vector Machines (SVM) and Artificial Immune System (AIS) to inspect network traffic to efficiently classify malicious events. A three-layer network architecture, shown in Figure 2.18, including Home Area Network (HAN), Neighborhood Area Network (NAN), and Wide Area Network (WAN) is considered for SGDIDS. In order to evaluate the proposed work, they have applied the SVM and AIS techniques located in every layer of the network architecture to the KDD Cup 1999 data set. The authors believe that the achieved results show that the proposed approach can considerably improve detection effectiveness. However, the proposed work does not consider the underlying contextual information in the intrusion detection process. Poor evaluation based on an old data set that does not cover recent attacks is another main drawback of this work.

Lo *et al.* in 2010 [113], proposed a cooperative intrusion detection system framework to mitigate the impact of DoS and DDoS attack in cloud environments. In the proposed framework, for every region in the cloud environment, an IDS is considered. These IDS cooperate together via exchanging alerts to prevent or mitigate the impact of DDoS attacks. Figure 2.19 illustrates the proposed cooperative IDS framework. The system has four main components including: intrusion detection, alert clustering and threshold computation and comparison, intrusion response and blocking, and cooperative operation. Additionally, every IDS has three modules:

— Block: drops bad packets received from source node
— Communication: sends warning messages about the detected attack by the IDS
— Cooperation: gathers alerts and analyzes the accuracy by majority voting

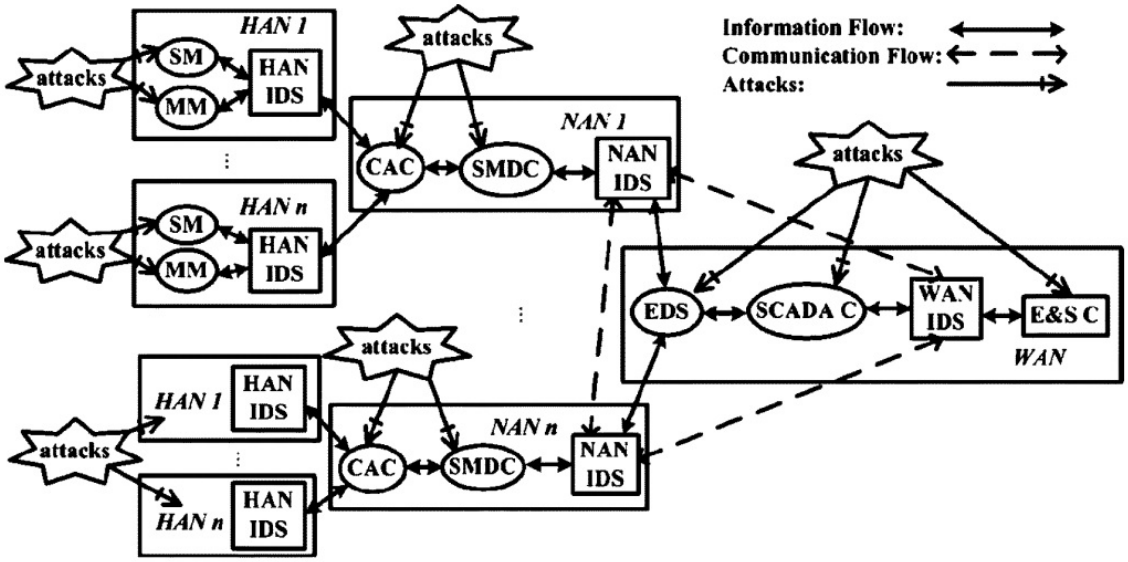

Figure 2.18 Zhang's proposed three-layer network architecture for SGDIDS [196]

The authors believe that one of the main advantages of their proposed framework is addressing the single point of failure problem. Furthermore, the proposed framework provides the intrusion prevention capability. However, this work is limited to only DoS and DDoS attacks. Moreover, this work has a potential to produce a large amount of false positives because it does not use contextual information to find out what kind of tools and vulnerabilities exist in the underlying context.

Abraham *et al.* in 2007 [11], proposed a Distributed Soft Computing-based IDS (D-SCIDS) that combines different machine learning classifiers to produce more efficient intrusion detection system. As mentioned in this paper, one of the most popular models for a distributed IDS is the master-slave model that is very efficient for small networks. This hierarchical model is also one of the prevalent models especially for wide networks. In this model, analysis and control tasks are performed at different layers because of their geographical distance. In such models, as Figure 2.20 illustrates, the alerts generated by different detection sensors are passed to analyzer/controller nodes. Analyzer/controller nodes may exist in different locations within the network. Agents residing in the individual analyzer/controllers include components responsible for the agent regeneration, dispatch, updating, and maintaining intrusion signatures, etc. In such models, Central Analyzer and Controller (CAC) is the key component of the whole model. It usually includes some databases and web servers which provide possibility of communication with system administrators and other components. Some other tasks of CAC are: attack aggregation, building statistics,

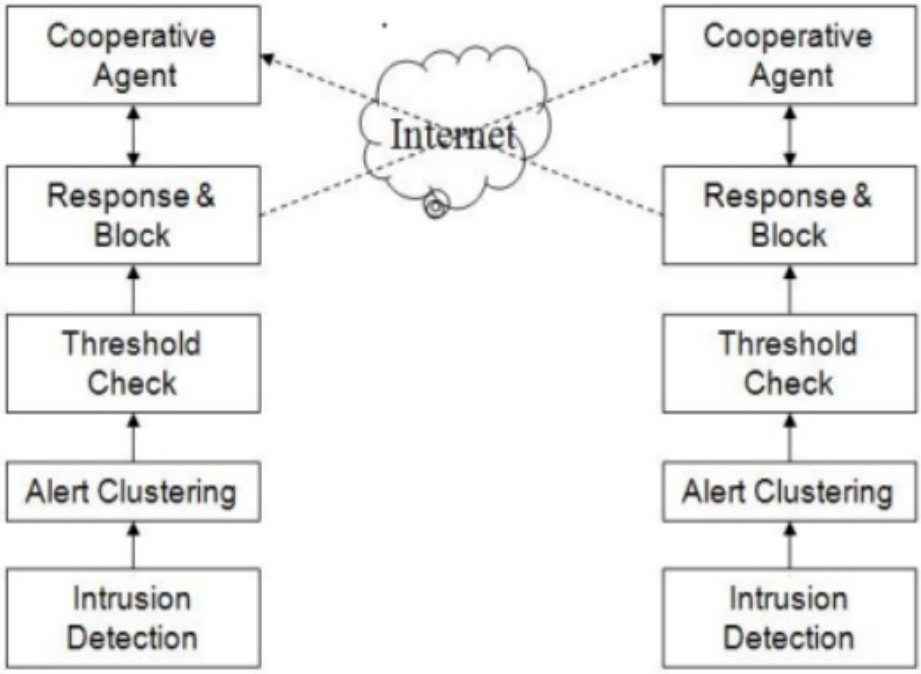

Figure 2.19 Cooperative IDS proposed by Lo [113]

identifying attack patterns, and performing rudimentary incident analysis. The authors in their experiments mention the importance of feature selection in order to model lightweight intrusion detection systems. In the base classifiers, they use soft computing approaches such as fuzzy logic and genetic algorithms. They also believe that their approach is useful for lightweight networks such as MANETs and Sensor Wireless Networks (SWN). However, the effectiveness of this work mainly depends on the selected feature set. If in the first level, an inappropriate feature set is employed, the final performance will be significantly reduced. Lack of automation is another shortcoming of this work that needs direct interaction with system administrators.

2.6 Host-Based Intrusion Detection Systems

In Host-based Intrusion Detection Systems (HIDS), events happening within an individual computer system are analyzed to determine malicious behaviors. Unlike NIDS, HIDS are able to watch the outcome of an attempted attack because it can directly monitor the data files and system processes which are target of the attack. HIDS usually use information of two sources: 1) operating system audit trails (system calls), and 2) system logs. As OS audit trails are usually generated in the kernel level, they are more detailed and reliable than

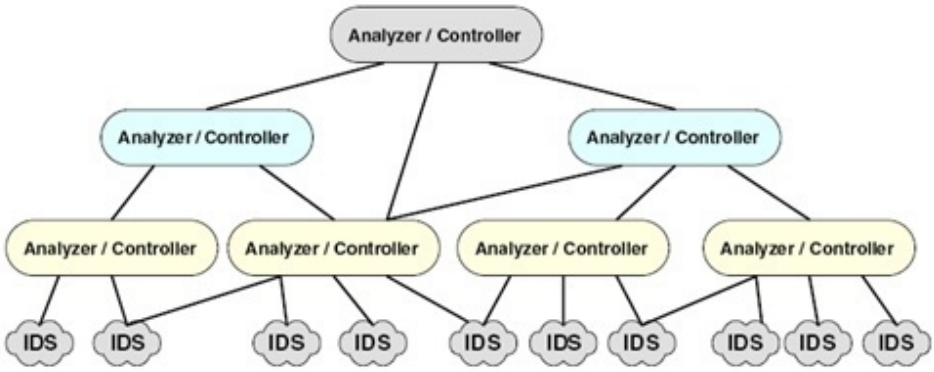

Figure 2.20 Abraham's hierarchical architecture with free communication between layers [11]

system logs. Figure 2.21 illustrates the general architecture of an anomaly-based HIDS [149]. As the figure shows, first the normal system call sequences are prepared. Then, based on the normal sequences, any deviation is considered as a malicious behaviour. In this process, the accuracy and comprehensiveness of the first step has direct impact on the efficiency of the whole process. In the following we describe some of the recently presented host-based intrusion detection approaches.

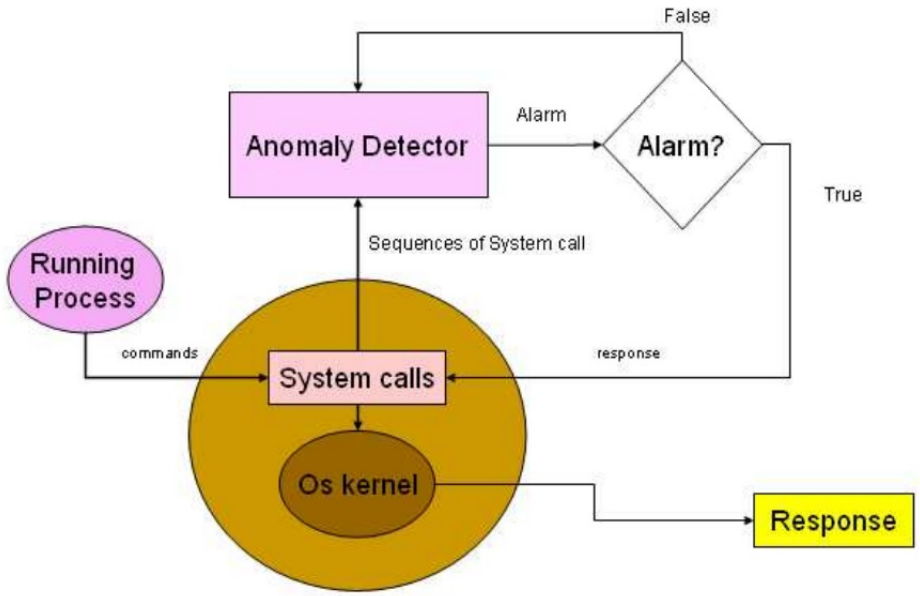

Figure 2.21 General architecture of an anomaly-based HIDS [149]

Sekeh *et al.* in 2009 [149], proposed an anomaly-based HIDS that uses the system call traces in order to detect malicious behavior. In the proposed approach, data mining and fuzzy logic techniques are employed to extract normal behavior, and promote the intelligence of

the system. Usually, in a HIDS operating system events are monitored for further analysis. During the normal operation of the underlying system, the proposed approach collects system call sequences to build a normal behavior data set. Based on this data set, normal profiles are prepared so that the system can efficiently analyze new system call sequences, and detect abnormal events. Figure 2.22 illustrates the proposed anomaly-based HIDS that has the following steps:

1. Normal system call sequences are collected as a training data set (UNM [1] data set [5] has been used for this purpose).
2. Using data mining approaches, detection rules are extracted from the data set.
3. The extracted rules are inserted into a Finite State Machine (FSM).
4. Using the *strace* command, system call sequences are collected as a testing data set. This data set is the input of the anomaly detector component.
5. New system call sequences are compared with FSM normal rules, and using the fuzzy logic inference engine, abnormal behavior is detected.

The authors believe that their proposed model has some advantages, such as real-time intrusion detection, low false positive rate, higher visibility, higher information assurance, and independence from the operating system. However, the total efficiency of this work depends on the accuracy rate of its first phase. Otherwise, it may generate a huge amount of false positives. Lack of automation is another shortcoming of this work.

Chung-Ming Ou [132] proposed in 2012 an HIDS based on Agent-Based Artificial Immune Systems (ABAIS), called ABIDS. In this work, a Multi-Agent System (MAS) consisting of antigen agents, DC agents, T-Cell agents and responding agents that have been designed. Instead of packet analysis that is very time consuming, ABAIS directly uses system call sequences to detect malicious events. Figure 2.23 illustrates the ABIDS architecture. As the figure shows, ABIDS's agents are:

— Antigen agents (Ag agent): they represent data items from some data sets.
— Dendritic cell agents (DC agent): they are the kernel of the ABAIS that are installed and distributed at computer hosts.
— T-cell agents (TC agent): they are installed in each computer system, and they are activated by the signals received from DC agents when the Danger Values (DV) exceed some thresholds.
— Responding agents (RP agent): they generate appropriate responses to the detected malicious events.

1. University of New Mexico

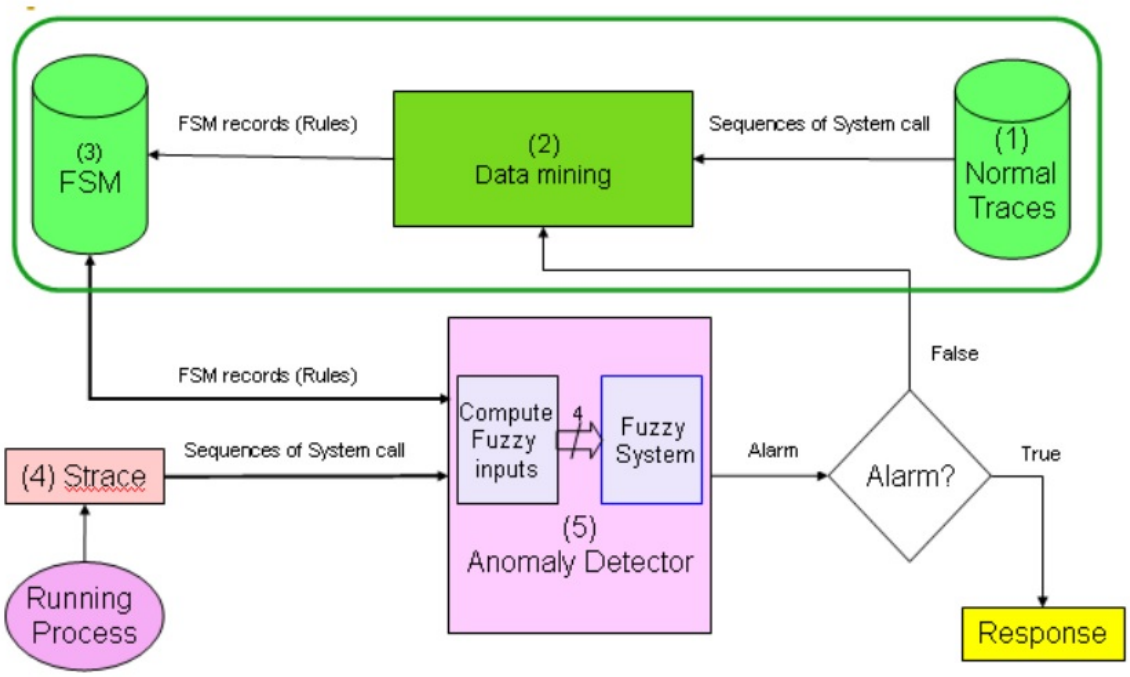

Figure 2.22 Sekeh's proposed model for intrusion detection [149]

In this work, the author believes that evaluation of three factors from systems calls including severity, certainty, and attacking time can significantly improve detection efficiency. However, if the thresholds are not defined accurately in this work, there is a potential to generate a significantly high false positive rate.

2.7 Intrusion Detection Evaluation Metrics

Every network-based IDS classifies each network event as normal or malicious. In order to select an IDS for a specific application, it must be compared with other existing candidates to choose the most appropriate one. For this purpose, we use evaluation metrics to compare IDS performance and effectiveness. During recent years, a number of IDS evaluation metrics, such as Bayesian detection rate, the expected cost, intrusion detection capability, etc. have been proposed. In the following we describe some of the mostly used metrics summarized in [84].

— **Receiver Operating Characteristic (ROC) Curves**. ROC curves plot the true positive rate versus the false positive rate [30]. They can be used to compare detection results of different IDSs. Figure 2.24 illustrates an example of how a ROC curve can be used to compare results of four different example analyses. One important limitation

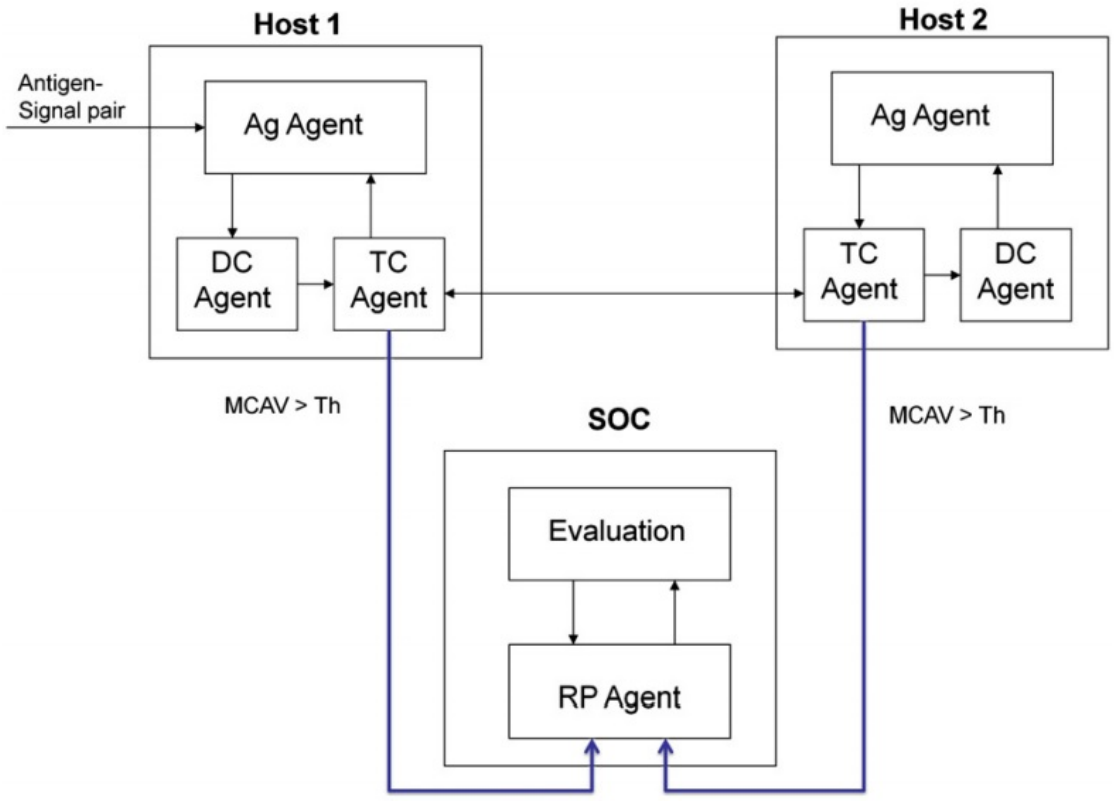

Figure 2.23 ABIDS architecture [132]

of this metric is that it calculates the overall performance of an IDS at all baselines. However, an IDS should be developed based on a best baseline.

— **Cost-Based Metrics**. In cost-based metrics based on the importance of the false positive rate and false negative rate, a cost measure is assigned to each one. It means that this metric considers the trade-off between the false positive rate and false negative rate. This cost measure can be individually adjusted to differentiate between the damage caused by a successful intrusion or the costs corresponding to a false alarm.

— **Information-Theoretical Metrics**. Intrusion Detection Capability (C_{ID}) is a metric based on the following observation [76]: an IDS at the abstract level classifies the network traffic X as normal or malicious. Therefore, from information theoretic point of view, less uncertainty should exist about X considering an IDS output Y. C_{ID} is the proportion of the mutual information between the IDS input and output, $I(X, Y)$, to the entropy of X, *H(X)*:

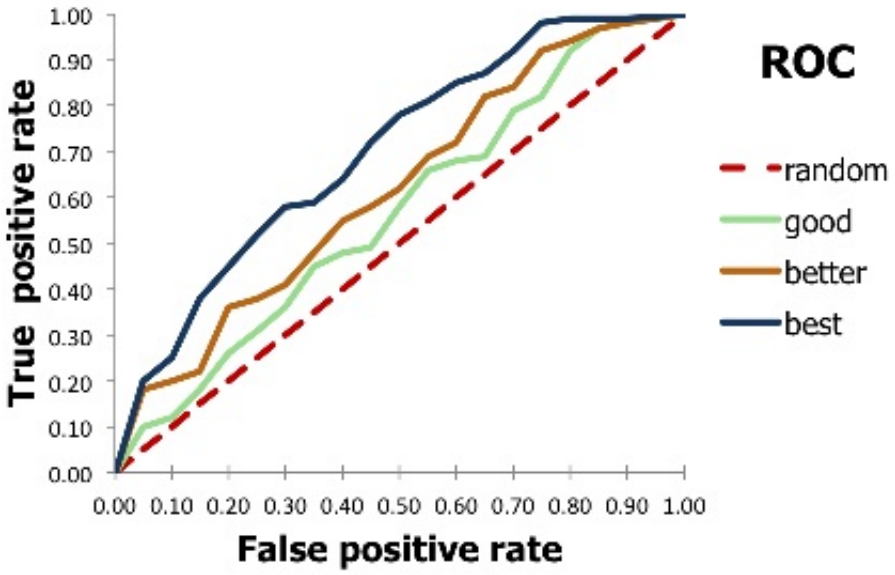

Figure 2.24 Comparing the efficiency of four different example analyses using ROC curve

$$C_{ID} = \frac{I(X,Y)}{H(X)} \tag{2.1}$$

— **Binary classification**: Each IDS classifies the network traffic into two main classes: normal or malicious. The most common metrics to compare IDS are False Positive (FP) rate and False Negative (FN) rate. FP rate is the probability of producing an alert even though the behavior of the system is normal (in such cases IDS detection is incorrect). In the other hand, FN rate is the probability of producing no alert even though the behavior of system is malicious. Equations 2.2 and 2.3 represent FP and FN rates:

$$FP = \frac{number\ of\ false\ positives}{number\ of\ negatives} \tag{2.2}$$

$$FN = \frac{number\ of\ false\ negatives}{number\ of\ positives} \tag{2.3}$$

Consequently, True Positive (TP) and True Negative (TN) rates can be defined: $TP = 1 - FN$ and $TN = 1 - FP$.

Essentially, there is a trade-off between false positives rate and false negative rate.

If the IDS detection policies become very sensitive, the risk of FP rate will be higher. Otherwise, the risk of FN rate will be higher. Therefore, a balance should be considered between these two risks in IDS configuration. Equation 2.4 represents IDS *sensitivity* that is defined as the proportion of normal behaviour:

$$Sensitivity = \frac{number\ of\ true\ positives}{number\ of\ true\ positives + number\ of\ false\ negatives} \tag{2.4}$$

However, sensitivity is not meaningful enough because it can be trivially achieved by classifying all behaviours as malicious. There is another metric called $specifity$ that solves this problem. Equation 2.5 represents this metric:

$$Specificity = \frac{number\ of\ true\ negatives}{number\ of\ true\ negatives + number\ of\ false\ positives} \tag{2.5}$$

However, by classifying all the traffic as normal, $specifity$ is completely achieved. $F - measure$ is a metric that combines $specifity$ and *sensitivity*:

$$F - measure = \frac{2 \times \text{sensitivity} \times \text{specificity}}{\text{sensitivity} + \text{specificity}} \tag{2.6}$$

Positive Predictive Value (PPV or Bayesian Detection Rate) is another metric that indicates the probability of an intrusion when an IDS reports an alert:

$$PPV = \frac{number\ of\ true\ positives}{number\ of\ true\ positivies + number\ of\ false\ positives} \tag{2.7}$$

Negative Predictive Value (NPV) is the probability of a normal event when an IDS does not report an alert:

$$NPV = \frac{number\ of\ true\ negatives}{number\ of\ true\ negatives + number\ of\ false\ negatives} \tag{2.8}$$

Unlike specificity, PPV outcomes the proportion of normal behaviours to the whole observation set. Essentially, the main difference between them is that specificity and sensitivity are independent from the total observation set. However, PPV and NPV are dependent on the total observation set. PPV is called Bayesian detection rate [19] and can be expressed via Bayes theorem:

$$PPV = P(actual\ intrusion|IDS\ alert) \tag{2.9}$$

In order to measure IDS efficiency based on these metrics, a common method is to employ some standardized data sets and apply the IDS to them. An example of such data sets is the DARPA 1998 [112]. We describe some of the most popular data sets in the next section.

2.8 Data Sets to Evaluate Intrusion Detection and Alert Correlation System

During the last two decades, one of the major challenges of many researchers was the evaluation of intrusion detection and alert correlation systems. Since there was no standard and comprehensive test framework consisting most of the attack and exploitation strategies, a common method to evaluate the efficiency of these systems is employing popular test data sets. In the following we describe some of these data sets:

— **DARPA 98** [112]. This was the first IDS evaluation data set sponsored by DARPA in 1998. This evaluation had two parts, off-line and real-time, that was conducted by MIT Lincoln Lab and Air Force Rome Lab. A simulated network was used to generate the network traffic. This data set includes real network traffic along with some artificial background and attack traffic. The major services to generate the traffic includes HTTP, FTP, SMTP, POP, DNS, X, IRC, SWL/Telnet, SNMP, Time and Finger. The data set includes around 38 attack types falling into four main categories:
  — Denial of service (DOS)
  — Remote to user (R2L)
  — User to root (U2R)
  — Surveillance / Probing

— **DARPA 99** [111]: After the initial success of their first data set, the DARPA 99 data set was released to cover more novel attacks than the previous one. Consequently, Windows NT systems were added to the simulation network, and 17 new attacks targeting these systems were injected to the network traffic.

— **KDD CUP 99** [81]: This data set is one of the most widely used data sets in order to evaluate network-based anomaly detection systems. It has been created based on DARPA 98. The KDD training data set includes 4,900,000 labeled (normal or attack) connections each of which contains contains 41 features, and the test data set contains about 300,000 connections.

— **DARPA 2000** [120]: The DARPA 2000 data set contains two attack scenarios,

LLDDOS 1.0 and LLDOS 2.0.2. Both attack scenarios are multi-step corresponding to a Distributed Denial of Service (DDoS) flooding attack. They have 4 phases:

— Probing
— Exploiting vulnerabilities to break into vulnerable hosts.
— installing DDOS software on the compromised system
— Launching the DDoS attack

The difference between these two scenarios is that the attacks inside LLDDOS 2.0.2 are more complicated and difficult to detect.

— **DEFCON 9**: DEFCON is another data set that is commonly used in IDS evaluation. It has been captured during a hacker competition in the Capture The Flag (CTF) format. Essentially, this traffic is completely different from "real world" network traffics because it includes only intrusive events without any normal background traffic. Therefore, it can be very appropriate for stress-testing IDS because it provides a worst-case scenario of the amount of attack traffic an IDS will receive.
— **ISCX data set** [151]: This data set was released in 2012 by the Information Security Centre of Excellence in University of New Brunswick. The major services that generated the traffic are: HTTP, SMTP, SSH, IMAP, POP3 and FTP. Various multi-stage attacks scenarios have been carried out to supply the anomalous portion of this data set:
  — Infiltrating the network from the inside
  — HTTP denial of service
  — Distributed denial of service using an IRC Botnet
  — Brute force SSH

2.9 Conclusion

In this chapter we reviewed the intrusion detection and alert correlation approaches proposed in recent years. Our main focus was on the works that had significant impact on the IDS and alert correlation advances. We started the chapter with machine learning based intrusion detection and alert correlation approaches. In that section we explained how machine learning techniques, such as Support Vector Machines (SVM), Bayesian Networks (BN), Artificial Neural Networks (ANN), Genetic Algorithms (GA), Hidden Markov Models (HMM), etc., have been employed in order to design intrusion detection systems. Next, we explained some of the valued alert correlation approaches, and how these works have addressed some of the major problems of IDS. Alert fusion approaches that apply fusion algorithms to the collected alerts from different sensors, context-aware IDS that use contextual information in order to

reduce false positives, Ontology-based or semantic-based IDS that target system automation, Host-based IDS (HIDS) that analyze events happening in a single system, and Distributed IDS (DIDS) that analyze events occurred in different part of a network, were some other topics that we reviewed in this chapter. Finally, in the last two sections, we mentioned some of the evaluation metrics and data sets that usually are used to compare and evaluate the efficiency of intrusion detection and alert correlation systems.

Additionally, in this chapter we described the major shortcomings and limitations of prior works. About machine learning based approaches, we explained that most of them only classify network events based on some features depending on behaviors observed with one type of event, and they do not take into account contextual information and events interrelationships. These works only classify events that happened, and generally they are not able to detect multi-steps or distributed attacks. In summary: 1) these techniques are not adaptable and flexible enough to be applied to various contexts, 2) choosing a suitable subset of features that is significantly critical to improve the performance of classifier, is another difficulty of these techniques, 3) these techniques do not take into account the semantic information of the alerts. It is also worth pointing out that most of these works have been evaluated via very old data sets, such as DARPA 98 or KDD 99. Because these data sets do not include recent complex attacks, they are not adequate to test and evaluate IDS efficiency.

About the described alert correlation approaches that apply correlation algorithms to the collected events from one or a number of IDS, we explained that most of these approaches only concentrate on one or few aspects of a comprehensive alert correlation process (e.g. alert fusion or attack session reconstruction), and they do not verify the relevancy of collected alerts based on the underlying context. Moreover, they mainly consider correlation only across multiple sensors of the same type having a common event and alert semantics (homogenous correlation), leaving it to security administrators to perform correlation across heterogeneous types of sensors. As a result, the problem as a whole has not been resolved.

About the described context-aware intrusion and alert correlation approaches, unfortunately, they only consider partial contextual information, such as target configuration or network traffic, and do not allow for inclusion of other types of context concepts.

The reviewed distributed IDS mostly rely on homogenous intrusion detection that employs similar IDS sensors as their detection agents. Lack of automation is another main shortcoming of these works that prevent the analysis of alert interrelationships to be performed by machines rather than security administrators.

In summary, previous intrusion detection and alert correlation approaches do not fully take into account all possible pieces of information, such as contextual, vulnerability and attack

information, from various heterogeneous sensors. In addition, they lack the flexibility that analyst requires to be able to add new types of information (e.g., contextual information) into their analysis process. While some intrusion detection and alert correlation approaches have attempted to address this problem by introducing security ontologies, their lack of extensibility and flexibility constitute important drawbacks. Motivated by the aforementioned shortcomings, and in order to provide a common solution encompassing the advantages of all of these approaches, we have designed and proposed henceforth a flexible and extensible ontology-based event correlation framework to incorporate various information (event logs, contextual, vulnerabilities, attacks or any other type of relevant information) from heterogonous sensors, such as NIDS, HIDS, operating systems, anti-viruses, databases, etc., in the event correlation process in order to improve IDS performance (higher detection rate) and automation (less human interaction) while reducing the number of duplicate, non-relevant, and false positive event logs.

CHAPTER 3 BASIC KNOWLEDGE ON ONTOLOGIES

The term ontology based on Gruber's definition is: "An Ontology is a formal, explicit specification of a shared conceptualization" [75]. According to Sowa's definition, Ontology is "the study of categories of things that exist or may exist in some domain. The product of such study, called an ontology, is a catalogue of types of things that are assumed to exist in a domain of interest **D** from the perspective of a person who uses a language **L** for the purpose of talking about **D**" [159]. Ontologies provide description for the following elements:

— Classes or "Things" in various domains
— Relationships among "Things"
— Properties or attributes for "Things"

Ontologies are useful for representing and interrelating many knowledge types. Hence, they have been employed in various areas of computer science, such as machine learning, computer security, knowledge representation, semantic web, etc. However, the major question that everybody asks about the concept of Ontology is on the difference between an ontology and a database schema. Unlike data models, the main capability of ontologies is their relative independence of particular applications which means that an ontology provides a relatively generic knowledge structure usable by different applications [161]. Table 3.1 illustrates a comparison of the concepts of "ontology" and database schema. Open world assumption, consistency with heterogeneous data, flexible structure, and deductive reasoning are a number of the added capabilities of ontologies in comparison to database schemas.

In order to design and develop ontologies, a number of languages have been proposed in recent years. The most popular ontology representation languages are RDF, RDF Schema, and OWL. These languages have received considerable attention since the emergence of Semantic Web. To make ontologies understandable for both machine and human, the ontology language should have precisely defined semantics accessible by automated processing. For this purpose, Description Logic (DL) can be an ideal choice, because it has formal logic-based semantics and it is equipped with inference procedures with the goal of implementing automated reasoning systems. This view of the potential place of DL in the semantic web resulted in the emergence of a number of languages that brought DL concepts to the semantic web. The World Wide Web Consortium (W3C) recommended the Web Ontology Language (OWL) for the semantic web, which exploits many of the strengths of DL including well defined semantics and practical reasoning techniques. The basic elements in OWL-DL are: class, property, restriction, individual.

— Class: classes provide an abstraction mechanism for grouping resources with similar

Table 3.1 Comparison of the concept of "ontology" vs. database schema

	Ontology	**Database Schema**
Class hierarchy and instances	Yes	Just Object oriented databases
Constraints	Intended meaning	Data integrity
Execution	Executable	Non-executable
Assumption	Open-world	Close-world
Reasoning & inference	Yes	NO
Extendibility	High	Low
Consistency with heterogeneous data	High	Low

characteristics.

— Property: OWL has two types of properties: *i*) object properties which are relations between instances of two classes, *ii*) datatype properties which are relations between instances of classes and RDF literals or XML schema datatypes.

— Property restriction: property restriction describes an anonymous class, namely a class of all individuals that satisfy the restriction. OWL provides two types of property restrictions: value constraints and cardinality constraints.

— Individual: An individual corresponds to an instance of a class.

Although OWL provides a relatively rich set of class constructors, it provides much weaker constructors for roles. A solution to address this limitation would be to extend it with rules. Semantic Web Rule Language (SWRL) [87] is a rule language combining the OWL and the Rule Markup Language. Using SWRL we can design various rules to extract information from ontologies. Drilling-down into very detailed and specific classes, and rolling-up toward very generic classes of the ontologies are two common and most useful operators that SWRL provides to extract required information from various ontologies. In the next chapters of this thesis, we use SWRL in order to extract and correlate information of various security-related ontologies, and reconstruct attack scenarios.

The rest of this chapter is organized as follows. In section 3.1, we provide an introduction to OWL Web Ontology Language. In section 3.2, we briefly introduce Semantic Query-enhanced Web Rule Language (SQWRL), which is one of the popular ontology query languages. In section 3.2, we describe the role of ontology in computer security research. Section 3.3 reviews ontology- or semantic-based intrusion detection and alert correlation approaches.

3.1 Introduction to OWL Web Ontology Language

OWL is an unified standard language for encoding and exchanging ontologies. It particularly extends the Resource Description Framework (RDF) and RDF Schema. Therefore, OWL is processable by most of the available XML and RDF tools. From a semantic perspective, OWL is based on Description Logic (DL) [20] which is a decidable subfamily of first-order predicate logic.

OWL describes classes, properties, and relations in a machine interpretable way. OWL 1 has three increasingly expressive sublanguages:

— OWL Lite provides class and property hierarchy, and simple constraints with enough expressive power to model simple ontologies.

— OWL DL increases expressiveness while retaining decidability. It provides all OWL constructs, under certain limitations.

— OWL Full is the complete language without any limitation, but it ignores decidability issues.

OWL Full is considered as an extension of RDF, whereas OWL Lite and OWL DL are considered as extension of restricted version of RDF. OWL DL and OWL Lite have less power but reduce the computing requests on a processor.

3.1.1 Description of the OWL Language

OWL uses RDF XML-based syntax. The root element of an OWL ontology is *rdf:RDF* element which specifies a number of namespaces. Namespaces provide a means to unambiguously interpret identifiers and make the rest of the ontology presentation much more readable. For example, in the following, the first namespace says that elements prefixed with *owl:* should be understood as referring to things defined in the namespace called http://www.w3.org/2002/07/owl#.

```
<rdf:RDF
        xmlns:owl ="http://www.w3.org/2002/07/owl#"
        xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:xsd ="http://www.w3.org/2001/XMLSchema#">
```
(3.1)

In OWL, a class is defined using an *owl:Class* element. As an example:

```
<owl:Class rdf:ID="associateProfessor">
</owl:Class>
```
(3.2)

There are two predefined classes: $owl : Thing$ which is the most general class and contains everything, and $owl : Nothing$ which is the empty class.

OWL has two types of properties:

— **Object properties**: these properties relate objects to objects. For example $isTaughtBy$ and $supervises$:

```
<owl:ObjectProperty rdf:ID="isTaughtBy">
</owl:ObjectProperty>
```
(3.3)

— **Data type properties**: these properties relate objects to RDF literals or XML schema data types. For example, in the following we define a data type property *age* that takes only non-negative values:

```
<owl:DatatypeProperty rdf:ID="age">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema
    #nonNegativeInteger"/>
</owl:DatatypeProperty>
```
(3.4)

OWL provides class restriction capability. $rdfs : subClassOf$ a class to be subclass of another class; every instance of the first class is also an instance of the second class. $owl : allValuesFrom$ is used to specify the possible values that a property can take. $owl : hasValue$ states a specific value that a property must have. $owl : someValueFrom$ states that there exists some values that a property can take from a specific class.

Some special properties of property elements are:

— *owl:TransitiveProperty* defines a transitive property.

— *owl:SymmetricProperty* defines a symmetric property.

— *owl:FunctionalProperty* defines a property that has at most one value.

— *owl:InverseFunctionalProperty* defines a property for which two different elements objects cannot have a same value.

OWL provides boolean combinations of classes, such as union *owl:unionOf*, intersection

owl:intersectionOf, and complement *owl:complementOf*.

In OWL instances of classes are defined as in RDF:

```
<rdf:Description rdf:ID="949352">
     <rdf:type rdf:resource="#academicStaffMember"/>
</rdf:Description>
```
(3.5)

In logics with the unique name assumption, different names always refer to different entities in the world. Unlike database systems, OWL does not respect *unique-names-assumption*.

3.2 Semantic Query-enhanced Web Rule Language (SQWRL)

In order to extract information from OWL ontologies, a concise, readable, and semantically robust query language is required. SPARQL is one of the current languages that is employed for querying OWL-based ontologies. However, SPARQL only operates on the RDF serialization of OWL and has no native understanding of OWL.

The Semantic Query-enhanced Web Rule Language (SQWRL) [131] is one of the popular RDF query languages that provides the mentioned characteristics. It is built on the SWRL rule language [87]. SQWRL employs built-in facilities of SWRL as an extension point. Using these built-ins, and without adding any syntactic extensions, a set of operators has been defined in SQWRL to construct retrieval specifications. Therefore, all the SWRL editors can generate and edit SQWRL queries.

3.2.1 Basic Querying

sqwrl:select is the core operator of SQWRL. This operator takes one or more arguments, and constructs a table making the arguments as table column. Example:

$$\texttt{Person (?p)} \wedge \texttt{hasAge(?p, ?a)} \wedge \texttt{swrlb:lessThan(?a, 9)} \longrightarrow \texttt{sqwrl:select(?p, ?a)} \tag{3.6}$$

This query returns pairs of individuals with age less than 9.

The results of every query can be ordered using by *orderBy* and *orderByDecending* built-ins. Example:

$$\texttt{Person (?p)} \wedge \texttt{hasAge(?p, ?a)} \longrightarrow \texttt{sqwrl:select(?p, ?a)} \wedge \texttt{sqwrl:orderBy(?a)} \tag{3.7}$$

SQWRL supports basic counting using a built-in called *sqwrl:count*. As an example, the following query returns the number of people in the underlying ontology:

$$\texttt{Person (?p)} \longrightarrow \texttt{sqwrl:count(?p)} \tag{3.8}$$

And as an example of grouped count, the following query returns people and the number of cars each one has:

$$\texttt{Person (?p)} \wedge \texttt{hasCar(?p, ?c)} \longrightarrow \texttt{sqwrl:select(?p)} \wedge \texttt{sqwrl:count(?c)} \tag{3.9}$$

Basic aggregation is also supported. This feature is provided by built-ins called *min*, *max*, *sum*, and *avg*. As an example, the following query returns the average age of the people in the ontology:

$$\texttt{Person (?p)} \wedge \texttt{hasAge(?p, ?age)} \longrightarrow \texttt{sqwrl:avg(?age)} \tag{3.10}$$

SQWRL supports the use of OWL class description. The following query retrieves all individuals that are associated with a restriction on a *hasChild* property. In this case, the query returns all the people having more than one child:

$$\texttt{hasChild} >= \texttt{1(?i)} \longrightarrow \texttt{sqwrl:select(?i)} \tag{3.11}$$

SQWRL queries can operate in conjunction with SWRL rules to retrieve knowledge inferred by other rules. Assume that an ontology has the following restriction to classify people as adult:

$$\texttt{Person (?p)} \wedge \texttt{hasAge(?p, ?age)} \wedge \texttt{swrlb:greaterThan(?age, 17)} \longrightarrow \texttt{Adult(?p)} \tag{3.12}$$

Hence, a query to list adult people in the ontology can be:

$$\begin{aligned} &\texttt{Adult(?p)} \\ &\longrightarrow \texttt{sqwrl:select(?p)} \end{aligned} \tag{3.13}$$

Intermediate inferences provided by SWRL rules make SQWRL able to decompose very complex queries. Because defining sub-queries is not provided in SQWRL, these intermediate inferences is an appropriate alternative. These inferences can be used by other queries and rules.

3.2.2 Set Operators: Closing the World

SQWRL supports some degree of closure in its queries considering OWL's open world assumption. SQWRL using set operators provides such additional requirements. A built-in called
sqwrl:makeSet is provided to construct a set:

$$\texttt{sqwrl:makeSet(<set>, <element>)} \tag{3.14}$$

The first argument specifies the set and the second one specifies the element to be added to the set. In SQWRL the set operator is shown with '$\circ$' character. An example of a query to list the number of persons in an ontology can be written:

$$\begin{aligned} &\texttt{Person (?p)} \circ \texttt{sqwrl:makeSet(?s, ?p)} \wedge \texttt{sqwrl:size(?size, ?s)} \\ &\longrightarrow \texttt{sqwrl:select(?size)} \end{aligned} \tag{3.15}$$

The set operator provides a closure mechanism. Essentially, to process such queries, two phases are required: 1) constructing the set, 2) Applying the other rule elements to the already constructed sets.

Some basic set operators provided in SQWRL are: sqwrl:union, sqwrl:difference, sqwrl:intersection. For example, a query to list the number of non beta-blocker drugs in an ontology with a class *Drug* and its subclass *BetaBlocker* is as follows:

$$\begin{aligned} &\texttt{Drug(?d)} \wedge \texttt{BetaBlocker(?bbd)} \circ \texttt{sqwrl:makeSet(?s1, ?d)} \wedge \\ &\texttt{sqwrl:makeset(?s2, ?bbd)} \wedge \texttt{sqwrl:difference(?s3, ?s1, ?s2)} \wedge \\ &\texttt{sqwrl:size(?n, ?s3)} \longrightarrow \texttt{sqwrl:select(?n)} \end{aligned} \tag{3.16}$$

This query lists the number of *non* beta-blocker drugs.

3.2.3 Ontology Traversing Operators: Drill-Down and Roll-Up

In order to navigate among various levels of class hierarchies of an ontology, we introduce two important operators that SQWRL provides:

— *Drill-Down* allows navigating among levels of data ranging from the most summarized to the most detailed concepts. For example, in the Figure 3.1, if our current position is in the level of class *Human*, in order to concentrate our analysis into a specific type of human, by one level drilling-down, we can move to the class *Male* or the class *Female*.

— *Roll-Up* allows navigating among levels of data ranging from the most detailed to the most summarized concepts. For example, in the Figure 3.1, if we want to generalize our analysis in order to cover more people, by one level rolling-up toward to the higher levels of the class hierarchy, we will reach to the considered level.

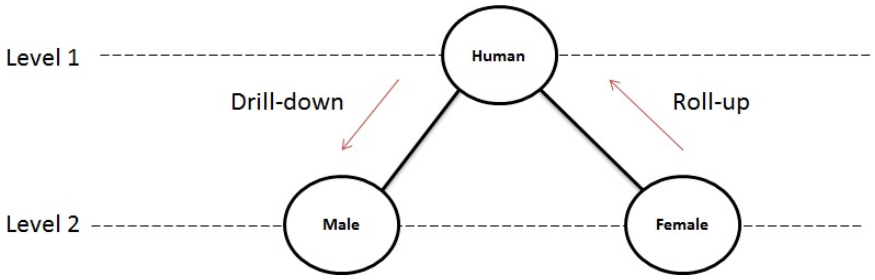

Figure 3.1 Drilling-down and rolling-up operators

3.3 Ontologies for Security Requirements

As the knowledge of computer security is growing in both science and industry, many security ontologies have been proposed during recent years. Some of these ontologies include all computer security aspects and they are considered as general ontologies. On the other hand, some other ontologies tackle a specific aspect of security, such as risk analysis, event correlation, attack analysis, etc. Figure 3.2 illustrates 8 families of security ontologies. Here we describe some of these families.

A taxonomy can be considered as an ontology in the form of hierarchy. However, these types of ontologies are considerably limited, and they only cover the generalization relationship. In the computer security domain, we can find many taxonomies representing various fields. For instance, [18] provides a taxonomy including classes of *faults*, *fault modes*, *fault tolerance techniques*, and *verification* approaches. In this taxonomy, the main threats are defined as

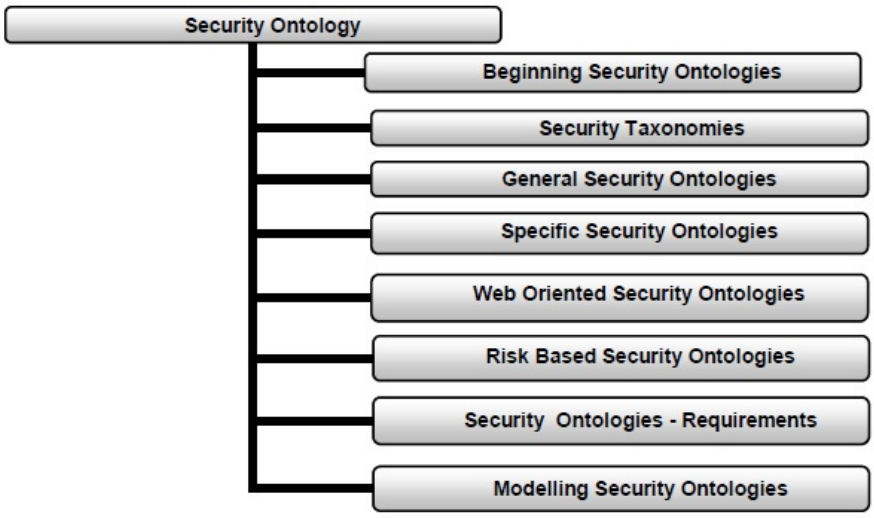

Figure 3.2 Classification of security ontologies into 8 families [158]

failure, errors, and faults. Landwehr *et al.* [104] proposed a taxonomy for security flaws. The taxonomy is based on three basic questions about each observed flaws: genesis, time of introduction, and location.

General ontologies try to cover all security aspects. For instance, Herzog and colleagues [80] proposed an OWL-based ontology for security information. The proposed ontology is based on *assets*, *threats*, *vulnerabilities*, and *countermeasures* concepts. Fenz and Ekelhart in [65] have proposed even broader ontology covering a larger part of the information security domain.

Specific security ontologies describe specific aspects of security, such as intrusion detection, security vulnerabilities, etc. Undercoffer *et al.* in 2003 [175], proposed a specific ontology for computer attacks and intrusions and represented it with an ontology representation language. In this ontology there is a class called *attack* described by the *Directed to*, *Effected by*, and *Resulting in* properties, and the *System Component*, *Input*, and *Consequences* are corresponding objects. Figure 3.3 illustrates this ontology. Viljanen in 2005 [178], proposed an ontology modeling security trust. The proposed model is classified as *identity-aware*, *action-aware*, *business value aware*, *capability-aware*, *competence-aware*, *confidence-aware*, *context-aware*, *history-aware*, and *third-party-aware*. Geneiatakis and Lambrinoudakis in [71] proposed an ontology for SIP-VoIP based services. This ontology covers attacks targeting VoIP service on internet. The ontology which contains two main concepts SIP_attack and $SIP_message$ can be applied to find both attack countermeasures and testing the security robustness of SIP-VoIP infrastructure.

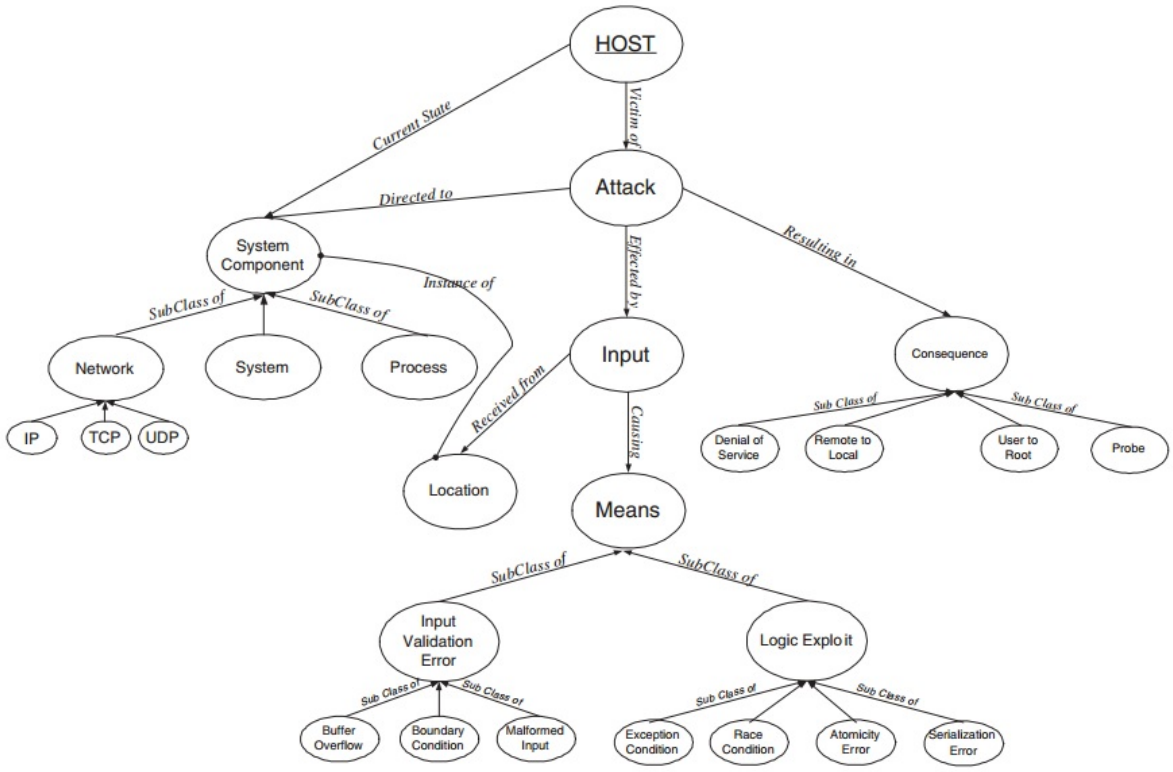

Figure 3.3 The proposed ontology by [175]

Denker *et al.* in [52], [53], [54] proposed a number of OWL ontologies for security annotations of web services. The proposed ontologies have two sub-ontologies including "*security mechanisms*" that describes high-level security notations and "*credential*" describing on authentication methods. The major goal of these ontologies is to enable high-level markup of web resources, services, and agents. The NRL security ontology proposed in [99] is based on 7 security ontologies including "*Main Security Ontology*", "*Credential Ontology*", "*Security Algorithms Ontology*", "*Security Assurance Ontology*", "*Service Security Ontology*", "*Agent Security Ontology*", and "*Information Object Ontology*". Artem Voroiev and Jaun Han in [180] proposed an attack ontology for web services. The ontology includes most of the popular web services attacks.

Risk analysis is one of the most important security processes. Ekelhart *et al.* in 2007 [60], proposed a security ontology covering risk analysis methodologies. Assali *et al.* in 2008 [17], also proposed an ontology for industrial risks analysis.

Some research work have proposed ontologies covering security requirements. Tsoumas *et al.* [172] have used *Asset*, *Stakeholder*, *Vulnerability*, *Countermeasure* and *Threat* concepts in

their proposed ontology. In [97], the authors propose an OWL ontology in order to develop secure applications. The ontology includes classes, such as "*Countermeasures*", "*objectives*", and "*threats*". They evaluate the proposed ontologies using nRQL queries. Firesmith in [66] propose a taxonomy of safety-related requirements including *Safety Requreiments*, *Safety-Significant Requreiments*, *Safety constraints*, *Safety system requirements*.

3.4 Previous Work in Ontology-Based Intrusion Detection Systems

Security researchers need to have a seamless mechanism to integrate various information generated by heterogeneous sources into a common data store that allows them to query the combined sensor alerts and contextual information. In order to solve this problem, several researchers have proposed semantic-based alert correlation approaches in order to automate and improve the flexibility of the correlation process [39, 92, 107, 146, 175, 179, 182].

Ontologies are knowledge representation models that allow the description of concepts, their attribute and the inheritance and association relationships between these concepts, in a way very similar to object-oriented modelling frameworks such as the Unified Modelling Language (UML). This unified yet user-extensible representation of concepts, definitions and relationships also allows the use of reasoning logic formalisms, that can be used to retrieve information in a generic and class structure-agnostic fashion. In addition, various types of ontologies have formal description languages that allow for the definition of complete reasoning logic that are machine-interpretable and solvable. Hence, they can be suited for representing concepts and for automated reasoning on domain-specific applications with a limited number of concepts. For that reason, some researchers have proposed ontology-based alert correlation approaches in alert correlation. In our case, we will be using these reasoning logic formalisms to design event correlation algorithms.

Vorobiev *et al.* [179] proposed security ontologies to improve IDS capabilities for detecting new generation of attacks such as multi-step distributed attacks and various distributed denial of service (DDoS) attacks. In their paper, they introduced several ontologies including Security Attack Ontology (SAO), the Security Defense Ontology (SDO), the Security Asset-Vulnerability Ontology (SAVO), the Security Algorithm-Standard Ontology (SASO), and the Security Function Ontology (SFO). The proposed ontologies encompass basic security concepts including attacks, defenses, functions, vulnerabilities, etc. Here, we briefly describe SAVO which is the major ontology of the proposed work. This ontology is located at a higher level than the other proposed ontologies. Figure 3.4 illustrates its general structure. In this ontology, the term "Asset" consists of all the entries in the underlying environment that need to be securely protected, such as data ($'Data'$), software ($'Software'$), accounts

($'Accounts'$), and resources ($'Resource'$). The class $'Asset'$ consists of several subclasses including $'ClientData'$, $'SystemData'$, $'Component'$, $'Service'$, $'CPU'$, $'Memory'$, $'Storage'$, etc. The term $'Threat'$ concerns any unwanted event which is dangerous for system elements. $'ThreatAgent'$ concerns any agent using threat to abuse system resources. $'Vulnerability'$ consists of any flaw or weakness that could bring the whole system under threat. Finally, $'defense'$ represents the security methods protecting the system against threats. The authors evaluated their proposed ontological approach against Mitnick, a multi-phased distributed attack [176] that is an instance of man-in-the-middle attacks. However, the proposed ontologies encompass only general security concepts and there is no detailed description of how these ontologies can be utilized in different contexts.

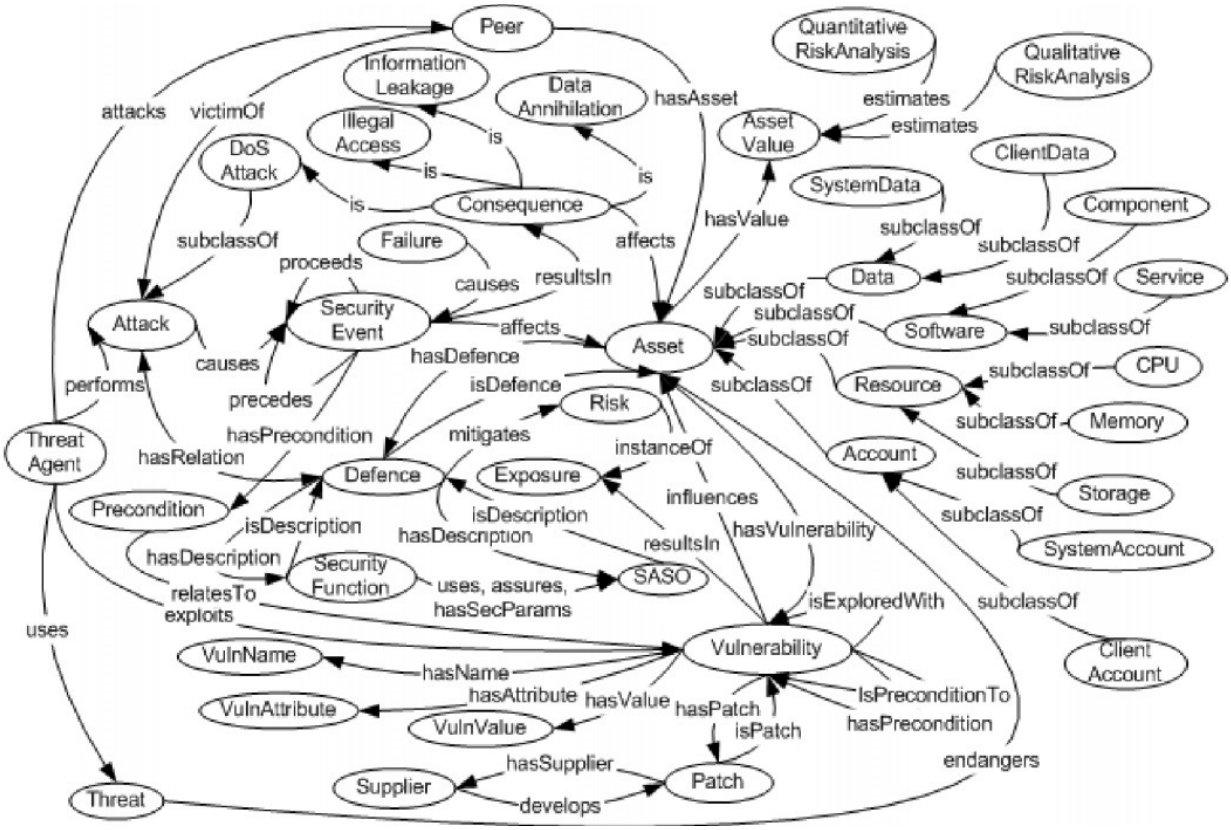

Figure 3.4 The Security Asset-Vulnerability Ontology [179]

Coppolino *et al.* proposed the Intrusion Detection and Diagnosis System (ID2S) using an ontology-based detection approach that correlates detection information at several architectural levels for further intrusion symptom analysis [39]. In the proposed approach *Diagnosis* means the ability to identify the reasons of the happened attack that took place, and the ability to estimate consequences of the attack accurately. As currently available tools

only suport intrusion detection or prevention, the lack of an Intrusion Diagnosis approach is significant. To do a diagnosis, attack related information should be collected from several architectural levels such as networks, operating systems, databases, and applications. These collected information using a Complex Event Processing (CEP) technology are correlated and analyzed to result an appropriate and efficient detection. The authors have developed two types of ontologies: the first one takes attack symptoms and detects which attacks are related to these symptoms, and the second one estimates potential damages which are consequence of the attack. Finally, appropriate reactions and responses are issued based on the output of this process. The authors also emphasize that their proposed approach is able to detect new types of stealthy attacks [94]. These attacks represent a major threat to not only the network level, but also the levels of underlying systems. Some of the other advantages of their work are improving detection performance, providing diagnostic capabilities, and reducing false alarms. We also believe that providing a diagnosis, because of using detection information at several architectural levels, can improve the detection efficiency. Moreover, using ontologies can improve automation of the detection process, and has potential to detect unknown attacks. The main shortcoming of the proposed work is their evaluation part which is limited to only limited attacks: SQL Injection and Cross Site Scripting. In addition, the proposed work does not take into account contextual information that can be very useful in reducing the false positive rate.

Wang *et al.* in 2009 [182], proposed an ontology-based approach called Ontology for Vulnerability Management (OVM) to model security vulnerabilities listed in the NVD [129] with additional inference rules, data-mining mechanisms, and knowledge representation. Figure 3.5 illustrates the proposed ontology. The top level concepts of the ontology are: *Vulnerability*, *IT_Product*, *Attacker*, *Attack*, *Consequence*, and *Countermeasure*. An existing vulnerability in a *IT_Product* may be exploited by an attacker using a related attack. Countermeasures are employed to protect the *IT_Product* through mitigating the Vulnerability. The authors believe that the achieved results provides an appropriate path to make the security automation successful.

Unfortunately, most of these works only cover only one or limited aspects of information required within the correlation process. For instance, some of these works do not take into account the contextual information, or some others are not able to correlate heterogenous event generators.

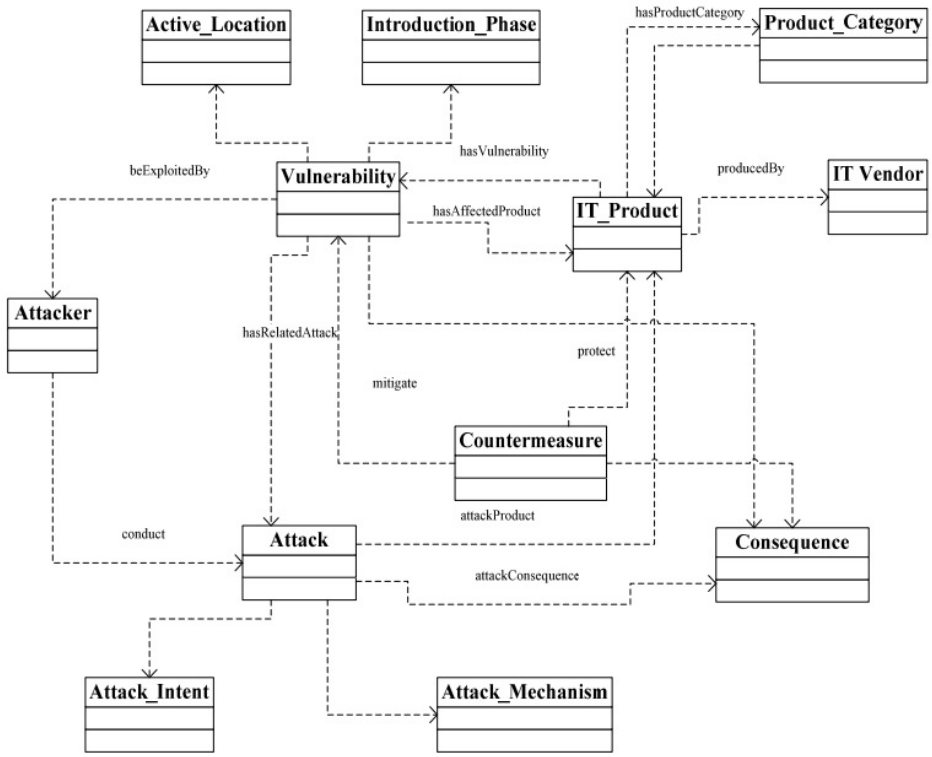

Figure 3.5 Wang's proposed ontology for vulnerability management [182]

3.5 Summary

In this chapter, we introduced the concept of ontology, its features and its role in computer security research. We described that the concept of "Ontology" is a significantly promising new paradigm in computer security. It represents knowledge in a formal and structured form providing a better tool for communication, reusability, and organization of the knowledge. We also provided a brief introduction to OWL which is a popular language to design and develop ontologies. Next, we introduced the SQWRL which is the de facto standard OWL query language. We described that drilling-down into very detailed and specific classes to analyze only a small portion of instances, and rolling-up toward very generic classes of the ontologies to cover a larger portion of instances, are two common and most useful operators that both SWRL and SQWRL provide to extract our considered information from various ontologies. In section 3.3, we described some a classification of ontologies proposed for security requirements. Finally, in section 3.4, we reviewed recently proposed ontology-based intrusion detection and alert correlation approaches. We highlighted that the reviewed ontology-based approaches have not taken full advantage of ontologies expressive power in terms of data modelling and logic reasoning. They also mostly focus on generic security

concepts rather than the detailed steps of a comprehensive alert correlation approach. Lack of appropriate implementation by mentioning the employed tools and ontology querying languages, and efficient evaluation are other limitations of most of these approaches.

CHAPTER 4 *PASARGADAE*: A CONTEXT-AWARE AND ONTOLOGY-BASED EVENT CORRELATION FRAMEWORK

Intrusion detection and alert correlation systems play a key role in the surveillance and monitoring of computer network infrastructures. Considering the current advances of the hacker capability, these systems are increasingly important in any computer network in order to protect the information systems of any organization. However, as described in chapter 1, one of the main shortcomings of these systems is that they produce a large number of false positives, and duplicate and non-relevant alerts. Our goal in this research work is to present to the network analyst only interesting alerts. Once an alert is produced, we want the alert to be actionable. For this purpose, we leverage the vast quantities of data available in the various sensors already present in the network to improve the value of the alerts we present to the administrator. In addition, any IDS and alert correlation system that can automate some of the correlation and contextualization that (good) security analyst are forced to do manually today will greatly enhance their productivity and security posture of their organizations.

In order to provide a method to gather event logs from various heterogenous sensors distributed in a computer network, and automate the analysis of the various information resources available to the security analyst, while preserving maximum flexibility and power of abstraction in the definition and use of such concepts, we propose the *Pasargadae* ontology-based context-aware event correlation framework in this chapter. *Pasargadae* uses ontologies to represent and store information on events, context and vulnerability information, and attack scenarios, and uses simple ontology logic rules written in Semantic Query-Enhance Web Rule Language (SQWRL) to correlate and filter out false positives, duplicate and non-relevant alerts.

The rest of this chapter is organized as follows. In section 4.1, we explain the *Pasargadae* event correlation framework and its components including event, context, attack and vulnerability ontologies along with its context and event integration components, and correlation engine. Next, in section 4.2, we propose a semantic-based and context-aware event correlation approach that employs *Pasargadae* as its main framework for the event correlation purpose. In section 4.3, in order to show the flexibility and extendability of *Pasargadae*, we explain how *Pasargadae* can be employed to implement other alert and event correlation approaches. Finally, we give a short summary of this chapter.

4.1 *Pasargadae* Event Correlation Framework

In this section, we describe *Pasargadae*, an automated and context-aware event correlation framework that relies heavily on ontologies and ontology description logic. *Pasargadae* performs heterogenous event correlation which means that it gathers event logs from not only NIDS, but also from other sensors, such as HIDS, operating systems, databases, anti viruses and some applications. The *Pasargadae* framework was made context-aware in order to take full advantage of the context information to which security analysts have typically access to prioritise alerts, and ontology-based in order to provide a technological solution to the problem of heterogeneous data integration and retrieval. The *Pasargadae* framework is depicted in Figure 4.1.

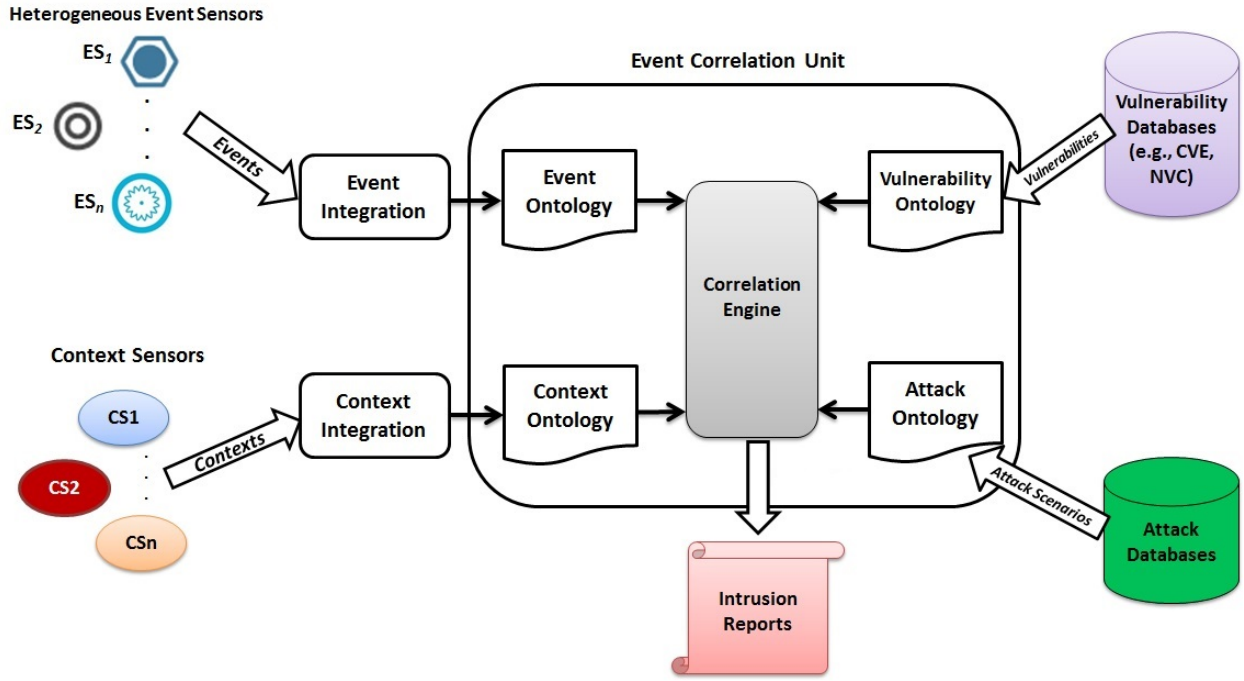

Figure 4.1 The Passargade ontology-based context-aware event correlation framework

As Figure 4.1 illustrates, in its first step, the alerts generated via distributed heterogeneous sensors are collected and transferred into the event integration component. Also in this step, all the information required for reasoning on these events is gathered from three different information resources namely:

— Context Sensors (CS)
— Common vulnerability databases, and
— Attack databases

The second step consists of the following two tasks:

— integrating and converting all the events generated by the heterogenous sensors into a unified format analysable by the event correlation unit.

— integrating all the contextual information received implicitly or explicitly from the various tools implemented in the system.

In the third step, the event and context ontologies are populated based on the integrated and converted events and context information. In order to fully automate the event correlation process, we have designed a group of comprehensive and extensible ontologies, namely event, context, attack and vulnerability ontologies. The explicit relationships between these ontologies allow reasoning on the information gathered from various resources, including the (mostly) static attack and vulnerability databases.

The last step consists of correlating the existing information within the ontologies, which is done via the correlation engine using ontology description logic. In the rest of this section, we describe each of the above components in detail.

4.1.1 Information Resources

For information resources, we consider the following sensors as our event and contextual information resources:

Event sensors: These sensors generate a specific type of logs based on the events that they analyze on the systems they monitor. The most typical and commonly deployed type of sensor are NIDS that generate alerts by examining individual network traffic packets. They also include host-based IDS and antiviruses that generate alerts and events based on system or application activity observed on a particular machine. Finally, it also includes other types of non security-related sensors such as operating systems, databases, proxy servers, routers, access points, applications that are not generating alerts per se, but rather system events that the security analysts consider important enough to be correlated with other sources of events. The difficulty here is that while many NIDS and HIDS will generate IDMEF-compliant alerts by filling generic attributes (e.g. time, severity, etc.), there might be some sensor- or log-specific attributes that we might want to correlate on, and that must therefore be integrated also. This is what ontologies are particularly suited for.

Context sensors: A *context sensor* is a generic term for any information source that can provide contextual information about the systems that are being monitored. The concept of context is purposefully vague to allow analysts to define and use the particular aspects that

they think is suitable for monitoring of their systems. This can include different types of information such as configuration (network, host or application), vulnerabilities, user role and profile, location, and even criticality of the corresponding IT asset. Contextual information can be implicitly collected by methods such as vulnerability scanning, network fingerprinting, passive network monitoring tools, or they can be explicitly provided by system administrators through tools such as Configuration Management Systems (CMS), for example.

Known vulnerabilities: At first, we gather information about vulnerabilities from the well-known public databases such as the Common Vulnerabilities and Exposures (CVE) [46] or the NVD [129]. Then, vulnerabilities can be associated to context instances (e.g. hosts, networks, applications) through vulnerability scanning or asset management. Severity information from these databases, in combination with information on asset criticality, can then be used to help prioritise alerts occurring in these contexts.

Attack scenarios and models: An attack depending on its type, may exploit a specific vulnerability to proceed. Attack information and models can be obtained from standardised databases such as the Common Attack Pattern Enumeration and Classification (CAPEC) [26] or expert knowledge. In order to model attacks, any of the existing attack modelling languages such as LAMBDA [44], STATL [59], Adele [119] or SHEDEL [118] could be used. These languages model attacks based on various techniques, such as scenario or sequence-based modeling, state/transition-based modeling, temporal modeling, etc. However, it is outside of the scope of this work to implement these formalisms within the ontology description logics that we use. In the rest of this chapter, and without loss of generality, we will illustrate our framework using a simplified attack model comprising a linear sequence of steps.

4.1.2 Event and Context Integration

Different types of event sensors produce event logs in various formats that might not be natively interpretable by the event correlation unit. Hence, it is necessary to preprocess these alert streams and export them in a format that is understandable by the event correlation unit. In production environments, this would be done by sensor specific drivers that would match event fields with class attributes at the appropriate abstraction level. In following good ontological engineering practises, all event sensor-specific fields should be translated into class attributed at the highest possible class in the taxonomy of events, i.e. that where all subclasses contain that type of information (or an equivalent one). The use of standard

representations such as IDMEF [47] or the Common Event Expression (CEE) [121] should be encouraged, but not at the detriment of not integrating sensor-specific information that is not standard-compliant; that is what sensor-specific event subclasses are for. For simplicity of presentation and for illustrative purposes, we use an IDMEF-specific ontology.

Additionally, one of the main tasks of the event integration component is event refinement. Obviously, each detection sensor has different logging format and logging attributes. Therefore, some of the sensors may not support particular attributes or logging features. This component covers missing data and attributes (specially the attributes which are important in following components) existing inside collected events. Moreover, this component smoothes the received data in case of any noise. Type transformation is another important task of this module. The detection sensors may generate the value of event attributes in different types (string, integer, etc.). This component transforms all the received data (attribute values) to a unified type consistent with our defined policies.

The context integration component of our framework also integrates all the contextual information in various formats received implicitly or explicitly from various tools implemented in the system. In this component, the contextual information gathered using our designed drivers is converted into a unified format analysable by the other components, i.e. into instances in the context ontology. Once the integration process is complete, the correlation process can start.

4.1.3 Description of the Ontologies

We chose to use ontologies because they provide a powerful knowledge representation information structure in a unified format that is understandable by both machines and humans. Ontologies also allow the use of reasoning logic formalisms, that can be used to retrieve information in a generic and class structure-agnostic fashion. In our case, we use these reasoning logic formalisms to design alert correlation algorithms, that will attempt to reconstruct possible attack scenarios while eliminating improbable ones, while making abstraction of irrelevant sensor- or system-specific details. The use of ontologies and ontology description logic thus enables us to fully automate the correlation process that is typically done manually by security analysts, and this uniformly considering all relevant information, no matter what its original format or source.

In order to integrate the data inputs to the correlation process and allow generic correlation reasoning, independent of specificities of information resources, we have constructed basic ontologies capturing the essence of the concepts of event, context, vulnerability, and attack. Essentially, they correspond to the following intuitive security facts:

1. Attack scenarios will generate system events that might in turn trigger sensors to cause related alerts. Depending on the attack model, an attack scenario might be described as linear sequence of events, or a partial ordering of events with pre- and post-conditions, an attack graph, etc.
2. Events happen in a context, whether this is an IT asset, network location, application, user, etc. In our case this relationship will be made explicit through information provided by the sensor with the event (e.g. IP address).
3. Vulnerabilities are always associated to a context, whether to high-level context concepts (e.g. an asset or service type) or to lower-level context subclasses (e.g. particular versions of OS or applications). Conversely, explicit context instances can be linked to specific or generic vulnerabilities, through vulnerability assessment or CMS information.
4. Most attack scenarios will require certain vulnerabilities to be present on the systems (context) so that they can exploited by that attack.

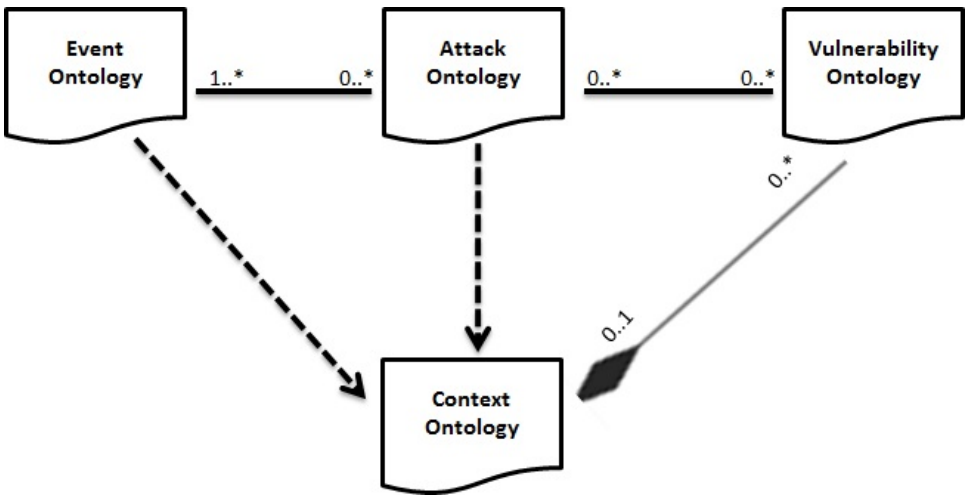

Figure 4.2 Conceptual relationship of the proposed ontologies

Figure 4.2 illustrates the conceptual relationships of the ontologies, and Figure 4.3 illustrates the class relationships and some of the subclasses of the basic ontology. These "starter" ontologies are not meant to be the end state of knowledge representation that security analysts would need in using our framework, but rather a starting point or template from which to build on, depending on the kind of sensors, context information or granularity of vulnerabilities and attack modelling desired. We now describe each of these ontologies in more detail.

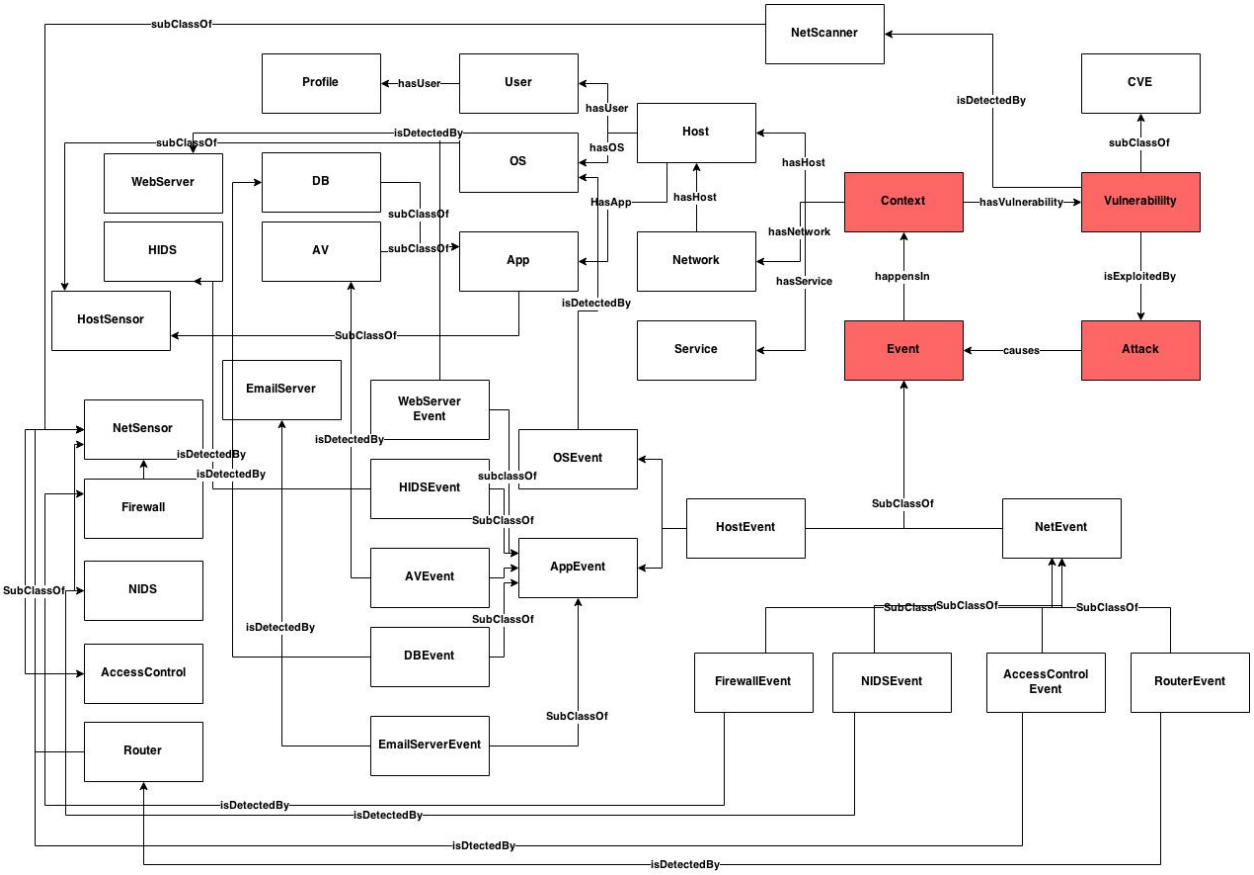

Figure 4.3 Class diagram relationship of the designed ontologies

Event ontology. All events in the integrated event logs are transferred into this ontology as its instances. As explained above, it has dependency relationship with the context ontology and an association relationship with the attack ontology, since usually each malicious event e is typically generated by a (suspected) attack at in a particular context c. The generic base class Event includes generic event attributes, such as time, and analyser (i.e. sensor). As Figure 4.4 illustrates, the Event class has three subclasses NetEvent, HostEvent, and AppEvent, corresponding to events generated by NIDS, HIDS, and application-based IDS or application logs, respectively, each containing specific sensor-generated attributes. It is important to note that because the concept of context is potentially very rich and multifaceted, it is likely that a single event might have to be linked to multiple context instances from various subclasses (e.g. a user, a network segment, an application), and thus the association between event and context will be many-to-many at the Context base class level.

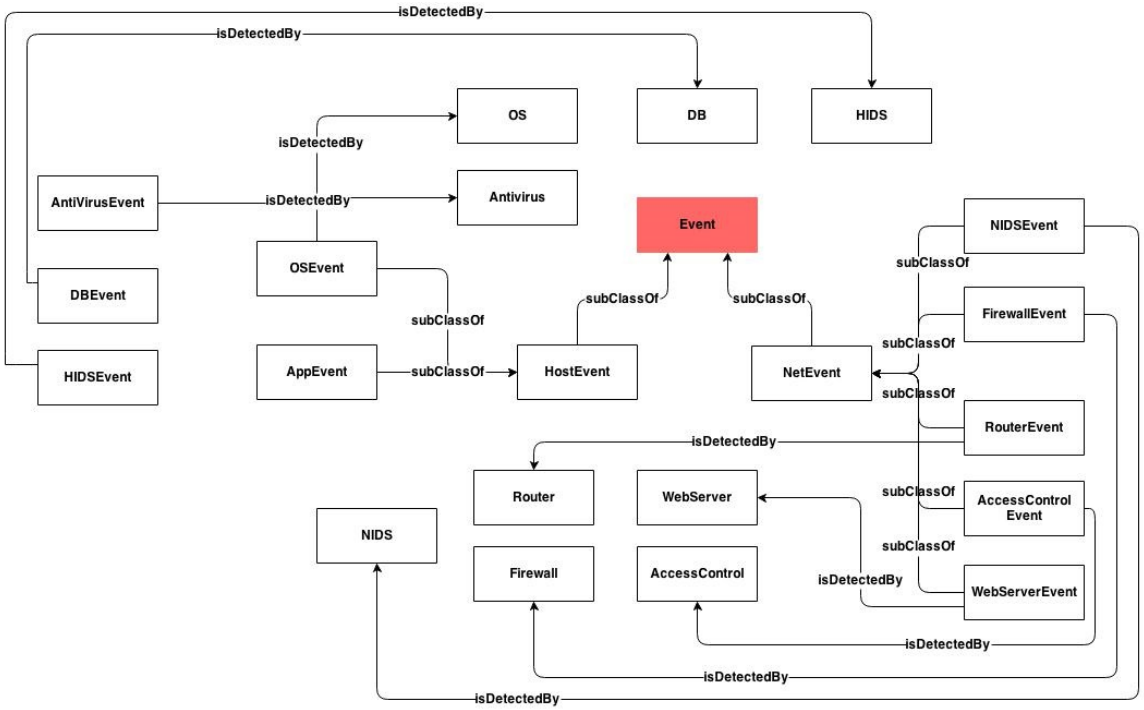

Figure 4.4 Class diagram relationship of the Event ontology

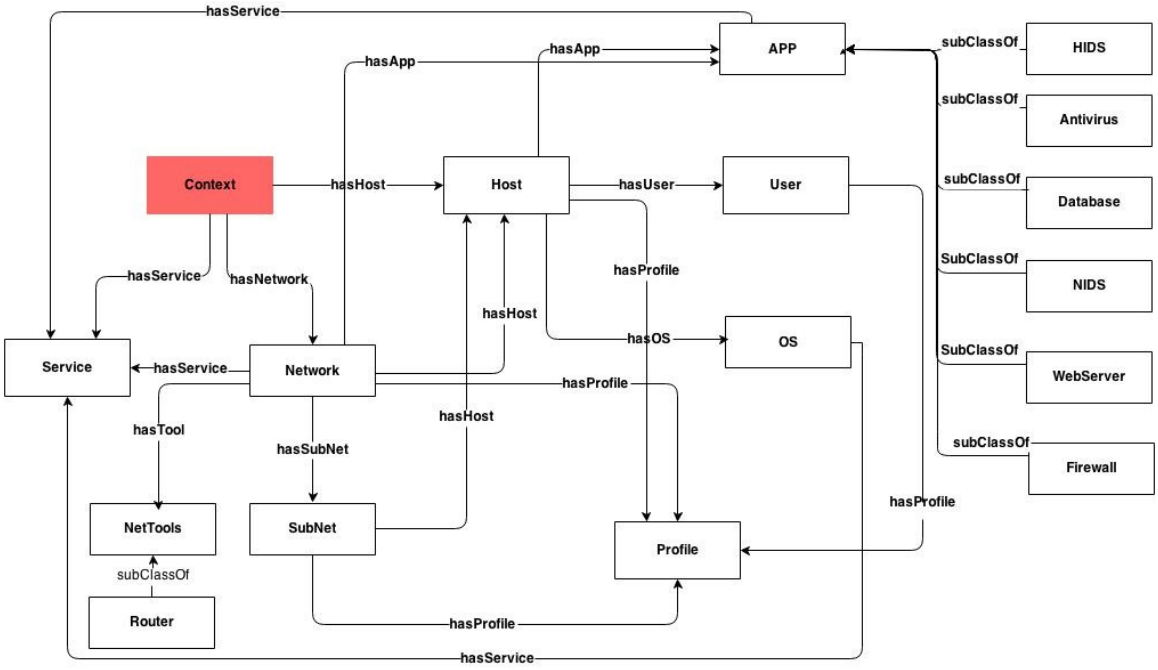

Figure 4.5 Class diagram relationship of the context ontology

Context ontology. The integrated contextual information is transferred into the context ontology. We split contextual information into two categories: *i*) static context information

that rarely changes over time (e.g. network architecture, host/user profiles, and OS type), and *ii*) dynamic context information that changes continuously over time (e.g. traffic type, system usage, time of day/week). Table 4.1 and 4.2 lists some of the possible subclasses of context class and their static and dynamic attributes. As depicted in Figure 4.5, the context ontology includes a Context base class and User, Host, Network and Service subclasses with their corresponding attributes. As mentioned, the implicit and explicit context information appear as instances of the context class in our proposed ontology, that will be populated through static information from CMS or system administrators, dynamic information from network profiling tools and even alert sensors themselves (e.g. when reporting on previously unknown IT assets/contexts).

Table 4.1 A list of static attributes of the context ontology classes

Organization	**Network**	**Host**	**User**	**OS**	**Application**
ID	Topology	Name	ID	Platform	Name
Product	Protocol	IP Address	Role	Type	Version
Client	Address Range	MAC Address	Location	SPVersion	
Location	Firewall	Role	Access rights	Version	
Network	Switch	Location			
	Host	User			
	User	Service			
	# hosts	OS			
	#Firewall	Application			
	#Subnet	CPU			
	#Switch	Memory			
		#User			
		#OS			

Vulnerability ontology. This ontology represents the list of vulnerabilities related to the existing assets in the underlying context, typically populated from a public vulnerability source such as CVE or NVD. This ontology has a part-whole relationship (composition) with the context ontology since every vulnerability v is specific to a particular type of system, which is represented as a part of Context (typically Host). Thus, v can be associated with all the asset (context) instances $\{c_1, ..., c_n\}$ that are vulnerable to it, by querying the ontology for Host instances whose applications (App) or OS are those associated to that vulnerability. This ontology also has an association relationship with attack ontology, since usually every vulnerability v is exploitable by 0 or more (0..$*$) attacks.

Table 4.2 A list of dynamic attributes of the context ontology classes

Date&Time	**Network**	**Host**	**OS**	**Application**	**Usage Profile**
Date	Traffic Type	CPU Usage	Syscall	Started State	PerUser
Time		Memory Usage	Table		PerHost
		Disc Usage	Processes		PerNet
		Open ports			
		Started Apps			
		Current Users			
		Connection			
		Status			
		Application			
		CPU			
		Memory			
		Sent (Bytes)			
		Received (Bytes)			

Attack ontology. The attack ontology includes information related to the known attack scenarios, and it includes generic attack attributes such as vectors, objectives, and so on. The Vector class represents the method that can be used by an attack to compromise computer systems, with example subclasses including social engineering, phishing, removable media, etc. The Objective class can include example subclasses such as information leakage, remote core execution, spamming and privilege escalation. The attack ontology has dependency relationships with the context ontology, and association relationships with the event and vulnerability ontologies, since basically every attack a needs a particular context c to proceed, and it might need to exploit particular vulnerabilities $\{v_1, ..., v_n\}$, and it results in triggering some events $\{e_1, ..., e_n\}$.

Rolling-up and drilling-down in the designed ontologies. In order to navigate among various levels of class hierarchies in the designed ontologies, here, we introduce two important operators that we can employ within the correlation engine of the *Pasargadae* framework. These operators are:

— *Drill-Down* allows to navigate among levels of data ranging from the most summarized to the most detailed concepts.

— *Roll-Up* allows to navigate among levels of data ranging from the most detailed to the most summarized concepts.

Drill-Down and *Roll-Up* do not remove any data that exists in the ontologies but change the level of granularity of a particular dimension. As an example, in Figure 4.3, lets assume that we are in the *event* class level. If in a particular ontology rule we want to only analyze the information of the *FirewallEvent* class, by two level navigation using the *Drill-Down* operator, we can reach the *FirewallEvent* class, and analyze its data. In order to return to the previous level (*event* class level), we can use *Roll-Up* operator for two times to reach the considered level.

These two operators are always employed in the correlation engine of the *Pasargadae* framework in order to correlate the information of the various ontologies in different levels of granularity.

Algorithm 1 Event Correlation Pseudocode

INPUT: Event Ontology (E), Context Ontology (C), Vulnerability Ontology (V), Attack Ontology (A).
OUTPUT: Attack Scenarios: AS[]
BEGIN

{Context- and vulnerability-based-filtering}
for all $Alert \in Event - Class$ **do**
  $targetSystem \leftarrow AlertTarget$
  $targetApp \leftarrow AlertApp$
  $targetVulnerabilities \leftarrow AlertVulnerability$

  if $targetApp \in targetSystem$ **then**
    $targetAppVulnerability \leftarrow V$

    if $targetVulnerabilities \in targetAppVulnerability$ **then**
      $finalAlertList \leftarrow Alert$
    end if
  end if
end for

{Attack reconstruction}
$AS[] \leftarrow EventCorrelationAlgorithm(finalAlertList)$

END

4.1.4 Correlation Engine

In order to implement the correlation logic, we employ the Ontology Web Language-Description Logic (OWL-DL) to design and populate an ontology for each of

the above four inputs. The usage of a generic language like OWL-DL provides significant flexibility to the framework by allowing the reuse or adaptation of data queries expressed in that logic to various deployment and security monitoring scenarios, e.g. on-line detection or after-the-fact network forensics analysis. Additionally, it provides sophisticated capabilities such as drilling-down and rolling-up in the class hierarchies of the ontologies to express very specific or very generic concepts.

Generally speaking, the correlation engine will have information about only three of the four ontologies that we have defined, i.e. event, context and vulnerabilities. It is by following the above-mentioned relationships between the corresponding classes that the correlation engine will be able to infer whether there is an attack instance that could match a particular subset of linked alerts, contexts and vulnerabilities. This correlation process is two-fold and can be viewed as two independent traversals on the core ontology classes:

— *Context and vulnerability-based filtering.* Given an event (or events) determine which contexts instances are involved, what are their associated known vulnerabilities, and finally determine which attack scenarios could be exploiting them. For instance, assume that there are some alerts issued by the Suricata IDS/IPS reporting malicious activities exploiting buffer overflow vulnerability of Microsoft SQL Server 2010. If in the underlying network there are some computers having Microsoft SQL server 2010, these alerts are considered context-relevant. Otherwise, they will be discarded from the final analysis because of their non-relevancy to the underlying context.

— *Attack reconstruction.* In this step, the information of the attack ontology that includes a comprehensive list of known attack scenarios is employed. Therefore, when there is an event or a group of events reported by the event sensors, for each possible attack scenario related to this (or these) event(s), try to match the sequence of previous alerts with the steps of the attack. We can use various alert and event correlation, and attack reconstruction algorithms in this step due to the considerable flexibility of the framework.

The pseudocode for the generic correlation process is presented in Algorithm 1. The outcome of this process should hopefully provide the security analyst with a reduced list of high level descriptions of potential ongoing (or completed) attacks that includes few redundancies, non relevant scenarios and false positives. In order to implement both steps of this event correlation process we use a set of logic rules expressed in Semantic Web Rule Language (SWRL) and Semantic Query-Enhanced Web Rule Language (SQWRL).

As mentioned earlier, in order to have a significant flexility, the most important capabilities that *Pasargadae* framework provides are easy drill-down and easy roll-up into the most

specific or most generic classes and attributes. Using these considerable capabilities, we can write very generic rules covering generic concepts, or very specific rules covering detailed concepts. Having such significant flexibility and adaptability makes the correlation engine capable to implement various alert and event correlation, alert fusion, and attack reconstruction approaches. Furthermore, within its correlation engine, *Pasargadae* can add some sophisticated features, such as context-awareness, extendability, and object orientation, to the existing correlation, fusion, and attack reconstruction approaches.

In the following sections, first, we propose a new semantic-based event correlation approach that employs *Pasargadae* to do its task. Next, we describe how *Pasargadae* is able to represent some of the existing alert correlation steps proposed in [177].

4.2 A Semantic-Based Event Correlation Approach Based on *Pasargadae*

In this section, we describe a new semantic-based event correlation approach that employs *Pasargadae* as its main framework. The proposed event correlation approach that has 3 main steps, can be implemented easily in the correlation engine of *Pasargadae*. In the proposed approach, after eliminating non-relevant alerts, events based on their attributes, are classified into a number of groups, and for every group a meta-event is created. Finally, these meta-events are logically analyzed to reconstruct existing attack scenarios. In the following, we describe the proposed event correlation process in detail:

1. *Non-relevant alerts elimination.* As described earlier, non-relevant alerts are those alerts that even though they correctly recognize an intrusion, those intrusions fail to reach their objectives because of unavailability of required contextual configuration (network-related context and target configuration). In order to verify relevance of an alert, we inspect targeted vulnerabilities of the alert using the information extracted from context and vulnerability ontologies. If the targeted vulnerabilities of the alert were not related to the underlying context, the alert is reported as a non-relevant alert. For example, suppose that there is an alert reported by Snort IDS which targets CVE-2012-2531 vulnerability. This vulnerability is a password disclosure vulnerability of Microsoft Internet Information Services (IIS). In case that Microsoft IIS is not used in the underlying context, this alert is reported as a non-relevant alert. Rules 4.1 and 4.2 written in SQWRL within the correlation engine of *Pasargadae* perform Non-relevant alerts elimination based on the target system vulnerabilities:

$$\texttt{ALERT(?a)} \wedge \texttt{HOST(?h)} \wedge \texttt{OS(?o)} \wedge \texttt{VULNERABILITY(?v)} \wedge \texttt{CLASSIFICATION(?cl)} \wedge \texttt{REFERENCE(?ref)} \wedge \texttt{hasTarget(?a,?h)} \wedge \texttt{hasClassific(?a,?cl)} \wedge \texttt{hasOS(?h,?o)} \wedge \texttt{hasReference(?c1,?ref)} \wedge \texttt{hasVulnerability(?o,?v)} \wedge \texttt{hasName(?ref,?n1)} \wedge \texttt{hasName(?v,?n2)} \wedge \texttt{stringEqual(?n1,?n2)} \longrightarrow \texttt{sqwrl:select(?a)} \tag{4.1}$$

$$\texttt{ALERT(?a)} \wedge \texttt{HOST(?h)} \wedge \texttt{APP(?ap)} \wedge \texttt{VULNERABILITY(?v)} \wedge \texttt{CLASSIFICATION(?cl)} \wedge \texttt{REFERENCES(?ref)} \wedge \texttt{hasTarget(?a,?h)} \wedge \texttt{hasClassific(?a,?cl)} \wedge \texttt{hasApp(?h,?ap)} \wedge \texttt{hasReference(?c,?ref)} \wedge \texttt{hasVulnerability(?ap,?v)} \wedge \texttt{hasName(?ref,?n1)} \wedge \texttt{hasName(?v,?n2)} \wedge \texttt{stringEqual(?n1,?n2)} \longrightarrow \texttt{sqwrl:select(?a)} \tag{4.2}$$

2. *Meta-events creation.* After eliminating non-relevant alerts, in this phase, first we classify the events into various groups based on their attributes similarity. Next, for every group of events, we generate an appropriate meta-event. We create separate meta-events for host-based and network-based events because the event attributes are different in each type. The structure of meta-events, as Figure 4.6 illustrates, contains 3 major parts:

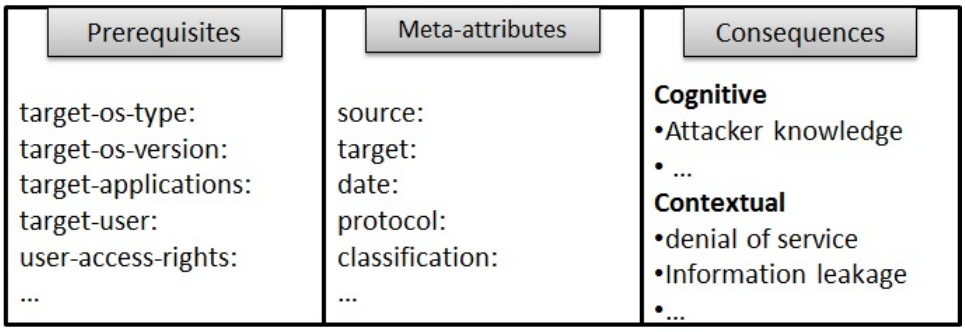

Figure 4.6 Meta-event structure

— *Prerequisites*: This part indicates the conditions that should be satisfied by a computer system or a computer network for an attack to succeed. This information can be extracted mostly from the context ontology. For example, for a telnet

related attack, the telnet service should be listening on the target host on the targeted port.

— *Meta − attributes*: This part includes any lower level event attributes such as source, target, time, classification, etc., and a number of new attributes such as meta-event ID, name and description.

— *Consequences*: This part indicates the impacts of a successful execution of an attack. These consequences can be physical effects that change contextual status of a system, or cognitive effects, which improve attacker knowledge. For example, a TCP scan has cognitive effects because it improves the attacker's knowledge of the underlying context.

Based on this structure, we correlate network-based and host-based events to create meta-events. For network-based events, we mainly use *source* and *target* of the events as their main attributes in this level of class hierarchy. For this purpose, we consider two major steps in order to correlate them:

— *One-to-one-event correlation*: we correlate those malicious events having same source, and same target in a specified time period. Our goal is to correlate events caused by an attacker testing multiple exploits or using the same attack tool multiple times against the same target. Brute force password cracking and blind SQL injection are example of such tools. A SQWRL rule to correlate such events can be:

$$\begin{aligned}&\texttt{NetEvent(?ne)} \wedge \texttt{Host(?h1)} \wedge \texttt{Host(?h2)} \wedge \texttt{hasSource(?ne, ?h1)} \wedge \\ &\texttt{hasTarget(?ne, ?h2)} \wedge \texttt{hasTime(?ne, ?t)} \wedge \texttt{biggerThan(?t, ?t1)} \wedge \\ &\texttt{lessThan(?t, ?t2)} \longrightarrow \texttt{sqwrl:select(?ne)}\end{aligned} \tag{4.3}$$

In this rule, $t1$ and $t2$ are time thresholds to correlate events happening within a specific time period.

— *Many-to-one and one-to-many event correlation*: we correlate those events originating from multiple sources against a single victim or events having a single source attacking multiple victims in a specified time period. These events are replaced by a meta-event including the sources and targets involved in the attack. Distributed denial of service attack (DDOS), spam distribution, and wide network scanning are examples of malicious scenarios which would produce this kind of events. Example of SQWRL rules to correlate such events can be:

$$\begin{aligned}&\texttt{NetEvent(?ne)} \wedge \texttt{Host(?h)} \wedge \texttt{hasTarget(?ne, ?h)} \wedge \\ &\texttt{hasTime(?ne, ?t)} \wedge \texttt{biggerThan(?t, ?t1)} \wedge \texttt{lessThan(?t, ?t2)} \\ &\longrightarrow \texttt{sqwrl:select(?ne)} \wedge \texttt{sqwrl:count(?ne)}\end{aligned} \tag{4.4}$$

$$\begin{aligned}&\texttt{NetEvent(?ne)} \wedge \texttt{Host(?h)} \wedge \texttt{hasSource(?ne, ?h)} \wedge \\ &\texttt{hasTime(?ne, ?t)} \wedge \texttt{biggerThan(?t, ?t1)} \wedge \texttt{lessThan(?t, ?t2)} \\ &\longrightarrow \texttt{sqwrl:select(?ne)} \wedge \texttt{sqwrl:count(?ne)}\end{aligned} \tag{4.5}$$

Regarding host-based events, we need to recognize that those events that indicate the same malicious behavior. For this purpose, we consider the following host-based event attributes:

— Node (N): Information about the host or device that appears to be causing the events (network address, network name, etc.).
— User (U): Information about the user that appears to be causing or is involved the event(s).
— Process (P): Information about the process that appears to be causing the event(s).
— Service (S): Information about the network service involved in the event(s).
— File (F): Information about file(s) involved in the event(s).

Based on these attributes we correlate those host-based events that have same node, user and process ($[N, U, P]$) or same node, user and file ($[N, U, F]$) within a specified time period. Example of SQWRL rules to correlate such events can be:

$$\begin{aligned}&\texttt{HostEvent(?he)} \wedge \texttt{Node(?n)} \wedge \texttt{User(?u)} \wedge \texttt{Process(?p)} \wedge \\ &\texttt{hasNode(?he, ?n)} \wedge \texttt{hasUser(?he, ?u)} \wedge \texttt{hasProcess(?he, ?p)} \wedge \\ &\texttt{hasTime(?he, ?t)} \wedge \texttt{biggerThan(?t, ?t1)} \wedge \texttt{lessThan(?t, ?t2)} \\ &\longrightarrow \texttt{sqwrl:select(?he)} \wedge \texttt{sqwrl:count(?he)}\end{aligned} \tag{4.6}$$

$$\begin{aligned}&\texttt{HostEvent(?he)} \wedge \texttt{Node(?n)} \wedge \texttt{User(?u)} \wedge \texttt{File(?f)} \wedge \\ &\texttt{hasNode(?he, ?n)} \wedge \texttt{hasUser(?he, ?u)} \wedge \texttt{hasFile(?he, ?f)} \wedge \\ &\texttt{hasTime(?he, ?t)} \wedge \texttt{biggerThan(?t, ?t1)} \wedge \texttt{lessThan(?t, ?t2)} \\ &\longrightarrow \texttt{sqwrl:select(?he)} \wedge \texttt{sqwrl:count(?he)}\end{aligned} \tag{4.7}$$

3. *Attack reconstruction.* After preparing meta-events, we correlate them to build attack scenarios. For this purpose, we create a graph of the meta-events representing every

attack scenario. In order to create meta-event graphs, we use a modified version of the topological sort algorithm [41]. In this algorithm, we define a factor called *pre-degree* as the main criterion to sort meta-events. The *Pre-degree* of each meta-event is computed based on the prerequisites, consequences and date/time attributes of the meta-event. For example, if the consequences of two meta-events are the prerequisites of a meta-event m regarding their time consistencies, m's pre-degree will be 2. Generally, the algorithm has two steps:

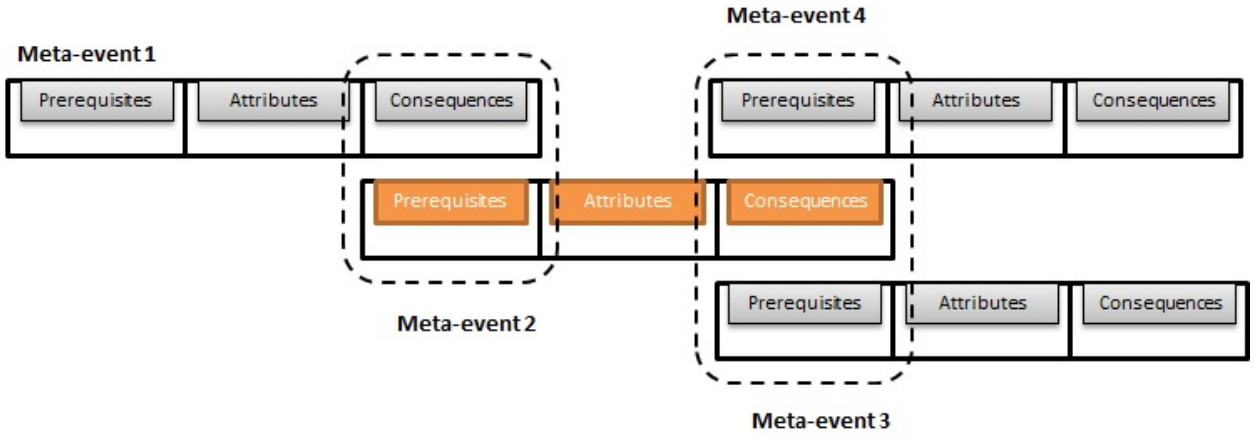

Figure 4.7 An example of meta-event graph to reconstruct attack scenarios

(a) Computing the *pre-degree* of all the meta-events and storing them in an array.

(b) Outputing the meta-events having *pre-degree* zero, and updating the *pre-degree* of the remaining meta-events.

Figure 4.7 illustrates the graphical representation of the building of an attack scenario based on these steps. the consequences part of Meta-event 2 constitutes the prerequisites parts of the Meta-event 3 and Meta-event 4. The *pre-degree* of Meta-event 1 is zero, and the *pre-degree* of Meta-events 2, 3 and 4 are 1.

The above steps are repeated until the meta-event list gets empty. The resulting graph corresponds to the attack scenario. The pseudocode for the proposed event correlation approach is presented in Algorithm 2. Once the attack scenarios are built, the outcome can be presented to the network security analysts.

Figure 4.8 illustrates how the proposed event correlation approach maps to the designed ontologies of the *Pasargadae* framework. As the figure illustrates, in the first step (non-relevant alerts elimination) and second step (meta-event creation), only three of the four ontologies

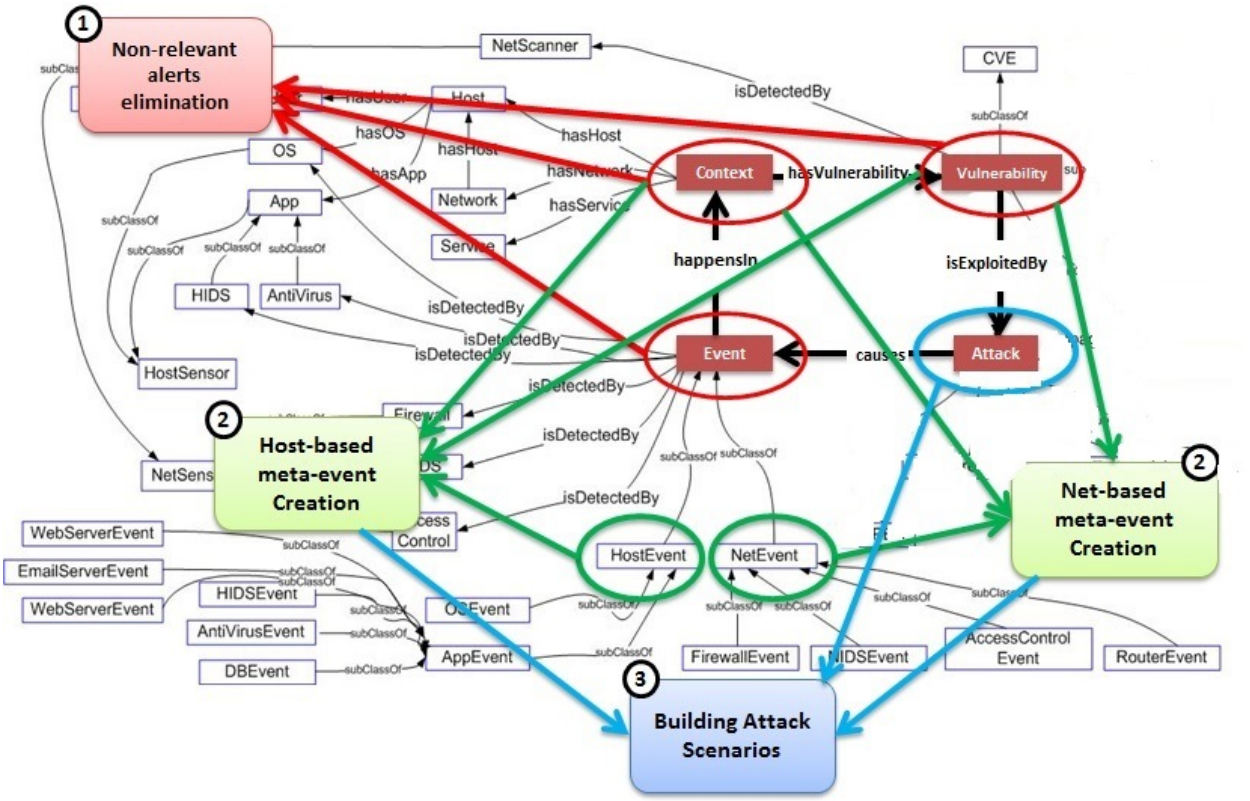

Figure 4.8 Mapping between the attack scenario reconstruction and the designed ontologies

(event, context and vulnerability) are involved. In the last step (building attack scenarios), the information of the attack ontology is applied to the results of the previous steps to reconstruct attack scenarios. The outcome of this process should hopefully provide the security analyst with a reduced list of high level descriptions of potential ongoing (or completed) attacks that includes few redundancies, non relevant scenarios and false positives.

In order to implement both components of this alert correlation approach we use a set of logic rules expressed in Semantic Web Rule Language (SWRL) and Semantic Query-Enhanced Web Rule Language (SQWRL). While various specific correlation approaches could be implemented within the above generic model, we use certain aspects of the approach described in [177] to illustrate the use of our framework.

4.3 *ONTIDS* Alert Correlation Framework as a Subset of *Pasargadae*

As described earlier, the alert correlation and the event correlation are two distinct concepts. The difference originates from the type of sensors that each one employs as their event log

generators. In the alert correlation process, only alerts generated by various IDS are taken into the analysis. However, the event correlation process analyzes not only IDS alerts but also event logs generated by other sensors, such as firewalls, routers, OS, antiviruses, web servers, etc. We consider the alert correlation as a specific version, or as a subset of the event correlation process. It means, the event correlation is more generic and more powerful than the alert correlation process because it has detailed view on every event happening on a computer network.

In this section, we describe how we can use a specific version of the *Pasargadae* framework to build a context-aware and ontology-based alert correlation framework, called *ONTIDS*. In the *ONTIDS* framework, instead of the generic *event ontology*, we define and develop an *alert ontology* which includes only IDS alerts as its instances that can be considered as a subset of the generic *event ontology*. This is the main difference between *Pasargadae* and *ONTIDS* frameworks. Therefore, in Figure 4.1, instead of *Heterogenous Event Sensors*, *ONTIDS* has *Heterogenous Alert Sensors*, instead of *Event Integration*, *ONTIDS* has *Alert Integration*, instead of *Event Correlation Unit*, *ONTIDS* has *Alert Correlation Unit*, and finally, instead of *Event Ontology*, *ONTIDS* has *Alert Ontology*.

Figure 4.9 illustrates the class relationship of the *ONTIDS* ontologies. As the figure illustrates, the class *Alert* has two subclasses *HostAlert* and *NetAlert* which includes NIDS and HIDS alerts. During the alert correlation process, these alerts are collected and correlated with the information existing in the context, vulnerability and attack ontologies within the correlation engine.

In the following subsection, in order to show significant flexibility of the *ONTIDS* framework in implementing various alert correlation approaches, we use it to implement various steps of the alert correlation approach proposed in [177].

4.3.1 Example Implementation of Valeur Approach Using *ONTIDS*

The alert correlation approach proposed in [177] includes a comprehensive set of steps that covers various aspects of an alert correlation process. In order to show how *ONTIDS* is able to automatically implement these steps, in the following, we explain the implementation details of some of these steps using *ONTIDS*.

— **Alert fusion.** Alert fusion is the process of merging alerts that represent the independent detection of the same malicious event by different IDS. It helps to reduce duplicate alerts and false positives. An important condition in order to fuse two or more alerts is that they should be reported in a same time window by different alert

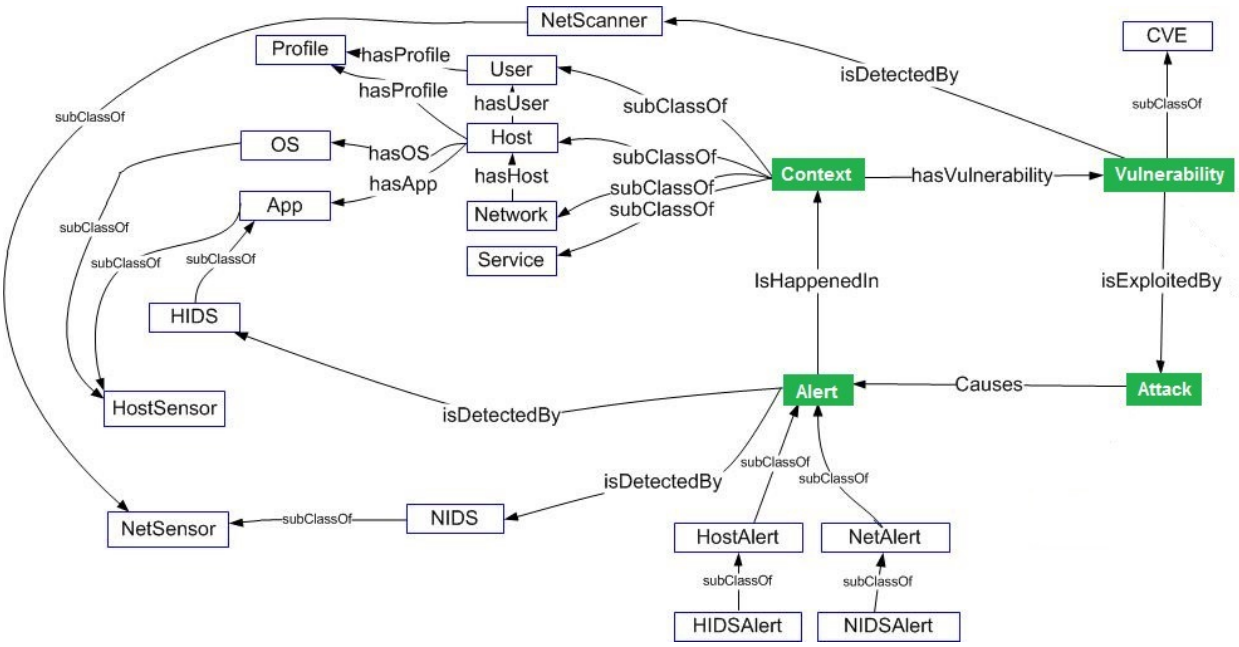

Figure 4.9 Class diagram relationship of the *ONTIDS* ontologies

sensors. We have defined Rule 4.8 within the correlation engine in order to perform alert fusion:

$$
\begin{aligned}
&\texttt{ALERT(?a1)} \wedge \texttt{ALERT(?a2)} \wedge \texttt{ANALYSER(?an1)} \wedge \texttt{ANALYSER(?an2)} \wedge \\
&\texttt{DetectTime(?dt1)} \wedge \texttt{DetectTime(?dt2)} \wedge \texttt{SOURCE(?s1)} \wedge \texttt{SOURCE(?s2)} \wedge \\
&\texttt{TARGET(?tar1)} \wedge \texttt{TARGET(?tar2)} \wedge \texttt{CLASSIFICATION(?cl1)} \wedge \\
&\texttt{CLASSIFICATION(?cl2)} \wedge \texttt{ASSESSMENT(?as1)} \wedge \texttt{ASSESSMENT(?as2)} \wedge \\
&\texttt{stringEqual(?s1,?s2)} \wedge \texttt{stringEqual(?tar1,?tar2)} \wedge \\
&\texttt{stringEqual(?cl1,?cl2)} \wedge \texttt{stringEqual(?as1,?as2)} \wedge \\
&\texttt{subtractTimes(?td,?dt1,?dt2)} \wedge \texttt{lessThan(?td,"5s")} \\
&\longrightarrow \texttt{sqwrl:select(?a1)}
\end{aligned}
\tag{4.8}
$$

The Rule 4.8 lists all the alerts having same attributes but generated by different IDS in a same time window.

— **Alert Verification**. Alert verification is the process of recognising and reducing non-relevant alerts which refer to the failed attacks. The major reason of attack failure is the unavailability of the contextual requirements of the attack, i.e. the

absence of required vulnerabilities in the attack context. Identifying failed attacks allows the correlation engine to reduce the effects of non-relevant alerts in its decision process. Rules 4.9 and 4.10 within the correlation engine of *Pasargadae* perform alert verification based on the target system vulnerabilities:

$$\begin{aligned}&\texttt{ALERT(?a)} \wedge \texttt{HOST(?h)} \wedge \texttt{OS(?o)} \wedge \texttt{VULNERABILITY(?v)} \wedge \texttt{CLASSIFICATION(?cl)} \wedge \\ &\texttt{REFERENCE(?ref)} \wedge \texttt{hasTarget(?a,?h)} \wedge \texttt{hasClassific(?a,?cl)} \wedge \texttt{hasOS(?h,?o)} \\ &\wedge \texttt{hasReference(?c1,?ref)} \wedge \texttt{hasVulnerability(?o,?v)} \wedge \texttt{hasName(?ref,?n1)} \wedge \\ &\texttt{hasName(?v,?n2)} \wedge \texttt{stringEqual(?n1,?n2)} \\ &\longrightarrow \texttt{sqwrl:select(?a)}\end{aligned} \tag{4.9}$$

$$\begin{aligned}&\texttt{ALERT(?a)} \wedge \texttt{HOST(?h)} \wedge \texttt{APP(?ap)} \wedge \texttt{VULNERABILITY(?v)} \wedge \texttt{CLASSIFICATION(?cl)} \\ &\wedge \texttt{REFERENCES(?ref)} \wedge \texttt{hasTarget(?a,?h)} \wedge \texttt{hasClassific(?a,?cl)} \wedge \\ &\texttt{hasApp(?h,?ap)} \wedge \texttt{hasReference(?c,?ref)} \wedge \texttt{hasVulnerability(?ap,?v)} \wedge \\ &\texttt{hasName(?ref,?n1)} \wedge \texttt{hasName(?v,?n2)} \wedge \texttt{stringEqual(?n1,?n2)} \\ &\longrightarrow \texttt{sqwrl:select(?a)}\end{aligned} \tag{4.10}$$

The Rules 4.9 and 4.10 verify whether the targeted OS or application vulnerabilities exist in the underlying context. The only difference between these two rules is the functional properties *hasOS* and *hasApp*.

— **Attack thread reconstruction.** Attack thread reconstruction is the process of merging a series of alerts that refer to an attack launched by one attacker against a single target, and is another step in the alert correlation process of [177]. Similarly to the alert fusion process, the alerts should happen in the same time window to be correlated. Rule 4.11 performs the attack thread reconstruction task:

$$\begin{aligned}&\texttt{ALERT(?a)} \wedge \texttt{HOST(?h1)} \wedge \texttt{HOST(?h2)} \wedge \texttt{TIME(?t1)} \wedge \texttt{TIME(?t2)} \wedge \\ &\texttt{hasSource(?a,?h1)} \wedge \texttt{hasTarget(?a,?h2)} \wedge \texttt{hasDetectTime(?a,?dt)} \wedge \\ &\texttt{greaterThanOrEqual(?dt,?t1)} \wedge \texttt{lessThanOrEqual(?dt,?t2)} \\ &\longrightarrow \texttt{sqwrl:select(?a,?h1,?h2)}\end{aligned} \tag{4.11}$$

Rule 4.11 lists all the alerts having same source and same target, and generated in a

same time window.

— **Attack session reconstruction.** Attack session reconstruction is the process of linking network-based alerts to the related host-based alerts, and constructing attack scenarios. As network-based alerts has different attributes compared to host-based alerts, linking these alerts together is a difficult task. Two factors that can provide clues to correlate these alerts are: time and port (process). Hence, we present the following rule (4.12) based on these factors to do attack session reconstruction task:

$$\begin{aligned}&\texttt{NetAlert (?na)} \wedge \texttt{HostAlert(?ha)} \wedge \texttt{Node(?n)} \wedge \texttt{Process(?p)} \wedge \\ &\texttt{hasTarget(?na, ?n)} \wedge \texttt{hasNode(?ha, ?n)} \wedge \texttt{hasProcess (?na, ?p)} \wedge \\ &\texttt{hasProcess(?ha, ?p)} \wedge \texttt{hasTime(?na, ?t1)} \wedge \texttt{hasTime(?ha, ?t2)} \wedge \\ &\texttt{biggerThan(?t1, ?tMin)} \wedge \texttt{biggerThan(?t2, ?tMin)} \wedge \texttt{lessThan(?t1, ?tMax)} \wedge \\ &\texttt{lessThan(?t2, ?tMax)} \\ &\longrightarrow \texttt{sqwrl:select(?na,?ha)}\end{aligned} \tag{4.12}$$

Rule 4.12 lists all the related network-based and host-based alerts generated by NIDS and HIDS in a same time window.

In summary, we can see how the first component of our canonical description is implemented by the correlation engine by applying Rules 4.9 and 4.10 to reduce non-relevant alerts. For this purpose, it retrieves required information from the alert, context, and vulnerability ontologies. Next, and for those alerts and scenarios that are relevant, attack thread reconstruction and attack thread reconstruction are performed by applying Rules 4.11 and 4.12, where the engine attempts to make a mapping between the filtered alerts and the steps of attacks in the attack ontology. Once it finds any mapping between the two ontologies, it will output the whole attack scenario.

4.4 Alert Fusion Using *Pasargadae* Framework

Most current IDS cannot by themselves adequately detect a large number of attacks. Each IDS, based on its detection algorithm, is mostly able to detect a particular group of attacks. For example, signature-based IDS are only able to detect known attacks, but not new attacks. However, anomaly-based IDS are able to detect new attacks even though they generate higher false positives. One of the solutions that has been proposed to improve detection rate is combining or *fusing* the alerts generated from several different IDS examining the same data source. In its generic definition, sensor fusion is the process of combining the output of

several different sensors observing the same data source, and then making a decision based on this combined output. The data source is typically a stream of events, with each event possibly triggering an alert by each of the sensors.

Current alert fusion approaches generate large numbers of redundant and non-relevant alerts, thus exacerbating the false positive problem. They do not take into account the semantical relationship and the contextual information of happening events to perform more appropriate analysis. In order to address these shortcomings, in this section, we propose a semantic-based and context-aware alert fusion approach that employs *Pasargadae* framework as its main bed for the alert fusion purpose. The proposed approach incorporates contextual information in order to reap the benefits of multi-sensor detection while reducing false positives.

In the rest of this chapter, first, we describe the steps of the proposed alert fusion approach. Next, we describe in detail the decision making component that is the main component of the proposed alert fusion approach.

4.4.1 The Proposed Alert Fusion Approach

Now, we describe our proposed semantic-based and context-aware alert fusion approach. The proposed alert fusion approach employs *Pasargadae*'s components to perform its task. In the following, we describe this approach with more details.

Figure 4.10 illustrates the proposed alert fusion approach and its components. This approach employs alert ontology which is a subset of *Pasargadae*'s event ontology, context ontology which provides contextual information of the underlying network, and vulnerability ontology. The steps of the alert fusion process are as follows:

1. In this step, all the alerts generated by various IDS (alert sensors) that monitor same event steams in the underlying network, are collected into the alert integration component. We can employ different IDS having different detection methods, such as anomaly-based IDS (e.g. Bro [136]) and signature-based IDS (e.g. Snort [144]). In the same time, contextual information mostly generated by context sensors (CS) and provided by system administrators are collected into context integration component.
2. This step consists of two phases:
   (a) The integration and conversion of all alerts generated by various IDS into a unified format analyzable by the alert fusion unit.
   (b) The integration of all contextual information gathered implicitly via network administrators or explicitly via context sensors (CS). Today, the most popular format for integrating alerts from various IDS is the Intrusion Detection Message

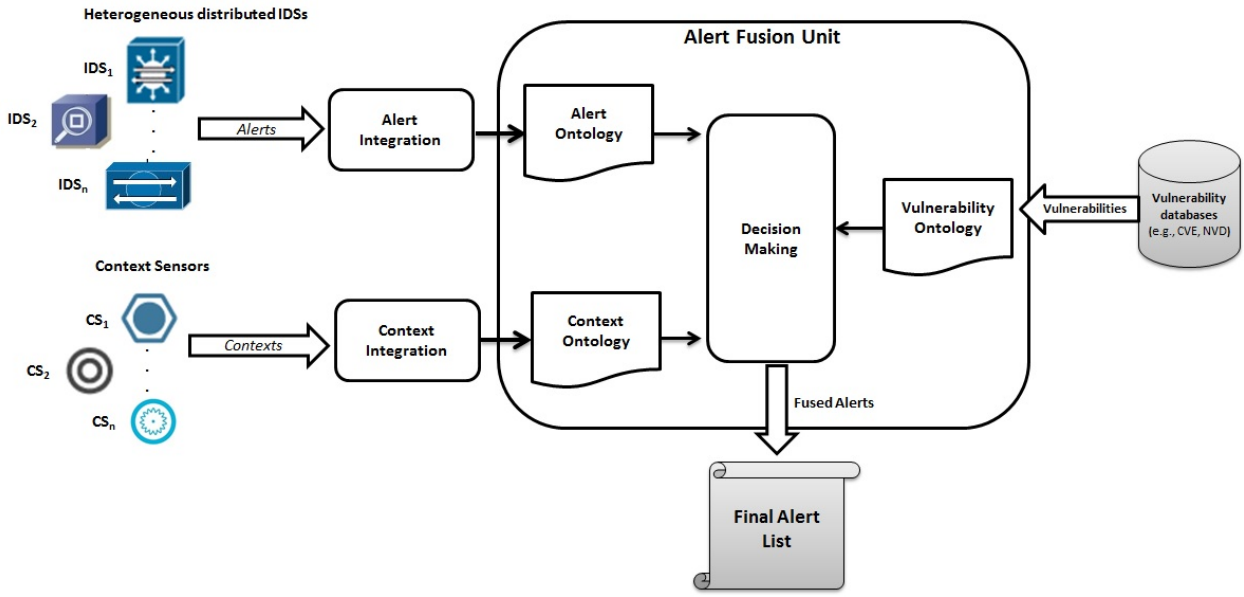

Figure 4.10 The proposed semantic-based context-aware alert fusion model

Exchange Format (IDMEF) [50] proposed by Internet Engineering Task Force (IETF). We use IDMEF in the alert integration component as the alert format. Figure 4.11 represents the IDMEF alert attributes.

3. This step consists of populating the ontologies with the integrated data from Step 2. In order to fully automate the alert fusion process, we designed a group of basic but extensible ontologies for alerts, context information, and vulnerabilities. These ontologies enable sharing and reasoning on the information gathered from various resources. Figure 4.12 illustrates the conceptual relationship among the ontologies and Figure 4.13 describes the class diagram relationship among the ontologies.
4. The key step is the decision making process, where contextual information is combined with alert information and vulnerabilities to filter out non-relevant alerts and false positives, and a final decision is made to report redundant alerts based on the same malicious event as a single alert instance.

As most of the components involved in the proposed alert fusion approach were described in detail in section 4.1, in the rest of this section, we only describe the decision making component of the proposed alert fusion approach.

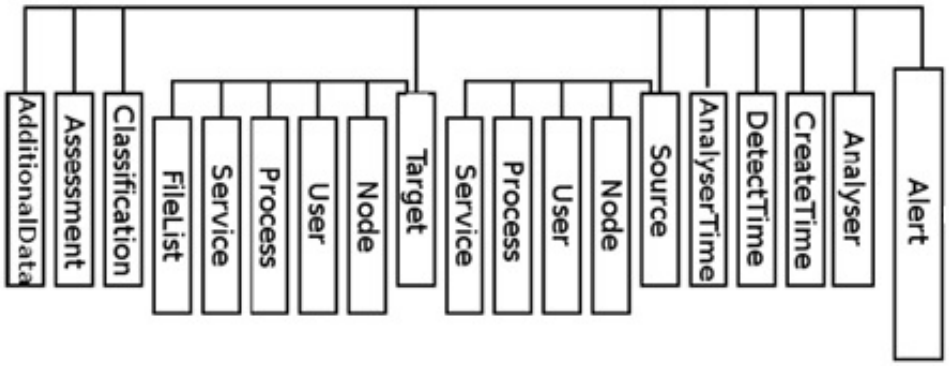

Figure 4.11 The IDMEF alert attributes [50]

Figure 4.12 The conceptual relationships among the proposed ontologies

4.4.2 The Decision Making Component

We now describe the decision making component of the proposed alert fusion approach. The outcome of this component provides the security analyst with a reduced list of alerts that hopefully includes mostly accurate alerts with fewer non-relevant ones and false positive.

We assume that N different IDS monitor the underlying system, and based on every single event e, each intrusion detection system IDS_i makes a decision $d_i(e)$. This decision can be alert or no-alert based on the predefined detection policies of IDS_i:

$$d_i(e) = \begin{cases} 1, & \textit{attack (alert)} \\ 0, & \textit{normal (no-alert)} \end{cases}$$

For every single event e, the alert fusion unit gathers all of the decisions from the different IDS along with the information provided from other resources, i.e. context sensors and vulnerability databases, and analyses them to make an appropriate final decision, alert or no-alert. Figure 4.14 graphically represents this process. In the following, we describe the decision making component of the alert fusion unit.

The decision making component essentially considers three conditions for each event e:

1. If all sensors report event e as a normal event, the decision making component adds

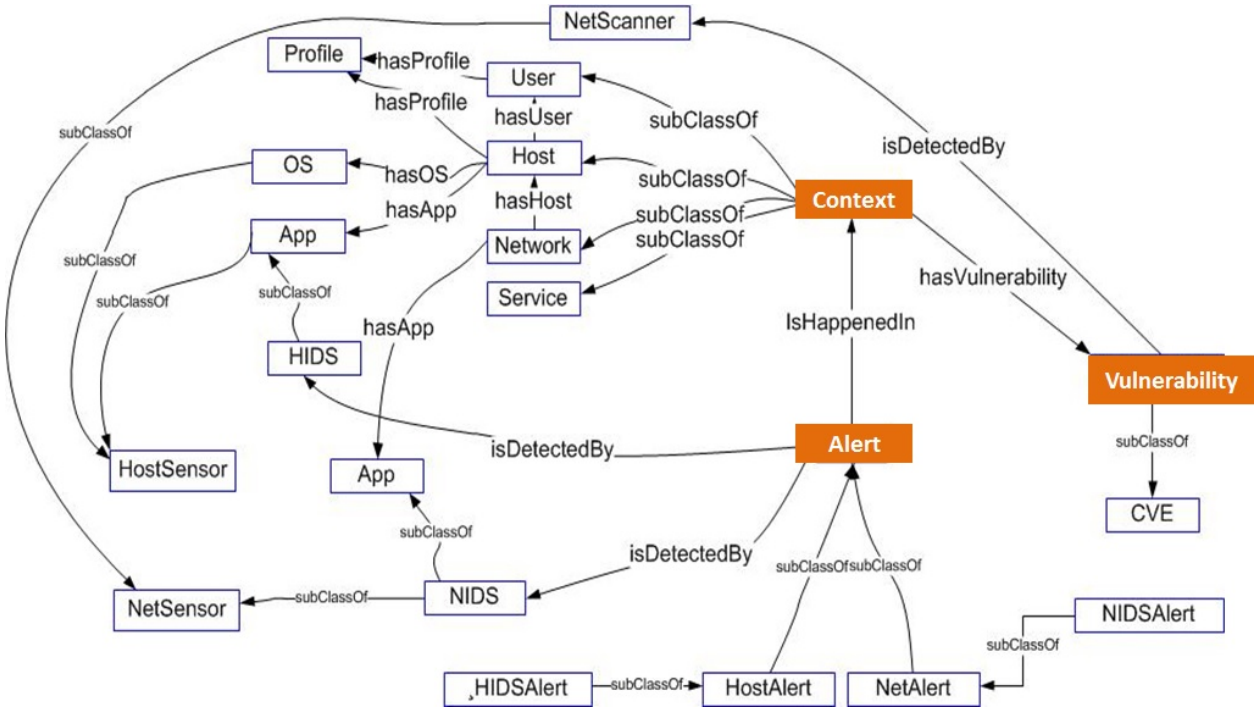

Figure 4.13 The class diagram of relationships among the proposed ontologies

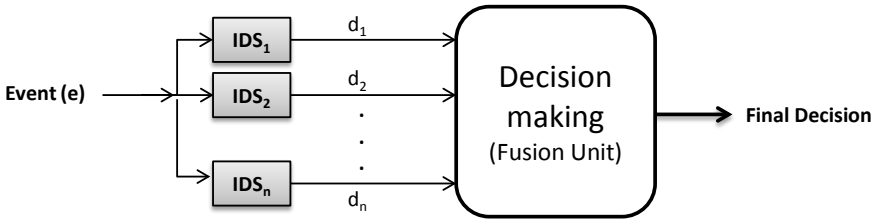

Figure 4.14 The alert fusion process.

it to the normal events list.

2. If all sensors report event e as an attack, the decision making component adds it to the attack events list.

3. If some sensors report event e as normal, and some report it as an attack, the decision making component employs the gathered contextual and vulnerability information to inspect the prerequisites and consequences of the event e in order to make an appropriate decision.

For the first two conditions, the decision making component trusts the decisions of all the IDS since they all generate the same alert for the same event e. In other words, in these two scenarios, there is no need for further analysis (additional complexity) by the decision

making component.

On the other hand, when the third condition occurs, our alert fusion approach kicks in. The decision making component of our approach first analyzes the characteristics of event e such as its source and destination IP addresses, targeted port, involved applications, exploited vulnerabilities, etc. Based on these characteristics, the decision making component retrieves required information from alert, context and vulnerability ontologies in order to make an appropriate final decision.

The key idea here is that as an attack progresses, it will generate a sequence of events $e_1, e_2, \ldots, e_n$ that might be detectable by the alert sensors. Normally, each attack event e_i will need specific preconditions to be met for it to occur, and if it occurs it will have certain effects on the attacked system. These preconditions and effects can be contextual or cognitive. Contextual preconditions of an event are the tangible conditions in the underlying system for that part of the attack to succeed. For example, for an ssh related attack, the ssh service should be active on the target hosts. Contextual effects are any tangible modification to the underlying system. For example, an event which is part of a SQL injection attack modifies a portion of data in the target database. On the other hand, cognitive requirements and effects are all those non-tangible conditions that are mostly related to the attacker's knowledge. For instance, an attacker who wants to start an ssh related attack, should know whether the ssh service is available in the target network or not, which is a cognitive requirement of the ssh attack. As such, a network vulnerability scanning event, which is part of the reconnaissance phase of typical attack scenarios, can be considered as a cognitive effect.

In essence, the decision making component considers for each event e the following two essential criteria:

1. **Event prerequisites**. The conditions that should be satisfied by system or network for the attack event e to succeed. These conditions can be both contextual which is related to the underlying context, and cognitive which is related to the attacker's knowledge. As an example, for the ssh related attacks, the appropriate context is when ssh service is listening on the target host to the ssh port (i.e. 22), and the cognitive condition is the attacker's knowledge about the existence of ssh services underlying context.
2. **Event consequences**. The consequences indicate the impacts of the successful execution of an attack event e. These consequences can be physical effects that change the contextual status of a system, or cognitive effects that improve the attacker's knowledge. For instance, a TCP scan has cognitive effects since it improves the attacker's knowledge of the underlying context (i.e. port scanning).

The decision making component first constructs the event's prerequisites based on the information retrieved from the ontologies using a number of pre-defined rules. For instance, suppose that there is an alert reported by Snort IDS which targets CVE-2012-2531 vulnerability. This vulnerability is a password disclosure vulnerability of the Microsoft Internet Information Services (IIS). The following rule, defined using the Semantic Web Rule Language (SWRL), extracts required information in order to constitute the event's prerequisites in order to check whether there are any vulnerable assets in the underlying system.

$$\begin{aligned}&\texttt{ALERT(?a)} \wedge \texttt{HOST(?h)} \wedge \texttt{APPLICATION(?ap)} \wedge \texttt{VULNERABILITY(?v)} \wedge \texttt{CLASSIFICATION(?c)} \\ &\wedge \texttt{REFERENCES(?r)} \wedge \texttt{hasTarget(?a,?h)} \wedge \texttt{hasClassification(?a,?c)} \wedge \\ &\texttt{hasApplication(?h,?ap)} \wedge \texttt{hasReference(?c,?r)} \wedge \texttt{hasVulnerability(?ap,?v)} \wedge \\ &\texttt{hasName(?r,"CVE-2012-2531")} \wedge \texttt{hasName(?v,?n)} \wedge \\ &\texttt{swrlb:stringEqualIgnoreCase("CVE-2012-2531",?n)} \longrightarrow \texttt{sqwrl:select(?a)}\end{aligned} \tag{4.13}$$

In addition to event prerequisites, the decision making component inspects the event consequences on the underlying network. If the event consequences provide cognitive or contextual gain for an attacker, the event is considered as a malicious event. For instance, if an event is part of a TCP scanning process which reveals some contextual information for an attacker, the decision making component considers the event as a malicious event. Consequently, the decision making component analyzes both prerequisites and consequences of every single event, and if both are available, the corresponding alert is considered as an attack. The pseudocode for the decision making process of our proposed alert fusion is presented in Algorithm 3.

4.5 Summary

In this chapter, we proposed an ontology-based context-aware event correlation framework, called *Pasargadae*, to overcome the shortcomings of current Intrusion Detection Systems (IDS) and alert correlation technologies. The main idea is to collect and correlate events generated by heterogeneous sensors (e.g., NIDS, firewalls, access control system, operating system, HIDS, antivirus, web server, and other applications) located on several architectural levels of a computer network to improve the systems' detection capability. The output of the system is a high-level description of the existing attack scenarios that is used to reduce false positives and non-relevant alerts. For this purpose, we evaluated contextual information to identify which aspects of context can be useful in improving IDS efficiency, and we proposed

an ontology of context information accordingly, with the goal of being able to import such context information from either explicit information in Configuration Management Systems (CMS) or from implicit information obtained by user and system profiling techniques. In order to allow IDS analysis automation, we designed comprehensive and extensible ontologies for events, vulnerabilities, attacks, and system context using Ontology Web Language (OWL). We designed and presented a number of rules using Semantic Query Web Rule Language (SQWRL) based on description logic (DL) to correlate the information of these ontologies within the correlation engine.

Next, we presented a new semantic-based and context-aware event correlation approach that employs *Pasargadae* as its main framework. The proposed event correlation approach has 3 main steps: non-relevant alerts elimination, meta-event creation, and attack-reconstruction. After eliminating non-relevant alerts, it classifies low-level events generated by heterogeneous sensors, and creates meta-events. Then, based on a modified topological-sort approach, it correlates these meta-events to reconstruct attack scenarios.

In section 4.3, we introduced *ONTIDS* an ontology-based automated alert correlation framework that is considered as a subset or a sub-framework of the *Pasargadae* framework. We described that the ontologies and correlation rules are generic enough to *i*) implement as special cases other existing correlation approaches such as that of Valeur *et al.*, and, *ii*) be applied with minimal changes to different analysis scenarios, such as in the case studies that are demonstrated in next chapters.

In section 4.4, we proposed a semantic-based, context-aware alert fusion approach to try to overcome some of the existing shortcomings of current alert fusion approaches. The proposed alert fusion approach employs *Pasargadae* as its main framework. The main idea of our approach is to collect alerts (decisions) made by different IDS sensors on the same event, and inspect them using contextual information to reduce redundant and non-relevant alarms and false positives. While employing several different IDS sensors having different methods can improve overall detection capabilities, the incorporation of contextual information allows us to reduce false positives and non-relevant alerts. Within the decision making component of the proposed alert fusion approach, based on the decisions received from a number of individual IDS, we inspect prerequisites and consequences of the particular event, and if both are acceptable according to the predefined policies, the event is considered as an attack. Otherwise, the decision making component reports it a normal event. These steps of the decision making component are implemented via a number of SWRL and SQWRL rules.

Pasargadae event correlation framework, compared to the proposed alert correlation framework (*ONTIDS*) and alert fusion framework, is more generic to cover not only IDS

alerts but also other event logs generated via various sensors, such as firewalls, antiviruses, OS, routers, databases, etc. Therefore, it is considered a more powerful and more generic framework than alert correlation and alert fusion frameworks. Furthermore, having more hierarchical levels in the *Pasargadae* framework, allows data analysts to easily drill-down into very specific concepts, or roll-up to cover very generic concepts. Consequently, we believe that significant flexibility to implement various similar approaches, ontological structure to provide system automation, context-awareness, and comprehensiveness of *Pasargadae* makes it powerful enough to be used in most of the real-world computer networks.

Algorithm 2 Semantic-based event correlation pseudocode

INPUT: Event Ontology (E), Context Ontology (C), Vulnerability Ontology (V), Attack Ontology (A).
OUTPUT: Attack Scenarios
BEGIN {Meta-Events Creation}
for all $e \in Event - Class$ **do**
  $Meta - Event - Creation(e)$
end for
{Attack Scenario Reconstruction}
$Attack - Scenario - Reconstruction(meta - event - list)$

Return $Attack - Scenarios$
END

Meta-Event-Creation
INPUT: Event e
OUTPUT: $meta - event - list$
BEGIN
{creating one-to-one, one-to-many and many-to-one meta-events}
if $e \in Net - Event$ **then**
  based on e's date, source and target assign e to an existing meta-event or create a new meta-event m
  $m - pre \leftarrow$ add e's prerequisites
  $m - con \leftarrow$ add e's consequences
end if
{creating meta event based on [node (n), user (u), process (p)] or [node (n), user (u), file (f)]}
if $e \in Host - Event$ **then**
  based on e's time, node, user, process and file assign e to a meta-event or create a new meta-event m regarding the two patterns [n, u, p] or [n, u, f]
  $m - pre \leftarrow$ add e's prerequisites
  $m - con \leftarrow$ add e's consequences
end if

Return $meta - event - list$
END

Attack-Scenario-Reconstruction
INPUT: $meta - event - list$
OUTPUT: $atack - scenario - list$
BEGIN
Compute the $preDegree$ of all meta-events
Find a meta-event m having minimum $preDegree$
Add m to $attack - scenario - list$
Remove m from $meta - event - list$
Update the $preDegree$ of remaining meta-events in the $meta - event - list$
Repeat the above steps while there are meta-events to be processed
Return $attack - scenario - list$
END

Algorithm 3 Decision making pseudocode

INPUT: Alert Ontology (A), Context Ontology (C), Vulnerability Ontology (V), Event List (E).
OUTPUT: Fused Alert List: FAL[]
BEGIN

{Fusing all the alerts generated by IDS for every event $e \in E$}
for all $e \in E$ **do**
  $finalDecision = normal$

  {Gathering decisions generated by IDS}
  $decisionList[1-n] \leftarrow IDSDecision[1-n]$

  {Inspecting the decisions made by IDS}
  if $decisionList[1-n] = normal$ **then**
    $finalDecision = normal$
  else
    if $decisionList[1-n] = attack$ **then**
      $finalDecision = attack$
    else
      $Prerequisites \leftarrow ContextOntology$
      if *Prerequisites provided* **then**
        $consequences \leftarrow ContextOntology$
        if *consequences malicious* **then**
          $finalDecision = attack$
        end if
      end if
    end if
  end if
  $FAL[e] = finalDecision$
end for
Return FAL[]
END

CHAPTER 5 REFERENCE IMPLEMENTATION

In Chapter 4 of this thesis, we proposed a number of new approaches related to the event and alert correlation and fusion. First, we proposed the *Pasargadae* context-aware and ontology-based event correlation framework. Next, we proposed a semantic-based event correlation approach that employs *Pasargadae* as its main framework. Finally, we proposed a semantic-based alert fusion approach based on *Pasargadae*.

In this chapter, we describe how *Pasargadae* which is the main framework of all the proposed approaches, was implemented in our lab and field test environment. For this purpose, we explain various tools and methods that were employed to implement *Pasargadae*'s components.

The rest of this chapter is organized as follows. In Section 5.1, we describe how we implement various parts of the designed ontologies, including their classes, and data type and object properties. In Section 5.2, we explain how the designed ontologies are stored, their querying process using SWRL, SQWRL and SPARQL, and their reasoning capability using Pellet reasoner. In Section 5.3, we describe how the designed ontologies are populated (class instantiation) with various information. Section 5.4 is dedicated to a discussion on the scalability of the proposed implementation. Finally, in Section 5.5, we describe a short summary of this chapter.

5.1 Implementing the Designed Ontologies

In this section, we describe how we implement our designed ontologies. As described in Chapter 4, the proposed event correlation framework consists of four ontologies including event, vulnerability, attack and context ontology. In order to design these ontologies, we use the Ontology Web Language Description Logic (OWL-DL) introduced in Chapter 3. For this purpose, we use the Protégé ontology editor and knowledge acquisition system [68].

Protégé is an open source software developed based on Java programming language. It provides the following features:

- Editing OWL 2 ontologies
- Full change tracking and revision history
- A customizable graphical user interface to define and populate ontologies
- Web-based ontology editor
- Deductive classifiers to validate consistency of an ontology's structure
- Inferring new information via analyzing ontologies

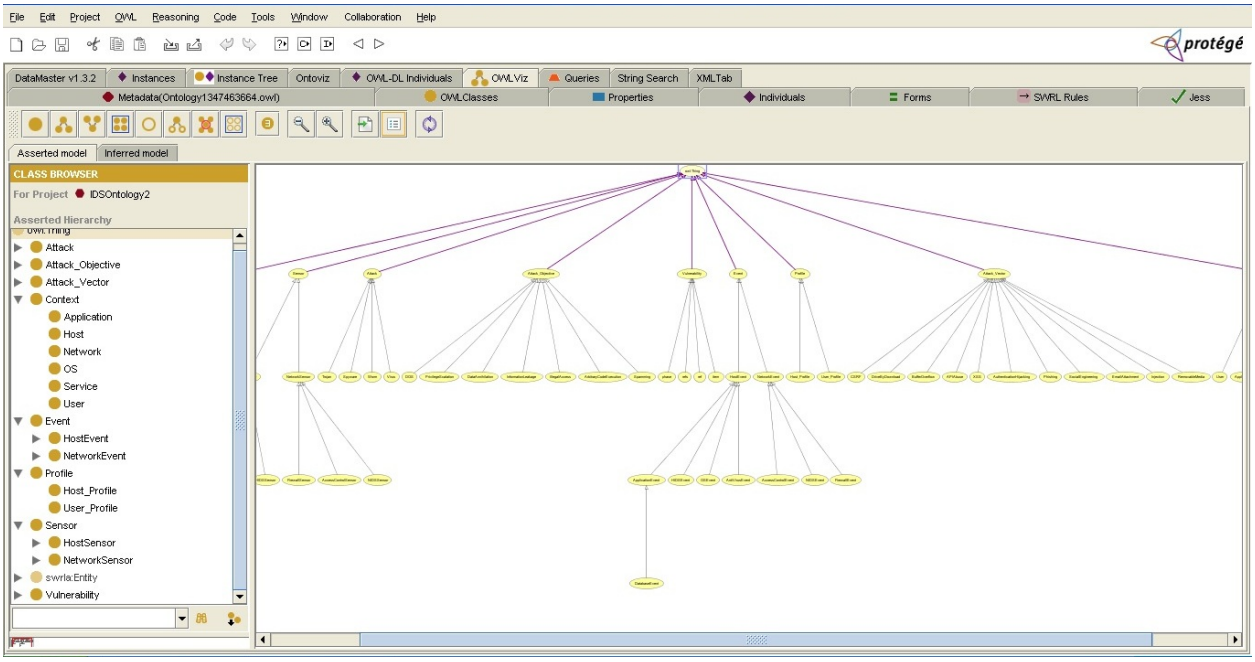

Figure 5.1 Hierarchical class diagram of the designed ontologies

— Editing Open Biological and Biomedical (OBO) Ontologies
— Multiple formats of uploading and downloading ontologies (RDF/XML, Turtle, OWL/XML, OBO, etc.)

Various plug-ins for different usages, such as Importing, Exporting, Inferencing, Reasoning, Querying, Visualization, etc., have been developed to integrate Protégé with other tools. Some of its most popular plug-ins are:

— DataMaster: importing schema structure and data from relational databases
— XML Tab: importing an XML document
— Excel Import: importing content and generating classes from Excel
— SWRL-IQ: editing, saving and submitting queries
— JessTab: provides a reasoner for ontologies
— SWRL Tab: editing and execution of SWRL rules
— SPARQL Query: composing and editing SPARQL queries

Using Protégé, we create the ontologies proposed in Chapter 4 (Figure 4.3) and their related classes, and object and data type properties. Figure 5.1 illustrates the hierarchical class diagram of the designed ontologies within Protégé, and Figure 5.2 illustrates the object properties of the designed ontologies.

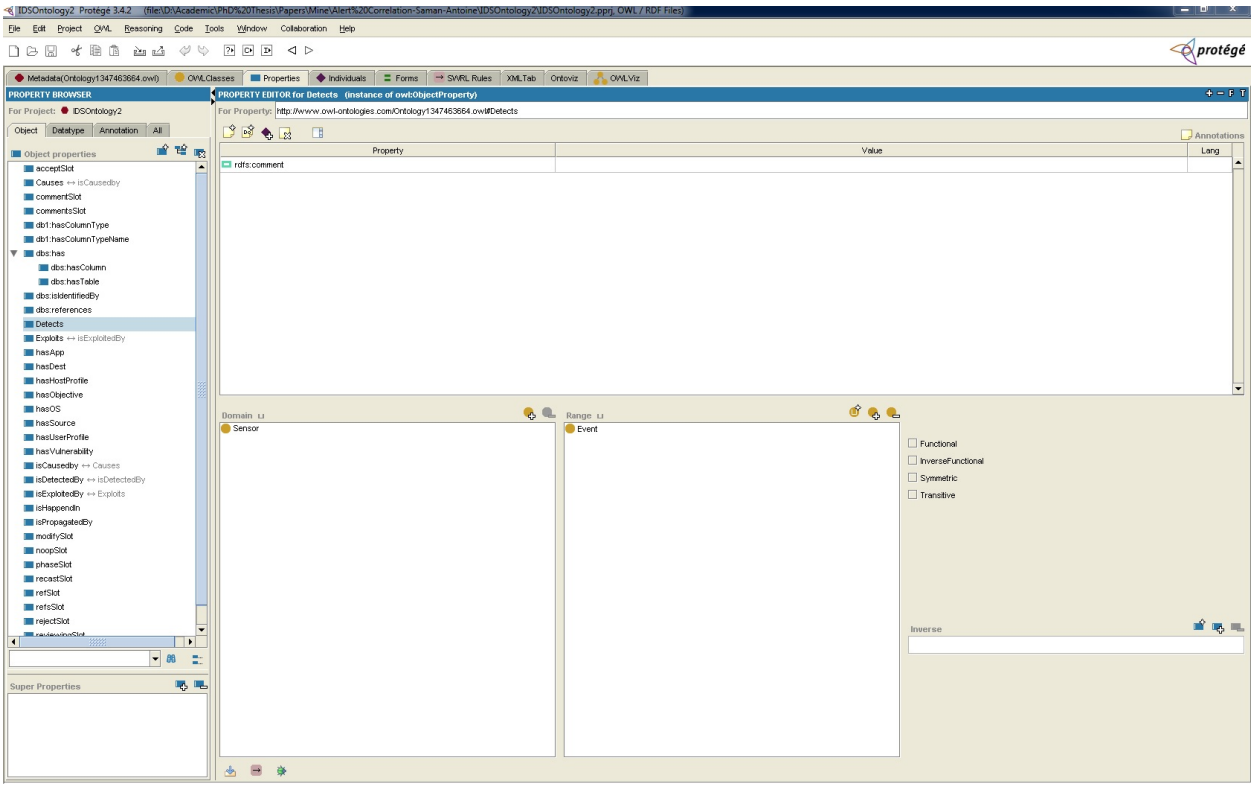

Figure 5.2 Object propoerties of the designed ontologies

5.2 Storing, Reasoning and Querying the Designed Ontologies

In order to store a designed OWL ontology and have persistent information, Protégé provides two types of storage OWL/RDF Databases and OWL/RDF Files. Figure 5.3 illustrates how we can specify the type of storage when creating an OWL-based Ontology. Essentially, the database storage has more advantages than file systems. However, it mostly depends on the volume of information that an ontology includes. As in this research, we analyze a large volume of event logs, the database storage provides better performance.

Protégé ontology editor has a number of reasoning plug-ins. Some of its most popular reasoners are HermiT [150], Pellet [134], NoHR [93], and FaCT++ [171]. In our implementation, we use the Pellet plug-in as a reasoner for OWL-DL. As Figure 5.8 illustrates, it provides sophisticated features, such as data type reasoning, SWRL support, consistency checking, etc.

In order to write and edit queries to correlate various information of the classes, we use Semantic Web Rule Language (SWRL) [87], Semantic Query-Enhanced Web Rule Language (SQWRL) [131], and SPARQL Protocol and RDF Query Language-Description Logic

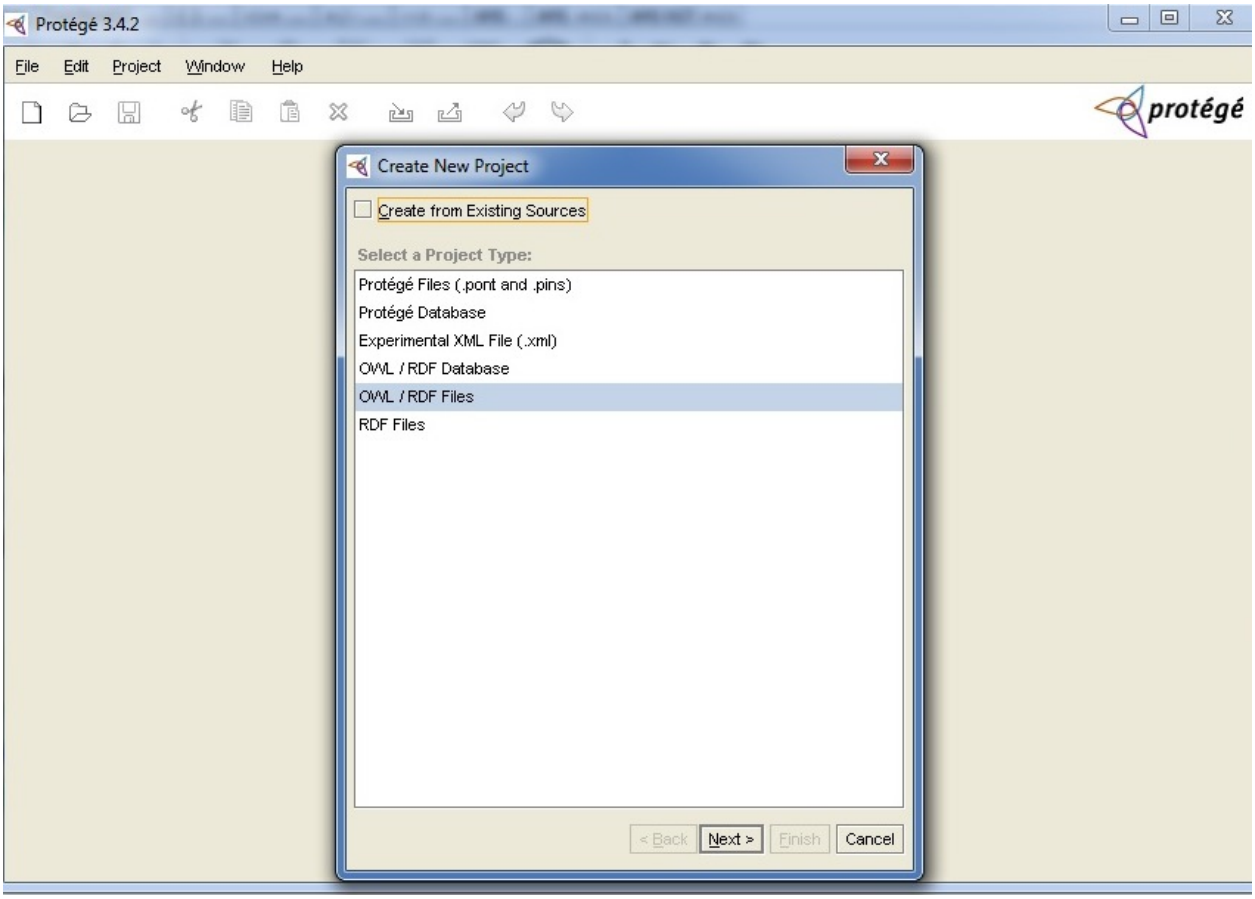

Figure 5.3 How to store an OWL Ontology in Protégé

(SPARQL-DL) [155]. We introduced these languages in Chapter 3 in detail, and we explained how they facilitate drilling-down to cover specific class instances, and rolling-up to cover generic class instances in the class hierarchies of ontologies. The Jess rule engine [69] is employed as SWRL rule compiler and SQWRLTab is employed as SQWRL compiler. Figure 5.5 illustrates SWRL Rles Tab of Protégé, and SQWRL and Jess compiler. After writing a rule, based on its type (SWRL or SQWRL), we can use SQWRL or Jess compilers to run it.

5.3 Populating the Designed Ontologies

In this section, we describe how we populate the designed ontologies (*Event*, *Context*, *Vulnerability* and *Attack*). For this purpose, first, the collected events, contextual information, and attack and vulnerability information are converted to appropriate formats analyzable by Protégé ontology editor. XML, Excel, and relational databases are the most popular

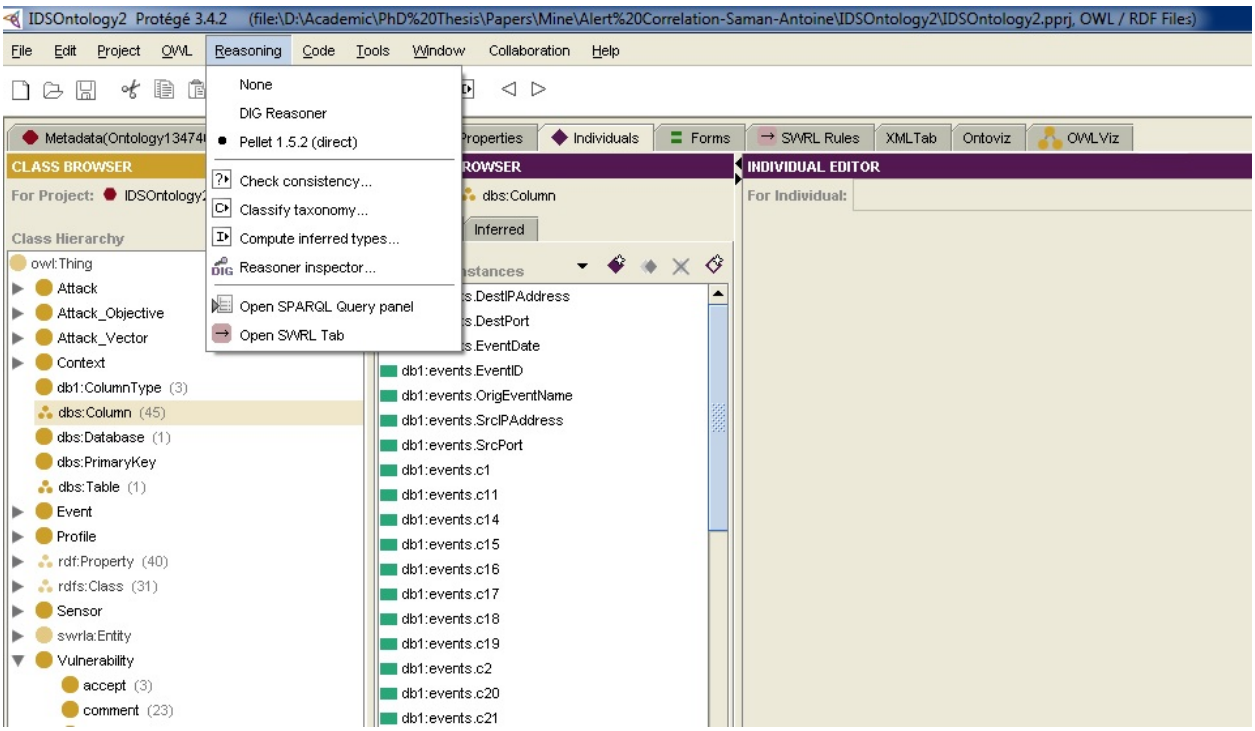

Figure 5.4 The Pellet reasoner of the Protégé ontology editor

formats that each one has its related plug-in, i.e. XML Tab, Excel Import and DataMaster [130] respectively, to import information into Protégé as the instances of classes. About the vulnerability and attack information, since the CVE vulnerability database and CAPEC attack database provide both XML and relational database format, we can use XML Tab or DataMaster to import the vulnerability and attack instances into the Protégé ontology editor. For the collected events and contextual information, *Event Integration* and *Context Integration* components perform this format conversion. In the following subsections, we describe what kind of event and context sensors we employ as the events and contextual information generators. Moreover, we describe how these event and contextual information are integrated within the *Event Integration* and *Context Integration* components.

5.3.1 Event Sensors and Event Integration Process

In order to populate the classes of the *Event Ontology*, first, the event logs are collected from heterogenous event sensors located in the various levels of a computer network into the *Event Integration* component. For this purpose, we considered the following event sensors for our lab and field experiments:

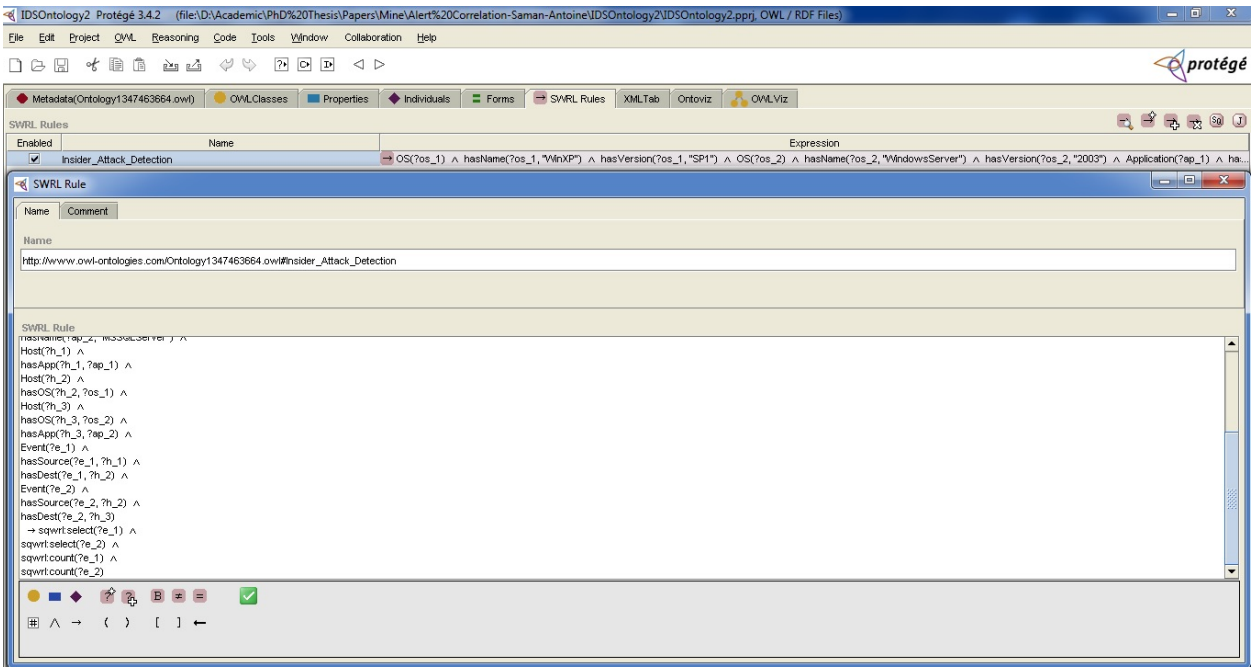

Figure 5.5 SWRL Tab in Protégé

— **NIDS:**
  — Suricata [4]: Suricata is a high performance Network IDS, IPS and Network Security Monitoring engine.
  — Snort [144]: Snort is an open source Network Intrusion Prevention System (NIPS) and Network Intrusion Detection System (NIDS).
  — ISS RealSecure [42]: ISS RealSecure is a commercial firewall/IDS providing malicious activity detection for large, complex networks.
— **HIDS**:
  — OSSEC [38]: OSSEC is an open source Host-based Intrusion Detection System (HIDS) performing log analysis, file integrity checking, rootkit detection, etc.
  — Auditd [6]: Auditd is the userspace component to the Linux Auditing System responsible for writing audit records to the disk.
— **Firewall**:
  — IPTables [7]: IPTables is a command line utility for configuring Linux kernel firewall implemented witthin the Netfilter project.
  — pfSense [186]: pfSense is an open source firewall/router computer software distribution based on FreeBSD.
— **Antivirus**:

  - ClamAV [186]: ClamAV is an open-source anti virus able to detect many types of malicious software.
- **Operating systems**: Windows 7, Windows Server 2003, Ubuntu 12.04, CentOS 6
- **Access point**: Linksys
- **Databases**: MSSQL, MySQL
- **Web server**:
  - Apache [96]: Apache is a web server application supporting a variety of features, many implemented as compiled modules which extend the core functionality.
- **Proxy server**:
  - Squid [184]: Squid is a proxy server and web cache daemon having variety of applications, mainly used for HTTP and FTP protocols.
  - SquidGuard [91]: SquidGuard is a URL redirector software that can be used for content control of websites users can access.
- **VoIP server**:
  - Asterisk [116]: Asterisk is a software implementation of a telephone private branch exchange (PBX) including many features, such as voice mail, conference calling, call redirection, etc.
- **VPN server**:
  - Openswan [187]: Openswan provides a complete IPSec implementation for Linux.
  - OpenVPN [64]: OpenVPN is an open source software application that implements virtual private network (VPN) techniques for creating secure point-to-point or site-to-site connections in routed or bridged configurations and remote access facilities.
- **Email server**:
  - iRedMail [8]: iRedMail is a free, open source mail server solution for Linux/BSD, provides services POP3/IMAP/SMTP, anti-spam, anti-virus, etc.

After collecting event logs into the *Event Integration* component, these events are preprocessed and converted into the IDMEF which is a XML-based format. We use Prelude Hybrid IDS [193] as our event integrator.

Prelude Hybrid IDS [193] is an agentless universal Security Information Management System (SIM, a.k.a SIEM) released under the terms of the GNU General Public License. Prelude collects, normalizes, stores and aggregates event logs from various event generators. Moreover, it converts the collected events into the IDMEF standard format. As Figure 5.6 illustrates, Prelude consists of several sophisticated components [193]:

- Prelude-Manager: Prelude-Manager is the central component of Prelude that can

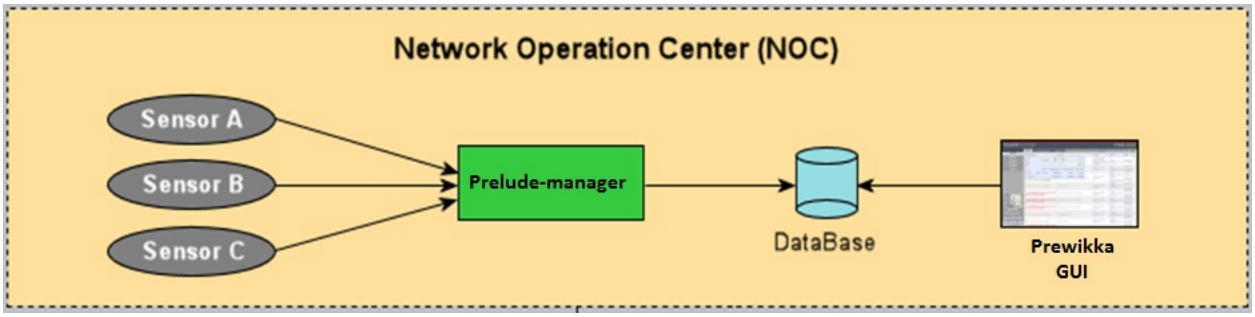

Figure 5.6 Prelude SIEM architecture

connect to both event sensors or other managers. It support various output formats, such as DB, Xmlmod, Textmod, Relaying and SMTP.

— Libprelude: Libprelude provides Application Programming Interfaces (API) that allows third parties to communicate with Prelude-Manager.

— LibpreludeDB: LibpreludeDB is a library that provides an abstraction layer for storing IDMEF alerts into the database.

— Prelude-LML: Prelude-LML provides the capability of analyzing different types of event logs.

— Prewikka Interface: The Prewikka Interface is the web-based Graphical User Interface (GUI) for Prelude.

— Prewikka-PFLogger: The PFLogger collects event logs from OpenBSD's PF software.

Therefore, Prelude directly or using its Prelude-LML component, collects event legs from the event sensors mentioned above, and converts them into the IDMEF format. As Prelude-LML does not provide Perl Compatible Regular Expressions (PCRE) rules for all these sensors, we developed new PCRE rules for those sensors which are not supported by Prelude-LML.

Next, when all the collected events were preprocessed and converted into the IDMEF format, using the XML Tab plug-in, we import them into the corresponding classes in the *Event Ontology*. Moreover, as Prelude allows storing the collected events into MySQL relational database, in such situations, we can use DataMaster plug-in to import them into the Protégé ontology editor. Figure 5.7 illustrates how we populate the *Event Ontology* classes with the integrated event instances.

5.3.2 Context Sensors and Context Integration Process

In order to populate the classes of the *Context Ontology*, first, the contextual information are collected form various context sensors into the *Context Integration* component. For this

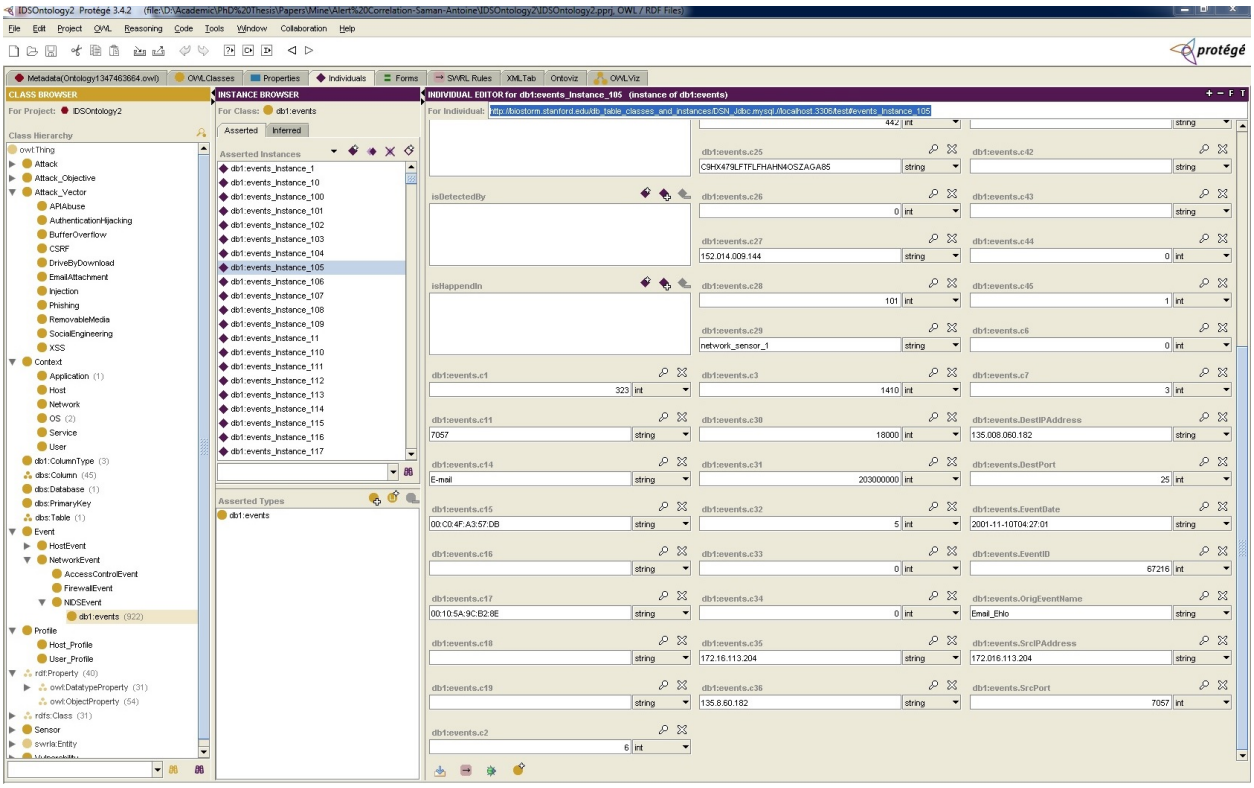

Figure 5.7 Populating the *Event Ontology*

purpose, we considered the following context sensors for our lab and field experiments:

- Nessus [55]: Nessus is a comprehensive vulnerability scanner allowing scans for the following types of vulnerabilities:
  - Vulnerabilities that are exploited by hackers having malicious objectives
  - Misconfiguration (e.g., missing patches)
  - Default passwords
  - Denials of service against the TCP/IP stack by using mangled packets
- Nmap (Network Mapper) [115]: Nmap is a security scanner used to discover hosts and services on a computer network and create a map of the network. Some of its main features are:
  - Host discovery
  - Port scanning
  - Version detection
  - OS detection
  - Scriptable interaction with the target

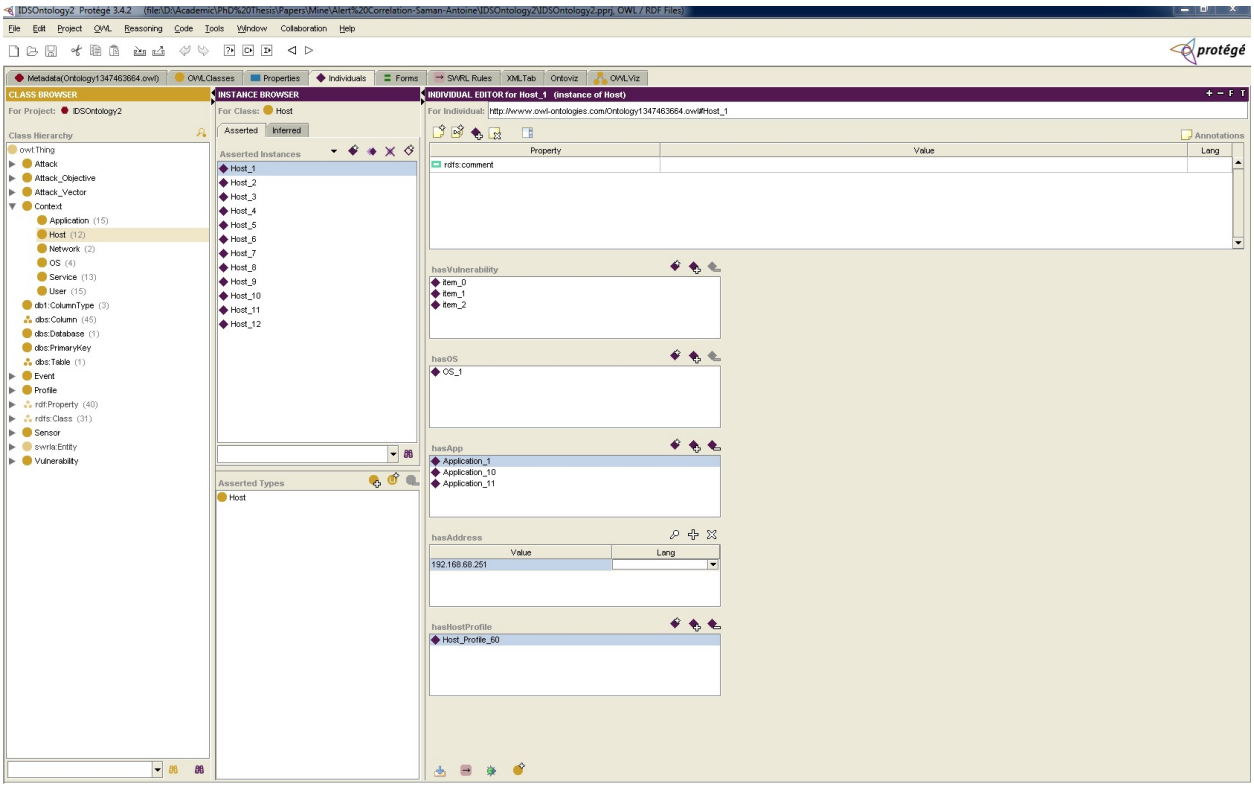

Figure 5.8 Populating the *Context Ontology*

We complement the contextual information received from the above sensors with the information that network and security administrators provide. Next, within the *Context Integration* component, all these information are converted into XML or Excel format, and using XML Tab or Excel Import plug-ins, they are imported into the *Context Ontology* as its class instances. Figure 5.8 illustrates how we populate the *Context Ontology* with the underlying contextual information.

5.4 Discussion on Scalability

We have not addressed at all the issue of scalability and performance of our approach. One of the main challenges that we faced during implementing *Pasargadae* event correlation framework is importing a large volume of information (e.g., 1G) into Protégé ontology editor. For this purpose, we tested various data format (e.g., XML, Excel, MySQL database), and various storage formats, such as Files and databases. However, this challenge is not completely addressed yet.

Generally, while ontologies are quite flexible and readily provide the benefits of abstraction,

they are not always efficient at updating and quickly providing access to stored data. In the case of alert correlation systems of large IT infrastructures, the vast amounts of data involved are likely to make typical XML flat file or relational database storage unwieldily and inefficient for quick on-line alert correlation. While some specific data storage solutions such as object-oriented databases might help alleviate these problems, significant engineering challenges would have to be solved to make *Pasargadae* perform at the same line speeds as some current commercial-grade Intrusion detection and alert correlation systems.

5.5 Summary

In this chapter, we described the implementation process of the proposed *Pasargadae* event correlation framework. We started by describing how we implemented the designed ontologies using the Protégé ontology editor and knowledge acquisition system. Next, we described how to store the designed ontologies, reasoning on the ontologies, and how to query them in order to correlate various classes instances. In Section 5.3, we described the population process of the designed ontologies. In this section, we described various event and context sensors that were employed, and we explained in detail the event and context integration process. Finally, in Section 5.4, we described the challenges that we faced during implementation process, and a brief discussion on the scalability of the proposed ontologies was explained.

CHAPTER 6 CASE STUDY-BASED EVALUATION

In the Chapter 4 of this thesis, we proposed the *Pasargadae* ontology-based context-aware event correlation framework as well as a new event correlation approach and a new alert fusion approach. Next, in Chapter 5, we described in detail the implementation process of *Pasargadae* framework in our lab and field test environment.

In this chapter, in the form of some case studies, we describe how the proposed event correlation and alert fusion approaches employ *Pasargadae* to perform a semantic-based and context-aware event correlation and alert fusion. We selected these case studies to evaluate our proposed approaches because they include attacks that are based on recently emerged sophisticated techniques making current IDS and correlation techniques unable to detect them. Hence, we evaluate the proposed correlation and fusion approaches from various perspective and based on different criteria, and we mainly show the significant flexility and efficiency of *Pasargadae* in implementing various correlation and fusion approaches.

The rest of this chapter is organized as follows. In Section 6.1, we start by describing the first case study that consists of an Island-Hopping attack. In Section 6.2, the second case study covering a Recon-Breakin-Escalate attack is described. Our final case study will be explained in Section 6.3 based on DARPA 2000 data set. Section 6.4 is dedicated to the discussion on flexility and extendability of the proposed approaches. Finally, in Section 6.5, we describe a short summary of this chapter.

6.1 Case Study 1: Island-hopping attacks

As our first case study, we describe an instance of Island-Hopping attack scenario which is part of the UNB ISCX Intrusion Detection Evaluation Dataset [151]. Island-Hopping is a technique of penetrating a network through a weak link, and then hopping around systems within that network [177]. As shown by Figure 6.1, in this scenario the attacker employs the Adobe Reader `util.printf()` buffer overflow vulnerability (CVE-2008-2992) to execute arbitrary code with the same privileges as the user running it.

First of all, let us follow the steps taken by the attacker and let us consider what kind of artifacts would be picked up by event sensors. To launch an attack, the attacker creates a malicious PDF file using Metasploit, and embeds a Meterpreter reverse TCP shell on port 5555 inside it. Then, the attacker sends a system upgrade email including the PDF file on behalf of admin@[...] to all the users of the testbed. Through user5, which initiates the first

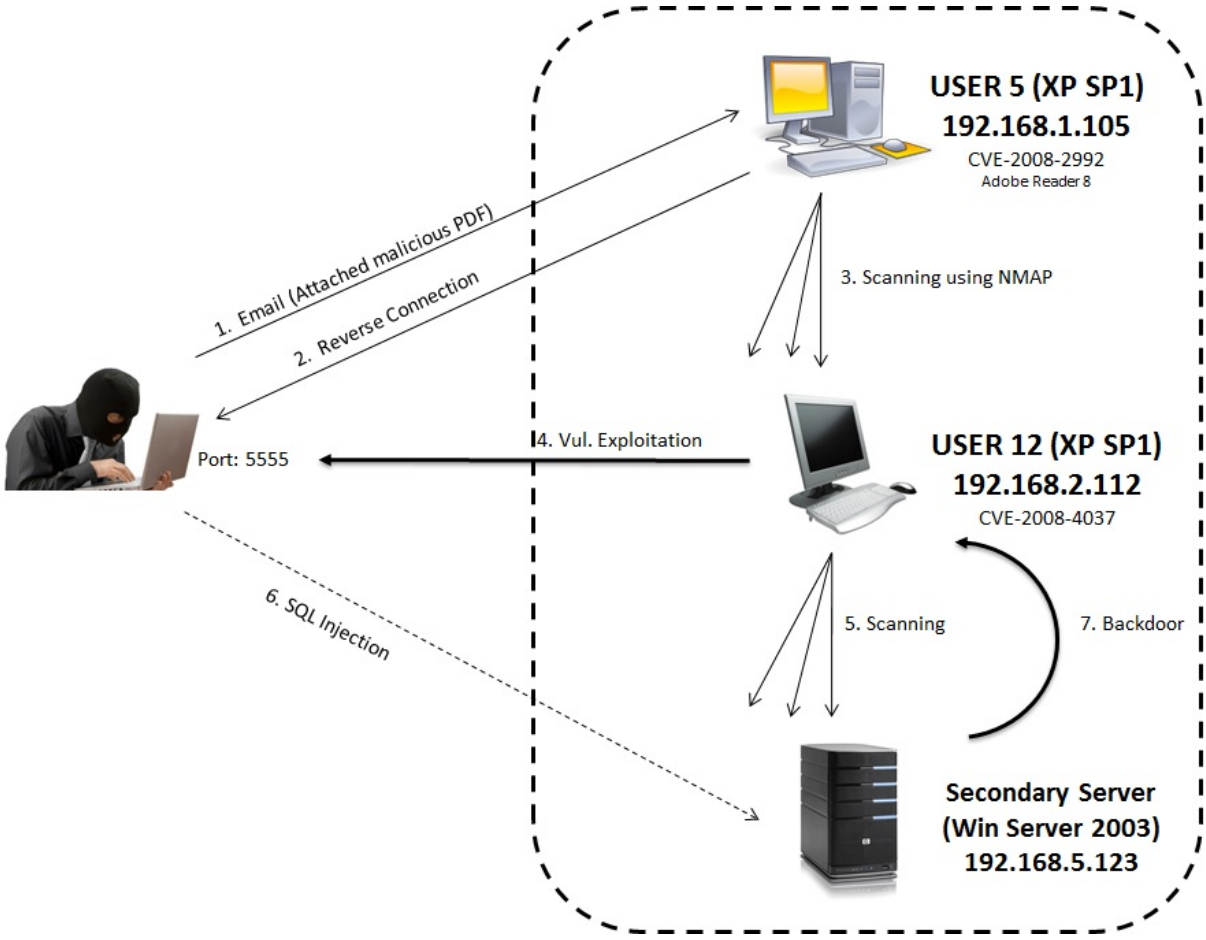

Figure 6.1 An instance of Island-hopping attack

session (events 1, 2, 3), the attacker starts to scan potential hosts on two consecutive subnets 192.168.1.0/24 and 192.168.2.0/24 (event 4). User12 is identified as running Windows XP SP1 with a vulnerable SMB authentication protocol on port 445 (CVE-2008-4037) (events 5, 6). The attacker exploits this vulnerability to capture user12 (event 7) and a scan is performed from this user to the server subnet (192.168.5.0/24) (event 8). This scan identifies a Windows Server 2003 running an internal Web application using MSSQL Server as its backend database with only port 80 opened. This leads to the use of Web application hacking techniques like SQL injection (event 9). Finally the attacker, using SQL injection techniques, compromises the target system (event 10, 11). Table 6.1 presents a summary of the events.

Now that we have a specific example attack with defined attributes, we can expand the base attack class of the proposed attack ontology, and drill down into some specific classes for our analysis. For this purpose, we expand this class into some specific classes including attack

Table 6.1 Event logs generated by the sensors in the island-hopping attack scenario

Event ID	**Name**	**Sensor**	**Date**	**Source**	**Target**	Tag
1	Local Exploit	HIDS	6/13/10 16:02:20	192.168.1.105	192.168.1.105	Step 1
2	Local Exploit	Antivirus	6/13/10 16:02:22	192.168.1.105	192.168.1.105	Step 1
3	Reverse Connection	Packet Capturer	6/13/10 16:17:32	192.168.1.105	192.168.1.105	Step 2
4	Scanning	NIDS	6/13/10 16:42:24	192.168.1.105	192.168.1.0/24 192.168.2.0/24	Step 3
5	Windows File Sharing	NIDS	6/13/10 17:20:32	192.168.1.105	192.168.2.112	Step 3
6	Windows File Sharing	NIDS	6/13/10 17:34:32	192.168.1.105	192.168.2.112	Step 3
7	Local Exploit	HIDS	6/13/10 17:50:24	192.168.2.112	192.168.2.112	Step 4
8	Scanning	NIDS	6/13/10 18:02:37	192.168.2.112	192.168.5.0/24	Step 5
9	HTTPWeb	NIDS	6/13/10 18:19:41	192.168.2.112	192.168.5.123	Step 6
10	SQLInjection	HIDS	6/13/10 18:20:19	192.168.5.123	192.168.5.123	Step 7
11	Bad Request	MSSQL	6/13/10 18:20:21	192.168.5.123	192.168.5.123	Step 7

vector, attack objectives, and a number of subclasses showing the type of example instances of this class. Figure 6.2 illustrates this customized expansion of the base attack class. We will use this specific version of the attack ontology in the next case studies, and also in the field test. As explained, in this case study, the attack vector is "system upgrade email having a malicious attachment", and the attack objectives can be "data annihilation and information leakage".

In order to preprocess the events generated by event sensors during the above scenario, first, the event integration component integrates all received events. Then, the integrated events are transferred into the event ontology. Additionally, we manually populate vulnerability and context ontologies based on the published documents related to the UNB ISCX dataset. Therefore, the Adobe Reader `util.printf()` vulnerability and others that might be present in the IT infrastructure are input into the vulnerability ontology. Contextual information about the existing hosts (IP addresses, open ports, available services, etc.), services and users are also manually input into the context ontology. In this case, this includes the

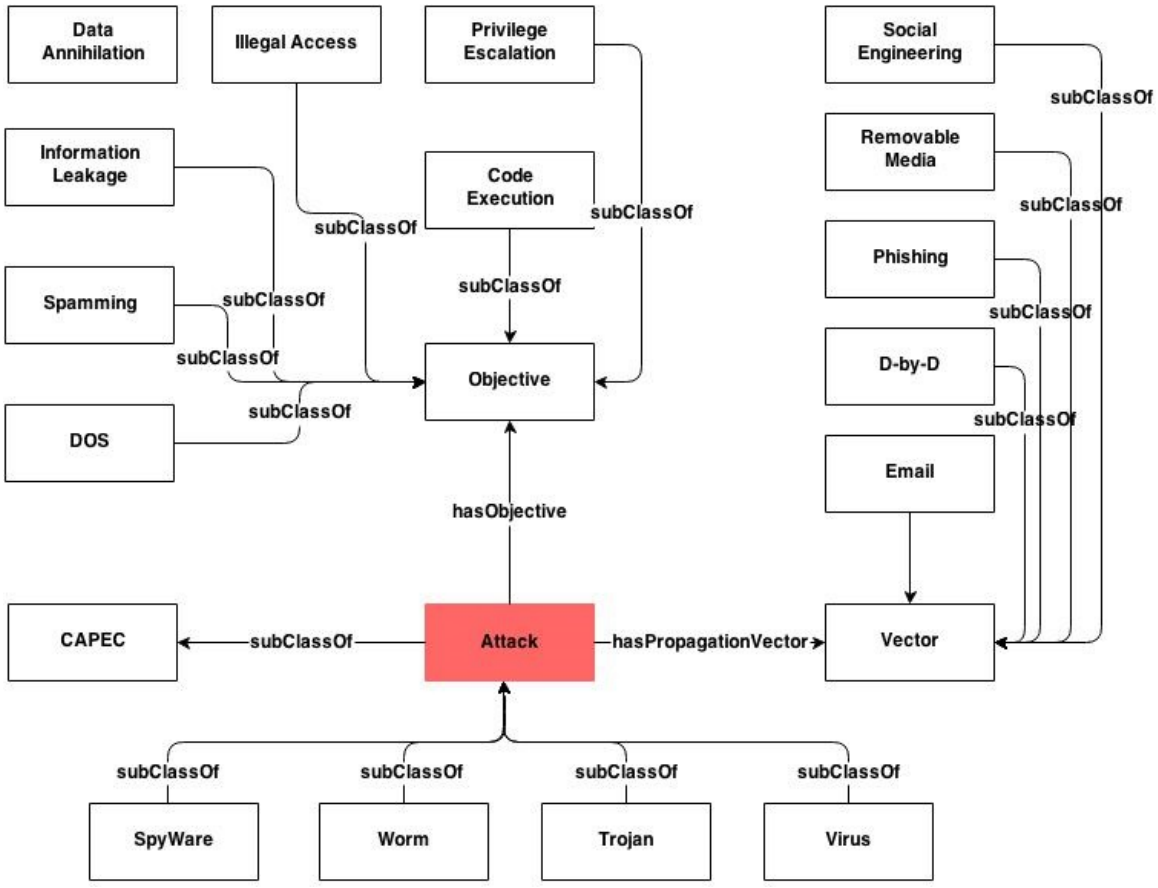

Figure 6.2 Class diagram relationship of the attack ontology

information about the three compromised hosts (IP addresses 192.168.1.105, 192.168.2.112, and 192.168.5.123), their open ports (i.e. 5555 and 445).

After preprocessing the events and populating the ontologies, we start the event correlation process. In this scenario, we can see that Table 6.1 only lists the event relevant to our attack scenario. In this case, the first step of the process, filtering out non-relevant alerts, does not reduce the number or events that must be processed. So, after gathering the generated events from the heterogonous sensors, we jump to the first level meta-event creation phase, and we create the corresponding meta-events. In our case, the correlator will regroup events 1 and 2, two sensors reporting on the same occurrence, in a meta-event. Also, because they originate from the same source and hit the same target, event 5 and 6 are correlated as being related to each other in the same meta-event. Here, even though event 7 is related to event 5 and 6, representing the event generated on the host by the SMB exploit we see moving on the network in event 6, the event is not correlated because of the Divide & Conquer technique we use to correlate network and host based event separately. In a similar way, event 10 and

Table 6.2 The meta-event list created from the events in Table 6.1

Meta-Event ID	**Name**	**Sensor**	**Date**	**Source**	**Target**	Tag
1	Local Exploit	[HIDS, Antivirus]	6/13/10 16:02:20-22	192.168.1.105	192.168.1.105	correlates with 2
2	Reverse Connection	Packet Capturer	6/13/10 16:17:32	192.168.1.105	192.168.1.105	correlates with 1
3	Scanning	NIDS	6/13/10 16:42:24	192.168.1.105	192.168.1.0/24 192.168.2.0/24	
4	Windows File Sharing	NIDS	6/13/10 [17:20:32 - 17:34:32]	192.168.1.105	192.168.2.112	correlates with 5
5	Local Exploit	HIDS	6/13/10 17:50:24	192.168.2.112	192.168.2.112	correlates with 4
6	Scanning	NIDS	6/13/10 18:02:37	192.168.2.112	192.168.5.0/24	
7	HTTPWeb	NIDS	6/13/10 18:19:41	192.168.2.112	192.168.5.123	correlates with 8
8	[SQLInjection, Bad Request]	[HIDS, MSSQL]	6/13/10 [18:20:19 - 18:20:21]	192.168.5.123	192.168.5.123	correlates with 7

11 will be correlated together because of the shared source IP, target IP, and application. As Table 6.2 shows, the first level meta-events are the result of this phase. We make a second pass on the meta-events to correlate all the events, both host based and network based, to form a sequence of events based on consequence and prerequisite relationships. For example, the packets flying on the network in meta-even 1 and 2 have consequences in the form of scanning event 3 and so on. We can then rebuild the entire attack scenario using the Rule 6.1:

$$\begin{aligned}
&\texttt{OS}(?os_1) \wedge \texttt{hasName}(?os_1, "WinXP") \wedge \texttt{hasVersion}(?os_1, "SP1") \wedge \texttt{OS}(?os_2) \wedge \\
&\texttt{hasName}(?os_2, "WindowsServer") \wedge \texttt{hasVersion}(?os_2, "2003") \wedge \texttt{Application}(?ap_1) \wedge \\
&\texttt{hasName}(?ap_1, "AdobeAcrobate") \wedge \texttt{hasVersion}(?ap_1, 8) \ \wedge \texttt{Application}(?ap_2) \wedge \\
&\texttt{hasName}(?ap_2, "MSSQLServer") \wedge \texttt{Host}(?h_1) \ \wedge \texttt{hasApp}(?h_1, ?ap_1) \wedge \texttt{Host}(?h_2) \wedge \\
&\texttt{hasOS}(?h_2, ?os_1) \wedge \texttt{Host}(?h_3) \wedge \texttt{hasOS}(?h_3, ?os_2) \ \wedge \texttt{hasApp}(?h_3, ?ap_2) \wedge \texttt{Event}(?e_1) \ \wedge \\
&\texttt{hasSource}(?e_1, ?h_1) \wedge \texttt{hasDest}(?e_1, ?h_2) \wedge \texttt{Event}(?e_2) \wedge \\
&\texttt{hasSource}(?e_2, ?h_2) \wedge \texttt{hasDest}(?e_2, ?h_3) \wedge \\
&\longrightarrow \texttt{sqwrl:select}(?e_1) \wedge \texttt{sqwrl:select}(?e_2) \wedge \\
&\texttt{sqwrl:count}(?e_1) \wedge \texttt{sqwrl:count}(?e_2)
\end{aligned} \tag{6.1}$$

Rule 6.1 correlates event and attack ontologies, and attempts to discover corresponding events for each step of the attack. If it finds at least one match regarding each step, the rule will be successful in detecting the whole attack scenario. This can now be presented to the security administrator instead of flooding him with a huge number of events. Figure 6.3 represents the result of the event correlation process.

Figure 6.3 The island-hopping attack graph

6.2 Case Study 2: Recon-breakin-Escalate attacks

As the second case study, we describe an instance of Recon-Breakin-Escalate attack scenarios. The Recon-Breakin-Escalate scenario models an attacker who scans for vulnerabilities in a network or host, breaks into a vulnerable host, and escalates her/his privilege [177]. In our attack scenario, a malicious attacker scans a wide range of IP addresses to find a vulnerable target. Once a server is found, he enumerates the services to find a vulnerability and he exploits the vulnerability to get arbitrary code execution and gain control of the server. He can then proceed to extract information from the server. Figure 6.4 illustrates this scenario.

As with our first example, we will go through each of the attacker's step and explain what events would be generated by our sensors on each step. Table 6.3 summarizes the events generated during the attack. First, the attacker (152.63.146.6) using a vulnerability scanning

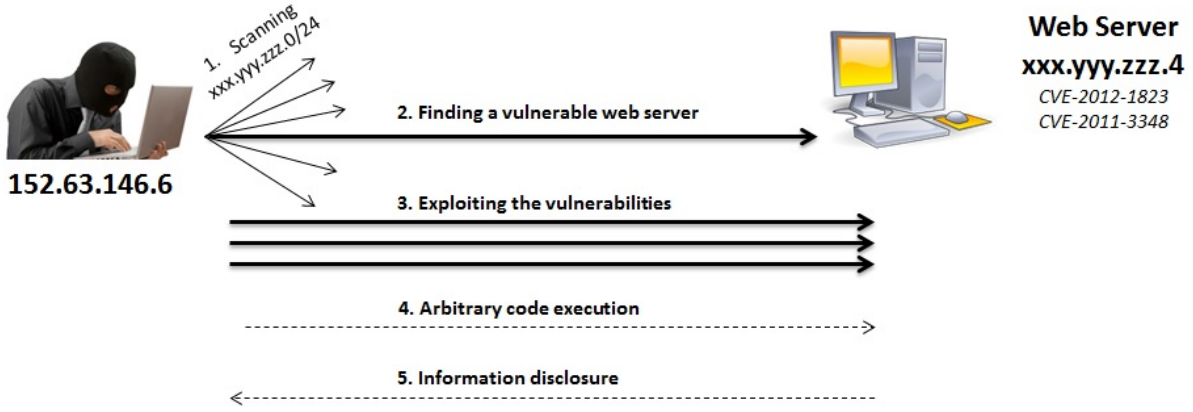

Figure 6.4 An instance of recon-breakin-escalate attack

Table 6.3 The meta-event list created from the events in Table 6.1

EventID	Name	Sensor	Date	Source	Target	Tag
1	Scanning	Router	May 31 09:27:44	152.63.146.6	xxx.yyy.zzz.1(80) xxx.yyy.zzz.n(80)	step 1
2	Scanning	NIDS	May 31 09:27:46	152.63.146.6	xxx.yyy.zzz.1(80) xxx.yyy.zzz.n(80)	step 1
3	Connection volume alert	Firewall	Jun 1 06:08:50	152.63.146.6	xxx.yyy.zzz.4	step 2
4	Apache Exploit	NIDS	Jun 1 06:08:51	152.63.146.6	xxx.yyy.zzz.4	step 2
5	IIS Exploit	NIDS	Jun 1 06:08:52	152.63.146.6	xxx.yyy.zzz.4	non-relevant
6	Local Exploit	HIDS	Jun 1 06:08:54	152.63.146.6	xxx.yyy.zzz.4	step 2
7	Access	Apache	Jun 1 06:08:55	152.63.146.6	xxx.yyy.zzz.4	step 3

tools launches a series of probes against the target network (xxx.yyy.zzz.0/24) searching for vulnerable hosts (events 1 and 2). The attacker finds that the host xxx.yyy.zzz.4 is vulnerable to CVE-2012-1823 and CVE-2011-3348. The attacker launches a web server exploit kit tool that exploits numerous vulnerabilities in a short period of time causing the firewall to register a large number of connections (event 3). A number of exploits are spotted by the IDS (events 4 and 5) and one makes it through and is registered on the host based IDS (event 6). Finally, the attacker accesses confidential information, such as the password file, on the web server

(events 7).

Table 6.4 The meta-event list created from the events in Table 6.1

MetaEvent ID	Name	Sensor	Date	Source	Target	Tag
1	Scanning	[Router, NIDS]	May 31 09:27:44-09:27:46	152.63.146.6	xxx.yyy.zzz.1(80) xxx.yyy.zzz.n(80)	step 1
2	[Connection volume alert, Apache Exploit]	[Firewall, NIDS]	Jun 1 06:08:50-06:08:51	152.63.146.6	xxx.yyy.zzz.4	step 2
3	[Local Exploit, Access]	[HIDS, Apache]	Jun 1 06:08:54-06:08:55	152.63.146.6	xxx.yyy.zzz.4	step 3

After gathering the events from the various sensors, the events are correlated. First, in the non-relevant events reduction step, we eliminate those events which are detected as non-relevant. For example, the attack tool attempts to exploit multiple vulnerabilities. Of those vulnerabilities, some are known to be ineffective. So, based on our information about the context (i.e. that the server is running a version of Apache) in the Host OS attribute and based on the information in the attack ontology, we can eliminate the IIS exploit (event 5) as being non-relevant event, i.e. as being impossible to start a chain of events. Therefore, we eliminate this event from event list. Then, we create the first level meta-events as presented in the island-hopping scenario. Table 6.4 shows the meta-events produced during this phase. Finally, we can rebuild the entire attack scenario using the Rule 6.2. Figure 6.3 represents the result of the event correlation process.

$$\begin{aligned}
&\texttt{Host(?h1)} \wedge \texttt{Host(?h2)} \wedge \texttt{Application(?a)} \wedge \texttt{hasApplication(?h2, ?a)} \wedge \\
&\texttt{Vulnerability(?v)} \wedge \texttt{hasVulnerability(?a, ?v)} \wedge \texttt{Analyzer(?an)} \wedge \\
&\texttt{hasType(?an, HIDS)} \wedge \texttt{Time(?t1)} \wedge \texttt{Time(?t2)} \wedge \texttt{Time(?t3)} \wedge \texttt{Event(?e)} \wedge \\
&\texttt{hasSource(?e, ?h1)} \wedge \texttt{hasTime(?e, ?t1)} \wedge \texttt{Event(?e1)} \wedge \texttt{hasSource(?e1, ?h1)} \wedge \\
&\texttt{hasDestination(?e1, ?h2)} \wedge \texttt{hasTime(?e1, ?t2)} \wedge \texttt{Event(?e2)} \wedge \\
&\texttt{hasSource(?e2, ?h2)} \wedge \texttt{hasDestination(?e2, ?h2)} \wedge \texttt{hasAnalyzer(?e2, ?an)} \wedge \\
&\texttt{hasTime(?e2, ?t3)} \wedge \texttt{swrlb:stringEqualIgnoreCase(?t1,?t2)} \wedge \\
&\texttt{swrlb:stringEqualIgnoreCase(?t2,?t3)} \wedge \\
&\longrightarrow \texttt{sqwrl:select(?e)} \wedge \texttt{sqwrl:limit(20)} \wedge \\
&\texttt{sqwrl:select(?e1)} \wedge \texttt{sqwrl:limit(5)}
\end{aligned} \tag{6.2}$$

Figure 6.5 The recon-breakin-escalate attack graph

6.3 Case Study 3: Alert fusion and alert correlation based on DARPA 2000 data set

In this case study, we evaluate *Pasargadae* as an alert correlation framework using DARPA 2000 data set [120].

The DARPA 2000 dataset contains two attack scenarios: LLDDOS 1.0 and LLDOS 2.0.2. We have chosen the first attack scenario (LLDDOS 1.0) for our evaluation. LLDDOS 1.0 is a multi-step scenario corresponding to a Distributed Denial of Service (DDoS) flooding attack [194]. The attack has 5 phases and it takes about three hours to be completed. Table 6.5 lists the attack phases, and Figure 6.6 illustrates a number of example events of every phase.

We use both the RealSecure and Snort NIDS as base our alerts sensors to detect all the steps of the attack. Snort outputs around 1,211 raw alerts for the LLDDOS 1.0 dataset, but it does not detect the installation phase of the DDoS attack (i.e. phase 4). On the other hand, and as is described in [90], RealSecure outputs 924 raw alerts for the same dataset, corresponding

Table 6.5 Five phases of DARPA's LLDDOS1.0 attack scenario

Step	Name	Time	Description
1	IP Sweep	09:45 to 09:52	The attacker sends ICMP echo-requests and based on the ICMP echo-replies finds out which hosts are active.
2	Sadmind Ping	10:08 to 10:18	Testing the existence of sadmind daemon on the live IPs.
3	Exploiting	10:33 to 10:34	Exploiting the sadmind vulnerability to break into vulnerable hosts.
4	Installation	10:50	Installation of the trojan mstream DDoS software on three hosts.
5	Launching DDoS	11:27	Launching the DDoS attack.

to the 22 alert types shown in Table 6.6. However, it does not output any alerts related to ICMP pings (i.e. phase 1). Consequently, the combination of Snort and RealSecure can detect all phases of the attack. Nonetheless, just using a combination of both IDS alerts with a simple `OR` rule will result in a significant number of redundant alerts and false positives, as we will see. With *Pasargadae*, we expect to have lower redundancy, and fewer non-relevant alerts and false positives.

In the second step, Prelude converts all the received alerts into the IDMEF format, and transfers the integrated alerts into the alert ontology as its instances. We manually populate the context and vulnerability ontologies based on the information existing in the published documents related to the DARPA 2000 dataset. Thus, the Solaris sadmind vulnerability (CVE-1999-0977) and others existing vulnerabilities in the underlying network are transferred into the vulnerability ontology. The same is done with contextual information about the existing hosts and users, in this case including the three compromised hosts (IP addresses 172.16.115.20, 172.16.112.50, 172.16.112.10), and their open ports (i.e. telnet port 23) and users (e.g. hacker2).

As more than one alert sensor have been employed in this scenario, first, we evaluate the performance and efficiency of our proposed semantic-based and context-aware alert fusion approach described in chapter 5. We start with eliminating redundant alerts related to the independent detection of same malicious events via different alert sensor using the following rule:

1	03/07/2000	09:51:36	00:00:01	eco/i	-	-	202.077.162.213	172.016.115.005	0	-
2	03/07/2000	09:51:36	00:00:05	ecr/i	-	-	172.016.112.194	202.077.162.213	0	-
3	03/07/2000	09:51:36	00:00:01	eco/i	-	-	202.077.162.213	172.016.115.020	0	-
4	03/07/2000	09:51:36	00:00:01	ecr/i	-	-	172.016.115.020	202.077.162.213	0	-
5	03/07/2000	09:51:38	00:00:01	eco/i	-	-	202.077.162.213	172.016.115.087	0	-
1	03/07/2000	10:08:07	00:00:01	sunrpc/u	54790	111	202.077.162.213	172.016.115.020	0	-
2	03/07/2000	10:08:07	00:00:01	32773/u	54792	32773	202.077.162.213	172.016.115.020	0	-
3	03/07/2000	10:08:07	00:00:55	sunrpc/u	54793	111	202.077.162.213	172.016.115.087	0	-
4	03/07/2000	10:08:07	00:00:25	dur/i	-	-	172.016.112.194	202.077.162.213	0	-
5	03/07/2000	10:08:37	00:00:01	dur/i	-	-	172.016.115.087	202.077.162.213	0	-
1	03/07/2000	10:33:10	00:00:01	sunrpc/u	60249	111	202.077.162.213	172.016.115.020	0	-
2	03/07/2000	10:33:10	00:00:01	32773/u	60251	32773	202.077.162.213	172.016.115.020	0	-
3	03/07/2000	10:33:12	00:00:01	sunrpc/u	60253	111	202.077.162.213	172.016.115.020	0	-
4	03/07/2000	10:33:12	00:00:01	32773/u	60255	32773	202.077.162.213	172.016.115.020	0	-
5	03/07/2000	10:33:14	00:00:08	telnet	46956	23	202.077.162.213	172.016.115.020	0	-
1	03/07/2000	10:50:01	00:00:05	telnet	47496	23	202.077.162.213	172.016.115.020	0	-
2	03/07/2000	10:50:02	00:00:01	shell	1023	514	172.016.115.020	202.077.162.213	0	-
3	03/07/2000	10:50:02	00:00:01	shell	1022	514	172.016.115.020	202.077.162.213	0	-
4	03/07/2000	10:50:03	00:00:01	shell	1022	514	172.016.115.020	202.077.162.213	0	-
5	03/07/2000	10:50:04	00:00:01	shell	1021	514	172.016.115.020	202.077.162.213	0	-
1	03/07/2000	11:26:15	00:08:06	telnet	49212	23	202.077.162.213	172.016.115.020	0	-
2	03/07/2000	11:27:04	00:00:01	1020	1022	1020	202.077.162.213	172.016.115.020	0	-
3	03/07/2000	11:27:09	00:00:01	9325/u	33799	9325	172.016.115.020	255.255.255.255	0	-
4	03/07/2000	11:27:51	00:00:01	7983/u	33800	7983	172.016.115.020	172.016.112.050	0	-
5	03/07/2000	11:27:51	00:00:01	7983/u	33800	7983	172.016.115.020	172.016.112.010	0	-

Figure 6.6 Alerts related to the 5 phases of the LLDDOS 1.0

$$
\begin{aligned}
&\texttt{ALERT(?a1)} \wedge \texttt{ALERT(?a2)} \wedge \texttt{ANALYSER(?an1)} \wedge \texttt{ANALYSER(?an2)} \wedge \\
&\texttt{CreateTime(?ct1)} \wedge \texttt{CreateTime(?ct2)} \wedge \texttt{DetectTime(?dt1)} \wedge \\
&\texttt{DetectTime(?dt2)} \wedge \texttt{AnalyzerTime(?at1)} \wedge \texttt{AnalyzerTime(?at2)} \wedge \\
&\texttt{SOURCE(?s1)} \wedge \texttt{SOURCE(?s2)} \wedge \texttt{TARGET(?t1)} \wedge \texttt{TARGET(?t2)} \wedge \\
&\texttt{TARGET(?t1)} \wedge \texttt{TARGET(?t2)} \wedge \texttt{CLASSIFICATION(?c1)} \wedge \\
&\texttt{CLASSIFICATION(?c2)} \wedge \texttt{ASSESSMENT(?as1)} \wedge \texttt{ASSESSMENT(?as2)} \wedge \\
&\texttt{swrlb:stringEqualIgnoreCase(?ct1,?ct2)} \wedge \\
&\texttt{swrlb:stringEqualIgnoreCase(?dt1,?dt2)} \wedge \\
&\texttt{swrlb:stringEqualIgnoreCase(?at1,?at2)} \wedge \\
&\texttt{swrlb:stringEqualIgnoreCase(?s1,?s2)} \wedge \\
&\texttt{swrlb:stringEqualIgnoreCase(?t1,?t2)} \wedge \\
&\texttt{swrlb:stringEqualIgnoreCase(?c1,?c2)} \wedge \\
&\texttt{swrlb:stringEqualIgnoreCase(?as1,?as2)} \wedge \\
&\texttt{swrlb:stringEqualIgnoreCase(?an1,"Snort")} \wedge \\
&\texttt{swrlb:stringEqualIgnoreCase(?an2,"RealSecure")} \\
&\longrightarrow \texttt{sqwrl:select(?a1)}
\end{aligned}
\tag{6.3}
$$

Table 6.6 Alert types generated by ISS RealSecure based on the DARPA 2000 dataset

ID	AlertType
1	RIPExpire
2	RIPAdd
3	*Email_Ehlo*
4	TelnetTerminaltype
5	*FTP_User*
6	*FTP_Pass*
7	*FTP_Syst*
8	*HTTP_Shells*
9	Admind
10	*Sadmind_Ping*
11	*Email_Almail_Overflow*
12	*HTTP_Java*
13	*Sadmind_Amslverify_Overflow*
14	*Mstream_Zombie*
15	Rsh
16	*HTTP_Cisco_Catalyst_Exec*
17	*SSH_Detected*
18	*Email_Debug*
19	TelnetXdisplay
20	TelnetEnvAll
21	*Port_Scan*
22	*Stream_DoS*

Rule 6.3 discovers all alerts having the same attributes but detected by different sensors. Based on our analysis, 32.7% of the gathered alerts have been generated by both Snort and IIS RealSecure. Therefore, we eliminate these alerts and keep the rest of the alerts for further analysis by the decision making component. Then, considering the five phases of the LLDDOS 1.0 attack scenario, the decision making component uses further rules to inspect the prerequisites (Rules 4.1 and 4.2) and consequences (Rule 6.4) of the event corresponding to each alert. Thus, the decision making component recognizes those alerts whose prerequisites are not met, or alerts whose consequences are not part of this particular 5-phase attack. These alerts are classified as non-relevant alerts and false positives, respectively.

$$\begin{aligned} &\texttt{ALERT(?a)} \wedge \texttt{CONSEQUENCES(?c)} \wedge \texttt{hasStep(?c,?s)} \wedge \texttt{hasName(?a,?n1)} \wedge \\ &\texttt{hasName(?s,?n2)} \wedge \texttt{swrlb:stringEqualIgnoreCase(?n1,?n2)} \\ &\longrightarrow \texttt{sqwrl:select(?a)} \end{aligned} \tag{6.4}$$

For instance, in the first phase of the attack, the alerts that are related to the ICMP echo requests are considered as attacks because their corresponding events are cognitive requirements for the next attack steps. As another example, in the third phase of the attack, the alerts that are related to the sadmind Remote-to-Root exploit are considered as attacks because their corresponding events have contextual effects on the underlying systems. Our results indicate that 91.08% of the alerts were false positives, and only 8.92% of the alerts were true positives.

Table 6.7 lists our results based on the mentioned evaluation metrics in the previous subsection. Since both base alert sensors (Snort and ISS RealSecure) only detect a few of the 33,787 attack events in Phase 5 (launching DDoS), their total false negative rates are quite high. The recall column in Table 6.7 consequently reports low rates for both sensors and our fusion approach. On the other hand, our fusion approach does well at reducing false positives: it reduces the false positive rate five-fold and four-fold for Snort and RealSecure, respectively, as can be seen in the *FP* column. Figure 6.7 shows that our model gives better results.

Next, we apply our proposed event correlation approach to the remaining alerts of the previous step. Therefore, alerts reported by only one IDS are then further analysed by attempting attack reconstruction on the 5 phases of the LLDDOS 1.0 attack scenario, by using the following rule:

Table 6.7 Alert fusion results of DARPA 2000 data set

IDS	**Redundant alerts (%)**	**FP**	**TP**	**FN**	**Precision**	**Recall**	**F-measure**
Snort	0.00%	1118	93	33814	0.07	0.002	0.002
RealSecure	0.00%	870	54	33853	0.05	0.001	0.001
The proposed fusion approach	32.7%	46	123	33784	0.836	0.003	0.005

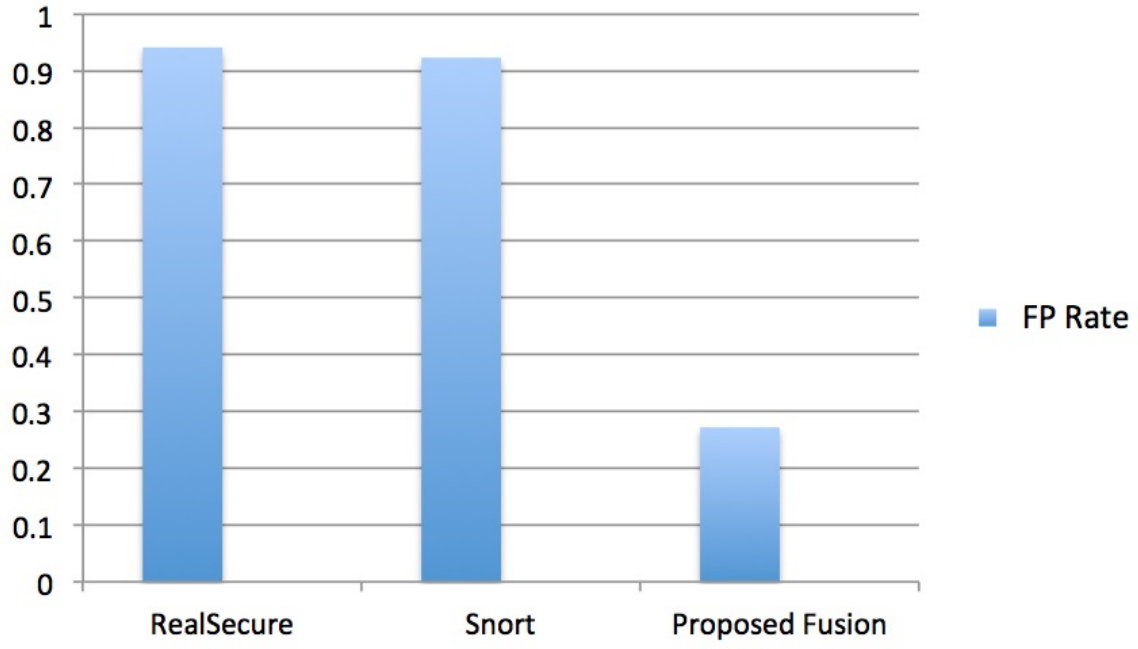

Figure 6.7 Improving the false positive rate of Snort + RealSecure using the proposed fusion approach

$$\begin{aligned}
&\texttt{ATTACK(?at)} \wedge \texttt{hasName(?at,"LLDDOS1")} \wedge \texttt{ALERT(?a1)} \wedge \texttt{ALERT(?a2)} \wedge \texttt{ALERT(?a3)} \wedge \\
&\texttt{ALERT(?a4)} \wedge \texttt{ALERT(?a5)} \wedge \texttt{ALERT(?a6)} \wedge \texttt{ALERT(?a7)} \wedge \texttt{HOST(?h1)} \wedge \\
&\texttt{hasName(?a1,"Scanning")} \wedge \texttt{hasTarget(?a1,?h1)} \wedge \texttt{hasService(?h1,"ICMP")} \wedge \\
&\texttt{hasName(?a2,"Sadmind_Ping")} \wedge \texttt{hasName(?a3,"Sadmind_Amslverify_Overflow")} \wedge \\
&\texttt{hasName(?a4,"Admind")} \wedge \texttt{hasName(?a4,"Rsh")} \wedge \texttt{hasName(?a4,"MStream_Zombie")} \wedge \\
&\texttt{hasName(?a4,"Stream_DOS")} \\
&\longrightarrow \texttt{sqwrl:select(?a1,?a2,?a3,?a4,?a5,?a6,?a7,?at)}
\end{aligned} \tag{6.5}$$

Table 6.8 Experimental results based on the DARPA 2000 dataset

IDS	Redundant alerts (%)	FP	TP	FN	Precision	Recall	F-measure
Snort	0.00%	1118	93	33814	0.07	2×10^{-3}	2×10^{-3}
RealSecure	0.00%	870	54	33853	0.05	10^{-3}	10^{-3}
ONTIDS	32.7%	0	123	33784	1.00	3×10^{-3}	5×10^{-3}

Rule 6.5 correlates alert and attack ontologies, and discovers corresponding alerts for the each step of the attack. If at least one match is found for each step, the rule will be successful in detecting the whole attack scenario. Table 6.8 summarises our results. The achieved results indicates that ONTIDS does considerably well at reducing false positives, in fact reducing it to 0. Figure 6.8 shows that our model gives better results.

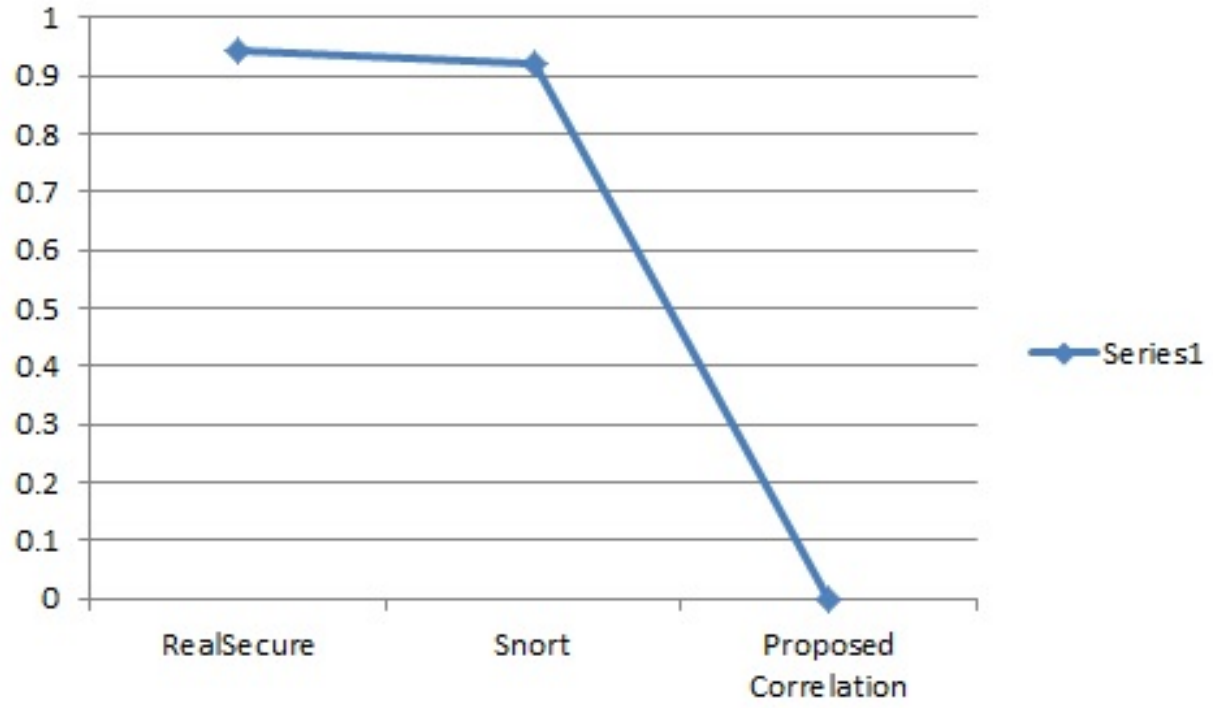

Figure 6.8 Improving the false positive rate of Snort + RealSecure using the proposed correlation approach

6.4 Discussion on flexibility

We have demonstrated the use of the *Pasargadae* event correlation framework in a number of quite different case studies involving considerably distinct attack scenarios. More important than the reduction in false positives (in this somewhat contrived evaluation scenario), the point of these tests was to show the level of significant flexibility of such a framework. The fact that the same correlation Rules 4.1 and 4.2 are used for the context-based alert filtering in both scenarios deceptively hides the fact that the vulnerability and context instances in both cases are quite different as they come from different sources, and hence have different attributes and properties. In fact, Figure 6.9 illustrates that the ontologies for every attack

scenario are different as they involve different subclasses of context, sensors, attacks and alerts. Nonetheless, it is the power of abstraction of ontologies (implemented through the construction of adequate information integration drivers) that allows us to design and use such generic correlation rules. As security analysts start to use *Pasargadae*, we expect that these ontologies will naturally expand to include new subclasses capturing the idiosyncrasies of the systems being monitored, the various types of sensors monitoring them, and richer and more complex attack models and vulnerabilities.

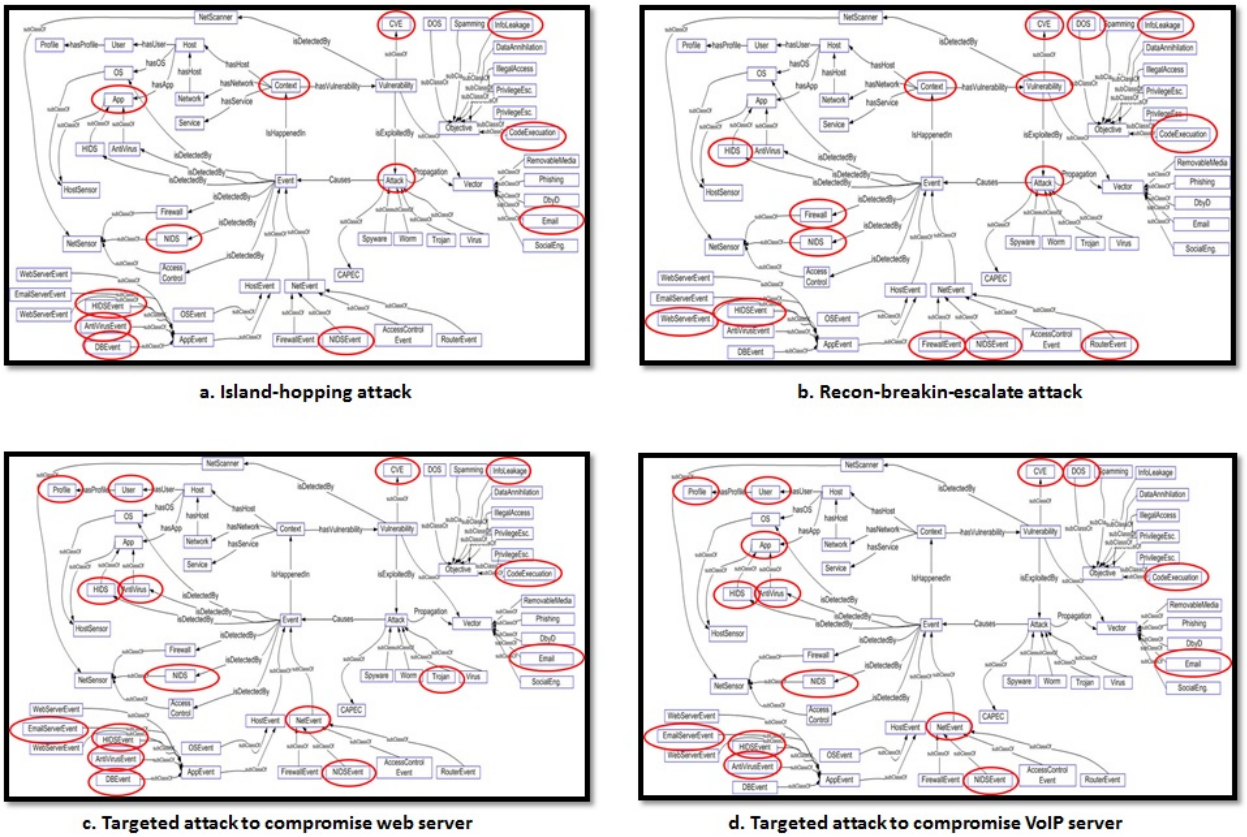

Figure 6.9 The involved classes of the designed ontologies on the proposed test cases

6.5 Summary

In this chapter, in the form of some case studies, we evaluated *Pasargadae* framework along with the proposed event correlation and alert fusion approaches. First, we evaluated the proposed approaches using UNB ISCX 2012 data set. Next, we evaluated the proposed approaches based on a Recon-Breakin-Escalate attack. Section 6.3 evaluated the proposed event correlation and alert fusion approaches using DARPA 2000 LLDDOS 1.0 data set. The

achieved results in these case studies, indicate that the proposed approaches are efficient enough to detect and reconstruct famous attack scenarios indexed in such publicly available data sets. Finally, we discussed about the significant flexibility, extendability and efficiency of the *Pasargadae* framework in implementing various correlation and fusion approaches, and inspecting different attack scenarios.

CHAPTER 7 FIELD TEST-BASED EVALUATION

In the Chapter 4 of this thesis, we proposed the *Pasargadae* ontology-based context-aware event correlation framework as well as a new event correlation approach and a new alert fusion approach. Next, in Chapter 5, we described in detail the implementation process of *Pasargadae* framework in our lab and field test environment. Chapter 6, using some popular IDS evaluation data sets, evaluated the proposed event correlation and alert fusion approaches in the form of some case studies. As the provided case studies were limited because of not including the underlying contextual information, in this chapter, we concentrate on applying the proposed event correlation and alert fusion approaches in a real network to analyze ground truth network traffic. We mainly evaluate the performance and efficiency of the proposed approaches, and we show how these approaches behave in real world usages.

The rest of this chapter is organized as follows. In Section 7.1, we start by introducing Groupe Access Company and their gateway product as our field test environment. In Section 7.2, we describe in detail our testbed network architecture. In Section 7.3, we describe the field test scenario which was concentrated on the event sensors functionality test. In Section 7.4, we describe the second field test which is related to a targeted attack that compromises a web server. Section 7.5 is dedicated to the third field test which is related to a targeted attack that launches an internal DoS attack against Astersik VoIP Server. Finally, in Section 7.6, we describe a short summary of this chapter.

7.1 Groupe Access Company as Our Field Test Environment

In this section, we describe how we implemented the *Pasargadae* framework in a real environment in order to evaluate the proposed event correlation and alert fusion approaches from various perspectives. To this end, we conducted our experiments in Groupe Access company which is one of the leading hardware and information technology (IT) services firms in Canada. Groupe Access provides cloud-based firewall and network security monitoring services to many clients located in Canada and USA. One of the upcoming products of Groupe Access is MCB [1] which is currently under tests and experiments and will be released from February 2015. MCB mainly plays a gateway role for small and medium sized computer networks and consists of several components which will be used as the event sensors in our experiments:

— Openswan as a Site-to-Site VPN tool

1. MyCloud in a Box

— OpneVPN as a Client-Server VPN tool
— Linux IPtables as a Firewall
— Suricata as an IDS/IPS
— Asterisk as a VoIP PBX
— Squid as a web proxy
— SquidGuard as a web filtering tool
— WFS as a failover tool

Groupe Access intends to provide various cloud-based services, such as firewall, VPN, IDS/IPS, VoIP PBX, web proxy, web filtering, etc. via MCB which will be located in the computer networks of clients. In order to remotely configure and monitor the behavior of MCB in the targeted network, there is a Command and Control (C&C) department situated in Groupe Access that collects/issues various monitoring/configuration information from/to MCB. We use the information collected in the C&C department to test and evaluate our event and alert correlation system. In the following, first, we describe the testbed network architecture and the sensors layout. Next, we explain our experiments and results.

7.2 Testbed Network Architecture

Figure 7.1 illustrates the network architecture of our experiments. This architecture includes four distinct networks:

— Groupe Access network: this network is the main network of Groupe Access company. About 60 employees having various roles are connected to this network. MCB is connected through its eth0 to this network as a typical computer system, but in the promiscuous mode that allows MCB to watch all the traffic passing the entire network.
— Test-LAN: This network is considered as a small LAN for our tests. It includes a traffic generator and 2 client machines. MCB is the gateway of this network, i.e., all the incoming traffic and outgoing traffic pass MCB.
— Test-DMZ: This network is considered as a DeMilitarized Zone (DMZ) for our tests. It consists of servers that provide web, email, VoIP PBX and access control.
— Test-WAN: This network plays WAN role in our tests. It includes a traffic generator, 2 client machines, and an attacker. Using this network, we send various network traffic (normal and malicious) into the test-LAN to evaluate the performance of MCB's sensors.

As the traffic generator, we use the IXIA 400 [1] and the Ostinato [2] traffic generators for our tests. Depending on the type of test, the traffic generators can send distinct types of traffic in both the sides. Figure 7.2 illustrates IXIA's GUI, and Figure 7.3 illustrates Ostinatio's

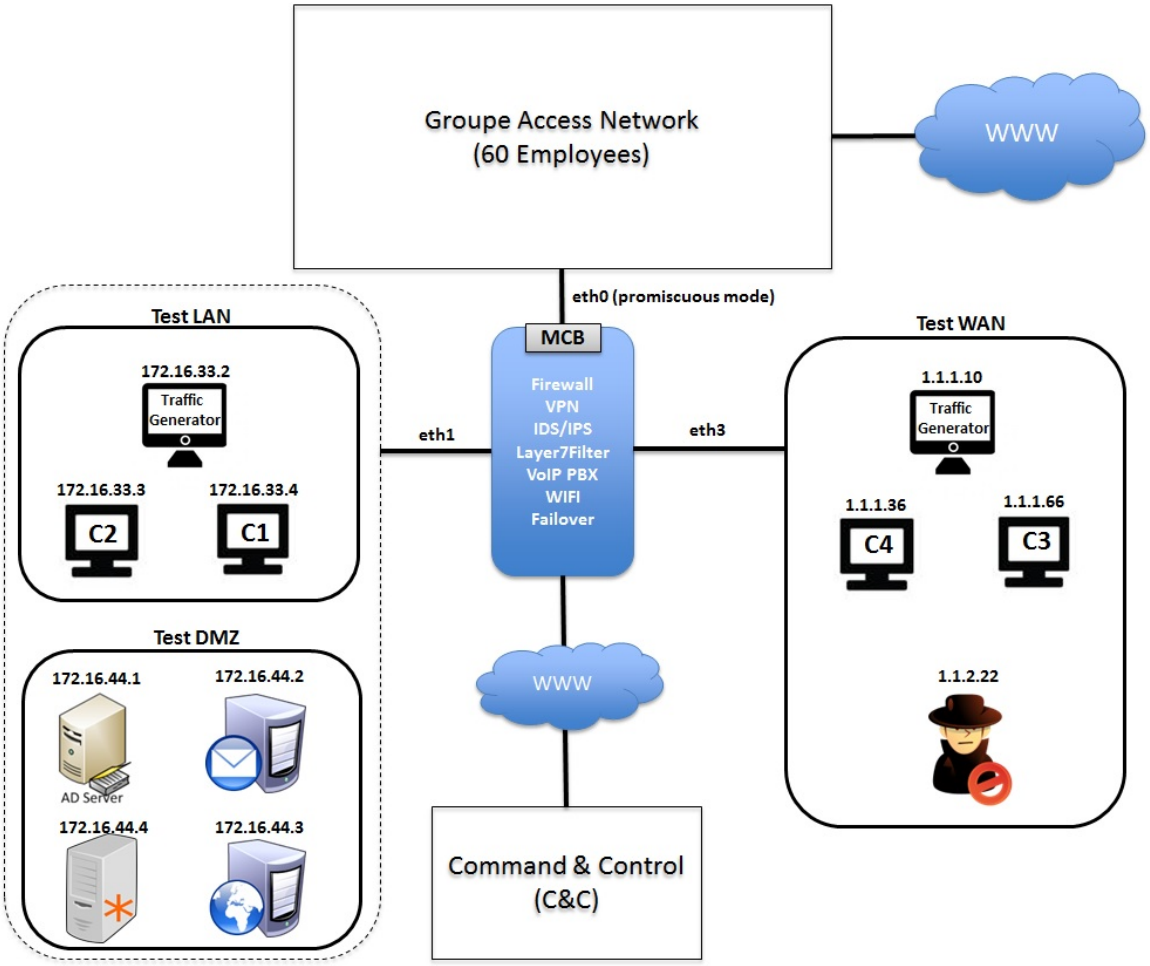

Figure 7.1 Field test network architecture

GUI configured for our experiments.

As the client machines, we use virtual machines with the following specifications:

- OS: Windows 7 SP1
- CPU: Intel (R) Core (TM) i5-2310 CPU @ 2.90GHz
- Memory: 2048

The Windows operating system is chosen because it would be possible to exploit a diverse set of known vulnerabilities against the testbed environment.

Table 7.1 presents a summary of the existing nodes in the Testbed network. The context ontology is populated using these information and some other information provided by network administrators. These information will be employed in the next sections during the event correlation process.

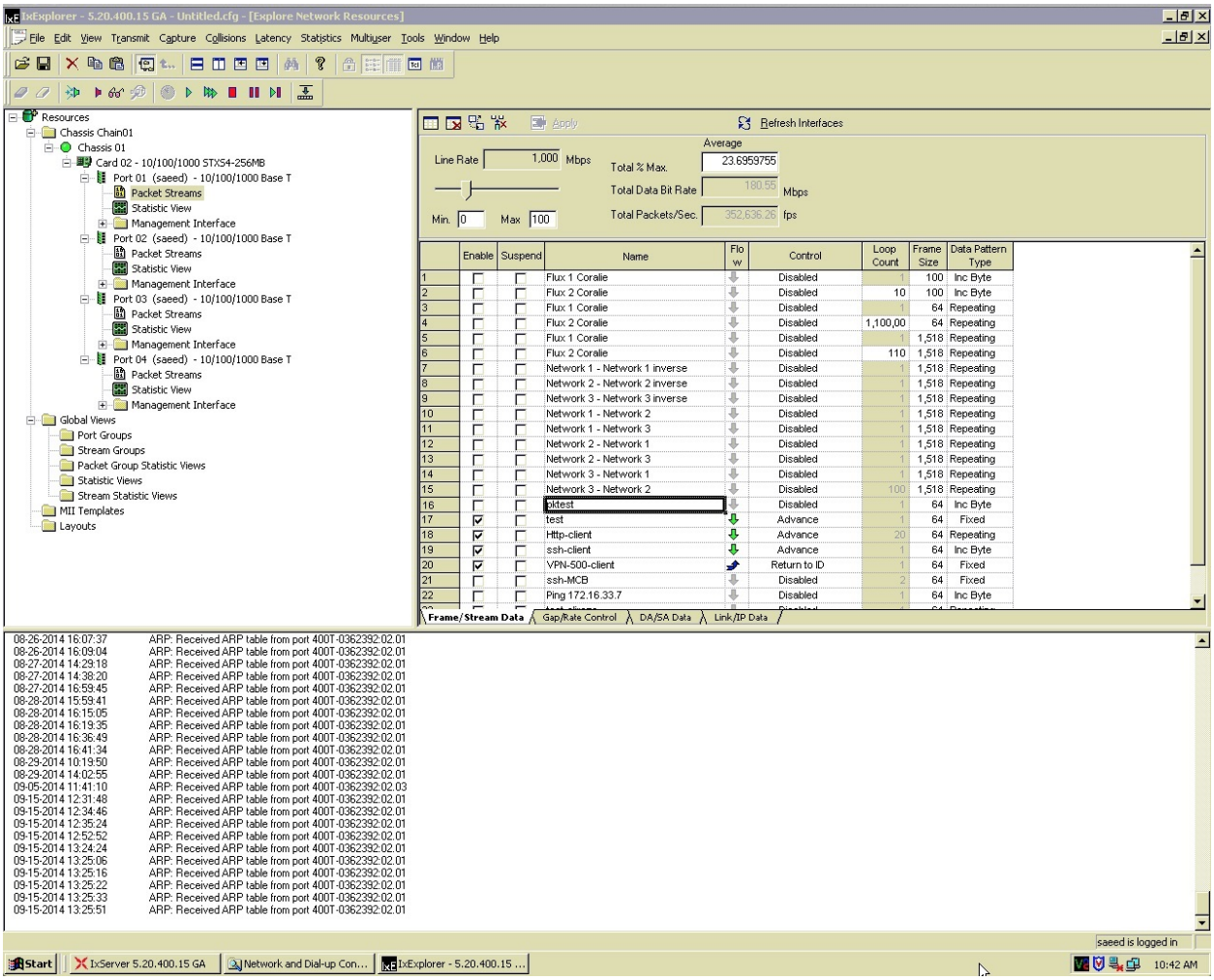

Figure 7.2 IXIA 400 GUI

7.3 Field Test 1: Sensors Functionality Test

In order to verify the functionality of the sensors integrated into MCB, we connected it to the described networks based on the testbed architecture. Next, considering the type of the sensors, various types of traffic were generated by traffic generators and client machines, and transferred into both the test LAN and test WAN. In the same time, Suricata IDPS was monitoring the whole traffic passing the Groupe Access Network. Table 7.2 lists some of the logs generated by these sensors. We received these event logs and alerts via Prelude Security Information Event Management system (SIEM) installed in the C&C servers. Receiving such event logs indicates on the sensors' correct functionality while inspecting their related events.

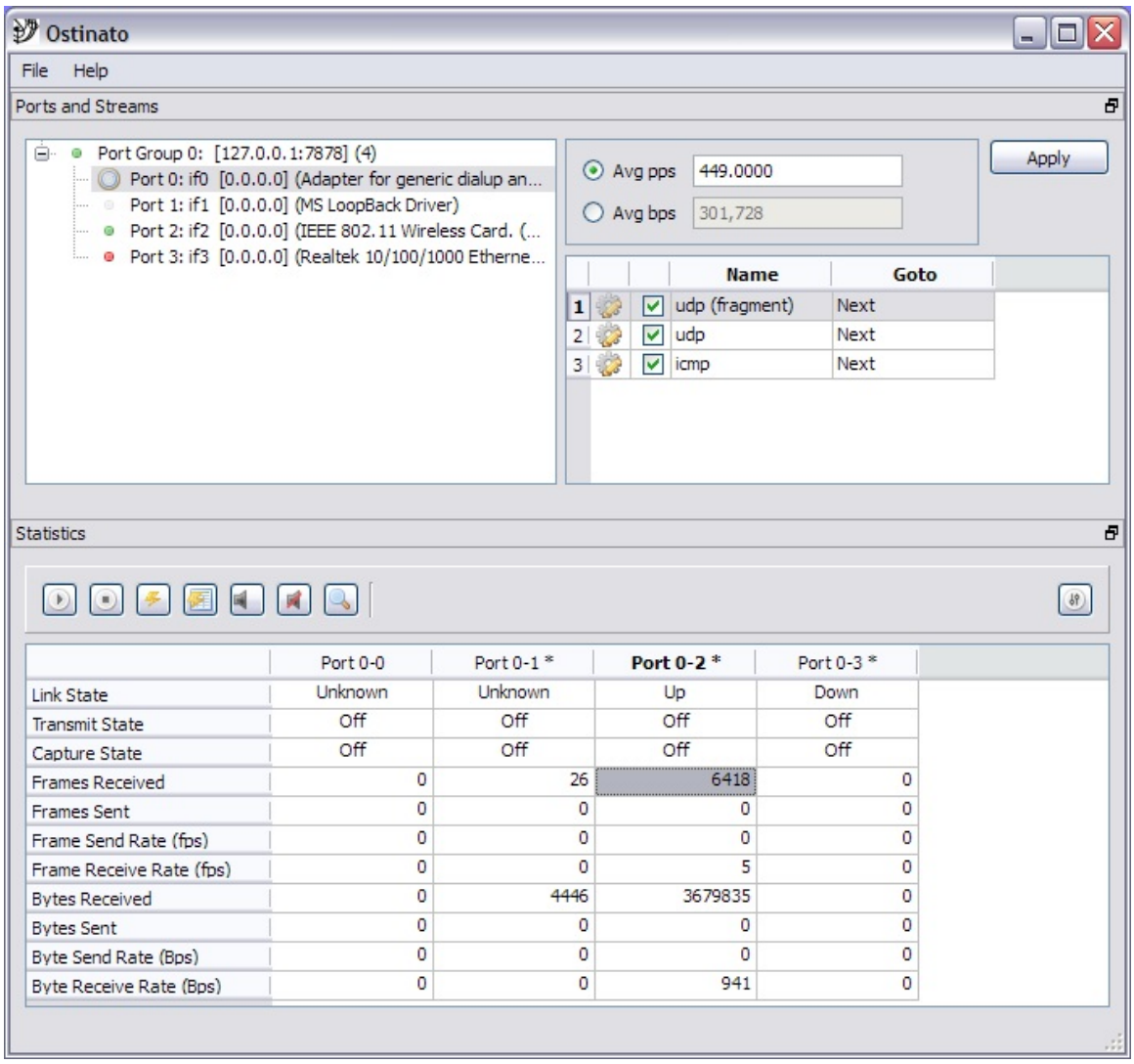

Figure 7.3 Ostinato's GUI

7.4 Field Test 2: A Targeted Attack to Compromise a Web Server

In this section, first, we describe a targeted attack scenario. Next, using the proposed event correlation system, we try to reconstruct the attack scenario. First of all, let us follow the steps taken by the attacker and let us consider what kind of artifacts would be picked up by event sensors.

As Figure 7.4 illustrates, to launch the attack, the attacker creates a malicious executable using Metasploit, and embeds a Meterpreter reverse TCP shell inside it. Next, the attacker sends a system upgrade email including the .exe file on behalf of admin@[...] to all the users of the test LAN (event 1, 2). Client 2 (C2) which gets compromised by the attacker, initiates

Table 7.1 Involved nodes in the testbed network architecture

Node	Role	OS	Application	Users	Usage Profile
172.16.33.2	Traffic Generator	Windows Server 2008	IXIA 400	Administrators	Permanent
172.16.33.4	Client Machine	Windows 7 SP1	ClamAV	Client 1	Login time: 8am - 5pm
172.16.33.3	Client Machine	Windows 7 SP1	Adobe Reader 8	Client 2	Login time: 8am - 5pm
172.16.44.1	AD Server	Windows Server 2012	Active Directory, OSSEC, ClamAV	Administrators	8am - 5pm
172.16.44.2	Email Server	UBUNTU 12.04	iRedMail, OSSEC, ClamAV	Administrators	Permanent
172.16.44.3	Web Server	Windows Server 2012	MSSQL, OSSEC, ClamAV	Administrators	Permanent
172.16.44.4	VoIP Server	UBUNTU 12.04	Asterisk, OSSEC, ClamAV	Administrators	8am - 5pm

the first session (events 4). Through this client, the attacker starts to scan the server subnet (172.16.44.0/24) collecting information (event 5). This scan identifies a Windows Server

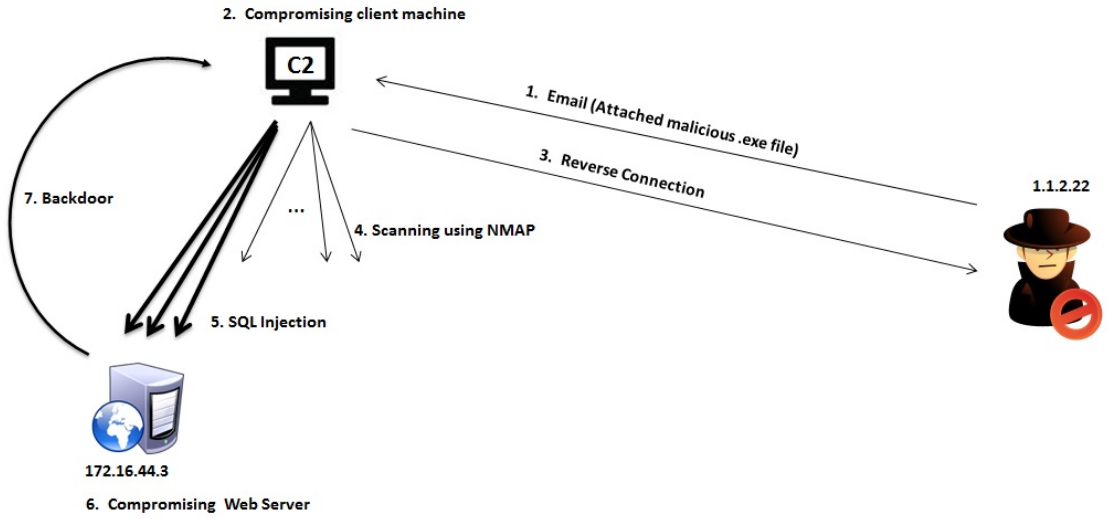

Figure 7.4 The targeted attack scenario to compromise a Web Server

Table 7.2 Some event logs generated by the employed event sensors

Classification	Source	Target	Analyzer	Time
ET POLICY Dropbox Client Broadcasting	172.16.51.32	172.16.51.255	Suricata	2014-08-21 17:21:47
SURICATA zero length padN option	fe80:0000:0000:0000: 14cb:c811:6202:a394	ff02:0000:0000:0000: 0000:0000:0000:0016	Suricata	2014-08-21 17:21:37
SURICATA ICMPv4 unknown type	172.16.52.212	224.0.0.1	Suricata	2014-08-21 17:23:05
UDP packet dropped	172.16.33.4	172.16.33.255	netfilter	2014-08-21 17:22:34
UDP packet dropped	172.16.33.4	172.16.33.255	netfilter	2014-08-21 17:22:34
UDP packet dropped	1.1.1.1	1.1.1.31	netfilter	2014-08-21 17:23:25
voip.ms	unknown	192.168.71.130	Asterisk	2014-08-21 18:41:50
voip.ms	unknown	192.168.71.130	Asterisk	2014-08-21 18:42:00
Log file rename		0xd79ef5ff.0x00000000	squid3	2014-08-21 06:25:04
Log file rename		0xd7a0477f.0x00000000	squid3	2014-08-22 06:25:03
VPN:.[AF_INET]42.2.2.2:81	local		OpenVpn	2014-08-18 21:16:10
VPN.Completed	unknown	unknown	OpenVpn	2014-08-18 21:16:10
Promiscuous mode detected		UBUNTU-MCB-PHY. developer.ca	kernel	2014-08-20 17:17:06
Promiscuous mode detected		UBUNTU-MCB-PHY. developer.ca	kernel	2014-08-21 17:03:45

2012 providing internal and external web services. The attacker identifies that there is an internal web application that uses MSSQL Server as its backend database, running on this server. This leads to the use of web application hacking techniques like SQL injection (event 6, 7, 8). Finally, the attacker using such techniques, compromises the web server (event 9, 10). Table 7.3 presents a summary of the generated events via the existing sensors in the underlying network.

After transferring the collected and integrated event logs into the event ontology, in this step, using Rules 4.1 and 4.2, we verify the relevance of these events to the underlying context in order to filter out non-relevant alerts. For this purpose, information of the context and vulnerability ontologies are employed. Table 7.1 presents a summary of the contextual information.

We continue to the first level meta-event creation phase, and we create the corresponding meta-events of the collected events. In our case, the correlation will regroup events 9 and 10 two sensors reporting on the same occurrence, in a meta-event . In a similar way, events 6, 7, 8 will be correlated together because of the shared source IP, target IP, and application. As Table 7.4 shows, the first level meta-events are the result of this phase. We make a second pass on the meta-events to correlate all the events, both host based and network based, to

Table 7.3 Event logs generated by the sensors during the Web Server attack

Event ID	Name	Sensor	Date	Source	Target	Tag
1	System Upgrade Email	iRedMail	2014-10-07 9:02:20	admin@ parseh.ca	testuser1@ parseh.ca	Step 1
2	System Upgrade Email	iRedMail	2014-10-07 9:02:24	admin@ parseh.ca	testuser2@ parseh.ca	Step 1
3	Local Exploit	Antivirus	2014-10-07 10:02:20	172.16.33.4	172.16.33.4	Step 2
4	Reverse Connection	TCPDUMP	2014-10-07 10:17:32	172.16.33.3	1.1.2.22	Step 3
5	Scanning	NIDS	2014-10-08 1:42:24	172.16.33.3	172.16.44.0/24	Step 4
6	HTTPWeb	NIDS	2014-10-08 2:19:41	172.16.33.3	172.16.44.3	Step 5
7	HTTPWeb	NIDS	2014-10-08 2:19:44	172.16.33.3	172.16.44.3	Step 5
8	HTTPWeb	NIDS	2014-10-08 2:19:47	172.16.33.3	172.16.44.3	Step 5
9	SQLInjection	HIDS	2014-10-08 2:20:19	172.16.44.3	172.16.44.3	Step 6
10	Bad Request	MSSQL	2014-10-08 2:20:21	172.16.44.3	172.16.44.3	Step 6
11	Reverse Connection	TCPDUMP	2014-10-08 2:25:30	172.16.33.3	1.1.2.22	Step 3

form a sequence of events (sessions) based on consequence and prerequisite relationships. We can then rebuild the entire attack scenario using the Rule 7.1.

Table 7.4 Event logs generated by the sensors during the Web Server attack

Meta-Event ID	Name	Sensor	Date	Source	Target	Tag
1	System Upgrade Email	iRedMail	2014-10-07 9:02:20	admin@ parseh.ca	testuser1@ parseh.ca	correlated with 3
2	System Upgrade Email	iRedMail	2014-10-07 9:02:24	admin@ parseh.ca	testuser2@ parseh.ca	correlated with 4
3	Local Exploit	Antivirus	2014-10-07 10:02:20	172.16.33.4	172.16.33.4	correlated with 1
4	Reverse Connection	TCPDUMP	2014-10-07 10:17:32	172.16.33.3	1.1.2.22	correlated with 2
5	Scanning	NIDS	2014-10-08 1:42:24	172.16.33.3	172.16.44.0/24	
6	HTTPWeb	NIDS	[2014-10-08 2:19:41, 2014-10-08 2:19:44, 2014-10-08 2:19:47]	172.16.33.3	172.16.44.3	correlated with 7, 8
7	[SQLInjection, Bad Request]	[HIDS, MSSQL]	[2014-10-08 2:20:19, 2014-10-08 2:20:21]	172.16.44.3	172.16.44.3	correlated with 6, 8
8	Reverse Connection	TCPDUMP	2014-10-08 2:25:30	172.16.33.3	1.1.2.22	correlated with 6, 7

$$\texttt{Event(?e1)} \wedge \texttt{Analyzer(?A1)} \wedge \texttt{hasAnalyzer(?e1, ?A1)} \wedge \texttt{hasType(?A1, "Email Server")} \wedge \texttt{User(?u1)} \wedge \texttt{hasEmail(?u1, ?em1)} \wedge \texttt{User(?u2)} \wedge \texttt{hasEmail(?u2, ?em2)} \wedge \texttt{hasSource(?e1, ?em1)} \wedge \texttt{hasTarget(?e1, ?em2)} \wedge \texttt{Event(?e2)} \wedge \texttt{Analyzer(?A2)} \wedge \texttt{hasAnalyzer(?e2, ?A2)} \wedge \texttt{hasType(?A2, "Packet Capture")} \wedge \texttt{Node(?n1)} \wedge \texttt{Node(?n2)} \wedge \texttt{hasSource(?e2, ?n2)} \wedge \texttt{hasTarget(?e2, ?n1)} \wedge \texttt{hasNode(?u2, ?n2)} \wedge \texttt{Event(?e3)} \wedge \texttt{Analyzer(?A3)} \wedge \texttt{hasAnalyzer(?e3, ?A3)} \wedge \texttt{hasType(?A3, "NIDS")} \wedge \texttt{hasSource(?e3, ?n2)} \wedge \texttt{Event(?e4)} \wedge \texttt{Analyzer(?A4)} \wedge \texttt{hasAnalyzer(?e4, ?A4)} \wedge \texttt{hasType(?A3, "HIDS")} \wedge \texttt{Node(?n3)} \wedge \texttt{hasSource(?e4, ?n3)} \wedge \texttt{Event(?e5)} \wedge \texttt{Analyzer(?A5)} \wedge \texttt{hasAnalyzer(?e5, ?A5)} \wedge \texttt{hasType(?A4, "DB")} \wedge \texttt{hasSource(?e5, ?n3)} \wedge \texttt{Event(?e6)} \wedge \texttt{hasSource(?e6, ?n3)} \wedge \texttt{hasTarget(?e2, ?n2)} \wedge \texttt{hasTarget(?e2, ?n2)} \longrightarrow \texttt{sqwrl:select(?e1, ?e2, ?e3, ?e4, ?e5, ?e6)} \tag{7.1}$$

Rule 7.1 correlates event and attack ontologies, and attempts to discover corresponding events for each step of the attack. If it finds at least one match regarding each step, the rule will be successful in detecting the whole attack scenario. This can now be presented to the security administrator instead of flooding him with a huge number of events. Figure 7.5 represents the result of the event correlation process.

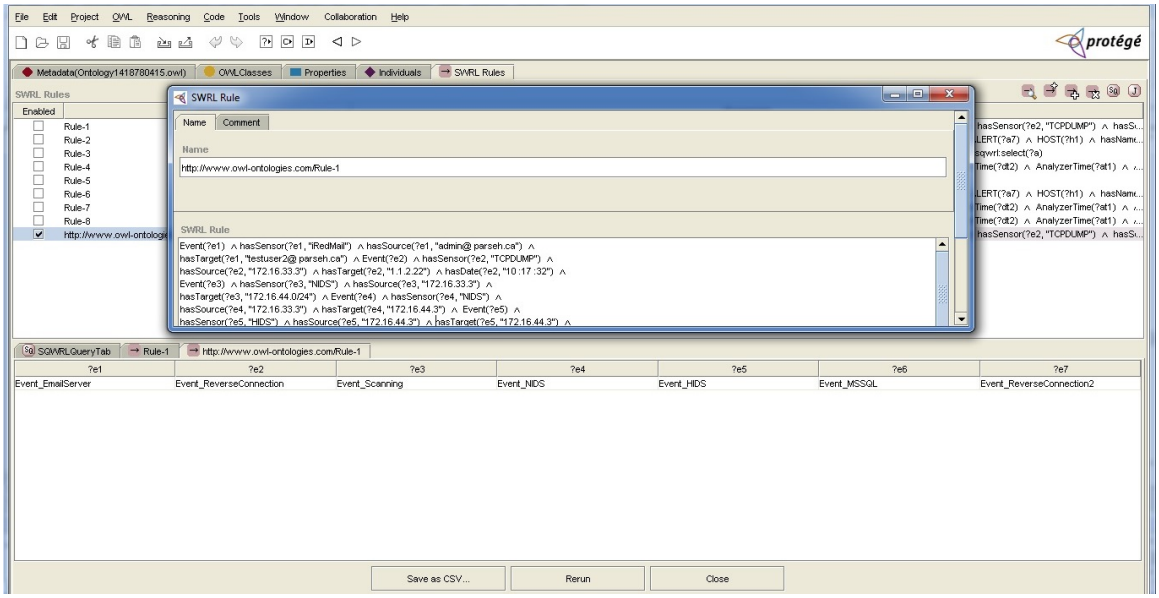

Figure 7.5 Correlating the event generated by the web server attack

7.5 Field Test 3: A Targeted Attack to Launch an Internal DoS Attack Against Asterisk VoIP Server

As another attack scenario, in this section, we describe a targeted attack causing an internal DoS attack against VoIP server. First of all, let us follow the steps taken by the attacker and let us consider what kind of artifacts would be picked up by event sensors.

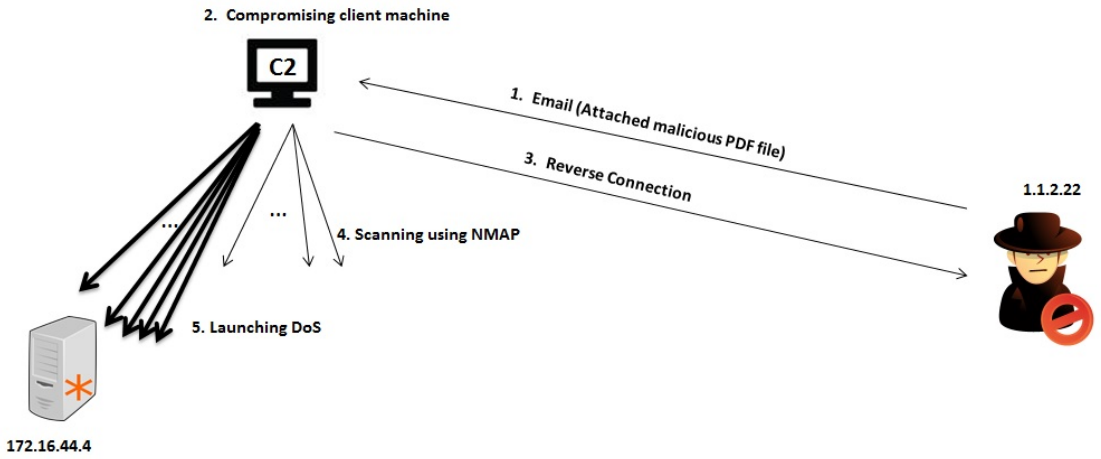

Figure 7.6 The targeted attack scenario to launch DoS attack against VoIP Server

$$\begin{aligned}
&\texttt{Event(?e1)} \wedge \texttt{Analyzer(?A1)} \wedge \texttt{hasAnalyzer(?e1, ?A1)} \wedge \\
&\texttt{hasType(?A1, "Email Server")} \wedge \texttt{User(?u1)} \wedge \texttt{hasEmail(?u1, ?em1)} \wedge \texttt{User(?u2)} \wedge \\
&\texttt{hasEmail(?u2, ?em2)} \wedge \texttt{hasSource(?e1, ?em1)} \wedge \texttt{hasTarget(?e1, ?em2)} \wedge \texttt{Event(?e2)} \\
&\wedge \texttt{Analyzer(?A2)} \wedge \texttt{hasAnalyzer(?e2, ?A2)} \wedge \texttt{hasType(?A2, "Packet Capture")} \wedge \\
&\texttt{Node(?n1)} \wedge \texttt{Node(?n2)} \wedge \texttt{hasSource(?e2, ?n2)} \wedge \texttt{hasTarget(?e2, ?n1)} \wedge \\
&\texttt{hasNode(?u2, ?n2)} \wedge \texttt{Event(?e3)} \wedge \texttt{Analyzer(?A3)} \wedge \texttt{hasAnalyzer(?e3, ?A3)} \wedge \\
&\texttt{hasType(?A3, "NIDS")} \wedge \texttt{hasSource(?e3, ?n2)} \wedge \texttt{Event(?e4)} \wedge \texttt{Node(?n3)} \wedge \\
&\texttt{hasSource(?e4, ?n2)} \wedge \texttt{hasTarget(?e4, ?n3)} \wedge \texttt{Event(?e5)} \wedge \texttt{Analyzer(?A4)} \wedge \\
&\texttt{hasAnalyzer(?e5, ?A4)} \wedge \texttt{hasType(?A4, "VoIP PBX")} \wedge \texttt{hasSource(?e5, ?n3)} \wedge \\
&\texttt{hasTarget(?e5, ?n3)} \wedge \texttt{Event(?e6)} \wedge \texttt{Analyzer(?A5)} \wedge \texttt{hasAnalyzer(?e6, ?A5)} \wedge \\
&\texttt{hasType(?A5, "HIDS")} \wedge \texttt{hasSource(?e6, ?n3)} \wedge \texttt{hasTarget(?e5, ?n3)} \\
&\longrightarrow \texttt{sqwrl:select(?e1, ?e2, ?e3, ?e4, ?e5, ?e6)}
\end{aligned} \tag{7.2}$$

As Figure 7.6 illustrates, to launch an attack, the attacker creates a malicious PDF file using Metasploit, and embeds a Meterpreter reverse TCP shell inside it. Then, the attacker sends a system upgrade email including the PDF file on behalf of admin@[...] to all the users of the test LAN (event 1, 2). Through client 2 (C2) compromised by the attacker, initiates the first session (events 4), the attacker starts to scan the server subnet (172.16.44.0/24) collecting information (event 5). This scan identifies an Asterisk v11 VoIP Server providing VoIP PBX services to the internal clients. This leads to launch a DoS attack against the VoIP Server.

For this purpose, the attacker installs SIPp tool on the client 2 machine. SIPp is an open source SIP traffic generator tool. Finally, using this tool, the attacker sends thousands of requests to the VoIP Server (event 6, 7, 8, 9, 10, 11) to make it unavailable for legitimate requests (events 12, 13). Table 7.5 presents a summary of the events generated by various sensors during this attack.

Table 7.5 Event logs generated by the sensors in the VoIP Server attack

Event ID	**Name**	**Sensor**	**Date**	**Source**	**Target**	**Tag**
1	System Upgrade Email	iRedMail	2014-10-08 9:02:20	admin@ parseh.ca	testuser1@ parseh.ca	Step 1
2	System Upgrade Email	iRedMail	2014-10-08 9:02:24	admin@ parseh.ca	testuser2@ parseh.ca	Step 1
3	Local Exploit	Antivirus	2014-10-08 10:02:20	172.16.33.4	172.16.33.4	Step 2
4	Reverse Connection	TCPDUMP	2014-10-08 10:17:32	172.16.33.3	1.1.2.22	Step 3
5	Scanning	NIDS	2014-10-08 10:42:24	172.16.33.3	172.16.44.0/24	Step 4
6	VoIP Requests	NIDS	2014-10-07 11:19:41	172.16.33.3	172.16.44.4	Step 5
7	VoIP Requests	NIDS	2014-10-07 11:19:41	172.16.33.3	172.16.44.4	Step 5
8	VoIP Requests	NIDS	2014-10-07 11:19:41	172.16.33.3	172.16.44.4	Step 5
9	Call Request	Asterisk	2014-10-07 11:19:42	172.16.44.4	172.16.44.4	Step 5
10	Call Request	Asterisk	2014-10-07 11:19:42	172.16.44.4	172.16.44.4	Step 5
11	Call Request	Asterisk	2014-10-07 11:19:42	172.16.44.4	172.16.44.4	Step 5
12	DoS attack	OSSEC	2014-10-07 11:20:10	172.16.44.4	172.16.44.4	Step 6

After transferring the collected and integrated event logs into the event ontology, in this step, using Rules 4.1 and 4.2, we verify the relevance of these events to the underlying context in order to filter out non-relevant alerts. For this purpose, information of the context and vulnerability ontologies are employed. Table 7.1 presents a summary of the contextual information.

We continue to the first level meta-event creation phase, and we create the corresponding

meta-events of the collected events. In our case, the correlation will regroup events 6, 7, 8 two sensors reporting on the same occurrence, in a meta-event . In a similar way, events 9, 10, 11, 12 will be correlated together because of the shared source IP, target IP, and application. As Table 7.6 shows, the first level meta-events are the result of this phase. We make a second pass on the meta-events to correlate all the events, both host based and network based, to form a sequence of events (sessions) based on consequence and prerequisite relationships. We can then rebuild the entire attack scenario using the Rule 7.1.

Rule 7.2 correlates event and attack ontologies, and attempts to discover corresponding events for each step of the attack. If it finds at least one match regarding each step, the rule will be successful in detecting the whole attack scenario. This can now be presented to the security administrator instead of flooding him with a huge number of events. Figure 7.7 represents the result of the event correlation process.

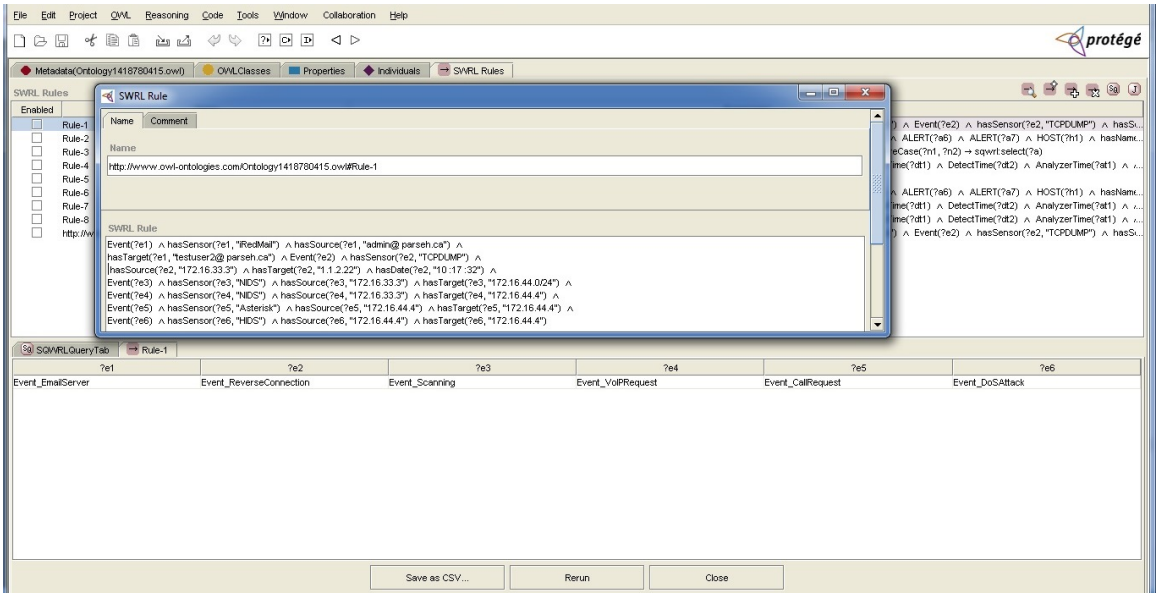

Figure 7.7 Correlating the events generated by the VoIP server attack

7.6 Summary

In this chapter, we evaluated *Pasargadae* framework along with the proposed event correlation and alert fusion approaches in the form some field tests in order to test them based on a grand truth network traffic. We started the chapter by introducing our field test environment (Groupe Access Company's network). Next, we described in detail our testbed network architecture. Event sensors function test was another step of our field test. Finally, we carried out two distinct attack scenarios (i.e. targeted attack against a Web Server and DoS

Table 7.6 Event logs generated by the sensors during the Web Server attack

Meta-Event ID	**Name**	**Sensor**	**Date**	**Source**	**Target**	**Tag**
1	System Upgrade Email	iRedMail	2014-10-08 9:02:20	admin@ parseh.ca	testuser1@ parseh.ca	correlated with 3
2	System Upgrade Email	iRedMail	2014-10-08 9:02:24	admin@ parseh.ca	testuser2@ parseh.ca	correlated with 4
3	Local Exploit	Antivirus	2014-10-08 10:02:20	172.16.33.4	172.16.33.4	correlated with 1
4	Reverse Connection	TCPDUMP	2014-10-08 10:17:32	172.16.33.3	1.1.2.22	correlated with 2
5	Scanning	NIDS	2014-10-08 1:42:24	172.16.33.3	172.16.44.0/24	
6	VoIP Requests	NIDS	[2014-10-07 11:19:41, 2014-10-07 11:19:41, 2014-10-07 11:19:41]	172.16.33.3	172.16.44.4	correlates with 7
7	[Call Request, DoS Attack]	[Asterisk, OSSEC]	[2014-10-07 11:19:42, 2014-10-07 11:19:42, 2014-10-07 11:19:42, 2014-10-07 11:20:10]	172.16.44.4	172.16.44.4	correlates with 6

attack against VoIP PBX Server), and employed our proposed event correlation approach to reconstruct the scenarios. We selected these scenarios to evaluate our proposed approaches because they include attacks that are based on recently emerged sophisticated techniques making current IDS and correlation techniques unable to detect them. The results obtained showed that the proposed contributions significantly improve IDS performance and capabilities in detecting currently existing complex multi-step attacks while reducing the rate of false positives, non-relevant and duplicate alerts.

CHAPTER 8 CONCLUSION

With the implementation and the employment of the *Pasargadae* ontology-based context-aware event correlation approach in a real computer network and under ground truth network traffic, and observing its significant flexibility and efficiency, we have finally reached our goal of improving current IDS, alert and event correlation research and technology.

In this research work, we discussed the shortcomings of current IDS, alert and event correlation systems technology, such as prone to alert flooding, false positives and false negatives, non-relevant alerts, continuous human interaction, limited scalability and flexibility, inability to detect zero-day and multi-step attacks, etc., and we explained that these shortcomings make IDS, alert and event correlation systems unreliable in real-world situations. To address these limitations, we proposed a number of contributions adding more capabilities and features to these systems, such as context-awareness, flexibility with various environments, automation, extendability, etc.

In Chapter 2, we reviewed research projects that have been proposed to address the mentioned shortcomings, such as machine learning and data mining based approaches, alert and event correlation based approaches, alert fusion based approaches, context-aware intrusion detection systems, distributed intrusion detection systems, ontology based intrusion detection systems, host-based IDS, etc. However, most of these works have focused on limited aspects of a comprehensive intrusion detection and alert correlation process, the problem as a whole has not been resolved. Machine learning based approaches mostly classify network events based on some features depending on behaviors observed with one type of event, and they do not take to account contextual information and events interrelationships. Alert and event correlation approaches mainly consider correlation only across multiple sensors of the same type having a common event and alert semantics (homogenous correlation), leaving it to security administrators to perform correlation across heterogeneous types of sensors. Ontology-based approaches have not taken full advantage of ontologies expressive power in terms of data modelling and logic reasoning. They also mostly focus on generic security concepts rather than the detailed steps of a comprehensive alert correlation approach. Context-aware intrusion and alert correlation approaches only consider partial contextual information, such as target configuration or network traffic, and do not allow for inclusion of other types of context concepts. Lack of automation is another main shortcoming of almost all these works that postpone the analysis of the alert interrelationships to be performed by security administrators.

Chapter 3 provided a background and basic knowledge required to follow the details of our proposed approaches. We presented an introduction about ontologies and their impact in computer security research. We also introduced Ontology Web Language Description Logic (OWL-DL) and Semantic Query-Enhanced Web Rule Language. A review on recent ontology-based intrusion detection systems was another part of this chapter.

Motivated by the aforementioned shortcomings, and in order to provide a common solution encompassing the advantages of all of these approaches, we introduced *Pasargadae* a context-aware and ontology-based automated event correlation framework that correlate across several heterogenous types of sensors and logs (e.g., NIDS, firewall, access control system, operating system, HIDS, antivirus, web server, and other applications), while providing a level of flexibility that would allow it to be used in the many different deployment scenarios that security analysts are likely to face. In order to make *Pasargadae* context-aware, we inspected contextual information to identify which aspects of context can be useful in improving IDS efficiency, with the goal of being able to import such context information from either explicit information in Configuration Management Systems (CMS) or from implicit information obtained by user and system profiling techniques. To make *Pasargadae* able to perform automated event correlation, and to improve its flexility, we designed a number of extensible ontologies incorporated within its correlation engine. In fact, the main idea behind *Pasargadae* is to use and leverage a template ontology containing base classes and some subclasses for the concepts of IT asset context, event logs, vulnerability and attack. These ontologies are then populated by instances either automatically through source-specific drivers (such as for IDMEF compliant event sensors), or manually for static information (such as context, vulnerability and attack information). The correlation engine is then implemented using logic rules written in Semantic Web Rule Language (SWRL) and Semantic Query-Enhanced Web Rule Language (SQWRL) based on the OWL description logic (OWL-DL). The ontologies and correlation rules described here are generic enough to i) implement as special cases other existing correlation approaches and, ii) be applied with minimal changes to different analysis scenarios.

Next, we introduced a context-aware and semantic-based event correlation and alert fusion approaches based on *Pasargadae*. The proposed event correlation approach, first, based on the network-based and host-based events' attributes, splits events into several groups. Then, it extracts pre-requisites and consequences of every group to create its correspondent meta-events. Finally, using topological sort algorithm, it prepares sequences of meta-events that outcomes existing attack scenarios. About the proposed alert fusion approach, the main idea is to collect alerts (decisions) made by different IDS sensors on the same event, and inspect them using contextual information to reduce redundant and non-relevant alarms and

false positives. While employing several different IDS sensors can improve overall detection capabilities, the incorporation of contextual information allows us to reduce false positives and non-relevant alerts. To do so in a manner that can be automated, but that yet can be easily extended to new concepts (richer concepts of context, alarm or vulnerability), we used ontologies and ontological engineering tools to represent knowledge and information about alerts, vulnerabilities, and contextual information.

In order to evaluate the proposed event correlation and alert fusion approaches, we implemented *Pasargadae* in our lab and field test environment. We employed various tools and methods during the implementation process. Protégé ontology editor and knowledge acquisition system was employed in order to design our ontologies. Protégé provides a collection of plug-ins in order for reasoning, storing, populating and querying the ontologies. Prelude Hybrid IDS was used as our event integration component to collects, normalizes, stores and aggregates event logs from a number of heterogenous event generators. Using this tool, we converted the collected logs into the IDMEF format analyzable by Protégé ontology editor. We described the challenges that we faced during implementation process, and a brief discussion on the scalability of the framework was explained.

Finally, in order to evaluate the performance (detection rate) and efficiency (detection accuracy) of the proposed context-aware and semantic-based event correlation and alert fusion approaches against realistic attack scenarios, we conducted two types of experiments: 1) case study, 2) filed study. First, we evaluated the proposed event correlation approach based on the UNB ISCX IDS evaluation data set. As another case study, we evaluated the proposed event correlation and fusion approaches based on the LLDDOS 1.0 attack scenario of the DARPA 2000 data set, on which we conducted a comparative evaluation of our approaches with Snort and ISS RealSecure, used separately and in combination. Next, as a filed study, we installed and configured the proposed event correlation approach in a real computer network for a two weeks period, and evaluated it against ground truth network traffic. The obtained results show that our proposed event correlation and alert fusion approaches significantly reduces false positives, duplicate, and non-relevant alerts, while improving the detection rate. Furthermore, more important than the reduction in false positives, duplicate, and non-relevant alerts, another important point of this work was to show the level of flexibility and extendability of such a framework. The sum of these contributions represents a significant improvement on the applicability and reliability of IDS in real-world situations.

Future works

We are concluding this thesis by presenting some applications and possible improvements of our proposed event correlation approach:

— We have not addressed at all the issue of scalability and performance of our approach. While ontologies are quite flexible and readily provide the benefits of abstraction, they are not always efficient at updating and quickly providing access to stored data. In the case of alert correlation systems of large IT infrastructures, the vast amounts of data involved are likely to make typical XML flat file or relational database storage unwieldily and inefficient for quick on-line alert correlation. While some specific data storage solutions such as object-oriented databases might help alleviate these problems, significant engineering challenges would have to be solved to make *Pasargadae* perform at the same line speeds as some current commercial-grade NIDS. Nonetheless, and even if future research does not readily solve these problems, we believe in the eventual applicability and usefulness of ontology-based approaches such as this, in situations where immediate online processing is not a key requirement, such as in network forensics in incident handling situations.

— Incorporating richer context into our analysis based on a sophisticated user and system profiling is another future research direction. To this end, we can employ machine learning techniques to analyze the behaviours of underlying users, systems and networks for a specific time period. Then, based on the collected information, a detailed profile can be considered for each of these entities. Importing these profiles into the context ontology, makes it complete enough to provide more reliable information for the correlation engine about the underlying context. As a result, the false positive and non-relevant alert rate will be significantly reduced.

— Adding an ontology of response (response ontology or action ontology) to issue appropriate responses in case of any ongoing attack is another considered future research direction. This ontology can be mostly dependent to the pre-defined policies of the underlying context. Because system automation is one of the main criteria that we have considered in our event correlation process, this ontology covers automated responses that a system can issue in case of any attack. Adding this ontology, makes the proposed event correlation system more applicable in Command and Control (C&C) units so that all the event logs from several branch are collected into a C&C unit for more analysis.

— Proposing an ontology-based and context-aware Intrusion Prevention System (IPS) to efficiently predict and prevent malicious behaviors is considered as another future research direction. To this end, we can combine machine learning-techniques and

ontological paradigms to analyze contextual information, vulnerabilities and happening events, and report potential malicious activities. This system is considered as a complementing part of the comprehensive *Pasargadae* Framework.

REFERENCES

[1] "Ixia leader in converged ip testing.". En ligne : http://www.ixiacom.com/

[2] "Ostinato traffic generator." 2013.

[3] "Security focus, url=http ://www.securityfocus.com".

[4] "Suricata : Open source ids / ips / nsm engine". En ligne : http://suricata-ids.org/

[5] "Unm intrusion detection system data set". En ligne : http://www.cs.unm.edu/~immsec/data-sets.htm

[6] "auditd : The linux audit daemon". En ligne : http://linux.die.net/man/8/auditd

[7] "iptables : Command line utility for configuring linux kernel firewall". En ligne : https://wiki.archlinux.org/index.php/iptables

[8] "iredmail email open source email server", 2007. En ligne : http://www.iredmail.org/

[9] T. Abbes, A. Bouhoula, et M. Rusinowitch, "Efficient decision tree for protocol analysis in intrusion detection", *International Journal of Security and Networks*, vol. 5, no. 4, pp. 220–235, 2010.

[10] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, et P. Steggles, "Towards a better understanding of context and context-awareness", dans *Handheld and ubiquitous computing*. Springer, 1999, pp. 304–307.

[11] A. Abraham, R. Jain, J. Thomas, et S. Y. Han, "D-scids : Distributed soft computing intrusion detection system", *Journal of Network and Computer Applications*, vol. 30, no. 1, pp. 81–98, 2007.

[12] S. Akbar, K. N. Rao, et J. Chandulal, "Implementing rule based genetic algorithm as a solution for intrusion detection system", *Int. J. Comput. Sci. Netw. Secur*, vol. 11, no. 8, p. 138, 2011.

[13] G. A. Ali et A. Jantan, "A new approach based on honeybee to improve intrusion detection system using neural network and bees algorithm", dans *Software Engineering and Computer Systems*. Springer, 2011, pp. 777–792.

[14] M. Almgren, U. Lindqvist, et E. Jonsson, “A multi-sensor model to improve automated attack detection”, dans *Recent Advances in Intrusion Detection*. Springer, 2008, pp. 291–310.

[15] N. B. Amor, S. Benferhat, et Z. Elouedi, “Naive bayes vs decision trees in intrusion detection systems”, dans *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 2004, pp. 420–424.

[16] D. Ariu et G. Giacinto, “A modular architecture for the analysis of http payloads based on multiple classifiers”, dans *Multiple Classifier Systems*. Springer, 2011, pp. 330–339.

[17] A. A. Assali, D. Lenne, et B. Debray, “Ontology development for industrial risk analysis”, dans *Information and Communication Technologies : From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*. IEEE, 2008, pp. 1–5.

[18] A. Avizienis, J.-C. Laprie, B. Randell, et C. Landwehr, “Basic concepts and taxonomy of dependable and secure computing”, *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, no. 1, pp. 11–33, 2004.

[19] S. Axelsson, “The base-rate fallacy and its implications for the difficulty of intrusion detection”, dans *Proceedings of the 6th ACM Conference on Computer and Communications Security*. ACM, 1999, pp. 1–7.

[20] F. Baader, *The description logic handbook : theory, implementation, and applications*. Cambridge university press, 2003.

[21] R. Bace et P. Mell, “Nist special publication on intrusion detection systems”, DTIC Document, Rapp. tech., 2001.

[22] M. Baldauf, S. Dustdar, et F. Rosenberg, “A survey on context-aware systems”, *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, no. 4, pp. 263–277, 2007.

[23] Z. Banković, D. Stepanović, S. Bojanić, et O. Nieto-Taladriz, “Improving network security using genetic algorithm approach”, *Computers & Electrical Engineering*, vol. 33, no. 5, pp. 438–451, 2007.

[24] Z. Bankovic, J. M. Moya, Á. Araujo, S. Bojanic, et O. Nieto-Taladriz, “A genetic algorithm-based solution for intrusion detection”, *Journal of Information Assurance and Security*, vol. 4, no. 3, pp. 192–199, 2009.

[25] D. Barbara, J. Couto, S. Jajodia, L. Popyack, et N. Wu, “Adam : Detecting intrusions by data mining”, dans *In Proceedings of the IEEE Workshop on Information Assurance and Security*. Citeseer, 2001.

[26] S. Barnum, “Common attack pattern enumeration and classification (capec) schema description”, *Cigital Inc, http ://capec. mitre. org/documents/documentation/CAPEC_Schema_Descr iption_v1*, vol. 3, 2008.

[27] E. Beqiri, “Neural networks for intrusion detection systems”, dans *Global Security, Safety, and Sustainability*. Springer, 2009, pp. 156–165.

[28] S. M. Botros, T. A. Diep, et M. D. Izenson, “Synthesis of anomalous data to create artificial feature sets and use of same in computer network intrusion detection systems”, Sep. 3 2013, uS Patent 8,527,776.

[29] S. T. Brugger, “Data mining methods for network intrusion detection”, *University of California at Davis*, 2004.

[30] A. A. Cárdenas, J. S. Baras, et K. Seamon, “A framework for the evaluation of intrusion detection systems”, dans *Security and Privacy, 2006 IEEE Symposium on*. IEEE, 2006, pp. 15–pp.

[31] A. Chan, W. W. Ng, D. S. Yeung, et E. C. Tsang, “Comparison of different fusion approaches for network intrusion detection using ensemble of rbfnn”, dans *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, vol. 6. IEEE, 2005, pp. 3846–3851.

[32] C.-C. Chang et C.-J. Lin, “Libsvm : a library for support vector machines”, *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[33] Z. Che et X. Ji, “An efficient intrusion detection approach based on hidden markov model and rough set”, dans *Machine Vision and Human-Machine Interface (MVHI), 2010 International Conference on*. IEEE, 2010, pp. 476–479.

[34] L. Chen et M. Aritsugi, “An svm-based masquerade detection method with online update using co-occurrence matrix”, dans *Detection of Intrusions and Malware & Vulnerability Assessment*. Springer, 2006, pp. 37–53.

[35] R.-C. Chen et S.-P. Chen, “Intrusion detection using a hybrid support vector machine based on entropy and tf-idf”, *International Journal of Innovative Computing, Information, and Control (IJICIC)*, vol. 4, no. 2, pp. 413–424, 2008.

[36] R.-C. Chen, K.-F. Cheng, et C.-F. Hsieh, "Using rough set and support vector machine for network intrusion detection", *arXiv preprint arXiv :1004.0567*, 2010.

[37] S.-B. Cho et H.-J. Park, "Efficient anomaly detection by modeling privilege flows using hidden markov model", *Computers & Security*, vol. 22, no. 1, pp. 45–55, 2003.

[38] C. A. D. B. Cid, "Ossec, open source host-based intrusion detection system", *Web, September*, 2008.

[39] L. Coppolino, S. D'Antonio, I. A. Elia, et L. Romano, "From intrusion detection to intrusion detection and diagnosis : An ontology-based approach", dans *Software Technologies for Embedded and Ubiquitous Systems*. Springer, 2009, pp. 192–202.

[40] E. Corchado et Á. Herrero, "Neural visualization of network traffic data for intrusion detection", *Applied Soft Computing*, vol. 11, no. 2, pp. 2042–2056, 2011.

[41] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein *et al.*, *Introduction to algorithms*. MIT press Cambridge, 2001, vol. 2.

[42] I. Corporation, "IBM RealSecure", `http://www-935.ibm.com/services/us/en/it-services/express-managed-protection-services-for-server.html`.

[43] F. Cuppens et A. Miege, "Alert correlation in a cooperative intrusion detection framework", dans *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*. IEEE, 2002, pp. 202–215.

[44] F. Cuppens et R. Ortalo, "LAMBDA : A language to model a database for detection of attacks", dans *Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection*, série RAID '00. London, UK, UK : Springer-Verlag, 2000, pp. 197–216.

[45] ——, "Lambda : A language to model a database for detection of attacks", dans *Recent advances in intrusion detection*. Springer, 2000, pp. 197–216.

[46] CVE, "common vulnerabilities exposures (CVE), the key to information sharing", `http://cve.mitre.org/`.

[47] H. Debar, D. Curry, et B. Feinstein, "The intrusion detection message exchange format (idmef)", 2007.

[48] H. Debar et A. Wespi, “Aggregation and correlation of intrusion-detection alerts”, dans *Recent Advances in Intrusion Detection*. Springer, 2001, pp. 85–103.

[49] H. Debar, M. Dacier, et A. Wespi, “Towards a taxonomy of intrusion-detection systems”, *Computer Networks*, vol. 31, no. 8, pp. 805–822, 1999.

[50] H. Debar, D. A. Curry, et B. S. Feinstein, “The intrusion detection message exchange format (idmef)”, 2007.

[51] D. K. Denatious et A. John, “Survey on data mining techniques to enhance intrusion detection”, dans *Computer Communication and Informatics (ICCCI), 2012 International Conference on*. IEEE, 2012, pp. 1–5.

[52] G. Denker, L. Kagal, T. Finin, M. Paolucci, et K. Sycara, “Security for daml web services : Annotation and matchmaking”, dans *The Semantic Web-ISWC 2003*. Springer, 2003, pp. 335–350.

[53] G. Denker, S. Nguyen, et A. Ton, “Owl-s semantics of security web services : A case study”, dans *The Semantic Web : Research and Applications*. Springer, 2004, pp. 240–253.

[54] G. Denker, L. Kagal, et T. Finin, “Security in the semantic web using owl”, *Information Security Technical Report*, vol. 10, no. 1, pp. 51–58, 2005.

[55] R. Deraison *et al.*, “The nessus project”, *see http ://www. nessus. org*, 2002.

[56] J. E. Dickerson et J. A. Dickerson, “Fuzzy network profiling for intrusion detection”, dans *Fuzzy Information Processing Society, 2000. NAFIPS. 19th International Conference of the North American*. IEEE, 2000, pp. 301–306.

[57] J. E. Dickerson, J. Juslin, O. Koukousoula, et J. A. Dickerson, “Fuzzy intrusion detection”, dans *Ifsa world congress and 20th nafips international conference, 2001. joint 9th*, vol. 3. IEEE, 2001, pp. 1506–1510.

[58] P. Dokas, L. Ertoz, V. Kumar, A. Lazarevic, J. Srivastava, et P.-N. Tan, “Data mining for network intrusion detection”, dans *Proc. NSF Workshop on Next Generation Data Mining*, 2002, pp. 21–30.

[59] S. T. Eckmann, G. Vigna, et R. A. Kemmerer, “STATL : An attack language for state-based intrusion detection”, *Journal of Computer Security*, vol. 10, no. 1, pp. 71–103, 2002.

[60] A. Ekelhart, S. Fenz, M. Klemen, et E. Weippl, “Security ontologies : Improving quantitative risk analysis”, dans *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on.* IEEE, 2007, pp. 156a–156a.

[61] H. T. Elshoush et I. M. Osman, “Alert correlation in collaborative intelligent intrusion detection systems—a survey”, *Applied Soft Computing*, vol. 11, no. 7, pp. 4349–4365, 2011.

[62] D. M. Farid, N. Harbi, E. Bahri, M. Z. Rahman, et C. M. Rahman, “Attacks classification in adaptive intrusion detection using decision tree”, *World Academy of Science, Engineering and Technology*, vol. 63, pp. 86–90, 2010.

[63] D. M. Farid, N. Harbi, et M. Z. Rahman, “Combining naive bayes and decision tree for adaptive intrusion detection.” *International Journal of Network Security & Its Applications*, vol. 2, no. 2, 2010.

[64] M. Feilner, *OpenVPN : Building and integrating virtual private networks.* Packt Publishing Ltd, 2006.

[65] S. Fenz et A. Ekelhart, “Formalizing information security knowledge”, dans *Proceedings of the 4th international Symposium on information, Computer, and Communications Security.* ACM, 2009, pp. 183–194.

[66] D. G. Firesmith, “A taxonomy of safety-related requirements”, dans *International Workshop on High Assurance Systems (RHAS’05)*, 2005.

[67] J. J. Flores, A. Antolino, et J. M. Garcia, “Evolving hmms for network anomaly detection–learning through evolutionary computation”, dans *Networking and Services (ICNS), 2010 Sixth International Conference on.* IEEE, 2010, pp. 271–276.

[68] S. C. for Biomedical Informatics Research, “Protege ontology editor and knowledge representation system”, `http://protege.stanford.edu/`.

[69] E. Friedman-Hill *et al.*, “Jess, the rule engine for the java platform”, 2003.

[70] F. Gagnon, F. Massicotte, et B. Esfandiari, “Using contextual information for ids alarm classification”, dans *Detection of Intrusions and Malware, and Vulnerability Assessment.* Springer, 2009, pp. 147–156.

[71] D. Geneiatakis et C. Lambrinoudakis, “An ontology description for sip security flaws”, *Computer Communications*, vol. 30, no. 6, pp. 1367–1374, 2007.

[72] L.-z. Geng et H.-b. Jia, "A novel intrusion detection scheme for network-attached storage based on multi-source information fusion", dans *Computational Intelligence and Security, 2009. CIS'09. International Conference on*, vol. 2. IEEE, 2009, pp. 469–473.

[73] G. Giacinto, F. Roli, et L. Didaci, "Fusion of multiple classifiers for intrusion detection in computer networks", *Pattern recognition letters*, vol. 24, no. 12, pp. 1795–1803, 2003.

[74] R. H. Gong, M. Zulkernine, et P. Abolmaesumi, "A software implementation of a genetic algorithm based approach to network intrusion detection", dans *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2005 and First ACIS International Workshop on Self-Assembling Wireless Networks. SNPD/SAWN 2005. Sixth International Conference on.* IEEE, 2005, pp. 246–253.

[75] T. R. Gruber, "A translation approach to portable ontology specifications", *Knowledge acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[76] G. Gu, P. Fogla, D. Dagon, W. Lee, et B. Skorić, "Measuring intrusion detection capability : An information-theoretic approach", dans *Proceedings of the 2006 ACM Symposium on Information, computer and communications security.* ACM, 2006, pp. 90–101.

[77] G. Gu, A. A. Cárdenas, et W. Lee, "Principled reasoning and practical applications of alert fusion in intrusion detection systems", dans *Proceedings of the 2008 ACM symposium on Information, computer and communications security.* ACM, 2008, pp. 136–147.

[78] M. Gudadhe, P. Prasad, et K. Wankhade, "A new data mining based network intrusion detection model", dans *Computer and Communication Technology (ICCCT), 2010 International Conference on.* IEEE, 2010, pp. 731–735.

[79] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, et D. Wolber, "A network security monitor", dans *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on.* IEEE, 1990, pp. 296–304.

[80] A. Herzog, N. Shahmehri, et C. Duma, "An ontology of information security", *International Journal of Information Security and Privacy (IJISP)*, vol. 1, no. 4, pp. 1–23, 2007.

[81] S. Hettich et S. Bay, "Kdd cup 1999 data", *UCI KDD Archive [http ://kdd. ics. uci. edu//databases/kddcup99/kddcup99. html]*, 1999.

[82] X. Hoang et J. Hu, "An efficient hidden markov model training scheme for anomaly intrusion detection of server applications based on system calls", dans *Networks, 2004.(ICON 2004). Proceedings. 12th IEEE International Conference on*, vol. 2. IEEE, 2004, pp. 470–474.

[83] X. D. Hoang, J. Hu, et P. Bertok, "A program-based anomaly intrusion detection scheme using multiple detection engines and fuzzy inference", *Journal of Network and Computer Applications*, vol. 32, no. 6, pp. 1219–1228, 2009.

[84] T. Holz, "13 security measurements and metrics for networks", *Dependability Metrics*, p. 157, 2008.

[85] M. S. Hoque, M. Mukit, M. Bikas, A. Naser *et al.*, "An implementation of intrusion detection system using genetic algorithm", *arXiv preprint arXiv :1204.1336*, 2012.

[86] S.-J. Horng, M.-Y. Su, Y.-H. Chen, T.-W. Kao, R.-J. Chen, J.-L. Lai, et C. D. Perkasa, "A novel intrusion detection system based on hierarchical clustering and support vector machines", *Expert systems with Applications*, vol. 38, no. 1, pp. 306–313, 2011.

[87] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean *et al.*, "Swrl : A semantic web rule language combining owl and ruleml", *W3C Member submission*, vol. 21, p. 79, 2004.

[88] J. Hu, X. Yu, D. Qiu, et H.-H. Chen, "A simple and efficient hidden markov model scheme for host-based anomaly intrusion detection", *Network, IEEE*, vol. 23, no. 1, pp. 42–47, 2009.

[89] W. Hu, J. Li, et Q. Gao, "Intrusion detection engine based on dempster-shafer's theory of evidence", dans *Communications, Circuits and Systems Proceedings, 2006 International Conference on*, vol. 3. IEEE, 2006, pp. 1627–1631.

[90] Y. Hu, "TIAA : A toolkit for intrusion alert analysis", 2004.

[91] T. Internordia *et al.*, "Squidguard filter".

[92] G. Isaza, A. Castillo, M. López, et L. Castillo, "Towards ontology-based intelligent model for intrusion detection and prevention", dans *Computational Intelligence in Security for Information Systems*. Springer, 2009, pp. 109–116.

[93] V. Ivanov, M. Knorr, et J. Leite, “Nohr : Querying sc with non-monotonic rules”, *ISWC 2013 Posters & Demos*, p. 17.

[94] M. Jakobsson, X. Wang, et S. Wetzel, “Stealth attacks in vehicular technologies”, dans *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, vol. 2. IEEE, 2004, pp. 1218–1222.

[95] H. Jiang et J. Ruan, “The application of genetic neural network in network intrusion detection.” *Journal of Computers*, vol. 4, no. 12, 2009.

[96] M. J. Kabir, *Apache Server Bible*. IDG Books Worldwide, Inc., 1998.

[97] M. Karyda, T. Balopoulos, S. Dritsas, L. Gymnopoulos, S. Kokolakis, C. Lambrinoudakis, et S. Gritzalis, “An ontology for secure e-government applications”, dans *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on*. IEEE, 2006, pp. 5–pp.

[98] R. Khanna et H. Liu, “System approach to intrusion detection using hidden markov model”, dans *Proceedings of the 2006 international conference on Wireless communications and mobile computing*. ACM, 2006, pp. 349–354.

[99] A. Kim, J. Luo, et M. Kang, *Security ontology for annotating resources*. Springer, 2005.

[100] L. Koc, T. A. Mazzuchi, et S. Sarkani, “A network intrusion detection system based on a hidden naïve bayes multiclass classifier”, *Expert Systems with Applications*, vol. 39, no. 18, pp. 13 492–13 500, 2012.

[101] C. Kruegel, D. Mutz, W. Robertson, et F. Valeur, “Bayesian event classification for intrusion detection”, dans *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*. IEEE, 2003, pp. 14–23.

[102] V. K. Kshirsagar, S. M. Tidke, et S. Vishnu, “Intrusion detection system using genetic algorithm and data mining : An overview”, *International Journal of Computer Science and Informatics ISSN (PRINT)*, pp. 2231–5292, 2012.

[103] C. M. Kuok, A. Fu, et M. H. Wong, “Mining fuzzy association rules in databases”, *ACM Sigmod Record*, vol. 27, no. 1, pp. 41–46, 1998.

[104] C. E. Landwehr, A. R. Bull, J. P. McDermott, et W. S. Choi, “A taxonomy of computer program security flaws”, *ACM Computing Surveys (CSUR)*, vol. 26, no. 3, pp. 211–254,

1994.

[105] W. Lee, S. J. Stolfo, et K. W. Mok, "A data mining framework for building intrusion detection models", dans *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on.* IEEE, 1999, pp. 120–132.

[106] W. Lee, S. J. Stolfo, P. K. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop, et J. Zhang, "Real time data mining-based intrusion detection", dans *DARPA Information Survivability Conference &amp ; Exposition II, 2001. DISCEX'01. Proceedings*, vol. 1. IEEE, 2001, pp. 89–100.

[107] W. Li et S. Tian, "An ontology-based intrusion alerts correlation system", *Expert Systems with Applications*, vol. 37, no. 10, pp. 7138–7146, 2010.

[108] W. Li, "Using genetic algorithm for network intrusion detection", *Proceedings of the United States Department of Energy Cyber Security Group*, pp. 1–8, 2004.

[109] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai, et K. Dai, "An efficient intrusion detection system based on support vector machines and gradually feature removal method", *Expert Systems with Applications*, vol. 39, no. 1, pp. 424–430, 2012.

[110] O. Linda, T. Vollmer, et M. Manic, "Neural network based intrusion detection system for critical infrastructures", dans *Neural Networks, 2009. IJCNN 2009. International Joint Conference on.* IEEE, 2009, pp. 1827–1834.

[111] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, et K. Das, "The 1999 darpa off-line intrusion detection evaluation", *Computer networks*, vol. 34, no. 4, pp. 579–595, 2000.

[112] R. P. Lippmann, I. Graf, D. Wyschogrod, S. E. Webster, D. J. Weber, et S. Gorton, "The 1998 darpa/afrl off-line intrusion detection evaluation", dans *Proc. of the First Intl. Workshop on Recent Advances in Intrusion Detection (RAID)*, 1998.

[113] C.-C. Lo, C.-C. Huang, et J. Ku, "A cooperative intrusion detection system framework for cloud computing networks", dans *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on.* IEEE, 2010, pp. 280–284.

[114] W.-Y. Loh, "Classification and regression trees", *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 14–23, 2011.

[115] G. F. Lyon, *Nmap Network Scanning : The Official Nmap Project Guide to Network Discovery and Security Scanning.* USA : Insecure, 2009.

[116] P. Mahler, *VoIP Telephony with Asterisk*. Signate, 2005.

[117] F. Massicotte, M. Couture, Y. Labiche, et L. Briand, "Context-based intrusion detection using snort, nessus and bugtraq databases." dans *PST*, 2005.

[118] M. Meier, N. Bischof, et T. Holz, "Shedel—a simple hierarchical event description language for specifying attack signatures", dans *Security in the Information Society*. Springer, 2002, pp. 559–571.

[119] C. Michel et L. Mé, "Adele : an attack description language for knowledge-based intrusion detection", dans *Trusted Information*. Springer, 2001, pp. 353–368.

[120] MIT Lincoln Laboratory, "2000 DARPA intrusion detection scenario specific data sets", 2000.

[121] Mitre Corporation, "A standardized common event expression (CEE) for event interoperability".

[122] B. Morin et H. Debar, "Correlation of intrusion symptoms : an application of chronicles", dans *Recent Advances in Intrusion Detection*. Springer, 2003, pp. 94–112.

[123] B. Morin, L. Mé, H. Debar, et M. Ducassé, "M2d2 : A formal data model for ids alert correlation", dans *Recent Advances in Intrusion Detection*. Springer, 2002, pp. 115–137.

[124] S. Mukherjee et N. Sharma, "Intrusion detection using naive bayes classifier with feature reduction", *Procedia Technology*, vol. 4, pp. 119–128, 2012.

[125] S. Mukkamala, G. Janoski, et A. Sung, "Intrusion detection using neural networks and support vector machines", dans *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, vol. 2. IEEE, 2002, pp. 1702–1707.

[126] S. A. Mulay, P. Devale, et G. Garje, "Intrusion detection system using support vector machine and decision tree", *International Journal of Computer Applications*, vol. 3, no. 3, pp. 40–43, 2010.

[127] A. P. Muniyandi, R. Rajeswari, et R. Rajaram, "Network anomaly detection by cascading k-means clustering and c4. 5 decision tree algorithm", *Procedia Engineering*, vol. 30, pp. 174–182, 2012.

[128] P. G. Neumann et P. A. Porras, “Experience with emerald to date.” dans *Workshop on Intrusion Detection and Network Monitoring*, 1999, pp. 73–80.

[129] NVD, “National vulnerability database (NVD), automatin vulnerability management, security measurement, and compliance checking”, `http://nvd.nist.gov/`.

[130] C. Nyulas, M. O’Connor, et S. Tu, “Datamaster–a plug-in for importing schemas and data from relational databases into protege”, dans *Proceedings of the 10th International Protege Conference*, 2007.

[131] M. O’Connor et A. Das, “Sqwrl : a query language for owl”, dans *Proc. of 6th OWL : Experiences and Directions Workshop (OWLED2009)*, 2009.

[132] C.-M. Ou, “Host-based intrusion detection systems adapted from agent-based artificial immune systems”, *Neurocomputing*, vol. 88, pp. 78–86, 2012.

[133] M. Panda et M. R. Patra, “Network intrusion detection using naive bayes”, *International journal of computer science and network security*, vol. 7, no. 12, pp. 258–263, 2007.

[134] B. Parsia et E. Sirin, “Pellet : An OWL-DL reasoner”, dans *Third International Semantic Web Conference-Poster*, 2004, p. 18.

[135] Z. Pawlak, “Rough sets : theoretical aspects of reasoning about data, system theory, knowledge engineering and problem solving, vol. 9”, 1991.

[136] V. Paxson, “Bro : a system for detecting network intruders in real-time”, *Computer networks*, vol. 31, no. 23, pp. 2435–2463, 1999.

[137] R. Perdisci, G. Gu, et W. Lee, “Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems”, dans *Data Mining, 2006. ICDM’06. Sixth International Conference on.* IEEE, 2006, pp. 488–498.

[138] P. A. Porras et P. G. Neumann, “Emerald : Event monitoring enabling response to anomalous live disturbances”, dans *Proceedings of the 20th national information systems security conference*, 1997, pp. 353–365.

[139] M. S. Prasad, A. V. Babu, et M. K. B. Rao, “An intrusion detection system architecture based on neural networks and genetic algorithms”, *International Journal of Computer Science and Management Research*, vol. 2, 2013.

[140] J. R. Quinlan, *C4. 5 : programs for machine learning*. Morgan kaufmann, 1993, vol. 1.

[141] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition", *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[142] E. Raftopoulos, M. Egli, et X. Dimitropoulos, "Shedding light on log correlation in network forensics analysis", dans *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2013, pp. 232–241.

[143] Š. Raudys et F. Roli, "The behavior knowledge space fusion method : Analysis of generalization error and strategies for performance improvement", dans *Multiple Classifier Systems*. Springer, 2003, pp. 55–64.

[144] M. Roesch *et al.*, "Snort : Lightweight intrusion detection for networks." dans *LISA*, vol. 99, 1999, pp. 229–238.

[145] S. Roschke, F. Cheng, et C. Meinel, "A new alert correlation algorithm based on attack graph", dans *Computational Intelligence in Security for Information Systems*. Springer, 2011, pp. 58–67.

[146] S. Saad et I. Traore, "Extracting attack scenarios using intrusion semantics", dans *Foundations and practice of security*. Springer, 2013, pp. 278–292.

[147] S. Sangeetha, S. Haripriya, S. M. Priya, V. Vaidehi, et N. Srinivasan, "Fuzzy rule-base based intrusion detection system on application layer", dans *Recent Trends in Network Security and Applications*. Springer, 2010, pp. 27–36.

[148] S. L. Scott, "A bayesian paradigm for designing intrusion detection systems", *Computational statistics & data analysis*, vol. 45, no. 1, pp. 69–83, 2004.

[149] M. A. Sekeh et B. Maarof, "Fuzzy intrusion detection system via data mining technique with sequences of system calls", dans *Information Assurance and Security, 2009. IAS'09. Fifth International Conference on*, vol. 1. IEEE, 2009, pp. 154–157.

[150] R. Shearer, B. Motik, et I. Horrocks, "Hermit : A highly-efficient owl reasoner." dans *OWLED*, vol. 432, 2008.

[151] A. Shiravi, H. Shiravi, M. Tavallaee, et A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection", *Computers & Security*, vol. 31, no. 3, pp. 357–374, 2012.

[152] C. Siaterlis et B. Maglaris, “Towards multisensor data fusion for dos detection”, dans *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 2004, pp. 439–446.

[153] S. Sinha, F. Jahanian, et J. M. Patel, “Wind : Workload-aware intrusion detection”, dans *Recent Advances in Intrusion Detection*. Springer, 2006, pp. 290–310.

[154] A. Siraj, R. B. Vaughn, et S. M. Bridges, “Intrusion sensor data fusion in an intelligent intrusion detection system architecture”, dans *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*. IEEE, 2004, pp. 10–pp.

[155] E. Sirin et B. Parsia, “Sparql-dl : Sparql query for owl-dl.” dans *OWLED*, vol. 258, 2007.

[156] S. S. Sivatha Sindhu, S. Geetha, et A. Kannan, “Decision tree based light weight intrusion detection using a wrapper approach”, *Expert Systems with applications*, vol. 39, no. 1, pp. 129–141, 2012.

[157] S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C.-L. Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance *et al.*, “Dids (distributed intrusion detection system)-motivation, architecture, and an early prototype”, dans *Proceedings of the 14th national computer security conference*, vol. 1. Citeseer, 1991, pp. 167–176.

[158] A. Souag, C. Salinesi, et I. Comyn-Wattiau, “Ontologies for security requirements : A literature survey and classification”, dans *Advanced Information Systems Engineering Workshops*. Springer, 2012, pp. 61–69.

[159] J. F. Sowa, “Knowledge representation : logical, philosophical, and computational foundations”, 1999.

[160] E. H. Spafford et D. Zamboni, “Intrusion detection using autonomous agents”, *Computer networks*, vol. 34, no. 4, pp. 547–570, 2000.

[161] P. Spyns, R. Meersman, et M. Jarrar, “Data modelling versus ontology engineering”, *ACM SIGMod Record*, vol. 31, no. 4, pp. 12–17, 2002.

[162] P. Srinivasulu, D. Nagaraju, P. R. Kumar, et K. N. Rao, “Classifying the network intrusion attacks using data mining classification methods and their performance comparison”, *International Journal of Computer Science and Network Security*, vol. 9, no. 6, pp. 11–18, 2009.

[163] G. Stein, B. Chen, A. S. Wu, et K. A. Hua, “Decision tree classifier for network intrusion detection with ga-based feature selection”, dans *Proceedings of the 43rd annual Southeast regional conference-Volume 2*. ACM, 2005, pp. 136–141.

[164] A. H. Sung et S. Mukkamala, “Identifying important features for intrusion detection using support vector machines and neural networks”, dans *Applications and the Internet, 2003. Proceedings. 2003 Symposium on*. IEEE, 2003, pp. 209–216.

[165] K. Swamy et K. V. Lakshmi, “Network intrusion detection using improved decision tree algorithm”, *IJCSIS) International Journal of Computer Science and Information Security*, vol. 10, no. 8, 2012.

[166] A. Tajbakhsh, M. Rahmati, et A. Mirzaei, “Intrusion detection using fuzzy association rules”, *Applied Soft Computing*, vol. 9, no. 2, pp. 462–469, 2009.

[167] S. Teng, H. Du, N. Wu, W. Zhang, et J. Su, “A cooperative network intrusion detection based on fuzzy svms.” *Journal of Networks*, vol. 5, no. 4, 2010.

[168] C. Thomas et N. Balakrishnan, “Improvement in intrusion detection with advances in sensor fusion”, *Information Forensics and Security, IEEE Transactions on*, vol. 4, no. 3, pp. 542–551, 2009.

[169] ——, “Advanced sensor fusion technique for enhanced intrusion detection”, dans *Intelligence and Security Informatics, 2008. ISI 2008. IEEE International Conference on*. IEEE, 2008, pp. 173–178.

[170] E. Totel, B. Vivinis, et L. Mé, “A language driven intrusion detection system for event and alert correlation”, dans *Security and Protection in Information Processing Systems*. Springer, 2004, pp. 208–224.

[171] D. Tsarkov et I. Horrocks, “Fact++ description logic reasoner : System description”, dans *Automated reasoning*. Springer, 2006, pp. 292–297.

[172] B. Tsoumas et D. Gritzalis, “Towards an ontology-based security management”, dans *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on*, vol. 1. IEEE, 2006, pp. 985–992.

[173] M. Tuba et D. Bulatovic, “Design of an intrusion detection system based on bayesian networks”, *WSEAS Transactions on Computers*, vol. 8, no. 5, pp. 799–809, 2009.

[174] W. Tylman, “Misuse-based intrusion detection using bayesian networks”, *International Journal of Critical Computer-Based Systems*, vol. 1, no. 1, pp. 178–190, 2010.

[175] J. Undercoffer, A. Joshi, et J. Pinkston, “Modeling computer attacks : An ontology for intrusion detection”, dans *Recent Advances in Intrusion Detection*. Springer, 2003, pp. 113–135.

[176] J. Undercoffer, J. Pinkston, A. Joshi, et T. Finin, “A target-centric ontology for intrusion detection”, dans *18th International Joint Conference on Artificial Intelligence*, 2004, pp. 9–15.

[177] F. Valeur, G. Vigna, C. Kruegel, et R. A. Kemmerer, “Comprehensive approach to intrusion detection alert correlation”, *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, no. 3, pp. 146–169, 2004.

[178] L. Viljanen, “Towards an ontology of trust”, dans *Trust, Privacy, and Security in Digital Business*. Springer, 2005, pp. 175–184.

[179] A. Vorobiev et N. Bekmamedova, “An ontology-driven approach applied to information security.” *Journal of Research & Practice in Information Technology*, vol. 42, no. 1, 2010.

[180] A. Vorobiev et J. Han, “Security attack ontology for web services”, dans *Semantics, Knowledge and Grid, 2006. SKG’06. Second International Conference on*. IEEE, 2006, pp. 42–42.

[181] G. Wang, J. Hao, J. Ma, et L. Huang, “A new approach to intrusion detection using artificial neural networks and fuzzy clustering”, *Expert Systems with Applications*, vol. 37, no. 9, pp. 6225–6232, 2010.

[182] J. A. Wang et M. Guo, “Ovm : an ontology for vulnerability management”, dans *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research : Cyber Security and Information Intelligence Challenges and Strategies*. ACM, 2009, p. 34.

[183] Y. Wang, D. Gu, W. Li, H. Li, et J. Li, “Network intrusion detection with workflow feature definition using bp neural network”, dans *Advances in Neural Networks–ISNN 2009*. Springer, 2009, pp. 60–67.

[184] D. Wessels *et al.*, “Squid web proxy cache”, 2001.

[185] G. B. White, E. A. Fisch, et U. W. Pooch, “Cooperating security managers : A peer-based intrusion detection system”, *Network, IEEE*, vol. 10, no. 1, pp. 20–23, 1996.

[186] M. Williamson, *PfSense 2 Cookbook*. Packt Publishing Ltd, 2011.

[187] P. Wouters et K. Bantoft, “Openswan : Building and integrating virtual private networks”, 2006.

[188] M. Wozniak, “Classifier fusion based on weighted voting-analytical and experimental results”, dans *Intelligent Systems Design and Applications, 2008. ISDA'08. Eighth International Conference on*, vol. 2. IEEE, 2008, pp. 687–692.

[189] C. Xiang, P. C. Yong, et L. S. Meng, “Design of multiple-level hybrid classifier for intrusion detection system using bayesian clustering and decision trees”, *Pattern Recognition Letters*, vol. 29, no. 7, pp. 918–924, 2008.

[190] F. Xiao, S. Jin, et X. Li, “A novel data mining-based method for alert reduction and analysis.” *Journal of Networks*, vol. 5, no. 1, 2010.

[191] D. Yu et D. Frincke, “Alert confidence fusion in intrusion detection systems with extended dempster-shafer theory”, dans *Proceedings of the 43rd annual Southeast regional conference-Volume 2*. ACM, 2005, pp. 142–147.

[192] R. Yusof, S. R. Selamat, et S. Sahib, “Intrusion alert correlation technique analysis for heterogeneous log”, *IJCSNS International Journal of Computer Science and Network Security*, vol. 8, no. 9, pp. 132–138, 2008.

[193] K. Zaraska, “Prelude ids : current state and development perspectives”, *URL http ://www. prelude-ids. org/download/misc/pingwinaria/2003/paper. pdf*, 2003.

[194] S. T. Zargar, J. Joshi, et D. Tipper, “A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks”, *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 4, pp. 2046–2069, 2013.

[195] F. Zeng, K. Yin, M. Chen, et X. Wang, “A new anomaly detection method based on rough set reduction and hmm”, dans *Computer and Information Science, 2009. ICIS 2009. Eighth IEEE/ACIS International Conference on.* IEEE, 2009, pp. 285–289.

[196] Y. Zhang, L. Wang, W. Sun, R. C. Green, et M. Alam, “Distributed intrusion detection system in a multi-layer network architecture of smart grids”, *Smart Grid, IEEE Transactions on*, vol. 2, no. 4, pp. 796–808, 2011.

[197] X. Zhao, H. Jiang, et L. Jiao, "A data-fusion-based method for intrusion detection system in networks", *International Journal of Information Engineering and Electronic Business (IJIEEB)*, vol. 1, no. 1, p. 32, 2009.

[198] B. Zhu, *Alert correlation for extracting attack strategies.* University of New Brunswick (Canada)., 2005.