Event correlation for detecting advanced multi-stage cyber-attacks

A. Spadaro

Delft University of Technology
M.Sc. Thesis

Event correlation for detecting advanced multi-stage cyber-attacks

Date	7/12/2013
Name	A. Spadaro
Student no.	4120388
E-mail	antspadaro@gmail.com
Faculty	Technology, Policy and Management
Section	Information and Communication Technology
Master Programme	*Management of Technology (MoT)*
Graduation committee	
Supervisors	Prof. dr. Y.H. Tan, Chair *Head ICT Section* Dr. ir. J. van den Berg *Associate Professor at ICT Section* Ir. M. Davarynejad *PhD Candidate at Systems Engineering Section* H. Hoogstraaten Ir. J. de Vries *Security Experts at Fox-IT B.V.*

Abstract

Rapidly evolving IT infrastructures bring beneficial effects to society and promote information sharing and use. However, vulnerabilities create opportunities for hostile users to perform malicious activities and IT security has gradually turned into a critical research area for organizations and governments. Processes of decision making in large organizations are widely influenced by their capability of detecting malicious activities effectively, and by the correctness in analyzing suspicious phenomena, which can be observed by a number of security sensors deployed in such large networks. Several techniques are currently employed to detect incidents starting from captured security-related events within networks and computer systems. However, the large volume of observable events, the continuous sophistication and changes in attack strategies make it challenging to provide effective solutions to detect and reconstruct cyber-security incidents. In particular, advanced multi-stage attacks tend to remain undiscovered because common security mechanisms can generally detect and flag harmful activity – sometimes with unsatisfactory false alert rates – but they are not able to draw a big picture of the incidents. Since such task is usually performed by security experts in full, it may be expensive and prone to errors. Therefore, it is essential to develop procedures for combining large heterogeneous datasets and system's information in meaningful way, and for supplying detailed information to IT security management. By examining realistic multi-stage incidents, this thesis proposes the design of a model to correlate detectable suspicious events by combining complementary state of the art methods, which perform correlation along different axis. Thus, it aims at providing standard data formats, prioritizing and clustering data, increasing confidence about threats, finding relations of causality between suspicious events and eventually reconstructing multi-stage incidents. In addition, reviewing the most influential scientific papers gives us the chance to categorize the techniques and suggest practices for further implementation.

Keywords: event correlation, IT security management, multi-stage attacks, machine learning, intrusion detection, big data

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I wish to express my very great appreciation to my research supervisors for their contribution to this Master Thesis project. I would like to express my deep gratitude to Dr.ir. Jan van den Berg for his helpful recommendations and his guidance in conducting this research. I have been very much appreciated his willingness to give his time, his enthusiastic encouragement along with his constructive critiques. My thanks are also extended to Ir. Mohsen Davarynejad for his valuable suggestions and patient guidance. I would also like to thank my research project supervisor at Fox-IT, Hans Hoogstraaten, for his useful critiques on this research work. His long-time experience in IT Security greatly contributed to the vision of this work as well as my growing interest in the area. I am also very grateful to Ir. Johannes de Vries for his willingness to give me constructive recommendations on the project.

I would also like to thank Fox-IT B.V. for offering me the opportunity and the resources for carrying out this research project in such an amazing environment. My thanks are extended to the employees and *stagiairs* with whom I have shared good times, experiences and enthusiasm, which I consider of great value for my personal and professional growth.

Finally, I want to express my profound gratitude to my parents and my sister for their persistent support throughout my studies. I extend my thanks to my family in Italy who also encouraged me greatly. I would also like to thank all my friends who regardless of the distance gave me exceptional support. Special thanks to Daniel, Judith and Vincenzo for their closeness.

Antonio Spadaro

Delft, June 26, 2013

MANAGEMENT SUMMARY

Today's world is interconnected. Individuals and organizations increasingly rely on IT to take advantage of this interconnection. A safe and resilient digital space is essential for economy and daily operations. Yet, risks posed by cyber-attacks are real and growing rapidly.

Nowadays, advanced cyber-attacks are performed in several stages by malicious users. These often utilize extensive resources and advanced skills to attack the complex IT infrastructures of large organizations. Broadly speaking, many types of security devices and software are capable to detect malicious activity. However, skilled attackers may also be able to change strategies frequently and devise new methodologies to mislead the security operators, who monitor and respond to security incidents. Therefore, detecting an advanced attack and understanding the 'big picture' of an IT security incident are difficult assignments.

We investigated the relation between incidents and events which are observable in an organization's network. Intelligent analysis of monitored events allows effective detection of suspicious events. Our solution allows identifying meaningful events. Events which occur in large computer networks can be monitored from diverse perspective, depending on the location of the security mechanisms. Their logging capabilities are also quite different. Hence, we suggest providing a standardized format to enable analysis.

Event correlation allows detecting advanced attacks and provides the 'big picture'. In regard to this, we suggest a format for event data which supplies detailed information to IT security management, allowing them to change the level of view. For example, a cyber attack which occurs in multiple stages can be seen at the level of security alerts, or at the level of the stages which contribute to the attack.

Researchers suggest the use of various and complementary techniques to correlate event-related data. Our event correlation model is based on the employment of modern correlation techniques – based on statistical analysis, expert signatures, and machine learning – in a modular way. Each technique is recognized to address specific needs effectively. We suggest a modular approach which exploits the benefits of each correlation technique. In some scenarios, for example, it may be necessary to reduce the false alarm rate or the volume and redundancy of events to analyze.

Continuous sophistication and persistent character of cyber-attacks make it urgent adopting flexible strategies to deal with them. One only needs to think of well-known attacks such as Stuxnet, Aurora or GhostNet, which have been widely reported by media. IT management of

large organizations needs to take adequate measures to protect their high-value assets against cyber-attacks and avoid late and inefficient incident response. Moreover, uncertainty around cyber security is driving the demand of new services and products despite the financial crisis. Security budgets are broadly expected to increase substantially in the next 3 years. In particular, revenues from security management tools are expected to grow.

The uniqueness of our solution is due to the flexibility of creating customized incident detection systems – while utilizing methods which are the most influential in the scientific community. Such flexibility is obtained through decomposing the problem of detecting advanced attacks into sub-problems. These are individually addressed by complementary modules. These components basically combine heterogeneous events with one another or with data which is not strictly related to any incident. For example, information about network topology, vulnerabilities and security policies may also be helpful for incident detection. Furthermore, our correlation model can be employed comprehensively or partially, according to the detection scope as well as an organization's business requirements. For this purpose, this research project presents design alternatives and some technical considerations. These are intended to provide direction to software developers and security analysts for deploying tailored solutions.

1. INTRODUCTION

Progress in Information Technology is the drive to productivity gains in our society [1]. Nevertheless, uncertainties surrounding technology makes IT infrastructures a target of security threats, leading organizations to face several security issues [2]. Nowadays, cyber-crime, cyber-warfare, cyber-terrorism are peremptorily emerging with a widespread belief that their impact will increase. Results from a survey show that the failure of critical infrastructures – e.g., electricity, water, gas, ICT, transport – is considered a matter of the utmost importance in the technological sector, with relatively low likelihood but high impact. Incidentally, respondents recognize cyber-attacks as the most relevant risk [3]. According to the results of a survey conducted by PricewaterhouseCoopers [4], cybercrime is one of the top four economic crimes. Despite that, most of the respondents do not have a cyber-crisis response plan or they are not aware of having one.

Most businesses have recognized the great importance of securing information systems from cyber-attacks, and information security has been a research area for a long time [5]. Security issues have become more and more frequent with the development of the Internet, leading to the pursuit of new techniques to detect malicious attacks. In this context, Intrusion Detection Systems (IDSs), which are hardware or computer programs designed to detect attacks on network and information systems, generally provide network administrators and operators with sufficient information about their health and the current threats against them, in order to handle security breach attempts [6]. Also, the continuous development of new reliable analysis techniques makes intrusion detection systems more efficient [7].

According to Gartner, worldwide spending on IT security products and service market is expected to reach $86 billion in 2016, growing with a compound annual growth rate (CAGR)[1] of 9%. The 2012 annual CIO survey shows an overall prioritization on security budgets, in spite of the extended financial crisis, which have had a small impact on IT security spending of emerging countries such as China, India, and Brazil. Nonetheless, recent market analyses forecast market for Security and Vulnerability Management tools to exceed revenues of $5.7 billion by the end of 2015, with a CAGR of 11.3%.

Demand of IT security services and products will be driven by the increasing sophistication of attack patterns and the persistent threat landscape [8];

[1] Compound Annual Growth rate (CAGR) is a measure of the annual growth rate of an investment, as it grows linearly over the considered period of time.

The key to success in the security management domain will be "the ability to provide proactive security protection and the knowledge and intelligence to provide comprehensive security assessment data" [9].

Operating system, application and service logs can be used for incident detection and analysis by correlating information about occurring events. Firewalls and routers provide logs of blocked connection attempts that can be used for incident detection when correlating events originated by other devices. Routers may also provide network flow information that can be used to detect anomalies within the network caused by malicious activity. Antivirus and anti-spam software may also indicate attack attempts if malware or other malicious contents are detected; this is possible only if signatures of malware are kept updated. Changes in relevant files (i.e., system files) and databases may also be detected by integrity checking software and used as attack evidence. IDS products detect suspicious events and record related data such as date and time of the detection, type of attack, IP source and destination, and so forth.

1.1. MOTIVATION

Cyber-attackers often change their behaviors and methods in order to conceal malicious activities, and they tend to use novel attack strategies as well as exploit new vulnerabilities in the targeted systems [10]. Moreover, it is largely accepted that most advanced attacks are made up of multiple stages; each step represents an individual attack scenario. Thus, identifying the scenarios belonging to a multi-stage attack can lead to identify attackers' actual strategies. It is therefore challenging to understand attacker's behavior, attack strategies and employed attack vectors. As a matter of fact, each attack can be usually represented by many IDS alerts. This makes it difficult to analyze intrusions because it is necessary to reconstruct attacks starting from IDS output; it may include persistent false alerts which potentially cause considerable financial losses to organizations.

According to Barry Hensley, director of the Counter Threat Unit/Research Group of Dell SecureWorks, industry leader in Information Security services:

"The tools, procedures and other controls used to defend commodity security threats are often ineffective against targeted Advanced Persistent Threats (APTs); when actors

> ***are focused on a specific target, they customize and adapt their tactics, techniques and procedures to predict and circumvent security controls and standard incident responses." [11].***

A traditional network often translates into a large volume of data produced by intrusion sensors, making it challenging to readily deal with intrusions; event-base data such as firewall and IDS alerts, log files, routing information and so forth, can be collected and used for security analysis. These data represent the activity at a certain moment and location, and attacks' evidence can be extracted from the overall amount of events within a distributed system. However, the large amount of events makes it challenging to find relationships between suspicious events and extract attack attempts from them [10; 12].

Security data need to be correlated by security analysts to gain in-depth understanding of an attack (i.e., high-level view of the incidents occurring in the monitored information system); however it is generally unfeasible to manually deal with massive amount of data. Although techniques based on expert rules produce accurate results, experts may be limited in processing information and solving complex problems.

Interviewing security experts showed that it is hitherto unfeasible to draw a complete picture of a high-sophisticated cyber-attack, even though it is possible to inspect the most relevant aspects. Many alerts are required to have an insight of the occurring incident despite a number of them might be not interesting – e.g., alerts raised by port scans may not be relevant for discovering some types of attack. Generally, alerts are useful to reconstruct an attack, but much other information, which does not come from IDSs, are required.

Technological designs of advanced reliable correlation systems have been emerging in literature only in the last decade. We have identified a number of previous design efforts towards comprehensive correlation models and architectures, such as M2D2 [13], Valeur et al. [14], Salah et al. [15]. However, they focus on producing surveys and taxonomies of techniques, validating their own correlation methods, or may not provide any practical implementation of state-of-the-art techniques.

Advanced techniques to detect intrusions focus on the use of multiple sources for supporting decision making. When considering data streams from heterogeneous sources, it is also necessary to take into account their high-frequency dynamics as well as their temporal nature: e.g., classifiers receive data streams asynchronously [16]. State of the art techniques

usually are based on statistical analysis, expert signatures, or make use of machine learning approaches to derive linkages among alerts.

Finding all possible causal relationships among alerts may be a time-consuming and excessively slow activity for being considered viable, bearing in mind the imbalanced distributions of anomaly and normal data to examine [17] [13; 18; 19]. Methods based on automatic inference are capable to find novel alerts, even though they are not able to discover the whole set of causal relationships between alerts. Therefore, it may be convenient to bring together the advantages of approaches based on inference and expert knowledge [6].

With the help of relevant scientific literature and subsequent interviews, we cope with the following problems, which also represent our criteria for the solution design:

- Current security mechanisms are not effective for detecting novel , stealth and multi-stage attacks;
- Advanced attack techniques and inadequate logging often make it impossible to identify actual attack complexity;
- Large amount of security-related data makes it costly to examine information concerning security incidents and difficult to take adequate decisions;
- Use of heterogeneous data sources leads to complex interpretation of the data;
- High rate of false alerts (i.e., both positive and negative) makes it generally hard to reconstruct a security incidents with a reasonable degree of confidence;
- Security operators often cope with lack of details in the available information.
- Lack of procedures for correlating effectively alert messages from malicious events with other information, such as system or application logs, network topologies, and so on.

1.2. RESEARCH GOALS

The previous section summarizes the challenges that must be faced in order to realize a model for correlating events. Such model is essential for the extraction of relevant data from the available sources and transforming it into knowledge about advanced security threats, knowledge which in turn is employed by security officers in order to improve security incident management. The objective of this thesis research is formulated as follows:

To produce an event correlation model for detecting advanced multi-stage attacks.

The scope is to detect security incidents which are caused by attacks performed in multiple steps. The agents initiating the attacks target IT infrastructures of large organizations, use advanced skills and extensive resources.

In order to gain a deep knowledge about the objective of the research, the following questions are addressed:

Q1. What is the domain in which advanced cyber-incidents occur?
Q2. What are the characteristics of defense mechanisms against cyber threats in today's organizations, in terms of scenarios?
Q3. What approaches to data correlation are suggested in literature for detecting and analyzing advanced IT security incidents?
Q4. What tasks should an event correlation model perform in order to detect multi-stage attacks, in terms of scenarios?
Q5. What are the considerations to take into account when developing a tailored approach to event correlation?

1.3. RESEARCH METHODOLOGY

The research will be conducted using the Design Science in Information Systems Research methodology [20]. Although the design of our correlation model plays an important role, a primary goal is to produce new knowledge and understanding that is inaccessible without designing a solution [21]. Design Science is "improvement research" using design as method, and focusing on its problem-solving and performance-improving nature [22].

The research is conducted in phases, each of which aims at answering the relative sub-question, formulated in the previous section. The phases are the following:

1. **Problem conceptualization**. This phase is based on investigation of the domain in which advanced computer security incidents occur. Describing such environment better frames the problem (i.e., "business needs") which our model aims at solving. The problem domain is examined by taking into account modern incident trends and formulating three multi-stage attack scenario cases.

2. **Problem analysis**. This phase provides understanding of fundamental concepts which are related to detection and correlation of advanced cyber threats by analyzing data sources, attributes, structures. Thus, problem is better defined through the formalization of key principles and the requirements and functionalities – which are based on the identified issues – for the development of the desired model.

3. **Synthesis of current developments in the field**. This phase summarizes the current state-of-art of solutions for the problem at stake. Literature review will be conducted by looking through collections of published scientific research, such as ScienceDirect, IEEE Xplore, Emerald Insight, ACM Digital Library, and so forth[2]. Relevant academic articles which are referenced on relevant journals are also reviewed. In order to consider the most authoritative studies in the field and rank them, we created a reputation index based on features such as average number of citations per year as well as citations in survey reports.

4. **Design**. This stage is "an essentially creative step wherein new functionality is envisioned based on a novel configuration of either existing or new and existing elements" [23]. The solution of the problem is drawn from existing knowledge base and theories; acquired knowledge from the previous phases is combined with further literature research and review of prior development efforts, in order to provide a solution design which overcomes the research problem. Specifically, the model is designed by considering the real scenarios which are conceived in the problem conceptualization phase. The approach to implement the solution design may vary, depending on many factors: e.g., the preferences of organizations, economic and computational resources available, types of threats which are to be detected, and so on. Therefore, we also aim at providing a deeper insight into the state-of-the-art, trade-offs and suggestions for the real implementation of our model.

The above phases lay the foundations for personal reflections and limitations, which may add requirements for future design phases. Subsequently, conclusion and future research are depicted.

[2] Keywords for search engines include combination of the following ones: a*lert correlation, fusion, aggregation, attack scenarios, intrusion detection, multi-stage, multi-step, feature selection, machine learning, inference, reasoning engine, computer security, cyber-security, computational intelligence, neural network, Bayesian network, data mining, hyper-alert, meta-alert, support vector machine, genetic algorithm,* and so forth.

Generally, Design Science Research also takes into account an evaluation phase; however, few evaluation techniques for alert correlation systems have been proposed, and the lack of reliable benchmarking datasets raise a fundamental issue about the rigorousness of an evaluation method for this model. A desirable dataset for a quantitative evaluation involves labeled host- and network-based data, with the description of complex attack scenarios [24].

1.4. THESIS OUTLINE

The outline of the thesis in Fig. 1 reflects the above mentioned research phases along with inputs and outputs for each phase.

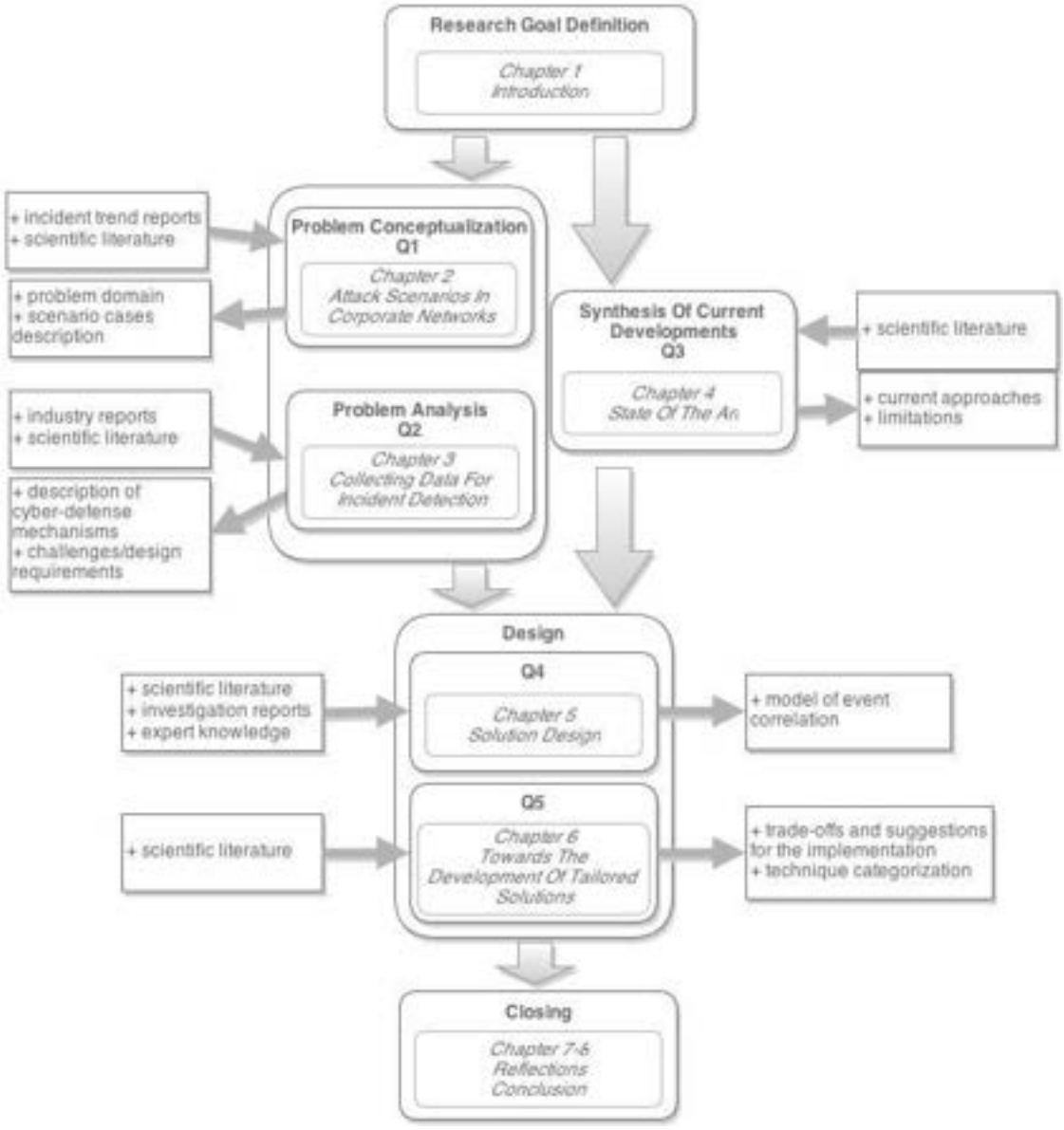

FIG. 1 THESIS OUTLINE

2. SCENARIOS OF ATTACKS ON CORPORATE NETWORKS

In this chapter, after introducing the context under analysis in our research project, we discuss the issue of cyber-security incidents and the main attack vectors. Subsequently, we use findings of an authoritative study about incidents occurred in corporate networks in 2011 – giving particular attention to the rising of multi-stage attacks targeting large organizations – to make up three significant scenario cases. These will be examined for the design process in the section 5.3.

2.1. INFORMATION SYSTEMS OF TODAY

Nowadays, IT systems are crucial to conduct business. Information systems are employed by firms to improve productivity and speed up processes, allowing new service and product development, thus new business models. Exploiting their capabilities of support in the decision making process and help managing relationships with suppliers and customers efficiently provides a potential competitive advantage. Moreover, as Critical Infrastructures increasingly rely on IT, information security is playing a key role on nations' health, economy and public security.

Most businesses have recognized the great importance of securing information systems from cyber-attacks, and information security has been a research area for a long time [5]. Security issues have become more and more frequent with the development of the Internet, leading to the pursuit of new techniques to detect malicious attacks. In this context, intrusion detection systems (IDSs), which are hardware or computer programs designed to detect attacks on network and information systems, generally provide network administrators and operators with sufficient information about their health and the current threats against them, in order to handle security breach attempts [6].

Let us consider the computer network of a large organization. For clarity, an example of the network architecture and of the Intrusion Detection System deployment is illustrated in Fig. 2. In a corporate network, a central site contains resources such as Mail servers, Web servers, DNS servers, and so forth, which deliver services and information to both internal and external users. Given that the hosts that provide such key resources are vulnerable to attacks from outside the local area network (LAN), they are placed into a sub-network known as DMZ[3]. The aim of DMZ is to protect the rest of the internal corporate network if attacks are performed on it. Servers in the DMZ provide services to both internal and external networks.

[3] Demilitarized zone.

The perimeter firewall/router provides NAT[4] services and allows access to a reduced set of ports only – e.g., TCP port 80 on the Web server, TCP port 25 on the Mail server, TCP and UDP ports 53 on the DNS server, TCP ports 20/21 on the FTP server – on each host in the DMZ sub-network. The internal firewall works as proxy for web traffic, thus blocking direct connections from the internal LAN to the Internet while allowing only some connections to the DMZ.

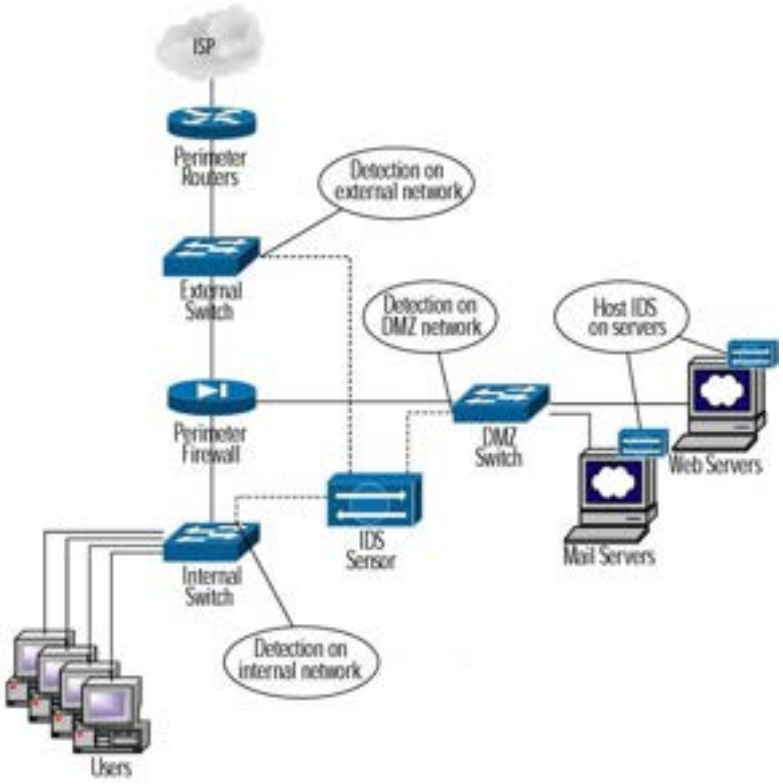

FIG. 2 EXAMPLE OF NETWORK CONFIGURATION; ADAPTED FROM [25].

Network-based IDS sensors – such as Snort[5] – monitor DMZ, external and internal sub-networks, and warn about possible threats that are not caught by the firewall. For best protection, host-based IDS software is also installed on each server located in the DMZ.

2.2. CYBER-SECURITY INCIDENTS AND ATTACK SCENARIO CASES

In this section, the nature of a security incident will be discussed. First, we provide an insight into the general trend for the means used by attackers and the key characteristics of documented breaches, focusing on incidents occurred in large organizations in the last few years. Finally, we adapt some attack stages extracted from real breaches targeting high-value organizations – i.e., hospitals, digital certificate authorities (CAs), governments, news media,

[4] Network Address Translation modifies IP address information in IP packet.
[5] Open source network intrusion prevention and detection system - http://www.snort.org

NGOs, defense contractors, international organizations, and other industries – to produce three attack scenario cases, which will be further examined in chapter 5.

There are many definitions of computer security incident; here, we limit to quote a formal definition contained in a special publication by NIST[6]:

> ***"A violation or imminent threat of violation of computer security policies, acceptable use policies, or standard security practices" [26].***

Imminent threats refer to having evidence for believing that certain incidents will occur. For example, we could think of sensitive information exposed through file sharing services, an attempt of extorting money by threatening the public release of sensitive information, a botnet[7] which is commanded with the intention to cause a web server's crash, a PDF document which is actually a malware[8] that infects a computer and establishes connections to an external host on the Internet. It is considered relevant to keep fairly low the amount of incidents through sufficient security controls in order to avoid adverse business impact [26].

There are innumerable ways for incidents to occur and organizations should be prepared to handle them, especially when common methods of attacks are utilized. According to NIST, usual attack vectors can be classified as follows [26]:

- Removable media (e.g., infected USB flash drive)
- Attrition (e.g., brute-force attack for password discovery; DDoS[9] to deny access to a service)
- Web (e.g., redirect to a website that exploits a vulnerability of the browser to install malware)
- Email (e.g., email message with a link to a malicious website; attached document containing malware)
- Impersonation (e.g., SQL injection[10]; spoofing[11]; MITM[12] attack)

---

[6] National Institute of Standards and Technology is an agency of the U.S. Department of Commerce which promotes innovation and industrial competitiveness.
[7] A botnet is a network of computers controlled without the owners' knowledge.
[8] Broad term which is used to refer to malicious software of any sort.
[9] Distributed Denial of Service (DDoS) is network traffic flooding by multiple systems which aims at saturating the resources of targeted web servers, thus causing service breakdown.
[10] SQL injection is a technique to exploit vulnerability in a database by injecting invalid SQL commands.

- Improper usage (e.g., users perform illicit activities or install file sharing software incurring in the loss of sensitive data)
- Loss or theft of equipment (e.g., organization's laptop, mobile phone, authentication token)
- Other than previous categories

In order to give an insight into our problem space, some representative cyber-intrusion scenarios are depicted in this section. The choice of such cases is strictly related with the characteristics of the domain under discussion. Thereby, we relied on a study recently conducted by Verizon RISK Team in collaboration with Australian Federal Police, Dutch National High Tech Crime Unit, Irish Reporting and Information Security Service, Police Central e-Crime Unit, and United States Secret Service [27], as it takes into consideration 855 incidents occurred in corporate networks in 2011, involving more than 170 million compromised records. Such large amount of compromised data is almost entirely composed by all kinds of personal details – i.e., names, addresses, and so on – despite it only accounted for about 30 incidents in the data sample. While most of the incidents aim at stealing authentication credentials, payment card information and bank data, also misappropriation of sensitive organizational data, system information, trade secrets, and so forth, considerably affect large organizations[13].

It is relevant to note that – in 79% of the reported incidents – victims are selected based on opportunity, as they are found to have exploitable vulnerabilities, rather than on choice. It can be seen as a way to simplify business processes: targeting the weakest and unsuspecting link, and then reproducing it on a large scale. However, 50% of the reported attacks against large organizations are targeted.

Besides those facts, 96% of the overall intrusions are not very complex, often showing highly sophisticated[14] techniques only in later stages. It implies that preventing early unauthorized access by malicious users is the most desirable situation to avoid more severe incidents – e.g., unauthorized release of data. However, for almost half of the compromised data examined in the report, the attack complexity is unknown, due to inadequate logging

[11] Spoofing is the use of false source IP address to masquerade the real source of Internet Protocol (IP) packets.
[12] Man-in-the-middle (MITM) attack occurs when an attacker is able to read, interpolate, and modify communication between two unaware parties.
[13] By 'large' the study refers to firms having at least 1000 employees.
[14] The authors consider high the degree of sophistication if advanced skills and customization, as well as extensive resources are required.

and unclear attack techniques. Also, the collected data shows that cybercriminals may put at risk large organizations with no need to deeply amend their techniques.

Nearly all the reported security incidents are caused by agents which are outside the concerned organization, such as criminal organizations, unknown hackers, as well as former employees and government entities, which mainly look for financial or personal gain; although it has been recorded an increasing number of 'hacktivists' threatening large organizations, mostly driven by political disagreement.

Multi-stage cyber-attacks

In 2011, incidents are primarily caused by hacking[15] and malware, often combined: they have been leveraged in 9 out of 10 multi-stage attacks. Most of the incidents involve the use of login credentials which are stolen through hacking, captured from user activity by malware such as key-loggers, or guessed – it usually occurs when default passwords are used. Other recurrent threat actions are brute force attacks, installation and exploitation of backdoors which allow access (and control) of remote machines, and data transfer to external entities. The latter may occur either automatically driven by malware or, with growing frequency, keeping footholds on internal system through backdoors.

In the last few years, it has been noted the upward trend in the use of remote installation of malware or injection by exploiting web application vulnerabilities. These facts, which represent almost the entirety of malware infection vectors, occur in scenarios in which intruders are able to get unauthorized access to remote systems. However, for some large organizations, intruders may find it easier to send malware via email, auto-executed web code, and malicious URLs, rather than gaining remote access. It is noteworthy to mention that the infection vector remains unknown in 3% of all the incidents involving malware. That may occur for many reasons such as the lack of logs, use of anti-forensics techniques by attackers, and so forth.

Known anti-forensics approaches – e.g., timestomping[16], packing[17], encryption – show up in about one third of the records and may vary, but they all aim at altering or hiding digital evidence.

[15] In the report DBIR 2012, it refers to "all the attempts to intentionally access or harm information assets without authorization or in excess of authorization by thwarting logical security mechanisms" [27].
[16] Timestomp is a tool which allows changing time attributes of computer files: 'create', 'access', 'modify', 'entry modified'.
[17] File packing is a technique to rebuild and alter the aspect of executable files.

As regards vectors for hacking activities, intruders prefer exploiting remote access services, such as RDP[18] and VNC[19]. Typically, opportunity targets are discovered by using malicious scripts which look among known remote access service ports, and then compromised thanks to the exploitation of default vendor credentials. Whereas, more than 90% of the reported record losses involves the installation of backdoors at some point during the attack.

Another substantial fraction of intrusions targets web applications as they are often required to be publicly accessible through the Internet, so as to be potentially used as point of entry to large organizations' data, which increasingly rely on web servers. Web and database servers are also the most compromised corporate assets in nearly all the security incidents involving large amount of data.

Regarding time measures, malicious users are generally able to start compromising the victim's assets some minutes, or only seconds, after the initial attack attempt; the bright spot is that, the period between the initial compromise and the information extraction is often on the order of days or months for large organization; thus making incident detection feasible within a reasonable time span.

It is also worthy to mention that 98% of the cases in which malware was used to extract information, it was paired with key-loggers, representing over 250 incidents; in almost each of them, stolen user credentials were used too. Moreover, 57% of the times when default or guessed credentials were exploited, a key-logger was founded, and in 47% of the above case, a backdoor was present too. These facts all exhibit the versatility of current malware.

As for the impacts of the incidents, it is not possible to provide an overall estimation of the economic loss caused by security breaches. For guidance, it is reported that fraud-related breaches have led to individual losses that have exceeded 70 million Euros. It is relevant to bear in mind that the impacts may also be suffered by third-party organizations; for instance, breaches involving payment cards do not only affect customers, but also the banks which issue payment cards. In addition, most of the reported security incidents showed no considerable long-run impacts on reputation, loss of competitive advantage, or market value.

The following sections of this chapter present three attack scenarios, which are derived from the above trends and include attack stages from real breaches targeting high-value

[18] Remote Desktop Protocol (RDP) is a Microsoft proprietary protocol which provides a graphical interface to a remote computer running RDP server software.
[19] Virtual Network Computing (VNC) utilizes remote frame buffer protocol to provide remote access through a graphical interface.

organizations. The cases will be further analyzed in chapter 5 in order to design a solution to our problem.

2.2.1. Scenario 1: Gaining unauthorized access and Disclosure of sensitive data

In this section, we present the case of a malicious user who tries to gain unauthorized access to an account on a local machine. The above depicted corporate network holds a server connected to the Internet. As previously discussed, it is located in the DMZ. The server at issue runs a Web service and a vulnerable FTP service. FTP services are usually utilized in Web servers for accessing files remotely. An instance of NIDS (e.g., Snort) is running on the perimeter firewall with default configuration. FTP (File Transfer Protocol) is extensively used in business due to its ease of use and relatively little cost. However, file exchanges through FTP expose to major risk of security breach. In order to ponder on the potential impacts of such intrusion, it is noteworthy to mention the breach discovered in 2012 which targeted a large Dutch hospital, the Groene Hart Ziekenhuis. An 'ethical hacker' has shown that the hospital, which relied on FTP provided by third-party, has been keeping sensitive data of about 500,000 patients such as letters, x-ray, laboratory results, diagnoses, and other personal information unprotected – or rather, not adequately protected – for over 4 years. Recent investigation has pointed out that medical records of at least 50 patients were actually stolen.

Attack steps

1. The attacker performs a port scan with Nmap (Network Mapper security scanning tool) on externally visible IP addresses in order to find open ports in the targeted organization's computer systems.
2. The attacker scans the servers' public IP address from the Internet by using a vulnerability scanning tool.
3. The tool identifies vulnerability in the FTP server, whose port has resulted open. The attacker is able to exploit it, and launches a remote-to-local (R2L) attack to the FTP server, using a different IP address than the one used in steps 1 and 2.
4. The attacker downloads a large database file from the FTP server.

2.2.2. Scenario 2: Data Exfiltration

The scenario in this section is loosely taken from the security breach which DigiNotar suffered in mid-2011. To sum up, DigiNotar B.V. was a Dutch certificate authority which provided worldwide-trusted digital certificates for various purposes, from securing websites

to e-Government services. The attack has consequently led to issuing over 500 rogue certificates and the fraudulent certificates had been used in a large-scale man-in-the-middle (MITM) attack. Due to the latter, network traffic directed to Google sub-domains of about 300,000 Iranian users was intercepted and credentials were stolen. Subsequently, Fox-IT B.V. had been asked to investigate the incident [28].

A vulnerable Web service located in DMZ has been compromised by an attacker, who has used it to extend his foothold, and compromised an important database server in a back-end network enclave, which contains highly confidential data. However, the firewall is configured in such a way that the intruder is not able to exfiltrate data from the database to the Internet directly.

ATTACK STEPS

1. The attacker utilizes a remote access service (e.g., RDP) toolkit to create a network tunnel between the database in the internal network and the Web server in the DMZ.
2. The attacker utilizes both the Web server in the DMZ and the internal SMTP[20] server as a stepping stone to transfer files between the critical system and arbitrary systems on the Internet.
3. The attacker erases a number of server access logs.

2.2.3. SCENARIO 3: SOCIAL ENGINEERING, ZERO-DAY EXPLOIT, BACKDOOR, C&C EXPLOITATION

This scenario is inspired by Shady RAT and Gh0stNet, two large-scale cyber-attacks reported in 2011 and 2009 respectively, reasonably associated with Advanced Persistent Threats (APTs). Operation Shady RAT [29] started in 2006, targeting over 70 large firms, NGOs, and governments worldwide in about 5 years. Petabytes[21] of data were stolen "by one specific actor" and, even though it is still not possible to know how it has been used, we are able to state that the losses harmed the national security of the involved countries, and whole industries within today's complex global economic landscape. Gh0stNet [30] operations led to the infection of about 1,300 computer systems, of which almost 30% are considered high-value target. The infected systems had been steered to download a trojan (i.e., gh0st RAT) that gave real-time complete control to the attackers, located in People's Republic of China.

[20] Simple Mail Transfer Protocol is a standard for transmission of e-mail over Internet Protocol (IP).
[21] 1 PB = 10^{15} Bytes (= 1 million gigabytes)

A malicious user utilizes social engineering techniques to craft an e-mail that is most likely to be opened by the CFO of the company. However, the name of the sender and the mail address do not match (i.e., spelling errors in the e-mail address). Moreover, the e-mail originates from a free webmail address (e.g., Yahoo! Mail). The e-mail includes a Microsoft Excel file with a zero-day exploit. As the CFO opens the attachment, a vulnerability is exploited.

ATTACK STEPS

1. The attacker sends a socially engineered e-mail containing a zero-day exploit to the CFO.
2. The CFO launches the attachment; the exploit automatically connects to a Dynamic DNS domain (e.g., DynDNS) and downloads a malware which disables the antivirus scanner (and its automatic updates) on the CFO's laptop;
3. A Trojan horse is downloaded on the laptop, giving the attacker complete control over it and connects to a command and control (C&C) server on the Internet, whose domain name was recently registered. The malware checks for net commands at random time intervals.

2.3. SUMMARY

In this chapter, we have depicted the problem domain starting from the brief description of a corporate network and the deployment of key security elements, such as network and host intrusion detection systems, and firewalls. We provided, then, a broad definition of computer security incident, as it is formally intended by U.S. Department of Homeland Security, which takes into account violating – even partially – standard security practices as well as individual organizations' security policies. Subsequently, we have listed the main vectors of attack, i.e., removable media, attrition, web, e-mail, impersonation, improper usage, loss or theft of equipment.

We have examined a major report about worldwide cyber incidents in 2011, looking for interesting trends in data; results have been utilized to compose three multi-stage attack cases to be further analyzed in the following chapters; i.e., (a) unauthorized access and disclosure of sensitive data, (b) data exfiltration, (c) social engineering, zero-day exploit, backdoor, C&C exploitation.

Our cases share with real scenarios some attack stages, hence, we have reduced to some degree the complexity of highly sophisticated attacks. However, the real cases and the

findings in Verizon's report confirm the socio-economic relevance, and the criticality of detecting these "simplified" scenario cases.

3. COLLECTING DATA FOR INCIDENT DETECTION

After the description of the context in which detection of advanced threats takes place, in this chapter we introduce effective security incident detection based on monitoring and intelligent analysis. We intend to show the nature of the relationship between information security incident and events that occur in computer networks and the related computer systems. In section 3.1, we give a deeper insight into the common tools which allow detection of suspicious events in a corporate network; the importance of utilizing such tools adequately lies in the need of providing proper incident response, within a reasonable time. With respect to these mechanisms, we discuss the logging capabilities and the characteristics of security-related data. Finally, we state the key definition of event correlation.

3.1. SECURITY INCIDENT DETECTION

In this section, we intend to outline security mechanisms – i.e., devices and software – which have been previously introduced in 2.1. As a matter of fact, the portrayed domain implements the security systems which we are going to discuss. Moreover, such systems are widely utilized to identify the suspicious events which characterize computer security incidents.

3.1.1. INTRUSION DETECTION SYSTEMS

The goal of an IDS is to identify signs for potential incidents by analyzing monitored events occurring in a computer system or network, becoming an imperative for every organization's security infrastructure [31]. IDSs are classified as either host- or network-based. Host-based IDSs (HIDS) detect misuse at the computer system level, whereas network-based IDSs (NIDS) detect misuse at the network level. For example, the NIDS which monitor our DMZ network in Fig. 2 may be able to detect an attack against a web server while monitoring data flows in the network, instead the HIDS installed on the web server may detect the attack as soon as it reaches the host. Generally, the analysis of an HIDS is more detailed than the one of NIDSs, as the former is able to observe both attempt and outcome of an incident, while the latter may not know whether the incident actually occurred [32].

3.1.2. ANTIVIRUS

Antivirus software is deployed on each computer system in order to mitigate malware threat, including viruses, worms, Trojan horses, blended threats, attacker tools (e.g., backdoors, keystroke loggers). This type of technology looks for signs of malware in file systems, applications, and critical components of operating systems through heuristic and signature detection. When files containing malware are detected, antivirus software try to disinfect or

quarantine them. Usually, they complement IDSs, as antivirus software can detect less common malware than IDSs. However, antiviruses often are not able to recognize new threats until updates with new signatures are released by vendors and then installed [31].

3.1.3. FIREWALLS/ROUTERS

Firewalls rely on source and destination IP addresses, transport layer protocol, and basic protocol information for filtering network traffic. Their purpose is to block unauthorized access attempts and they can also be reconfigured by some IDSs to block particular threat. Some models of router (e.g., Cisco, Huawei, and many others) can also monitor network traffic behavior and supply flow data – data containing header and statistics information of a set of packets with similar characteristics – which can be utilized to detect unusual flows – e.g., distributed denial of service (DDoS), worms, backdoors, and so on. Examples of standard formats for flow data are NetFlow[22], IPFIX[23], and sFlow[24] [31].

3.1.4. LOG MONITORING TOOLS

To deal with detection and response of IT security incidents is necessary to monitor events from different systems in the network, such as operating systems, applications, network devices, and so forth. It results into the collection of a huge amount of logs which is, however, very valuable when correlated with other security-related data. Such tools may also be able to detect the occurrence of any misuse or anomalies in the logs, and raise warning messages.

3.1.5. VULNERABILITY SCANNERS

In this category is included software which aims at investigating weaknesses in computer systems, networks and applications for IT Security Assessment. Likewise, such tools are most likely to be used by malicious users to perform the so-called “reconnaissance”; in other words, attackers may use them to gain information about vulnerability to be further exploited and gain unauthorized access to a system. Therefore, it is crucial using scanners for defense purpose.

3.2. LOGGING CAPABILITIES OF SENSORS

In this section, logging capabilities of sensors yielding security data will be analyzed. Collecting data at various locations increases the chances to detect sophisticated cyber-attacks, such as espionage activities and advanced persistent threats (APT). Diverse probes

[22] http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html
[23] http://tools.ietf.org/wg/ipfix
[24] http://www.sflow.org

are deployed in the network segments for gathering data in the form of physical appliances or software [33]. The use of different probes provides complementary advantages, for instance by detecting events that another local security element is not capable to detect accurately. Behavior analysis and signature detection are performed locally, thus shaping a distributed system with no single point of failure, and preserving the performance of the protected hosts. Events that are logged by the security probes need to be correlated and analyzed to detect the most sophisticated attacks [33; 31].

Table 1 – derived from a special publication by NIST [31] – compares to a considerable extent the primary technologies which we have discussed above.

TYPE	TYPE OF ACTIVITY DETECTED	SCOPE	STRENGTH
NIDS	Network, transport, and application TCP/IP layer	Multiple sub-network and groups of hosts	Able to analyze the widest range of application protocols;
HIDS	Host application and OS activity; network, transport, and application TCP/IP layer	Individual host	Able to analyze activity that was transferred in end-to-end encrypted communications
Firewall/Routers	Network, transport, and application TCP/IP layer activity that causes anomalous flows	Multiple sub-network and groups of hosts	Very effective in identifying reconnaissance scanning and DoS attacks, and at reconstructing major malware infections

TABLE 1. COMPARISON OF PRIMARY SOURCES OF SECURITY ALERT

Generally, a Security Information and Event Management (SIEM[25]) system should gather data from logs and correlate events among them, including those generated by IDSs, firewalls, antivirus, operating system, application servers, and other security devices. Initially, collected data is normalized by converting the logs into standard fields that allow them to be analyzed. Then, SIEM software generates global alerts based on the data analysis, being capable to detect events that individual probes cannot as well as to verify the accuracy of the alerts. However, the presence of a central analysis element might introduce a delay in performing analysis, as the logs have to be transferred from the local detection systems for correlation.

[25] SIEM denotes a broad class of commercial solutions for managing security-related logs.

3.2.1. IDS

Usually, IDSs and other security technologies log extensive data related to observable events, which can be used for correlating events among all the logging sources.

3.2.1.1. NIDS

Widely used data attributes logged by NIDS include:

- Session ID
- Timestamp (i.e., date and time)
- Event type
- Importance rating (e.g., impact, priority, confidence)
- Application, transport, and network layer protocols
- Source and destination IP address
- Source and destination TCP or UDP ports, or ICMP types and codes[26]
- Number of transmitted bytes
- Decoded payload data (e.g., application requests and responses)
- State-related information (e.g., authenticated username)

In addition, NIDS may log data related to packet capture as soon as an alert occurs [31].

3.2.1.2. HIDS

Common attributes that are logged by HIDS include [31]:

- Timestamp
- Event type
- Importance rating
- Source IP address
- Port
- Application information, filenames and paths
- User ID

3.2.2. ANTIVIRUS

Antivirus software typically logs the following information:

- Timestamp
- Event type

[26] Transmission Control Protocol, User Datagram Protocol, and Internet Control Message Protocol, respectively, are core protocols used for the Internet.

- Threat name
- Threat type
- Filename and path
- User ID

3.2.3. FIREWALLS AND ROUTERS

Usually, firewalls and routers record basic attributes of blocked connection attempts only, including the following [31]:

- Timestamp
- Source and destination IP addresses
- Transport layer protocol
- Source and destination TCP or UDP ports, or ICMP types and codes
- NAT[27] IP addresses

Latest routers may also log the following attributes:

- Event type
- Importance rating
- Application, and network layer protocols
- Number of bytes and packets transmitted by source and destination
- Other packet header fields

3.3. CHARACTERISTICS OF SECURITY DATA

In this section we introduce the structural characteristics of the data which contribute to incident detection. First, we can distinguish between **event-data** and **meta-data**. The former refers to attack-related data which is required for detecting IT security incidents; the latter represent the additional data not directly related to incidents, which, however, can reveal useful information for the detection. For example, knowledge of network topologies, vulnerabilities, security policies, users, and so on, is considered meta-data. Generally, meta-data can be generated passively – i.e., manually or in time intervals – or actively after the occurrence of a given event [34]. We can classify event data in:

[27] Network address translation is the process of mapping private LAN addresses to public IP addresses.

- **Alert:** warning message which shows the occurrence of a suspicious event in monitored systems (i.e., through analysis of raw data such as logs, packet payloads, flow data, and so on).

- **Intrusion Alert:** alert with higher likelihood of being malicious.

- **Meta-Alert:** global warning message obtained by grouping more alerts together.

- **Event Report:** message warning about the occurrence of a cyber-attack; it shows the causal relationships among suspicious events.

- **Incident Report:** message containing the reconstruction of a multi-stage cyber-attack. The message is produced by identifying the correlation among the security threats enclosed in the event reports.

3.4. EVENT CORRELATION

In the previous section, we stated that attack-related data is needed to detect incidents; this data can be seen as a subset of the entire dataset, which consists of observable events in a network (and underlying computer systems). A user who sends an email, a server that receives a request for a webpage, a firewall that blocks a connection attempt are only few examples.

We define **event**:

> ***Any observable phenomenon from which it is possible to extract features for the detection of computer security incidents.***

The term “event” abstracts generic input data – e.g., TCP connections, log entries, IDS alerts – for any type of detection engine. Such mechanisms monitor and examine events, and produce alert messages when suspicious events are observed.

Each event is specifiable by a set of attributes. For example, an alert may be specified by sensor ID, timestamp, source IP, destination IP, port(s), user name, process name, attack

class, and so forth. An example of standardized data format for alerts is the *Intrusion Detection Message Exchange Format (IDMEF)* [35], which is discussed in the following chapters.

Again, it is worthy to set apart meta-data information from events. The former is made up of additional information that is useful to understand and evaluate an event – e.g., linking disconnected events – although it does not take part either in the event itself or other events.

Usually, the degree of incident detectability may vary and deep expert knowledge is strongly required for analyzing efficiently the signs of a potential incident. Cichonski et al. [26] define **precursor** the sign of a potential incident in the future, whereas **indicator** is a sign that an incident has occurred or may be occurring.

Examples of indicators are: alerts from a network intrusion detection sensor when an attempt of buffer overflow occurs against a DB server; alerts from antivirus when the system is infected; filenames with strange characters; multiple failed login attempts in application logs; rare network traffic flow. Precursors are, for instance: log entries containing evidence of the use of a vulnerability scanner; a verbal threat from a group of hackers. However, from a target's point of view (i.e., the system under attack), precursors are not always detectable. The most common precursors and indicators are extracted from software alerts, logs, people and public available information.

Large amount of event-base data such as firewall and IDS alerts, log files, routing information and so forth, can be collected and used for security analysis, since these data represent the network activity at a certain moment and location. Attack evidence can be extracted from the overall amount of event logs within a distributed system [26]. However, the large amount of heterogeneous data, which are distributed over time and locations, make it challenging to find relationships between events and extract attack attempt from them [10; 12].

We provide the following definition of **event correlation**:

> ***A set of techniques to investigate event patterns, combine suspicious events into meaningful entities and help detecting attack strategies.***

Event correlation can be seen as the cognitive process of human brain linking information from sensory organs, assessing situations, making decisions, and regulating actions [36]. By

identifying significant events within a large set of logged events, one is able to get a complete picture of an occurring incident. Event correlation should be able to work regardless of the type of sensor or system; IDS sensors and other security sources are intended to be functional components which may run individually. Also, it should be possible to correlate events in diverse scenarios [34].

Since heterogeneous security sensors are capable of observing events from multiple perspectives, it is necessary to analyze suspicious events and identify which event patterns refer to an identical incident. The use of multiple technologies is essential to detect security incidents as it allows gathering more extensive information than using a single technology or detection method; they provide various output data which specify different viewpoints of the same incident.

Given the large volume of possible suspicious events, correlation system also faces the challenge of decreasing the rate of false positives and false negatives, in order to detect advanced attack strategies with satisfying accuracy [18; 13]. In order to reduce the rate of false negatives, Morin et al. [13] also suggests the make use of heterogeneous analyzers and log resources to get many views of the attacks. Different log resources which utilize different analysis methods may reinforce each other and increase the degree of confidence about the occurrence of an incident [37].

However, many sensors may also trigger multiple alerts which are cause of the same event. Hence, one goal of a correlation system is to decrease alert redundancy in an effective way. Nonetheless, each sensor provides a large volume of suspicious events thus – at individual sensor level – correlation focuses on decreasing the number of alerts [37; 13; 19] [18].

Heterogeneous sensors also lead to a problematic interpretation of the data [18; 19]; another goal is therefore to provide similar data formats to each event, which even allow for the transmission of detailed information to security operators [38; 13].

3.5. SUMMARY

In this chapter, characteristics (and requirements) of modern organizations' defense mechanisms against cyber-threats are discussed. First, we discussed the sources and the common data attributes that are available for detecting computer security incidents. Second, we introduced the various types of structure of event data, i.e., alert, intrusion alert, meta-alert, event report, incident report; these are attack-related messages required for cyber-security incident detection. Third, we defined "event" as any observable phenomenon from which it is possible to extract features for the detection of computer security incidents. Alert

messages are particular events which are assessed as being suspicious by the relative detection engines. Events can be divided in precursors and indicators, depending whether they are related to potential or already occurred incidents respectively. Then, we defined "event correlation" as a set of techniques to investigate event patterns, combine suspicious events into meaningful entities and help detecting attack strategies. Based on previous considerations and challenges identified in literature (see page 6), we are finally able to summarize the goals of event correlation as follows:

- identifying significant suspicious events;
- identifying event patterns in heterogeneous datasets which refer to the same incident;
- providing similar data format to each event;
- decreasing false alert rates;
- reducing data redundancy;
- reducing data volume;
- supplying detailed information to security operators;
- providing a complete picture of an occurring incident.

4. STATE OF THE ART

According to several researchers, there is no consensus in literature about how the correlation process should be implemented and evaluated [14]; hence many models have been proposed in the last years. Some of the correlation techniques presented in literature are introduced in this section.

In literature, the term *alert correlation* suggests a method that matches event information from multiple sensors, allows to group different alerts into attack scenarios and helps network administrators to analyze attack patterns [39; 26]. It is sometimes utilized in literature as equivalent of *aggregation*, *clustering*, or *fusion*; yet these terms limit to describe the capability of grouping alert messages together.

Several correlation methods have been proposed with the goal of cutting down the overwhelming amount of alerts and to give an overview of the security status of IT assets within organizations. Correlation systems are hitherto based on either machine learning or other techniques such as statistical analysis, logical rules or graph algorithms. These methods depend upon the creation of a system and its regular maintenance by security experts, as attack scenarios mutate very frequently [39] [36].

A broad classification of correlation techniques in literature is provided by Ren et al. (2010):

- methods based on classification and *clustering of similar alerts*;
- methods to enhance *detection accuracy* based on filtering of false positive and low-interest alerts;
- methods that aim at regulating *alerts' priority* based on their harshness;
- methods based on *alert causality* analysis.

4.1. ANOMALY AND SIGNATURE DETECTION

The emergence of event correlation for the detection of security incidents is supported by the introduction of the basic elements which allow an early classification of event correlation methods. We already discussed the characteristics of security data and formats in 3.3. In terms of correlation techniques, an essential distinction is between signature- and anomaly-based techniques.

Attackers target vulnerabilities of computer systems and perform distinct sequences of activities per each attack, which are also known as *attack patterns*. Techniques of **signature**

(or misuse) detection focus on analyzing patterns in data to identify and stop an attack when a known misuse occurs. That is made possible by measuring the similarity between events and signatures of known attacks [17]. Therefore, a signature works as a filter for incoming events: their matching enables the notification of higher layers of an incident management system [34].

The signatures are constantly developed by public and private research teams. Hence, through misuse detection, known attacks are detectable rapidly and with a low rate of false positives, while novel attacks will not be identified [17], as they represent a new form of misuse. The latter might remain unknown even for months, until the corresponding signature will be discovered and embedded in the detection system [32].

Hence, the reliability of event correlation depends on detection rules, which may strongly affect correlation algorithms when inadequately defined. Moreover, large sets of signatures bear on the computational performance of a correlation system [34].

Signature-based detection takes into account features of well-known incidents to detect new ones; thus, producing a **Black List** of the characteristics of known attacks, it will be possible to detect new instances. Blacklists are generally obtained by gathering two types of knowledge:

- Reactive knowledge includes attacks detected by others (e.g., complaining customers), watching live attacks, luring attackers (e.g., honey pot), and so on.
- Proactive knowledge includes human intelligence intrusion techniques, trying the intrusion oneself (e.g., slight modifications of known instances, completion of new attack instances based on inside knowledge, black-box testing), and so on.

The gained knowledge is analyzed and used to extract rules, select raw data and data features on which intrusion detection will be subsequently performed.

Intrusive events also represent anomalous activity, which can be detected in two ways: by comparing observed behavior with a predefined range of parameters, or by using data mining techniques in which the correlation algorithms are trained with normal behaviors and then occurring events are monitored. The pattern in data which is not consistent with the built baseline is considered – and reported as – anomalous behavior [17; 34]

Hence, through **anomaly detection**, new and unusual attacks can be identified but the built profile should be adaptive in order to remain accurate and keep a low rate of false positives. It should be taken into consideration when developing a correlation system [34]. As a matter of fact, the rate of false tends to be raised because of a general imbalanced distribution of normal and anomaly data. It is also hard to define the detection sensitivity and the boundaries between the two types of behavior, which widely vary. In addition, given large volume of high-dimensional-feature data are difficult to be monitored and analyzed, it demands efficient data processing and pattern learning algorithms [17].

Anomaly-based detection considers well-known 'good' behavior – so a **White List** is utilized – to detect any deviant conduct which suggests the occurrence of a security incident. Applied to anomaly detection, knowledge is extracted by mining and analyzing normal behavior from a sample data set. Profiles are built statistically (e.g., monitoring statistical distribution of activities), using data mining methods to find meaningful activities and features (e.g., clustering and outlier detection, associate rule discovery), or using machine learning methods (e.g., system call sequence analysis, hidden Markov model) [40]. Gained knowledge is used to select raw data/probes, data features, algorithms and related parameters. However, the above mentioned process of selection relies on availability of data, system architecture, network layout, Operating System and application settings, corporate security policies, and so forth.

4.2. CORRELATION LAYERS

Event correlation can be considered as running in a layered security architecture and correlation techniques can be applied in every layer; Limmer and Dressler [34] are able to distinguish among raw data layer, event layer, and report layer. In the raw data layer, data from sensors is collected, processed and sent to the event layer, where IDSs and correlators classify, prioritize and discard non-relevant data. Then, obtained data is sent to the report layer, where it is post-processed. In the **raw data layer**, correlation methods mainly focus on aggregating the large amount of data which is produced by each sensor, extracting data features and detect simple incidents, such as port scans.

In the **event layer**, events that originate from lower layers – where traffic data analysis is performed – are processed, mainly aiming to aggregate alerts into meta-alerts, and gather as much information as possible to identify the event which causes the incident.

Meta-alerts (or *hyper-alerts*) are generally created by aggregating alerts with similar features caused by different sensors, in order to obtain high quality resulting information, minimize

redundancy, and reduce the volume of alerts. For example, alerts originating from the same source and having the same target IP address [18], alerts caused by the same event, alerts referring to the same vulnerability, alerts caused by events belonging to the same TCP/IP session, alerts that are related on a temporal basis, might be considered similar alerts and be grouped together [13]. Measuring and assessing similarities of attributes in alerts is the core of probabilistic reasoning methods [41].

A meta-alert can be seen as an intrusion report at a high-level of abstraction and its content is a function of the attributes of the merged alerts and contains references to them. Further alerts can also be merged with meta-alerts, thus creating a hierarchical structure. It can be represented as a tree in which the latest meta-alert is the root, sensor alerts are leaves, and references are followed to identify successor nodes [14]. Aggregation of alerts is possible if they have in common at least one causal event, by which is meant the event causing an alert. The association among causal events is heavily dependent on their timestamps that may diverge from the timestamps of the alerts which is raised by the detection systems [13].

In order to detect a security incident, causal connections between events must be identified at the event layer. According to Zhai et al. [42], Zhu and Ghorbani [10], and Ren et al. [6], it is also possible to classify correlation techniques into the following four categories, with respect to causality:

- **Similarity-based** approaches correlate events based on the similarity between data features. Alert messages are compared if they have similar attributes or features. Generally, the feature similarity between two events is computed by a function and the result is associated to their relation. This score will determine if two generic events can be correlated. If a high degree of similarity among features is identified, alerts are correlated [41]. Hence, similar alerts are shown to be clustered efficiently by techniques based on similarity. In other words, coarse-grained alert messages (e.g., meta-alerts) are created and the amount of alerts is considerably reduced. On the other hand, the main weakness of these techniques is that they are generally unable to discover causality between time-related events.

- **Scenario-based** techniques are based on known attack scenarios, which are specified by languages such as STATL [43] or LAMBDA [44], or learned from training datasets. Nonetheless, it can be used only to discover the causal relationship of alerts among known pre-specified situations. Attack

scenarios are either specified by experts or learned from a training dataset, although a database with this knowledge may be expensive to build and maintain. Therefore, significant expert knowledge is required to carry out the costly and fallible process of developing attack scenarios in advance. This set of techniques is generally very effective to detect well-known cyber-attack attempts, but incapable for the detection of novel and more sophisticated attack techniques.

- **Multi-stage-based** techniques correlate events utilizing well-known prerequisites and consequences of multistage attacks. They are based on the assumption that suspicious events are usually related to different steps of a multi-stage attack. Predefined rules based on pre-requisites and consequences of attack stages allow identifying related alerts. Despite this approach have the potential to discover unknown attack patterns, it is difficult to define all attack prerequisites and possible consequences beforehand (even for known attacks) [10]. These techniques aim at reconstructing attack scenarios by looking for individual steps that belong to the same attack attempt and linking them. The detection accuracy of these approaches strongly relies on the false alarm rate of each raw data stream which contributes to the correlation. This is due to the fact that most of the methods in this category focus only on events that can be correlated, while discarding all the others. It is relevant to consider that building a comprehensive knowledgebase which considers all the variations of incident steps may be an expensive process, as a large (and increasing) quantity of diverse incident types occurs.

- **Statistical-based:** relationships among alerts occurring within a certain time period is analyzed statistically and it may be independent of prior knowledge. Correlation engines based on Bayesian networks are presented in literature to discover dependency among alerts [45; 6].

Many approaches that rely on predefined knowledge of attacks – e.g., using modeling language or attacks' pre- and post-conditions – are not able to identify the correlation in case of novel attacks, similarly to misuse detection techniques. As a matter of fact, one cannot have an *a priori* knowledge of the possible matching conditions between attacks, as intruders always look for unexpected sequence of attack to be unpredictable. Therefore, it is preferable

to bring together the advantages of approaches based on inference and expert knowledge [6]. Knowledge of underlying networks (i.e., a type of meta-data) and relative flaws are also requirements for several event correlation techniques [45].

In the **report layer**, a final analysis allows the output of the lower layers to be abstracted, displayed and post-processed. It leads to a verification of incidents and a detailed view of the detected threats.

It is relevant to mention that such hierarchical structure – i.e., the three above mentioned layers – ensures that each correlation component may be seen both as analyzer and correlation engine, depending on the place from which they are viewed; from the top they are just general event analyzers (or detection engines), while downstream they look like event correlation engines [34].

4.3. LIMITATIONS

Several techniques have been proposed in the past years to correlate security data for different purposes, such as filtering data, finding causality, clustering, and so on. Correlation techniques based on misuse are generally not able to discover previously unknown attacks, for which relative signatures has not been produced yet. Moreover, inadequate and incomprehensive signatures may compromise the quality of correlation, whereas use of a large collection of signatures usually requires extensive labeling work, large storage and intense computing power. Techniques based on anomalies overcome the problem of detecting novel attacks but it may be generally difficult to determine detection sensitivity and criteria for differentiating between normal and anomalous behaviors. Also, pattern learning and adequate processing power are demanded for monitoring and analyzing the large sets of features of security data.

Each of the four methods which allow finding causal relationships between events present some limitations. In general, by looking at feature similarity it is impossible to discover causality between two events which exhibit temporal connections. Techniques based on predefined attack scenarios require extensive knowledge about possible attack patterns, thus introducing all the limitations that have been above discussed for signature-based techniques. Techniques based on pre- and post-conditions present the same weaknesses of scenario-based and – broadly speaking – signature-based techniques, although they potentially allow identifying previously unknown attack patterns. The detection accuracy of multi-stage correlation approaches strongly relies on the false alarm rate of each raw data stream which contributes to the correlation; this is due to the fact that most of the techniques in this

category focus only on events that can be correlated, while discarding all the others. It is relevant to consider that building a comprehensive knowledgebase which considers all the variations of attack steps is unattainable in reality, as well as being an expensive process as a large (and increasing) quantity of diverse incident types occurs. Many approaches that rely on predefined knowledge of attacks – e.g., using modeling language or attacks' pre- and post-conditions – are not able to identify the correlation in case of novel attacks, similarly to misuse detection techniques. As a matter of fact, one cannot have an a priori knowledge of the possible matching conditions between attacks, as intruders always look for unexpected sequence of attack to be unpredictable. Finally, techniques based on statistics usually do not require prior knowledge about threats to detect causality between events but they are more time consuming than the other methods as well as extremely sensitive to noise and time delays. Therefore, it is preferable to bring together the advantages of complementary approaches based on inference and expert knowledge [6]. Knowledge of underlying networks (i.e., a type of meta-data) and relative flaws are also requirements for several event correlation techniques [45].

Although a number of scientific papers proposed correlation approaches and taxonomies, comparative studies are not feasible. First of all, the scientific community blames the lack of public evaluative datasets, specific for testing alert correlation methods. As a matter of fact, the examined state-of-the-art techniques are evaluated by the respective authors using a large variety of datasets such as customized private datasets, DARPA, DEFCON, and so on. Publicly available datasets are not specifically designed for event correlators, rather they were conceived for evaluating intrusion detection systems (IDSs); besides some of them are largely considered obsolete or unrealistic. Moreover, each dataset implies the use of diverse background network traffic and threats, both for learning and testing, as well as different network topologies.

The lack of a common scientific vocabulary related to alert correlation, makes it complex to bring techniques into comparison. Therefore, it is generally accepted that the performance of every approach depends on specific situations; that is because each of them present strengths and weaknesses which cannot be assessed without an empirical analysis.

4.4. SUMMARY

A broad classification of correlation techniques depicted in literature allows differentiating among methods which group similar alerts together, improve detection accuracy, regulate event priority, and analyze event intercausality. Another distinction is between signature-based detection and anomaly-based; the former considers instances of known threats to

identify novel ones, while the latter detects behavior which deviates from a baseline of well-known ‘good’ behaviors.

Event correlation can also be divided into three layers: raw data layer, event layer, report layer. In the raw data layer, data from sensors is collected, processed and sent to the event layer, where IDSs and correlators classify, prioritize and discard non-relevant data. Then, obtained data is sent to the report layer, where it is post-processed. In particular, messages are aggregated and causality between events is identified within the event layer.

With respect to causality, correlation techniques can be classified as follows: similarity-based, scenario-based, pre/post conditions-based and statistical-based. Techniques based on similarity, predefined scenarios, and pre/post conditions require extensive expert knowledge to find correlation among events, whereas statistical methods are able to infer relationships among events by using statistical or machine learning analysis. Generally, methods based on expert knowledge produce accurate results but experts may be limited in solving complex problems and in processing information. Finding all possible causal relationships among alerts may be a time-consuming and excessively slow activity for being considered viable. Methods based on automatic inference are capable to find novel alerts, but they are not completely able to discover causal relationships between alerts though.

5. SOLUTION DESIGN

In this chapter, the aim is to design an innovative model for the detection of advanced security incidents by examining contemporary attack scenario cases. The first section introduces a generic tiered architecture which is usually employed to detect security incidents. In the second section, event correlation, which is broadly defined in literature, is explained through the identified essential components that participate in the correlation process. The final section shows how detection takes place for the cases introduced in chapter 2, which range from common low-sophisticated threats, such as port scan, to more complex and multi-stage attacks, such as cyber-espionage activities.

5.1. DETECTION CHAIN

The term 'detection chain' represents the sequence of tasks that are commonly performed during the process of incident detection. Generally, raw data, gathered from a detection probe, is preprocessed for the extraction of data features which are fed to a detection engine; the latter generates alerts which are then investigated – possible false alerts are useful to tune up the detection engine as well as to improve detection signatures. Then, intrusion alerts – i.e., alerts which are accepted as true positives – are reported and incident response may take place. Given that only one analysis engine identifies suspicious events, we call it (1-tier) detection chain (see Fig. 3 Example of 1-tier detection chain).

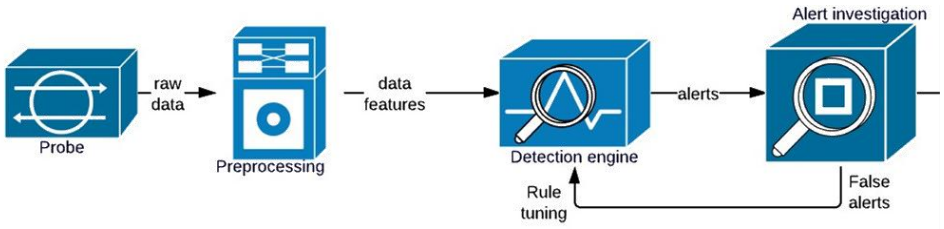

FIG. 3 EXAMPLE OF 1-TIER DETECTION CHAIN

In a **2-tier** detection chain, raw data, gathered from a detection probe, is preprocessed; the detection engine checks the relevant data features against intrusion rules (or looks for anomalies) and raises alerts. Then, alerts are further examined by a subsequent detection engine, in order to create more accurate and informative alerts. Again, investigation will set apart false alerts which in turn will help to tune up the detection rules. Positive alarms are reported and response may take place. Fig. 4 Example of 2-tier detection chain shows an example of 2-tier detection chain with sequential detection engines. In general, however, engines can also be deployed in parallel.

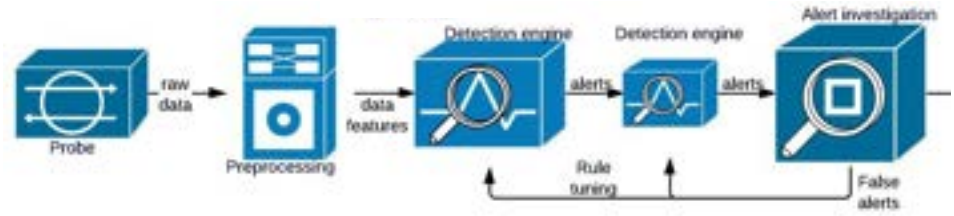

FIG. 4 EXAMPLE OF 2-TIER DETECTION CHAIN

In general terms, we refer to **n-tier** detection chain when each raw data stream, gathered from multiple information sources, is separately preprocessed. Features are obtained separately and checked against detection rules to trigger alerts. A subsequent detection engine will jointly inspect and correlate the separate alert streams, in order to create incident reports.

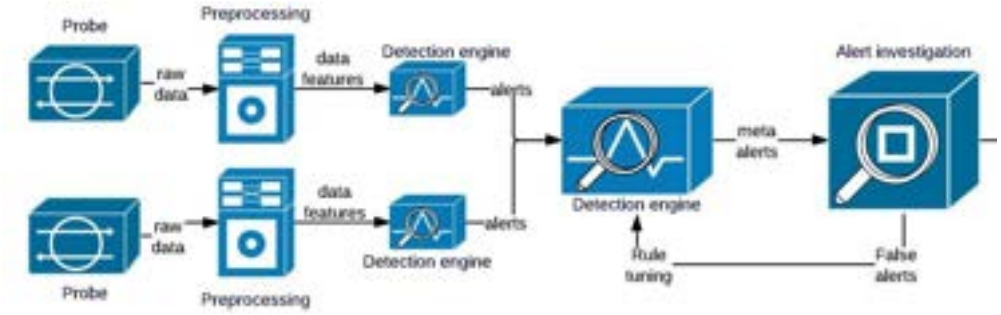

FIG. 5 EXAMPLE OF 3-TIER DETECTION CHAIN

5.2. THE 6 'COMPONENTS' OF CORRELATION

Each technique presented in literature emphasizes on one or more aspects of the broad concept of 'event correlation', and their relative goals (see chapter 4). Our project is grounded on the previous research efforts made by Cuppens [46], Valeur et al. [14], Qin [45], and Sadoddin and Ghorbani [47]; in essence, they identified six components which characterize an exhaustive correlation system: n*ormalization, aggregation, correlation, false alert reduction, prioritization, and intrusion strategy analysis*. These aspects can be categorized with respect to their input data, their goal as well as the location in the detection chain.

5.2.1. NORMALIZATION

It has been said throughout this document that heterogeneous security data sources supply different data formats, and it is necessary to standardize the inputs for correlating events, ensuring interoperability between security components.

Besides, normalization supplies incomplete or missing data attributes that may be relevant for correlation, thus augmenting events with further information. In that regard, the data content is reported, which – according to the IETF RFC4766 memorandum [48] – is the minimum requirement for the information exchange within computer security incident management systems, regardless of the information source:

- **Detected data**: the monitored event should be described in unambiguous way.
- **Event identity**: global categories should be created and used to classify events.
- **Background information**: meta-data such as system configuration, topology, sensor information, and so forth, is necessary to assess events adequately.
- **Event information** (source and target identities): information about the source of the event and the attack's target (i.e., IP address or domain).
- **Event impact and degree of confidence**: importance of an event. Knowledge of the event source affects the correlation.
- **Automatic response**: proposals for countermeasures to be adopted by the analyzer after an occurred event.
- **Time Synchronization**: common time basis is relevant for the deployment of a distributed architecture.

This component may not provide any tangible improvement such as reduction of the number of alerts; however, it is crucial for the correct functioning of the correlation process through the successive components.

5.2.2. AGGREGATION

Basically, a high-level view of what is currently occurring in a network is the desired outcome of the events fusion process. Therefore, correlation engines should be capable of recognizing scenarios through the intelligent fusion of data generated from different sources into *meta-alerts*. In order to perform this task, it is desirable to have some degree of awareness of what kind of events will lead up to others in certain scenarios and of what features are expected to be shared [41].

The component of aggregation is intended to cluster similar alerts together into meta-alerts, based on their features. *Meta-alerts* (also known in literature as *hyper-alerts*) are generally created by aggregating alerts with similar features caused by different sensors, in order to obtain high quality resulting information; for instance, alerts originating from the same

source and having the same target IP address [18], alerts caused by the same event, alerts referring to the same vulnerability, alerts caused by events belonging to the same TCP/IP session, alerts that are related on a temporal basis, are considered similar alerts [13]. Measuring and assessing similarities of attributes in alerts is the core of a probabilistic reasoning method, and aggregation may be controlled by configurable parameters [41].

A meta-alert can be seen as an intrusion report at a high-level of abstraction and its content is a function of the attributes of the merged alerts and contains references to them. Further alerts can also be merged with meta-alerts, thus creating a hierarchical structure. It can be represented as a tree in which the latest meta-alert is the root, sensor alerts are leaves, and references are followed to identify successor nodes [14].

Clustering of alerts is possible if they have in common at least one *causal event*, by which is meant the event causing an alert. Aggregation provides an undeniable advantage in reducing alert redundancy, easing later reduction of false positive alerts and causality as well as human analysis.

5.2.3. CORRELATION

In order to find the causality among events, a correlation component is required; it will provide a ‘big picture’ of the occurring incident. It is essential taking into account that false positive inputs may heavily compromise the accuracy of the correlation as it can result in complete but inexistent attack scenarios.

5.2.4. FALSE ALERT REDUCTION

It is known that low-level sensors may generate alerts with high false positive rate, making the final outcome of correlation process unreliable. Therefore, false alert reduction is intended to differentiate between true positive and false positive events. For this purpose, it is also crucial to combine the values of confidence supplied by various sensors and related to the same event into a global value.

5.2.5. PRIORITIZATION

Security incidents may have more or less severe impacts on targeted networks. As security policies, network topologies, vulnerabilities, and other meta-data are taken into account, alerts can be prioritized depending on their severity. Hence, it is essential to develop specific classification constructs and model domain knowledge adequately.

5.2.6. INTRUSION STRATEGY ANALYSIS

The importance of finding the causal relationship between events is long been recognized. However, correlated events usually show isolated attack patterns, and not the complete intention of the attacker. This may be due to events which are potentially missed by security systems – i.e., false negative alerts. Therefore, higher level correlation is desirable to identify a complete picture of attack patterns occurring in a network as well as predicting future attack steps. This component considerably improves the abstraction level of the meta-alerts.

5.3. DESIGN OF SOLUTIONS FOR THE CASE SCENARIOS

The objective of this section is to show how detection takes place in the scenario cases using the detection model which is depicted in 5.1. The blocks showed in the detection chain can be mapped to correlation methods in order to reason about cyber-attacks. The design approach is based on known attack scenarios, thus addressing the difficulty of reasoning about unknown threats. This complexity is typical of the reactive nature of cyber-security. Therefore, known scenarios will help providing the abstraction of a solution for incident detection based on correlation. The relevance of the following attack scenarios is previously reported in 2.2.

5.3.1. SCENARIO 1: GAINING UNAUTHORIZED ACCESS AND DISCLOSURE OF SENSITIVE DATA

In this section, we aim at detecting an attack scenario which is characterized by four attack steps. That is to say, the attacker's intrusion strategy is performing the following four steps in the attempt to gain remote access to a local machine (i.e., a public server in DMZ) and obtain sensitive data from the organization:

1. The attacker performs a port scan with Nmap (Network Mapper security scanning tool) on externally visible IP addresses in order to find open ports in the targeted organization's computer systems.
2. The attacker scans for vulnerabilities on the servers' public IP address from the Internet by using a vulnerability scanning tool.
3. The tool identifies vulnerability in the FTP server, whose port is open. The attacker is able to exploit it, and launches a remote-to-local (R2L) attack to the FTP server, using a different IP address from previous steps 1 and 2.
4. The attacker downloads a large database file from the FTP server.

In order to obtain a 'big picture' of the attack is necessary to detect the four steps individually. Each of them defines an attack pattern, which in complex multi-step scenarios may (or may not) be causally linked to the real intentions of the malicious user. Hence, let us proceed step-by-step.

5.3.1.1. STEP 1

Let us consider to detect the incident by using a simple detection system with one probe and one detection engine. As previously discussed, this relies on the use of a single probe placed behind the perimeter firewall of our corporate network; which is equivalent to say that we are taking into account only network packets for analysis as raw data.

It is generally believed that a port scan is not strictly an attack, but a precursor to a more complex attack, therefore it is widely accepted that detecting a scan is an important early warning detection process. In order to adequately detect attack precursors – and with a good sense of timing – network traffic should be collected real-time.

Port scan detection can be performed in two ways:

- By looking for anomalies – a baseline containing normal behavior is essential for this purpose;
- By matching characteristics of malicious patterns – characteristics of port scans must be well-known in order to create adequate signatures.

Let us assume that IDSs in use do not adequately detect port scans. A probe, located behind the perimeter firewall, gathers TCP packets by capturing network traffic[28].

A pre-processing phase will provide for the extraction and selection of data features for the incoming network packets; this task will be performed by summarizing packet information into higher level connection records. Each of the records will be a sequence of TCP packets which bidirectionally flow from a source IP (and port) to a target IP address (and port) within a pre-specified time window.

The goal of our detection system is to correlate TCP connections together and to provide a reliable event report warning about suspicious scan activities: aggregating connections, identifying causality, and reducing the volume to analyze by prioritizing them. Moreover, the aforementioned processes can be executed in different order. In other words, we want to implement a set of mechanisms that provide for the detection of port scans while not

[28] PCAP library for *nix systems http://www.tcpdump.org and Windows systems http://www.winpcap.org

flooding the incident response team with tens or hundreds of similar alerts. First, we can use a database to store and manage TCP connections.

PRIORITIZATION

In order to reduce the data set significantly, TCP connections can be prioritized by gathering contextual information which help assessing the severity of the network activity. One solution would be to create and maintain two databases: one including an *incident handling fact base,* and the other one containing topological information [49] (see Appendix B for further details).

FIG. 6 AN EXAMPLE OF PRIORITIZING TCP CONNECTIONS

This step results in a relevance score for each TCP flow, which is the degree of dependency between incident and the configurations of network and hosts. Security analysts evaluate the level of criticality of resources and assets and the success probability of the incident to compute an overall incident rank; then, users may prioritize TCP flows according to the ranking and dynamically filter out connections which are not relevant for the detection of port scans.

AGGREGATION

In this scenario, port scan packets have same source IP and target IP addresses, but different TCP destination ports. Alerting consists of producing high-quality information which show a number of scanning instances from an arbitrary remote location of given ports of the target system, in a certain time period. Hence, one possible solution is to group together similar TCP flows and count the instances; if the number of instances exceeds a threshold set by the expert, an alert is triggered.

An option to reduce the data sets to be analyzed is the use of temporal-based fusion and grouping events which belong to a certain time pattern together [50]. In general, we can face the following two possible cases: (1) the time gap which occurs between a scan probing and another is smaller or larger than the time window used for sampling. If the gap is small enough, flows allow fusing duplicate features (i.e., IP source and target addresses). On the contrary, (2) if the gap between scan probes is bigger than the time window, our system will

consider port scans independently, thus generating a number of similar alerts which will require further processing.

TCP flows can be clustered together if they have similar features [18; 49], according to rules produced by experts. First, a simple technique has been proposed to cluster data by similar source IP address [51]; however, this technique is not convenient for stealth scanning and does not take in any consideration the target.

Clustering process can also be based on predicate logic, a formal system in which similar features specify the similarity between TCP connections. Every cluster may be seen as a high level TCP flow, according to a *cluster stability* – different attacks set maximum delay times for cluster stability [52]. Similarly, a function can assess similarity between features and return a value between 0 (i.e., totally diverse TCP connections) and 1 (i.e., identical connections) in order to cluster connection records. Subsequently, similarity values are averaged [41]. Analogously, it has been proposed to group connections having identical feature values, regardless of the time [53], since slow scanning – say a day or more – may be used as stealth technique.

Alternatively, it is possible to organize connections in classes, and use a simple machine learning algorithm of nearest-neighbor (NN) [54] – TCP connections are assigned to the class of its nearest neighbor – to decide whether a flow can be considered unrelated to others or be clustered with related ones. In order to train this algorithm, it is generally required to compute the distance between instances, which is based on values of the feature set (see 6.3 for further details).

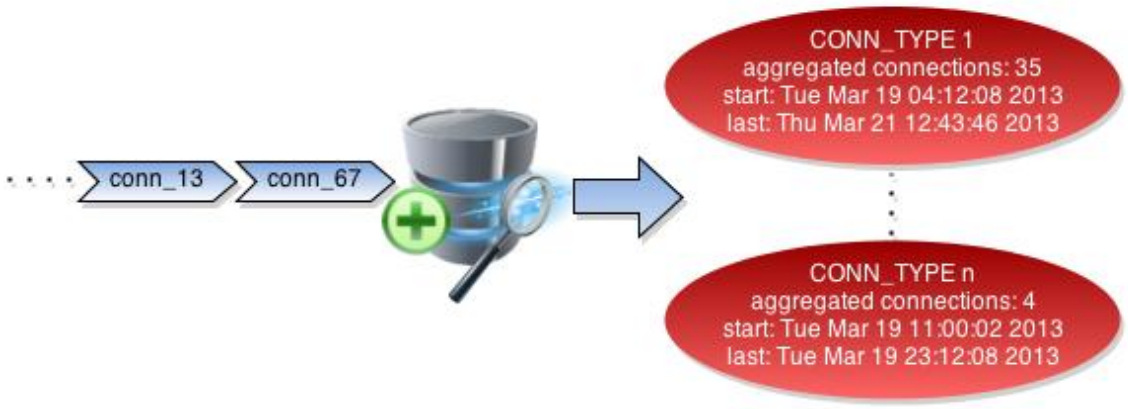

FIG. 7 AN EXAMPLE OF AGGREGATION OF TCP CONNECTIONS

This step results in the fusion of suspicious TCP connections into one or more clusters, thus reducing the volume of data to be further analyzed by the detection engine. Each cluster can

be considered as a ´global TCP connection´, which will be more informative and accurate than the ones forming it. Fig. 8 summarizes the detection system for step 1.

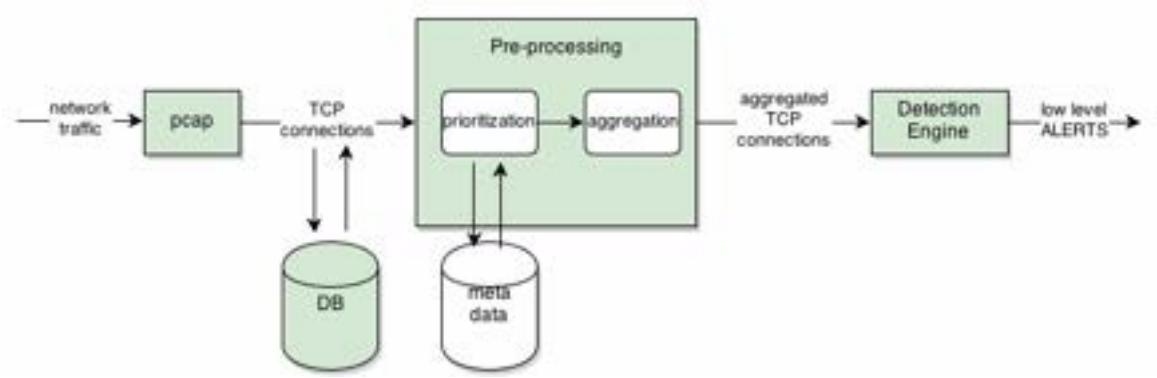

FIG. 8 1-TIER PORT SCAN DETECTION CHAIN

5.3.1.2. STEP 2

The attacker scans the server's public IP address from the Internet by using a vulnerability scanning tool. Let us assume to detect the vulnerability scans by using a detection system with a single detection engine. In other words, our NIDS – e.g., SNORT with signature detection engine – provides alerts showing the presence of vulnerability scans from a remote IP address, as well as other alert types. Analogously to port scanning, vulnerability scans are considered precursors to attacks, therefore the similar considerations previously done with port scan are valid in step 2 too.

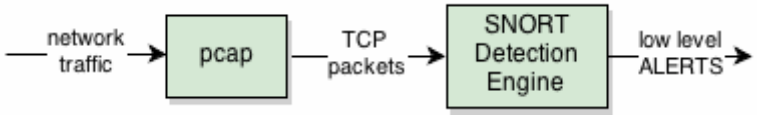

FIG. 9 1-TIER VULNERABILITY SCAN DETECTION CHAIN

Hereby, we do not pay close attention at the event trigger mechanism of the NIDS, which is based on expert rules. We want to further examine what is occurring in the network and correlate the alerts produced by our NIDS – located behind the perimeter firewall – with the alerts triggered in step 1. Hence, let us assume that we have port scan low-level alerts from the earlier detection engine, as well as low-level alerts from vulnerability scanning.

The goal of our detection system is to correlate the two types of alerts (i.e., port and vulnerability scan) and provide a reliable event report warning about suspicious scan activities. Essentially, we want to implement a set of procedures that provide for the

detection of port and vulnerability scan while not flooding the incident response team with tens or hundreds of similar alerts.

NORMALIZATION

First, a normalization process is required to provide a unified data model to our low-level alerts, according to the IDMEF requirements [35]. Then we can use a database to store and manage alerts.

FIG. 10 AN EXAMPLE OF NORMALIZATION OF HETEROGENEOUS ALERTS

PRIORITIZATION

Likewise prioritization in step 1, alerts can be prioritized by gathering contextual information which helps assessing the severity of the network activity. This step results in low-level alerts generated by Snort IDS, which are ranked in accordance with their predetermined impact over the monitored system.

FALSE ALERT REDUCTION

Apart from prioritizing the alerts, it is also crucial to filter out false warnings. Reduction of false alerts is largely assumed to be achieved by combining misuse and anomaly detection [55; 56]. As signature-based detection has been used to trigger our alerts, the remaining data (i.e., uncertain patterns) may be input to an episode mining engine, which compares unknown patterns to frequent episode rules. Episode rules are mined from normal network traffic and maintained in a database for the learning process. Highly mismatching episodes are labeled as anomalous and used to generate new signatures.

Alternatively, false positive reduction can also be achieved including meta-data such as network topology, semantics (i.e., information proving alert relevancy towards vulnerabilities of the monitored system), and other information which define the security state of the system [42; 57].

Since we are looking for anomalies, it is possible to filter data by employing novel machine learning algorithms which model the response of dendritic cells [58] of the autoimmune system, thus differentiating between danger and safe signals. Similarly, classification algorithms [57; 59] are capable to return a confidence score for input data, thus allowing to filter out low-level alerts which have values below a given threshold (see section 6.2 for further details).

FIG. 11 AN EXAMPLE OF ALERT PRIORITIZATION AND FALSE ALERT REDUCTION

At the end of this alert analysis process – i.e., prioritization and false alarm reduction – we will consider only those low-level alerts which might pose a threat to the monitored network, whereas discarded false alerts will be examined and used to tune the earlier detection engines.

AGGREGATION

In this scenario, IP packets of vulnerability scans have same IP source and target addresses as the port scan, but different destination ports. The final event report represents high-quality information which shows several (port and vulnerability) scanning instances from an arbitrary remote location, in a certain time period.

In order to lower the number of remaining alerts, it is possible to use temporal-based fusion techniques [50], such as SVO or *chronicles* [60], thus clustering events which belong to a certain time pattern together.

Alert aggregation could also be achieved by using either feature [53] similarity-based techniques [52; 18; 49; 41] or machine learning techniques such as nearest-neighbor [54].

For the purpose of aggregation, feature similarity may also be explained in terms of distance (i.e., path length) between feature values in the corresponding generalization hierarchy (or taxonomy) [61] [62]; the latter are hierarchical structures, which are built for each of the

features, with the goal of ordering them from general to specific values (see section 6.3 for details).

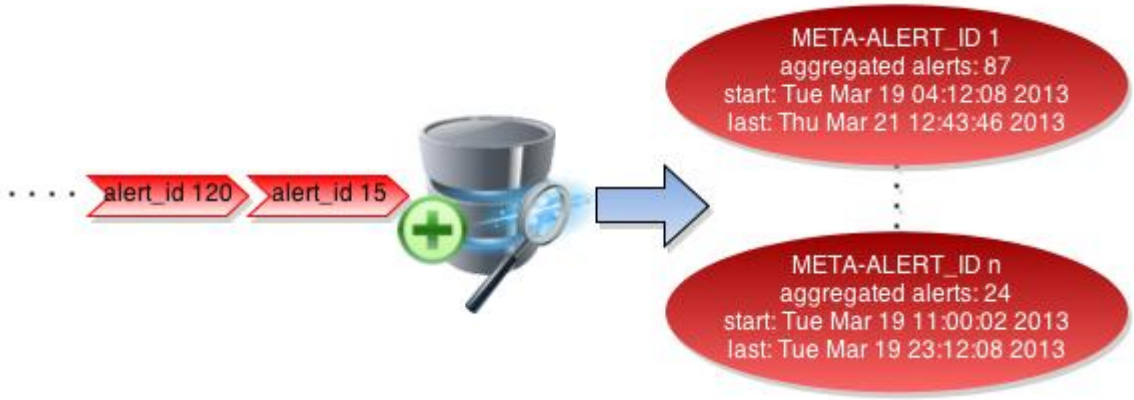

FIG. 12 AN EXAMPLE OF META-ALERT CONSTRUCTION

This step results in alert clusters; each of them can be seen as a global alert which includes a set of low-level alerts, thus eliminating redundancy and reducing the volume of data to be processed.

REPORTING THE EVENT

Alert reduction and production of meta-alerts can occur before [41] or be coupled with the search of causal relationships between alerts [51] [10] [6]; that is to say, correlation can be performed directly on low-level alerts, or else on meta-alerts after alert aggregation. In 1-tier detection chain, reasoning about intruder's strategy of the individual step 1 is the same as simply defining causality between alerts raised by port scans. Moreover, correlation techniques based on pre-requisites and consequences of attacks are generally useless for detecting the individual step within a multi-step attack.

If aggregation has occurred, in the best case scenario – i.e., our desirable condition – we will have just one global informative meta-alert which does not require further analysis and can be directly read as event report. However, if aggregation has produced various meta-alerts for the scan probes (e.g., time-based clustering) it may be necessary to provide a high-level view by examining mutual relationships between meta-alerts. On the other hand, if aggregation and correlation are coupled, we consider intrusion alerts as our input, and event reports will be the output.

FIG. 13 AN EXAMPLE OF META-ALERT CORRELATION

CORRELATION OF META-ALERTS

In order to perform correlation after meta-alerts have been produced, security experts may compute the probability that a feature matches if two alerts are correlated [41]; for example, if two probes from the same target scan the same ports on diverse parts of a sub-network, we have low expectation of matching target IP address.

Time-series analysis such as Granger Causality Test (GCT) [45] is used to check through a statistical hypothesis test whether two time series variables x and y correlate each other. However, the approach requires alerts to show a temporal relationship; therefore, it will not be able to correlate events having random time delays between them.

CORRELATION/AGGREGATION OF LOW-LEVEL ALERTS

As we mentioned above, in some cases correlation and aggregation techniques are coupled. Correlation between alerts can be based on heuristics, data mining and machine learning techniques such as decision trees, radial base function network (RBFN), multilayer perceptrons (MLP) [10; 51], support vector machine (SVM) [10], statistical approaches such as Bayesian Network (BN) [6] [45]; these techniques aim at correlating low-level alerts automatically by learning. Generally, training data is assumed to be used to tune parameters for the various techniques and extensive expert knowledge is required for the manual alert labeling.

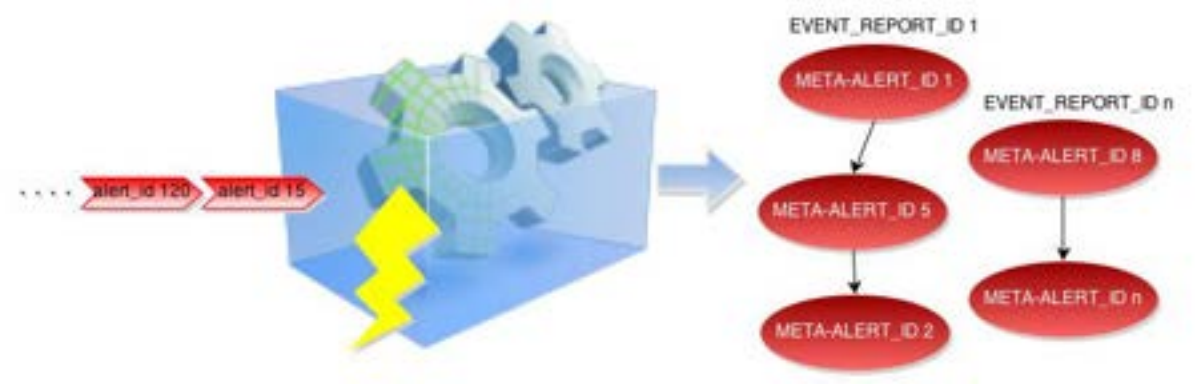

FIG. 14 AN EXAMPLE OF INTRUSION ALERT CORRELATION

This step results in the production of event reports, which can be considered as sets of correlated meta-alerts. Event reports are more informative meta-alerts, since they also supply causality among them. Fig. 15 depicts the architecture of the detection chain for step 2.

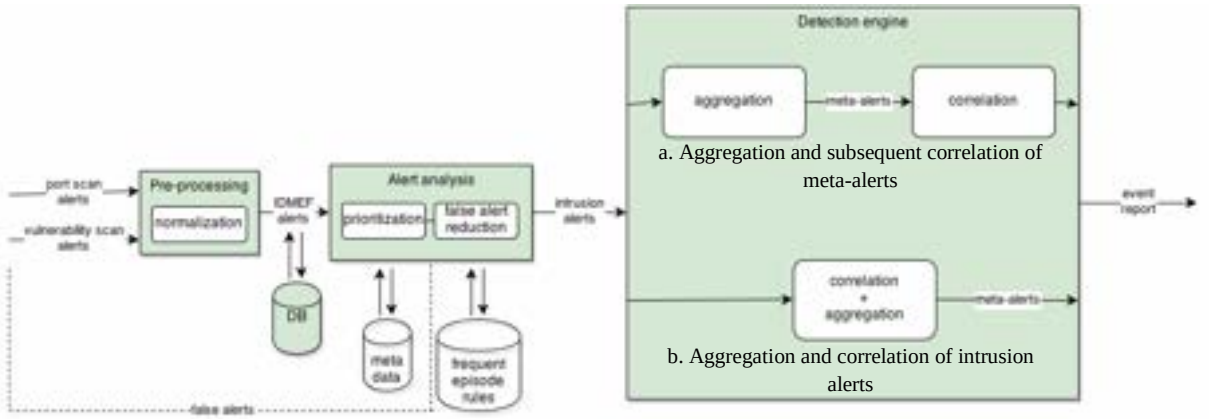

FIG. 15 STEP 2 DETECTION CHAIN (INCLUDING OPTIONS A AND B).

5.3.1.3. STEP 3

If the incident response has not been activated, after scanning the public IP address of the organization's servers, the attacker obtains a list of open ports as well as a list of vulnerable services. For example, the port scanning with *nmap* has resulted in the following:

```
PORT        STATE      SERVICE
21/tcp      open       ftp
22/tcp      open       ssh
80/tcp      open       http
646/tcp     filtered   ldp
1720/tcp    filtered   H.323/Q.931
9929/tcp    open       nping-echo
…           …          …
…           …          …
```

It is simple to understand that a series of TCP ports are open on the server at stake. What the attacker needs to know is whether or not he can actually intrude the system through any of the open ports. The vulnerability scan tools (e.g., Nessus) has probed open services and listed discovered vulnerabilities. In particular, the version of the FTP service, whose port is open, shows a few vulnerabilities; the attacker has found himself able to exploit one of them and launches a remote attack. The intruder has well-judged to use a different IP address than the previous one which is used for the scanning probes of step 1 and step 2.

As step 2 is in progress, security operators will be able to see a global informative event report which displays (and link) the two scan probes. As already stated, the event can be considered an attack *precursor*. Essentially, our system should be able to detect the occurring remote-to-local attack and causally link it to the precursor of step 1 and 2.

A R2L (remote-to-local) attack is usually accomplished by sending an anomalous server request, such as a buffer overflow, in the application payload. It is possible to detect this type of incident in the NIDS as well as in the HIDS of the server. Therefore, for the detection of the current attack scenario let us assume to correlate alerts generated from our NIDS (i.e., Snort located behind the perimeter firewall) and HIDS (i.e., IDS installed on the server), alerts from server log, and the events generated from the attack precursor.

Usually, there may be many exploits and techniques to attack a server, and we may not exactly know the specific technique used by the intruders in order to produce effective signatures. Hence, detection also based on anomalies is advisable; that is to say, let us synthesize normal behavior and create a mechanism which triggers alerts when anomalous events, which are logged in the server log, occur. Profiles can be built by statistical computation, mining activities, or using machine learning methods [17]. In order to find relevant entries, a log file monitor will periodically check server log entries against known profiles, to find anomalies and trigger alerts.

NORMALIZATION

Multiple events from heterogeneous sources – i.e., probe scan alerts, FTP server log entries, alerts from NIDS, alerts from HIDS – are initially normalized to provide them a common format (e.g., according to IDMEF); then, they are stored into a database.

PRIORITIZATION AND FALSE ALERT REDUCTION

The number of alerts from the database is reduced by filtering events by using meta-data (e.g., information about attacks, services, vulnerabilities, hardware, OS dependencies of alert types, network assets, IP to hostname mappings, active services, etc.) which help estimating

the severity of the network activity [42; 57; 49; 63]. Models based on human autoimmune system recognize and filter out ‘safe’ signals in such anomaly detection problem [58]. Alerts can also be reduced by using a classification algorithm which returns a confidence value for true and false alerts [59; 57].

At the end of this process, only alerts which pose a threat to the monitored system will be considered; discarded alerts will be used to enhance signatures (or training data) for the preceding detection engines.

AGGREGATION

As irrelevant events are filtered out, we could still have a large number of alerts to process; so, it can be convenient to cluster together similar alerts. Various techniques based on similarity of data feature are widely used for alert aggregation [52; 18; 49; 41] [53]; some consider the complete feature set, but similarity can also be expressed regardless of timestamps, or in terms of path length between feature values in the relative generalization hierarchy [61] [62]. Machine learning techniques such as nearest-neighbor are also suitable for the purpose of alert clustering [54].

A so-called *spacial fusion* [50] – spacial information relates to threat scores of various features characterizing the same event – aggregates alerts by using a two-stage algorithm which considers a security index based on the criticality of the application.

Since unauthorized access generally occurs in the order of either seconds or minutes after the initial attack attempt, temporal-based techniques (e.g., SVO [50] and chronicles [60]) can be applied to fuse alerts belonging to a defined time pattern. However, it can be necessary to separately evaluate the performance of the two approaches, considering the possibility that attackers might introduce hours or days of delay between attack steps.

This process results in a reduced number of meta-alerts; they will enclose intrusion alerts which somehow share features or have certain temporal relationships.

CORRELATION OF META-ALERTS

In order to find the causality among meta-alerts, a few alternatives have been proposed in literature. Likewise step 2, security operators can create a matrix which encodes the expectation of feature similarity between meta-alerts – i.e., the prior expectation that a feature matches if two meta-alerts are causally linked [41].

Meta-alerts can also be modeled in time series variables (i.e., alerts per time unit) and correlation is found through a statistical hypothesis test known as Granger Causality Test

(GCT) [53]. However, as previously mentioned, a temporal relationship among alerts is necessary for the technique to be applied.

Causal relationship between meta-alerts can be based on pre-requisites and consequences of cyber-attacks. In other words, as we are aware of some pre and post-conditions of given attack steps, security events can be correlated by matching the consequence of an old alert with the prerequisite of a new alert. For example, a pre-condition for a R2L attack to an FTP server is that FTP is an available service on TCP port 21.

AGGREGATION AND CORRELATION OF INTRUSION ALERTS

Pre-conditions of attack B and post-conditions of attack A can be known and specified in advance by using a state description language, which is a type of predicate logic. Comparing pre- and post-conditions allow for merging attack specifications, thus allowing delineating the evolution of an attack within a host and estimating the next steps to be performed by the malicious user.

Correlation and aggregation between alerts can also be based on same techniques described for step 2, such as heuristics, data mining and machine learning techniques – i.e., decision trees, radial base function network (RBFN), multilayer perceptrons (MLP [10; 51]), support vector machine (SVM [10]) – and statistical approaches such as Bayesian Network (BN [45; 6]). Other approaches to correlation are based on the knowledge of the given attack scenario [64; 51].

REPORTING THE EVENT

At the end of the previous process – i.e., meta-alert correlation or aggregation/correlation of intrusion alerts – we have been able to reduce the number of alerts as well as find some causality among events. However, potentially a few events might not be detected, thus making difficult the reconstruction of the security incident.

INTRUSION STRATEGY ANALYSIS

Similarity of source and target in meta-alerts can be quantitatively measured in order to reconstruct different stages in a multi-step attack [41]. Algorithms to extract attack strategies from sequences of correlated alerts have been largely proposed.

However, missing events (i.e., false negative alert) may make reasoning about multi-step attacks difficult; so, some correlation methods aim at hypothesizing missing events [52] [65; 66]. State description languages (e.g., LAMBDA [44]) allow the abduction of intermediate alerts from scenarios, even though they have not been detected.

Missing alerts can also be inferred by manually analyzing attack scenario graphs, which are constructed by utilizing various techniques such as taking into account system state transition [67; 50], attack goals [65], or satisfying conditions for the attack execution. Trees can be further transformed into Bayesian networks [65; 10; 6] and inference can be applied on it. Eventually, event report may be ranked based on the known relevance of the attack [66; 67]: for instance, we could consider of storing the probability of having following attacks (PFA) in the attack graph.

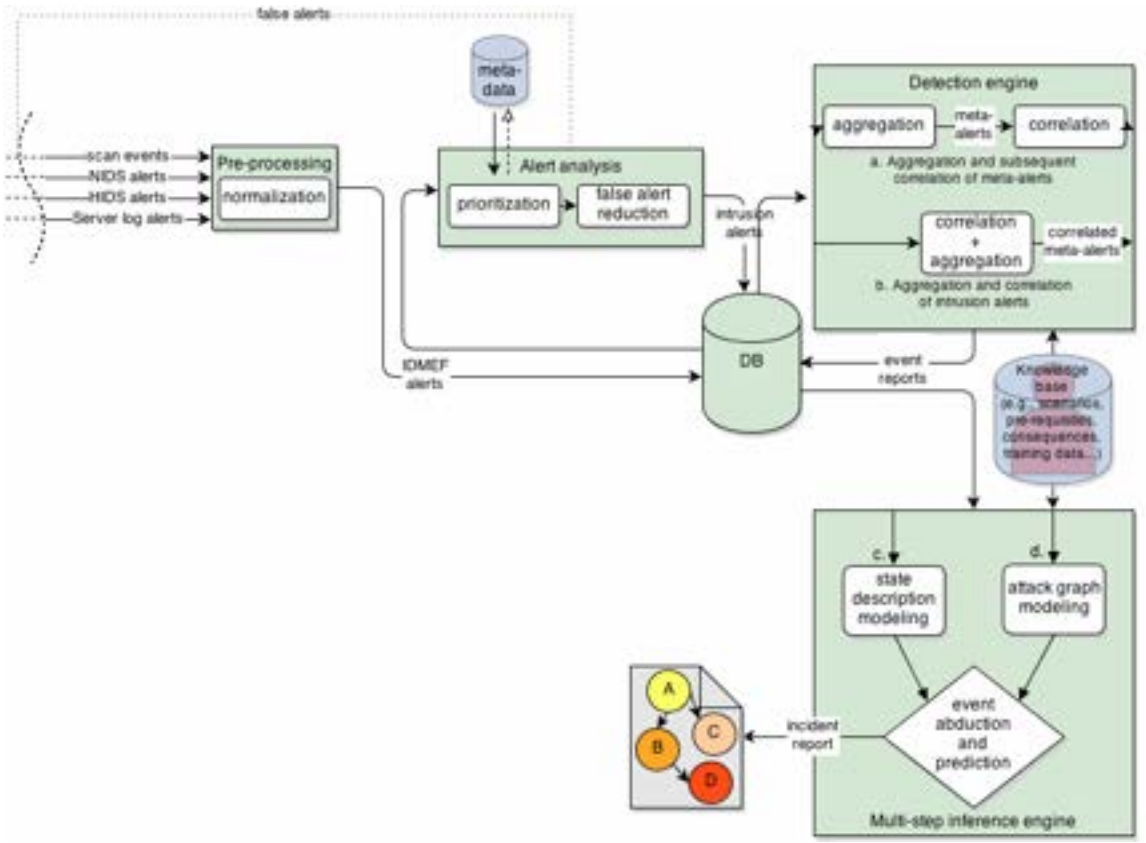

FIG. 16 N-TIER DETECTION CHAIN (N=7)

The obtained detection system will be constituted by (at least) 7 tiers, e.g., n = 8 if log analysis is performed using both anomaly and signature detection. It also includes a 3-tier chain for scan detection, 1-tier for NIDS alerting, 1-tier for HIDS alerting, 1- or 2-tier for server log alerting, and finally 1-tier for correlation of alerts from multiple sources. The letters *a* and *b* indicate two options for generating event reports: respectively, aggregation into meta-alerts and causal inference, and causal-driven aggregation of intrusion alerts. The letters *c* and *d* indicate two alternatives for preprocessing event reports before abducting missing attack steps and reasoning about a multi-step incident: respectively, specifying event reports through a system state description language, and through the creation of attack graphs or trees.

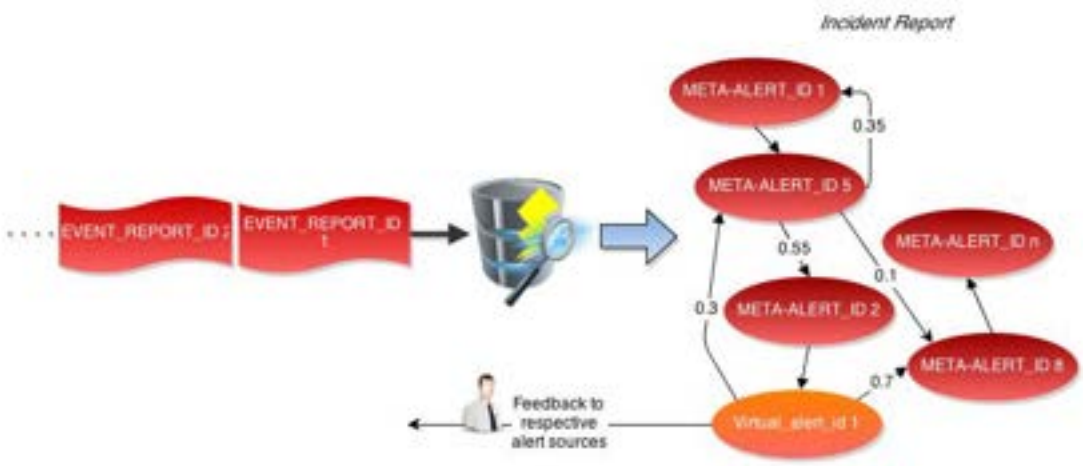

FIG. 17 EXAMPLE OF MULTI-STEP CORRELATION

The strategy analysis process results in incident reports, which can be represented graphically by attack trees. The colors in the picture are only suggestive of visualization techniques to deliver correlated events for expert analysis and incident response. Moreover, standardized information formats such as Incident Object Description and Exchange Format (IODEF) [68] allow incident reports to be stored and managed.

5.3.1.4. STEP 4

In step 4, the attacker attempts to download a large database file from the FTP server; thus our knowledge base can be extended by considering that events of step 1 to 3 are precursors for step 4. In order to detect and report the event at hand, we can take into account the FTP server log as well as alerts from NIDS and HIDS, which are placed similarly as in step 3. NIDS and HIDS will trigger alerts based on malicious TCP connections starting from and directed to the FTP server.

A server log monitoring tool is designated to check log entries and raise alerts when anomalies are encountered or entries match signatures of well-known malicious behavior. However, since log analysis may be much slower than the time employed by an IDS to trigger alerts, it is advisable to include logs as source for detecting the multi-stage incident system rather than reporting the single event. Essentially, for the purpose of incident detection we propose to use the analogous system which has been depicted in the previous section. Nonetheless, at the time of step 4, security operators may already have enough knowledge about the occurring security incident and take action for response.

5.3.2. SCENARIO 2: DATA EXFILTRATION

In this section, the goal is to detect an attack scenario which is made up of three main steps. The attacker's intrusion strategy, in other words, is to perform the following three actions to send data to external site, after he/she has extended his foothold on the internal network:

1. The attacker utilizes a remote access service (e.g., RDP) toolkit to create a network tunnel between the database in the internal network and the Web server in the DMZ.
2. The attacker utilizes both the Web server in the DMZ and the internal SMTP[29] server as a stepping stone to transfer files between the critical system and arbitrary systems on the Internet.
3. The attacker erases a number of server access logs.

5.3.2.1. STEP 1

At a given time, the attacker has compromised a vulnerable Web service in DMZ and then has got a foothold on an inside database management system (e.g., a database server such as Oracle MySQL, Microsoft SQL Server) which contains highly confidential data. We are not aware of any set of techniques or tool which has been exploited to get a foothold in the back-end network segment though.

We could hypothesize that after a few attempts – blocked by the perimeter firewall – of sending organization's data to an arbitrary Internet location, the attacker creates a network tunnel to exfiltrate data. This is an IP communication channel between the server located in DMZ, which will be used as a stepping stone (see step 2), and the internal network. Essentially, the firewall log can be examined to check dropped connections from the internal network segment to the Internet, and this evidence constitutes an attack *precursor*.

In order to build a tunnel, the attacker exploits a remote access service through a toolkit. Thus, anomalous traffic may be identified on the remote access ports – such as TCP 3389, RDP or VNC – of internal servers, directed to the server in DMZ; besides, files of such toolkit may be discovered on the systems in the compromised network enclaves. Tools, if identified, are most likely to contain the port numbers used for tunneling as well as external destinations used by the attacker; Web server logs might be examined for further connections to such IP addresses. The internal firewall, which is placed at the edge of the internal network to separate the latter from the rest of the network, will log traffic connections which originates in the internal network and points to DMZ.

[29] Simple Mail Transfer Protocol is a standard for transmission of e-mail over Internet Protocol (IP).

Hence, alerts for the detection of step 1 are produced from the five following sources:

- Perimeter firewall log (e.g., dropped TCP connections from DB server to any Internet location);
- NIDS monitoring the internal network (e.g., anomalous traffic on remote access ports of internal servers);
- NIDS monitoring the DMZ (e.g., anomalous traffic on remote access ports of the Web server);
- Web server log (e.g., external IP addresses matching the IP addresses of toolkit files);
- DB server log (e.g., anomalous traffic on remote access ports)
- Internal firewall log (e.g., TCP traffic from internal server to DMZ systems).

NORMALIZATION

Since the events logged from each source have various format, it is required a normalization process in order to provide them a standardized format. Subsequently, alerts are stored in a database.

PRIORITIZATION AND FALSE ALERT REDUCTION

Given that we will have a large number of low-level alerts, before grouping them together, it may be convenient to filter potentially false alerts and prioritize the most relevant ones. In other words, by utilizing meta-data such as historical information about incidents, active and vulnerable services, topological information and so forth, it is possible to prioritize alerts in accordance with their severity. In order to remove false alerts, we can apply a classification algorithm [57] to alerts' confidence values; if confidence score is not classifiable, experts can utilize their feedback for adaptive classification. Algorithms based on human immune system aim at preparing a set of mature dendritic cells which will be capable of classifying normal and anomalous patterns.

The process result in confirmed intrusion alerts, which are ranked depending on their severity. Discarded alerts can be analyzed by experts, used to select new features and tune the systems (e.g., log file monitor, NIDS, etc.) which contribute to the detection.

AGGREGATION

Then, resulting intrusion alerts are stored in a database and processed for reducing the large number and give a more condensed view of the malicious activity. Many data fusion techniques are suitable for this purpose such as time based and feature similarity based.

Given that the attack can be executed in days, it is not convenient to group events with respect to their time patterns; instead, similar feature values are able to specify the similarity between alerts from heterogeneous sources.

Hence, clusters (i.e., meta-alerts) can be produced by considering similarity of source IP address, features selected by experts, average values of the feature set, identity of feature values regardless of timestamps. Values of the feature set can also be used to calculate the distance between alerts and train a nearest-neighbor clustering algorithm which classifies intrusion alerts.

In order to cluster alerts, experts can also examine the number of different values for each alert feature and build taxonomies or hierarchical structures (i.e., from general to specific values) for each feature, and then computing feature similarity as the path length between values in the corresponding taxonomy.

CORRELATION OF META-ALERTS

At this point produced meta-alerts can be correlated in several ways. A static matrix can be created from the prior probability distribution with expectation of feature similarity; feature values match when two meta-alerts are causally linked together.

Known prerequisites and consequences of attacks can also be employed to model the causal relationship between meta-alerts by matching consequence of a meta-alert with the prerequisite of another one, taking into account their temporal characteristics.

CORRELATION AND AGGREGATION OF INTRUSION ALERTS

Comparing prerequisites and consequences may also be favorable to fuse attack specifications, thus explaining how an attack evolves within a host. Incoming intrusion alerts are checked against the ones stored in the database; if they show a certain degree of similarity of feature values, expert rules are utilized to trigger a single meta-alert from a tuple of linked alerts. Relative importance of network assets can also be used to find indirect causality among alerts through prerequisites and consequences of attacks.

Data mining and machine learning models, such as decision trees, RBFN, MLP, SVM, Bayesian networks, are advisable to correlate and merge intrusion alerts by learning, although it may be expensive to manually label alert data for training.

EVENT REPORTING

Fig. 18 illustrates the set-up of the n-tier detection chain for step 1; by considering all the detection engines which contribute to detect the event at hand, n will be equal or greater than

7. Such intelligent system will be capable to report a reduced number of meta-alerts which show a certain minimum degree of causality among them. Since the attacker has executed only one step, any further technique of multi-step correlation is not taken into account at this stage.

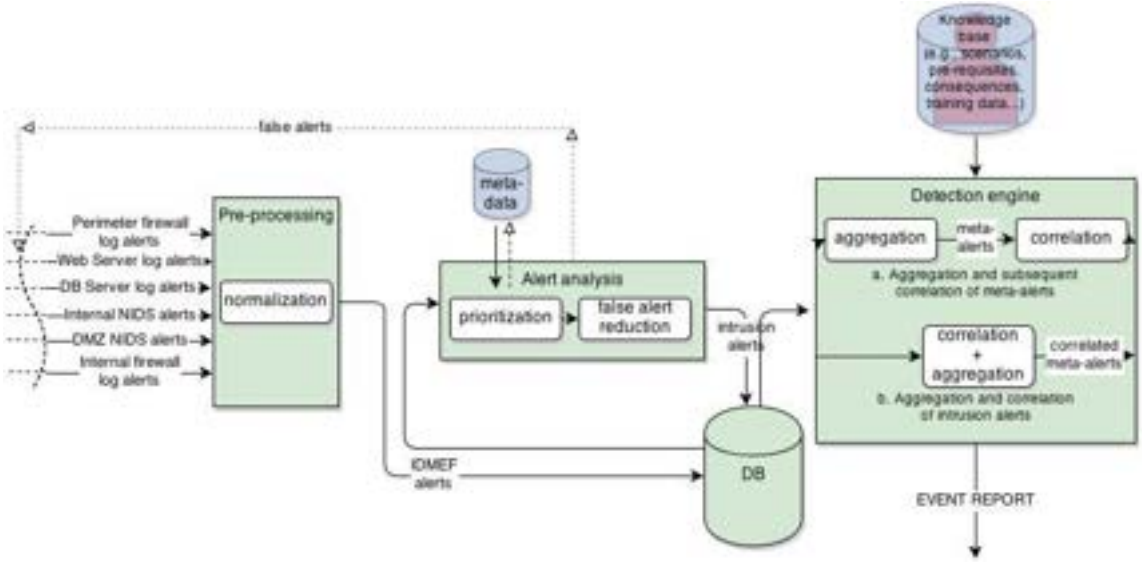

FIG. 18 STEP 1 N-TIER DETECTION CHAIN (N>=7)

5.3.2.2. STEP 2

As the attacker has set a network tunnel, he/she will utilize it to transfer some critical files from the Database server in the internal network segment to the Web server in DMZ. Other sensitive data is transferred from the DB server to a SMTP server in the same network enclave. And eventually, the attacker uses these stepping stones for sending data to an arbitrary external site.

Hence, the sources of indicators for the detection of the event at step 2 are:

- Internal NIDS (e.g., unusual TCP traffic from DB to SMTP server);
- DMZ NIDS (e.g., unusual TCP traffic from DB to Web server);
- HIDS on SMTP server (e.g., unusual TCP traffic from DB Server);
- HIDS on Web server (e.g., unusual TCP traffic from DB Server);
- HIDS on DB Server (e.g., unusual origins of queries);
- Web server log (e.g., anomalous connections to the external sites);
- DB server log (e.g., anomalous connections to the Web Server and SMTP server);
- SMTP server log (e.g., anomalies in mail sending);

- Perimeter firewall log (e.g., unusual TCP traffic from Web server and Mail server to external sites);
- Internal firewall log (e.g., unusual SMTP connections).

NORMALIZATION

Let us assume to employ the alerts generated by the previous eight sources of data. Normalization task will provide for a common format (e.g., IDMEF), so that alerts can be stored in a database and further processed, regardless of their origin.

PRIORITIZATION AND FALSE ALERT REDUCTION

Given that alerts are produced only for specific malicious connections, we can assume that alerts have the same priority (with regards to the impact over the targeted corporate network). The degree of confidence can increase and, hence the amount of false alerts can be reduced by combining anomaly and signature detection. For example, two types of episode rules can be mined: from suspicious behavior and from normal behavior; highly mismatching episodes are labeled as anomalous, and become input of an attack signature database, which is used to update signature for early detection engines [56]. Therefore, this technique allows lowering false positive rate and discovering unknown attacks.

Dendritic cells can be trained by utilizing a human immune system-based algorithm; cells are able to classify normal and suspect behavior. Mature cells associate their input patterns with anomalies, while semi-mature cells consider input patterns as normal [58].

Cost-sensitive algorithm based on RIPPER rule learner allows adaptive classification (i.e., adaptation towards previously unknown events by including experts' feedback in the form of labeled data) of alerts' confidence values for true and false alerts [57]. An evolutionary soft-computing technique based on adaptive neuro-fuzzy classifiers can also be used for the same purpose [59].

Nonetheless, confidence fusion is desirable for reducing false alerts when heterogeneous sensors report the same occurrence with different degree of confidence. In order to model conflict between sources, weighted Dempster-Shafer theory also allows weighting confidence values of alerts from heterogeneous data sources and obtaining a degree of belief from subjective probabilities [69].

Finally, conflicts in alert confidence from heterogeneous inputs can be solved by combining prior confidence of an attack type, collected from historical data, with the confidence for each system attribute and security state-based evidence, which can be considered as meta-

data provided by monitoring tools, vulnerability scanners, human observations and so on [42].

AGGREGATION

Intrusion alerts, which are obtained from the aforementioned process, should be aggregated in order to provide a concise view of the event, for which we have been able to get indicators throughout the network. A simple approach to fuse alerts is a time-based clustering, which nevertheless could fail offering a better representation – in terms of data reduction – if actions (e.g., data downloading) are performed with considerable time delay within this attack step.

An alternative to aggregation is by looking at similarity of features (see section 6.3 for further details). Moreover, distance between alerts (i.e., values of the feature set) can also be employed to train a nearest-neighbor algorithm, which group intrusion alerts into a predetermined number of clusters – i.e., security experts can decide how much meta-alerts should be created – regardless of the 'size' of each cluster.

EVENT REPORTING

The event report will be the output of the system at hand; it will provide a highly consolidate view of the occurring event either by correlating meta-alerts, or by correlating and aggregating intrusion alerts.

CORRELATION OF META-ALERTS

Causality among meta-alerts, which have been stored in a database, can be discovered by using prerequisites and consequences of attacks; that is to say by matching consequence of a meta-alert with the prerequisite of another meta-alert.

Security experts can also consider of encoding, in a static matrix, the prior expectation (i.e., probability distribution) that a feature value matches if two meta-alerts are correlated. As experts may have prior knowledge of meta-alerts affecting each other across different periods in time, meta-alerts can be modeled in time series variables on which a statistical hypothesis test (i.e., Granger Causality Test) is performed. However, as meta-alerts are required to show some temporal relationship, this technique can fail to find causality if random time delay is present among meta-alerts.

CORRELATION AND AGGREGATION OF INTRUSION ALERTS

The evolution of an attack within a host can be explained by merging attack specifications – i.e., prerequisites, post-conditions, events which allow detection, conditions proving that the

attack has succeeded, events which complete the attack; therefore, by comparing feature and specification values of two intrusion alerts, they can be considered to be causally related and hence merged into a single meta-alert.

Security experts can also describe the attack scenario in advance through its known consequences; such rules describe the correlation relationships between events. Related events can be considered to belong to the same attack pattern and can be processed together [64].

Machine learning methods based on decision trees, artificial neural network (RBFN, MLP), support vector machines, and Bayesian networks have been previously deployed and tested for clustering and inferring causality among alerts.

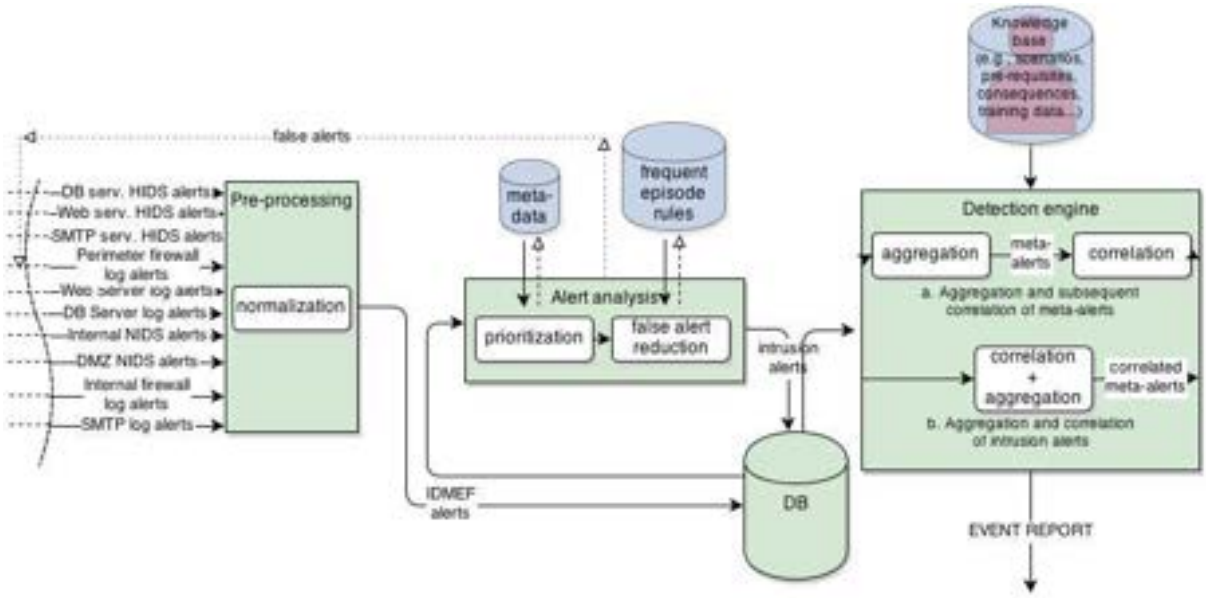

FIG. 19 N-TIER DETECTION CHAIN (N>=11)

5.3.2.3. STEP 3

Web, SMTP, and database logs are valuable sources of information with respect to activities performed by malicious users. In this final step, the attacker erases a number of server log entries. Such counter forensics technique aims at hindering incident detection and forensic analysis by removing entire log files or only critical entries related to a malicious set of activities. To be thorough, it is possible to prevent this type of attack by simply creating a logging server at another site or real-time exporting to a log management system.

Let us consider that Database, SMTP, and Web servers maintain activity logs within the host and a log management system is intended to monitor and analyze log files for incident detection based on well-known signatures. Given that the intruder has deleted some relevant log entries, we can presume that the monitoring tool will produce alerts.

NORMALIZATION

Encoding variants and different standards – supported by different log file manager vendors – requires a preprocessing phase in order to provide a common format for the warnings, which the log manager has produced, and verify the information integrity (e.g., valid timestamp, etc.).

PRIORITIZATION AND FALSE ALERT REDUCTION

The input data for the detection of the step is represented by normalized alerts generated by a log manager, which is assumed working by signatures. Alerts can be prioritized by taking into account meta-data such as security policies, which help assessing the possible impact of the warning events.

Since signature-based detection has been used to trigger alerts, we could assume that alerts are true positive alarms to a great extent, and any mechanism for detecting (and reducing) false alerts is not advisable due to time constraints; as a matter of fact, it is much more convenient to detect this incident indicator as quickly as possible in order to activate response immediately.

AGGREGATION

Temporal-based clustering – i.e., alerts grouped in the same temporal window belong to the same meta-alerts – might not work if every attempt of deleting (or modifying) entries in the server log files is performed with arbitrary lag. Similarly, nearest-neighbor clustering is not suitable, due to the fact that it requires determining the number of meta-alerts to produce beforehand.

Clustering techniques based on similarity of features are suitable in this specific case; meta-alerts may be created by looking at features selected manually by experts, average value of the whole feature set (and eventually rejecting temporal attributes from the feature set), path length for corresponding feature's generalization hierarchy (or taxonomy).

CORRELATION OF META-ALERTS

If one of the previous techniques for aggregation eventually create several meta-alerts, it is necessary to find a causal relationship among them. For this purpose, the prior (i.e.,

expected) probability that a feature value matches if two meta-alerts are correlated can be employed by security operators to construct a static matrix (so called “alert correlation matrix”), which is used to find correlated meta-alerts manually.

Using pre-requisites and consequences to correlate meta-alerts may not be convenient, as the latter ones – caused by deletion of log files (or entries) – may have multiple consequence rules, thus making rule building a costly process for security experts.

CORRELATION AND AGGREGATION OF INTRUSION ALERTS

Alternatively, decision trees, artificial neural network (RBFN, MLP), support vector machines, and Bayesian networks have been largely proposed in literature for alert clustering and causal inference. However, they require extensive manual alert labeling for training the above mentioned algorithms.

EVENT REPORTING

Let us assume that we have reduced the number of alerts, and causality among them has been identified by applying the previous methods; we obtained, in other words, a high-level view of the occurring event. Therefore, security operators, who are able to notice that server log entries have been removed, have perceived that an incident occurred as well as gained insight about the compromised network assets; namely, they can consider such event (i.e., detection of anti-forensics techniques) as *indicator* of more advanced attacks.

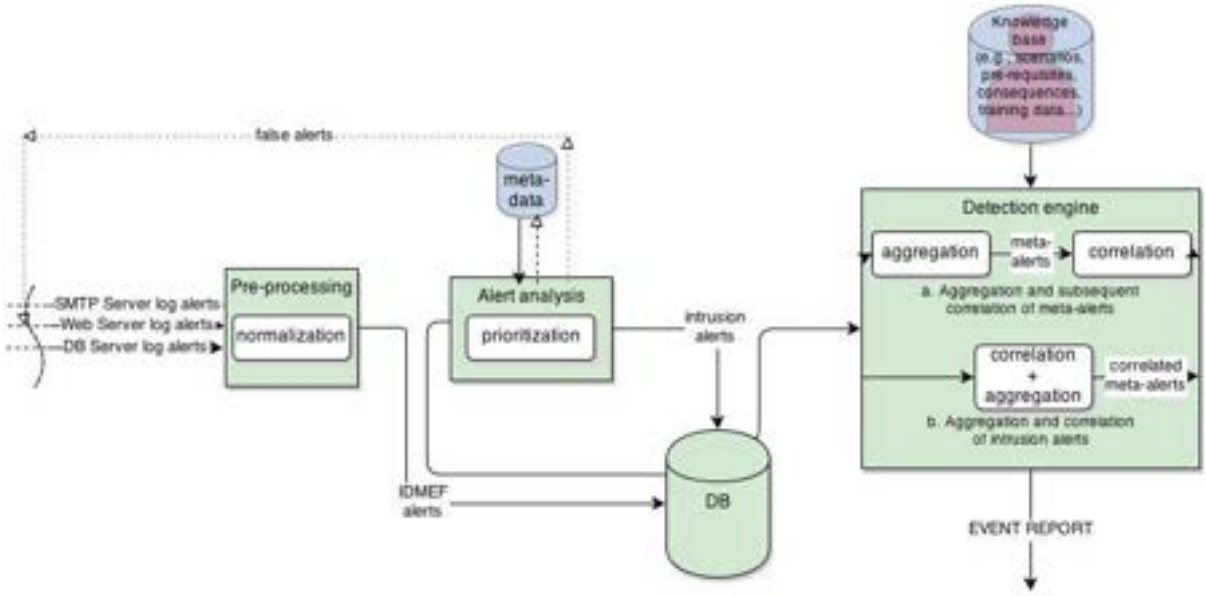

FIG. 20 N-TIER DETECTION CHAIN (N=4)

INTRUSION STRATEGY ANALYSIS

Nevertheless, in order to entirely understand the security incident, the following techniques for analysis of the intrusion strategy can be applied, taking into account the sources of information regarding detection of attack step 1 and 2.

Event reports from each of the previous three steps, which are constituted by correlated meta-alerts, depict the occurrence of diverse stages in a multi-step cyber-incident. Therefore, we could consider feeding the three types of event report (i.e., correlated meta-alerts from each detection step) into a multi-step inference engine which will be responsible for the incident reconstruction.

Known attack scenarios can be specified in a state description language such as LAMBDA; when an attack step is the known consequence of a (expected) set of events which have not triggered alerts, missing events are hypothesized. Given that event reports can be inaccurate or incomplete, thus making scenario reconstruction difficult, abduction of intermediate event plays a key role in the reconstruction process.

In order to find missing events, it is also possible to create correlation graphs for each event report, and manually analyze graphs in order to find missing virtual events which satisfy causal relationships.

As regards reasoning by graphical means, scenario trees can be built in various ways. Then scenario trees can be transformed into Bayesian networks, and either correlation probability is attached to every edge or probabilities of goals (and sub-goals) are derived from the state of parent nodes – satisfied sub-goals and observed events are used to apply inference to the Bayesian net.

Eventually, a data mining algorithm takes a set of candidate attack patterns and a minimum value of support, and returns a set of attack sequences and their probability of having following attacks (PFAs) [67].

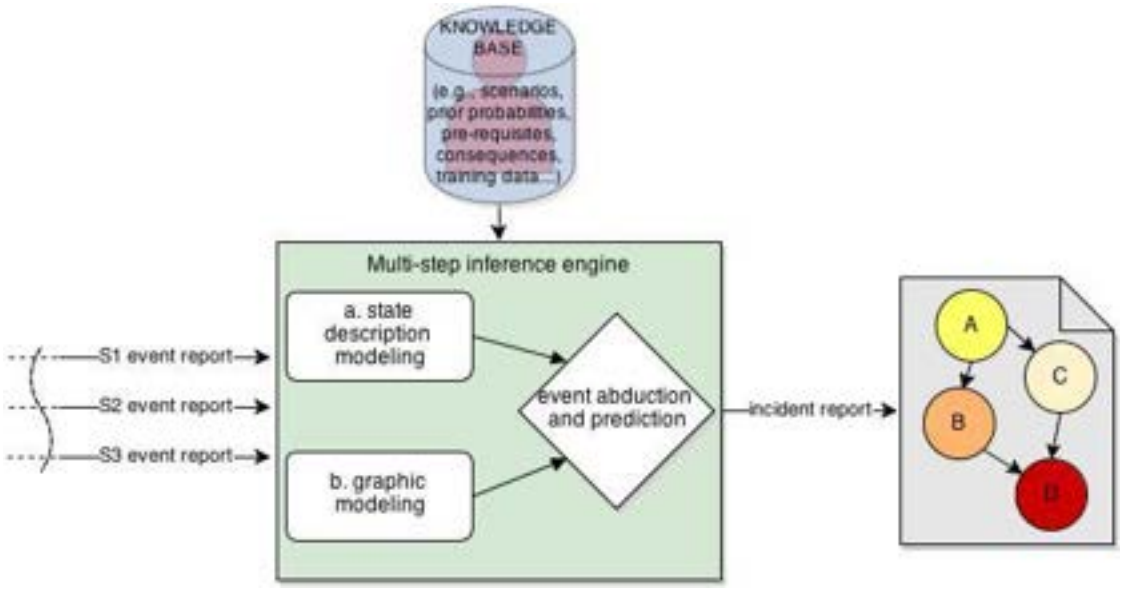

FIG. 21 STEP 2 N-TIER DETECTION CHAIN (N>=22)

5.3.3. SCENARIO 3: SOCIAL ENGINEERING, ZERO-DAY EXPLOIT, BACKDOOR, C&C EXPLOITATION

In this scenario, the goal is to detect a security incident scenario which is characterized by the following three steps:

1. The attacker sends a socially engineered e-mail containing a zero-day exploit to the CFO.
2. The CFO launches the attachment; the exploit automatically connects to a Dynamic DNS domain (e.g., DynDNS) and downloads a malware which disables the antivirus scanner (and its automatic updates) on the CFO's laptop;
3. A Trojan horse is downloaded on the laptop, giving the attacker complete control over it and connects to a command and control (C&C) server on the Internet, whose domain name was recently registered. The malware checks for net commands at random time intervals.

5.3.3.1. STEP 1

The malicious user employs the technique of pretexting for targeting the corporate officer, by using an e-mail attachment as attack vector. In other words, the attacker has invented a plausible scenario to trick the target into open the mail attachment. This event is very hard to detect since it essentially exploits a 'human vulnerability', circumventing most detection mechanisms; as a matter of fact, antivirus and antispyware software are usually able to detect malware as it arrives (or when it is executed) but keep the host vulnerable to zero-day exploit. Then, if we aim at detecting zero-day malware within e-mail messages, we consider looking only at received e-mail logs in order to detect suspicious events.

Let us assume that there are spelling errors in the mail address, so that it does not exactly match with the name of the sender. In addition, the attacker is using a free webmail. Therefore, a monitoring tool for inbound e-mail, which will employ adequate signatures (e.g., blacklisting messages originating from free mail accounts and/or containing attached documents) for detection of malicious activity, can signal suspicious messages through alerts. Such alerts are very valuable, as they can be considered as *precursors* of an incident; although they may exhibit a high false alarm rate, thus making indispensable identifying following events which are potentially related to email alerts.

5.3.3.2. STEP 2

Subsequently, the CFO executes the attached document, which appears normally, letting him unaware of the infection; the exploit automatically connects to a Dynamic DNS (DDNS)

domain (e.g., DynDNS) and downloads a malware which disables the antivirus scanner on the CFO's laptop, as well as automatic updates. Since the exploit connects to a control server on the Internet which is intended to change public IP address frequently, the DDNS provider enables the attacker's domain, which is configured into the malware, to update real-time and automatically point to changing IP addresses. DDNS services are largely utilized for malware distribution even though they cannot be considered malicious per se.

In substance, besides the antivirus will not produce any alert, network sensors can show connections to diverse IP addresses, thus making it ineffective correlation based on similarity of IP source address. Information to be examined for detecting the malicious action at hand may be represented by the following:

- anomalous system performance (e.g., CPU utilization);
- byte-distribution payload anomalies in inbound network traffic;
- anomalous network traffic (i.e., through IDS and firewall logs);
- changes in configuration settings (e.g., Windows registry);
- suspicious active system processes (e.g., vendor signatures, Windows process name-alike).

The previous items translate into the employment of the following sources of information:

- Process and registry monitoring tools;
- NIDS over the internal network segment;
- Firewall log file;

By initially profiling the system (i.e., CFO's laptop in this case), the first type of tools intends to produce alerts real-time when misuse is detected. Suspicious facts are, for instance, differences in CPU utilization, timing of API calls, active system processes, and so forth. A registry monitoring tool is able to create a baseline and real-time compare the state of the Registry at a given time to the baseline. Thus, it is able to produce alerts whenever a critical registry key is added or changed; security experts can set up the criticality of registry keys and locations for real-time monitoring.

Firewall log monitoring tool can produce alerts based on signatures such as outbound connections to IP addresses that point to dynamic DNS domains. NIDS, such as Snort, monitors inbound and outbound network traffic of the internal network segment, and trigger alerts based on a rule set. In addition, NIDS can examine incoming request packets sent to given services (or ports) and raise alerts if any deviation from a normal profile is identified. Essentially, since every application has its own protocol and then its own type payload, n-

gram payload analysis can be utilized. In other words, byte value distribution is computed for each port, service and direction of payload flow, and the centroid model is identified. The occurrence of each n-gram is counted by passing a sliding window with width n over the payload. The case with n=1 calculates the average frequency of each of the 256 possible byte values.

NORMALIZATION

In step 2, we propose the use of at least three heterogeneous sources of alerts, which require pre-processing in order to be further analyzed; it consists in the use of a common alerting format, integrity checking and possible correction of invalid event attributes.

PRIORITIZATION AND FALSE ALERT REDUCTION

Alerts can be weighted towards the severity of specific ports which are known to be often used by malware [70]; for this purpose knowledge of the monitored network and security policies are encoded in meta-data.

The degree of alert confidence can be used for decreasing the number of false alerts in several ways, for example:

- Cost-sensitive rule learner outputs confidence values for true and false alerts;
- Adaptive neuro-fuzzy inference systems (ANFIS) can be trained to detect different attack types and return fuzzy values of class membership, which are consequently aggregated by using a fuzzy inference engine (FIS) with parameters adjusted by a genetic algorithm (GA);
- Algorithm based on Dempster-Shafer theory (DST) weights confidence values of heterogeneous alerts and computes a degree of belief from subjective probabilities;
- Prior confidence values of historical attack types can be combined with meta-data, such as system attribute and security state-based evidence, in order to solve confidence conflicts.

AGGREGATION

Since malicious activity is most likely to exhibit evidence of anomalies at different location in a short time interval, temporal-based clustering (i.e., based on timestamp values) allows achieving satisfying event reduction.

Also, similarity of features can be employed to create meta-alerts, taking into account only features selected by experts, average values of the whole feature-set, path length in each alert feature´s *generalization hierarchy*, or even training a k-NN clustering algorithm (k=1).

EVENT REPORTING

The goal of step 2 is to detect the malware infection. At the end of the above mentioned processes we will be able to produce a report showing suspicious activity involving the personal computer of the CFO, even though it would not possible to get a complete picture of the occurring incident if several meta-alerts have been resulted.

CORRELATION OF META-ALERTS

In order to reason about the malicious occurrence, we can then consider using a technique based on prerequisites and consequences of meta-alerts. That is possible if alerts produced in step 1 are defined as prerequisite of meta-alerts in step 2, and oppositely step 2's meta-alerts are defined consequences of alerts in step 1.

In order to identify casual linkages between meta-alerts, security experts can also employ a matrix created by encoding the expectation of feature similarity (i.e., probability that a feature matches if two meta-alerts a correlated).

CORRELATION AND AGGREGATION OF INTRUSION ALERTS

The malware infection can be specified in terms of pre/post conditions, events which allow detection, conditions proving the attack succeeded, events which allow completing the attack. When a new alert is generated, it is compared with previous instances in the database, and alert pairs – showing a determined degree of feature and specification similarity – are merged into single meta-alerts.

Decision trees, radial basis frequency networks, multilayer perceptron, support vector machines, and Bayesian networks have been largely proposed to identify causality among alerts and group them together with regard to the root cause.

5.3.3.3. STEP 3

A Trojan horse is downloaded and activated on the laptop, giving the attacker complete control over it. Once the backdoor Trojan is on the system, it allows the attacker to control the victim machine. In order to activate the Trojan, one or more files must be copied to the targeted system; however, the antivirus will not be able to detect any harmful files, since it has been previously deactivated (see step 2).

A network port on the victim system is required to be opened for giving control to the *botmaster*; thus tools such as *netstat*[30] can be utilized for real-time monitoring of a list of ports, and then alerting when port status changes to "listening". At random time intervals, the victim machine connects to a Command and Control (C&C) server on the Internet, looking for net commands.

Most trojans add or modify keys in the Registry, for instance, to be able to run whenever the system reboots or the user logs in; user temporary directories and system folders can also reveal the presence of malicious files. Generally, unusual performance and behavior are observable on the victim machine due to the trojan infection; CPU usage, running processes on the host and network traffic can be monitored to identify anomalies (i.e., outliers).

In order to detect the download of the backdoor Trojan and the exploitation of the command and control channel, we can use the same techniques presented in step 2, by further adding a port status monitoring tool to the previous data sources. Essentially, the attack vector has changed (i.e., from *e-mail attachment* in step 1-2 to *drive-by download*) the processes of normalization, prioritization and false alert reduction, aggregation, and correlation are utilized in the same manner as in the previous step, given that the trojan is not directly detectable by the antivirus software, which has been disabled earlier.

In addition, it is convenient to avoid using time-based correlation given that this attack step can be executed with a random time delay, as well as excluding approaches based on the similarity of IP source address, due to the fact that the attacker is using a dynamic DNS service (i.e., fixed domain name pointing to changing IP address).

INCIDENT REPORTING

INTRUSION STRATEGY ANALYSIS

Likewise multi-stage attack detection approaches, in which predefined scenario patterns abduct several state transition sequences, a backdoor infection (and exploitation of C&C channel) can be detected through its communication sequences. When a sequence of correlated meta-alerts abundantly matches a predetermined infection dialog model, an incident report will enclose the events which contribute to the infection.

Correlation graphs for each generated event report (i.e., correlated meta-alerts) also come to the aid of security analysts in order to find missing (virtual) events which satisfy existing causal relationships, and provide a comprehensive incident report. Scenario graphs can be

[30] Command-line tool that shows network statistics concerning connections, routing tables, interfaces, and protocols.

drawn with system state (or goals) as nodes, and events which allow transitions (or malicious events) as edges. Subsequently, correlation probabilities (or probabilities of goals derived from parent nodes' state) are appended to trees, thus obtaining Bayesian networks (see section 6.6 for further details).

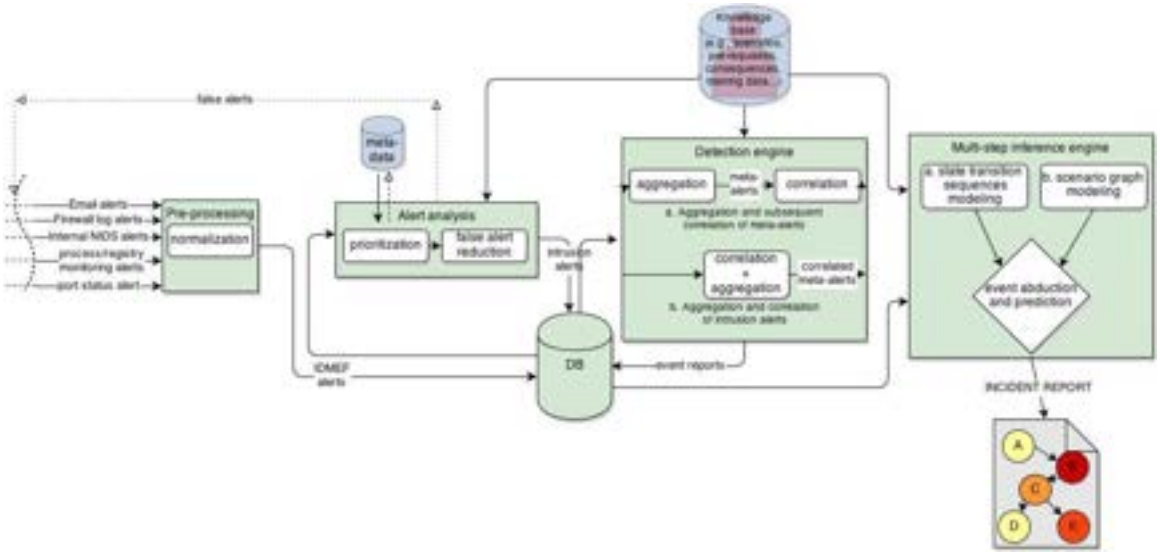

FIG. 22 SCENARIO 3 N-TIER DETECTION CHAIN (N>=6)

5.4. SUMMARY

In this chapter, after having discussed the "traditional" detection system architecture, we have examined various multi-stage attacks, and suggested options not only for identifying suspicious activities in the monitored information system, but also for finding relations of cause and effect between events generated from heterogeneous sources of information. Through a complex interaction between modules, the solution is capable to filter out irrelevant events, reduce the number of instances to be analyzed, and reconstruct – and to some extent predict – advanced security incidents occurring in multiple stages. The following table summarizes the modules and their own objectives, possible inputs, and the location in the typical intrusion detection architecture.

COMPONENT	INPUT	GOAL	DETECTION CHAIN LOCATION
Normalization	Alerts from multiple sources	Giving a standard format to inputs	Pre-processing
Aggregation	TCP connections	Extracting features and clustering	Pre-processing
	Alerts	Clustering alerts	Detection engine
Correlation	Alerts	Finding causality and cluster alerts	Detection engine
	Meta-alerts	Finding causality among meta-alerts	Detection engine
False alert reduction	Alerts	Filtering out false positives alerts	Alert analysis
	Meta-alerts	Filtering out false positives meta-alerts	Alert analysis
Prioritization	TCP connections	Ranking connections according to their severity	Pre-processing
	Alerts	Ranking alerts according to their severity	Alert analysis
	Meta-alerts	Ranking meta-alerts according to their severity	Alert analysis
Intrusion strategy analysis	Event reports	Providing highest-level view by reasoning about intruder's intentions	Multi-step inference engine

TABLE 2 SUMMARY OF CORRELATION MODULES

The proposed solution, which is summarized in the previous table, is also visually represented in Fig. 23.

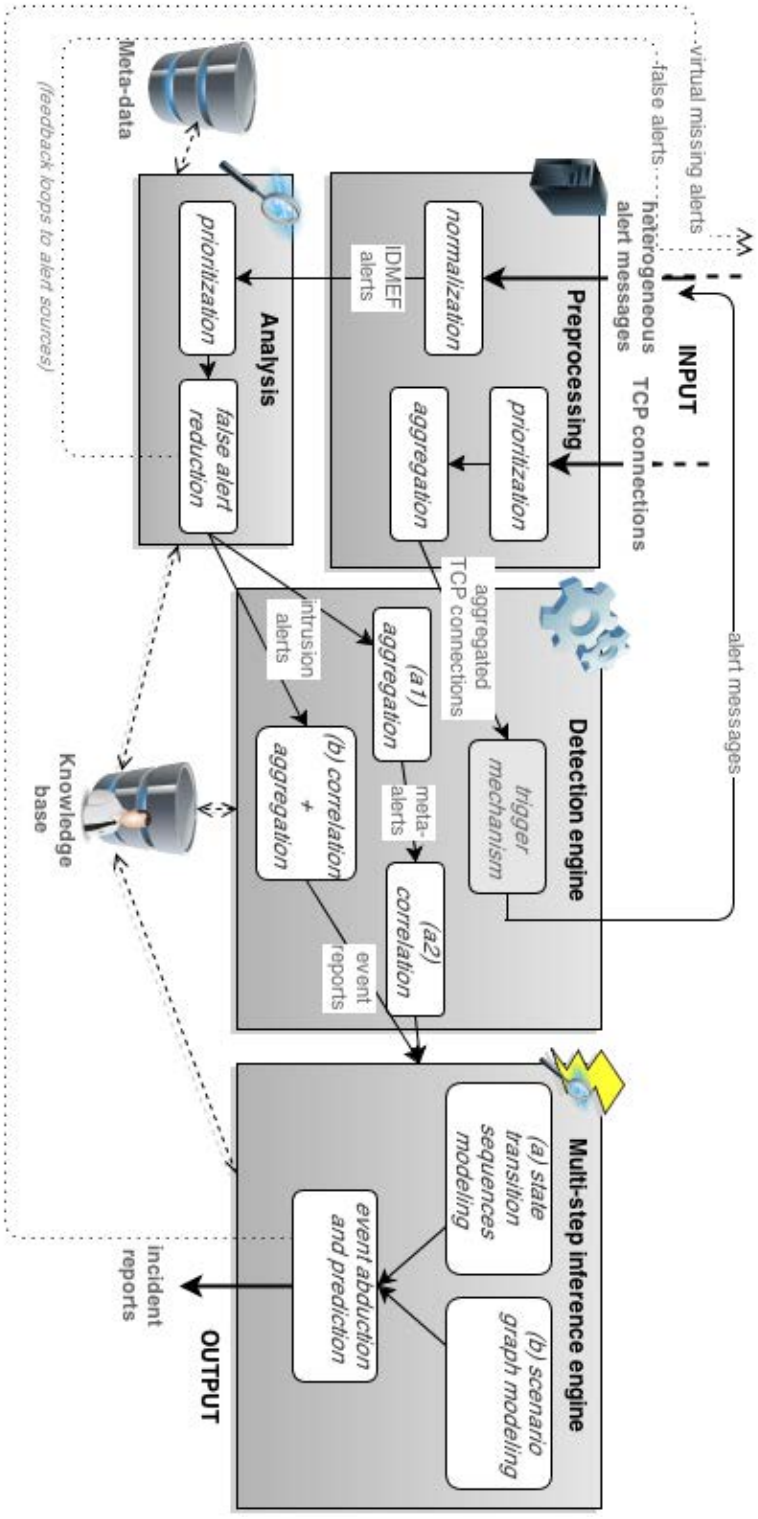

FIG. 23 EVENT CORRELATION MODEL

6. TOWARDS THE DEVELOPMENT OF TAILORED SOLUTIONS

In the previous chapter, it is shown how detection of advanced security incidents takes place by employing correlation techniques along different axis. In this chapter, after identifying the advantages of a design approach based on scenarios, we provide key elements to consider when developing a customized solution based on correlation of events.

General mathematical concepts behind the methods, strengths and limitations, and technical considerations disclosed by the corresponding authors, are reported in more detail in Appendices A to G. In order to offer advice for the real-life implementation of each technique, the structure of this chapter follows the previous subdivision in correlation modules. Deep understanding of the approaches in literature eventually allows an enhanced categorization of the correlation techniques in question.

6.1. SCENARIO-BASED APPROACH

In chapter 5, the design of solutions for our scenarios is shown. The use of such approach based on scenarios tackle several technical issues, which are summarized as follows:

- Scenario-based approaches allow integration into design science research methodology [20]; in this context, scenarios may demonstrate the utility of the designed solution.

- By simulating real situations that users want to avoid, it is possible to elicit information about actions to take for the design. In addition, by providing a context, our approach based on scenarios allows to evoke information about the system requirements, which otherwise would be difficult to achieve. This is due to the reactive nature of cyber-security which makes detection of unknown threats complex.

- Scenarios help understanding the goals of each actor and their different perspectives (i.e., intentions of attackers as well as steps followed by analysts for incident detection).

- Scenarios provide a way to work out what the system should do and how it looks like. This also makes it suitable to improve communication with end-users and other stakeholders.

- Scenarios provide pragmatism – as each offers a stable perspective of attacks and a specific solution – as well as flexibility to handle dynamic situations, thus enabling to make simple changes to the detail level.

- Scenarios help developing new skills and learning to ‘do’; design attempts provide a way to learn and reflect on tradeoffs and design consequences, which might not be fully understood at earlier stages.

- Scenarios-based design allows general design rules to be identified and provides abstraction of the solution; the resulting event correlation model is depicted in Fig. 23 Event correlation modelof Section 5.4.

The validity of the chosen scenarios is demonstrated by the cyber-attack trends and their impacts, which are discussed in Section 2.2. Moreover, recent studies towards the development of intelligent systems for detecting Advanced Persistent Threats (APTs) highlight the importance of considering advanced threats executed in separate steps, which are shown in Fig. 24 APT analysis framework .

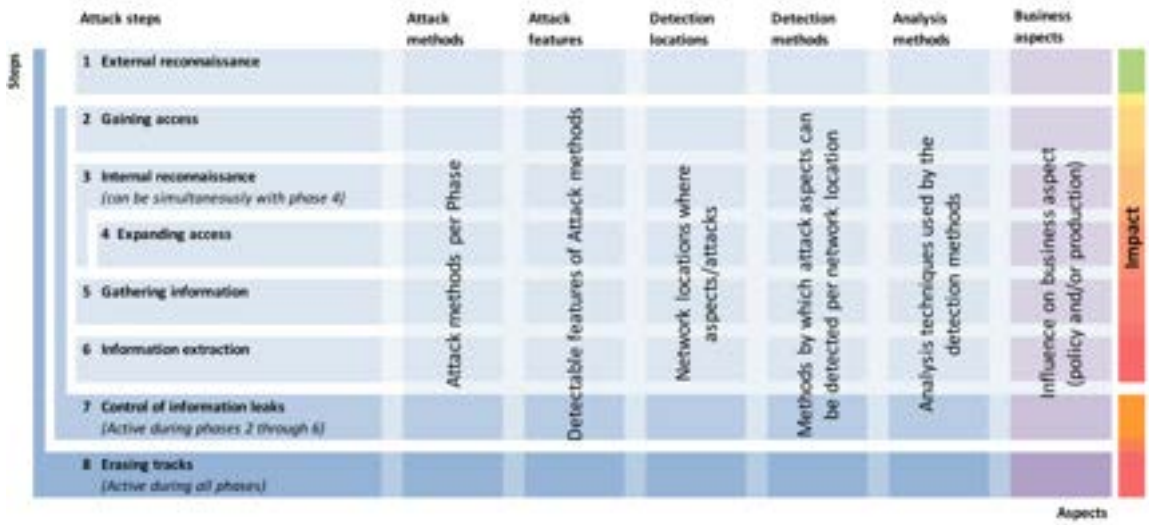

FIG. 24 APT ANALYSIS FRAMEWORK [33]

As different attack steps have different specific goals and it is executed with different methods, it is relevant to analyze individual steps and design components for detecting each of them. The modular approach of event correlation allows the design of tailored solutions. The next section will describe the components of the general event correlation model.

6.2. EVENT CORRELATION MODEL

In the previous section, we stated that scenario-based design provides the abstraction of our solution, which is shown at the end of the previous chapter in Fig. 23 Event correlation model. This section provides an insight into the design alternatives of a tailored event correlation solution.

6.2.1. DATA NORMALIZATION

The module for normalization is essential to correlate events which are generated by heterogeneous sources. It provides the minimum data contents for correlating security-related events, as well as providing a unified data model, regardless of the information source. Hence, syntactic analysis of security events may be convenient; Intrusion Detection Message Exchange Format (IDMEF) [35] is suggested to standardize the data model, and report events across a security information and event management system. In addition, alert messages may represent either individual or multiple events, thus allowing the representation of clusters or correlated alerts. In Appendix A, further technical details concerning the IDMEF model are presented.

6.2.2. DATA PRIORITIZATION

Prioritization of relevant security data is achieved by correlating events with meta-data such as services, hardware, vulnerabilities, and so on. The component for data prioritization can be generally located in two different locations of the detection chain, in an exclusive way or jointly. In other words, depending on what one aims at detecting, it can be more convenient ranking low-level alerts, prioritizing meta-alerts, event reports, or using a mixed approach.

The choice depends on the characteristics of a given cyber incident, as well as time constraints. For example, an attacker could perform some actions that trigger low-impact alerts, which however are – from a defensive perspective – helpful to determine alert causality and reason about the incident. Appendix B provides suggestions for the technical implementation of data prioritization module.

6.2.3. FALSE ALERT REDUCTION

Three main approaches have been identified for reducing the number of false alerts which are triggered by each sensor:

- Hybrid-based: combination of signature-based and anomaly-based intrusion detection reduces the volume of events by filtering out redundancies and increasing confidence about the security endangerment of a system.

- Meta-data-based: event data is combined with meta-data (e.g., vulnerability assessments) to achieve false positive reduction [63; 57; 42]. This is due to the fact that for the same type of attack, the threat can vary in different environments. The strength of this technique is that network and host sensors can perform faster, alert data is cleansed, thus providing quality information; the disadvantage is that this system warns only if attacks are directed towards specific vulnerabilities, ignoring all the other malicious activity.

- Confidence-based: multiple classifiers can be trained with different categories of attacks in order to perform an early classification of the logged security events. The outputs are combined and inference on confidence values of events is used to distinguish between normal or attacking activity, thus reducing the amount of false alerts.

Appendix C provides mathematical considerations, details for technical implementations and considerations based on empirical evidence.

6.2.4. DATA AGGREGATION

The component of data aggregation clusters alerts together into meta-alerts, based on similar characteristics. We can distinguish the following:

- Temporal-based: events which belong to a certain time pattern are grouped together. Temporal patterns can be modeled by known attack patterns, through series of events compelled to time (and context); this technique is capable of reducing the high number of alerts as well as false positives, due to the analysis of contextual events. However, effectiveness is based on the prior knowledge of phenomena which involve many alarm messages.

- Similarity-based: events which share similar features – e.g., with respect to network sessions, host activity, attack type – can be assumed to be correlated and they are clustered together.

- Threat score-based: different features can have a different threat score within one occurring event. Approaches in this category combine data fusion with prioritization towards the criticality of the features.

Mathematical background and information for technical implementation are reported in Appendix D.

6.2.5. CORRELATION OF META-ALERTS

This module performs analysis of clustered security events in order to find causality among them. We are able to distinguish three main categories:

- Temporal-based: statistical analysis can be utilized to discover causality relationships among events which are modeled in time series variables (e.g., based on alerts per time unit). The strength of these approaches is that correlation can be found without prior knowledge of attacks. However, expert knowledge is still needed insofar as further analysis is necessary to reconstruct attack scenarios.

- Probability-based: these techniques are generally able to find occurrence probability of alerts in a meta-alert B, given prior occurrence of alerts of meta-alert A. They generally do not require prior knowledge about predefined attack scenarios, or prior expectation that a feature matches if two alerts are correlated may be estimated by experts. Causal relationships among alerts can be inferred and additionally features which are relevant for the correlation can be discovered.

- Pre-requisite and consequence-based: meta-alerts can be specified as logical predicates by its pre-requisites and consequences, which are known and specified in advance. If there is a connection between the consequences of an attack A and the pre-requisites of attack B, one can correlate an occurrence of A with an occurrence of B; if there is a link between the consequences of A and the consequences of B, one can assume they have the same objective, thus it is possible to correlate them. Given that these techniques provide a further specification of the events, it is potentially able to decrease the impact of false positives and negatives. However, they will not adequately correlate two meta-alerts if the upstream detection engines are not able to detect critical attacks, or there is no defined relationship between them.

It is advised to consult Appendix E for mathematical considerations, examples of usage and suggestions for the technical implementation.

6.2.6. AGGREGATION AND CORRELATION OF RAW DATA

This module performs analysis of security events in order to aggregate them and discover causality relationships. We are able to distinguish three main categories:

- Scenario-based: these techniques rely on the knowledge of specific attack scenarios in order to determine causality between alerts and group them based on predefined relationships, thus reducing the number of suspicious events to be further examined by security analysts. Extensive expert knowledge is required for attack scenario specification, as well as manual alert labeling for training. The weaknesses of these techniques are that the models resulting from it may over-fit the training data; moreover, they are not able to correlate previously unseen scenarios.

- Pre-requisite and consequence-based: techniques based on pre-requisites and consequences of alerts can be employed for correlating and cluster alert messages. Causal relationships among security events are found by employing Neural Networks or Bayesian Networks. The key strength of these techniques is that they do not require an enormous amount of rules in order to find causality between alerts.

Appendix F provides the mathematical background behind this set of techniques as well as suggestions for actual implementation.

6.2.7. MULTI-STEP CORRELATION

Different stages in multi-step cyber incidents can be reconstructed in various ways. Reasoning about sophisticated threats may also require hypothesizing about events which have been discarded or not detected by the deployed security sensors. We can broadly categorize the techniques in:

- Similarity-based: similarity of source and target of the correlated meta-alerts are measured in order to reconstruct different stages in a multi-step attack. However, this approach is considered naïve and inadequate for attack strategy analysis and intrusion prediction.

- State Description-based: Likewise attack scenario patterns lead to sequences of state transitions, security incidents can be represented by state-based communication sequences. Attack description languages such as LAMBDA [44], which are based on a logical representation of the state of a given computer system, may also allow

deducing intermediate alerts from scenarios, even though they have not been detected, thus extracting correlation rules online and building up whole – and still unknown – attack scenarios. Essentially, if an arbitrary alert is the known consequence of an event that should have raised another alert too – which however is not received by the correlation engine – the missing event is hypothesized [52].

- Graph-based: there are several ways to construct attack scenario trees. Graphs which contain alerts with similar features may be further analyzed by security experts, and virtual missing alerts may be deduced in order to satisfy the causal relationships (e.g., graph inconsistencies, unpredicted attribute variations). However, that is not effective when none of the attacks in multi-stage incidents is known.

  Prior knowledge about attacks may be stored in causal networks converted from scenario trees, and employed to correlate isolated scenarios; this allows gathering useful information for further investigation, although it requires an updateable repertoire of known attack strategies to apply inference to the Bayesian network. Obtained probabilities can be attached to graphs and rank attack patterns. This allows identifying novel attack scenarios and predicting evidence for probable following attacks. However, probability of some sequences of attacks might not be retrieved or attack graphs may be incomplete.

  Assuming that the size of attack graphs is such as to make analysis feasible in reasonable time, similarity between sequences of events may also be seen as an error tolerant graph/subgraph isomorphism problem.

Appendix G provides examples, mathematical considerations, and recommendations to implement the techniques in question.

6.3. SUMMARY

The solution we designed intends to overcome the problems which have been previously identified in the earlier chapters of this thesis report, by examining all the components of event correlation process, and techniques and methods which are considered most authoritative by the research community. However, the implementation of all the components and the choice of the techniques should be adequately evaluated, in respect to each situation. This chapter provides some guidelines for actually implementing the solution, comprehensively or only partially. Eventually, deeper insight of the approaches offers a

chance to better categorize the correlation techniques in question, and map them to the correlation modules of our model, as shown in Table 3.

CORRELATION COMPONENT	TECHNIQUE	APPENDIX	REFERENCE
Data normalization	Parsing	A	[35]
Data prioritization	Meta-data-based	B	[49]
False data reduction	Hybrid-based	C	[56] [55] [71]
	Meta-data-based		[57] [42] [63]
	Confidence-based		[57] [58] [59] [69] [72]
Data aggregation	Temporal-based	D	[50] [60]
	Similarity-based		[18] [49] [51] [46] [41] [53] [54] [6] [61] [62]
	Threat score-based		[50]
Correlation of aggregated data	Temporal-based	E	[53]
	Probability-based		[41] [6]
	Pre/post conditions-based		[52] [73]
Aggregation/Correlation of raw data	Scenario-based	F	[51] [64]
	Pre/post conditions-based		[10] [6] [45] [42]
Multi-step correlation	Similarity-based	G	[41]
	State-description based		[70] [44]
	Graph-based		[50] [67] [65] [66] [74] [42] [6] [10]

TABLE 3 TAXONOMY OF EVENT CORRELATION TECHNIQUES, SELECTED FROM LITERATURE

7. REFLECTIONS AND LIMITATIONS

In this chapter, some personal considerations and research limitations which arose during the realization of this thesis are introduced. First of all, since there is no ultimate solution for the implementation of our model – e.g., selection of features, techniques, threshold parameters may vary – it may be crucial to perform a number of tests, which allow one to make trade-off decisions with respect to response time of the system, required computing power, detection and correlation accuracies, training time, labeling time, number of tiers of the detection system, data reduction rate, and so forth.

As a matter of fact, performance is not absolute but must be empirically assessed, depending on specific situations. One may think for example that, for detecting certain incidents, prioritization and false alert reduction can be better performed again after correlation. It might also be interesting to cluster alert messages iteratively in case that a satisfying data reduction is not achieved.

Previously, in section 3.5, design requirements have been conceptualized. The model must be capable of identify significant suspicious events; in particular, prioritization, false alert reduction, and correlation modules are responsible to select – among the 'big data' – the events which are meaningful.

In order to detect – in heterogeneous datasets – event patterns referring to the same incident, first the normalization module deals with providing data attribute homogeneity and similar data format to each event message; then, aggregation and correlation components identify events which refer to the same event, while the multi-step inference engine (or intrusion strategy analysis module) considers those events in order to detect patterns related to a same incident, and aims to provide complete pictures of occurring incidents.

As regards large volume of data and redundancy, the analysis component (prioritization and false alert reduction modules) work as early data filter, while aggregation and correlation modules – by detecting the intrusion alerts which are triggered by the occurrence of the same malicious event – cut off superfluous data.

In order to supply detailed information to security operators, the normalization module provides a data format (i.e., IDMEF) which is a flexible object-oriented representation of alert messages. In addition, the alert data model offers chances to be extended with further information, which may be required for using multiple complementary correlation techniques as well as to employ the most novel ones; for example, prerequisites and consequences might be embedded in it. The set of correlation operations extend the data

model while keeping it consistent. Each component of event correlation provides messages which vary from ‘simple’ alerts to meta-alerts to event reports, with no semantic differences; hence, given that IDMEF allows specifying relationship among events at different granularity level, security analysts are provided with backwards view of incidents.

Likewise a standard data model for alert messages enable the correlation of heterogeneous data, standard formats for incidents can further improve incident management. The Incident Object Description Exchange Format (IODEF) [68], for example, determines a standard model to represent incidents, which enables interaction between Computer Security Response Teams (CSIRTs) and improve their operational performance. An interesting point is the compatibility between IDMEF and IODEF, which allows including alert messages into Incident Objects. Advantages of using IODEF are also represented by the facts that IODEF objects are human-oriented, given that the main actors are CSIRTs, rather than security sensors.

With respect to the potentially high false alarm ratio (FAR), false alert reduction module implements techniques which are able to decrease the rate of false positives and negatives. However, in order to preserve potential false negatives, which are of utmost importance for reconstructing complex attack scenarios, it might be advantageous to loosen signatures in each downstream detection mechanism while reinforcing the policies of data prioritization and false alert reduction.

As regards the use of further security data for correlation, Trouble Ticketing System (TTS) provide information which may accelerate incident analysis and prioritize relevant data [15]. Moreover, a novel intelligent reputation system, which is recently under development by researchers at Google, has been proposed to integrate a content-agnostic malware protection (CAMP) [75] in the web browser, with the objective of assessing the level of risk of each file in download. As tests offer encouraging results – i.e., it outperforms any local or web-based antivirus with lower false alert rate and higher accuracy – in future one can consider of using further security data directly from web browsers.

We have seen that graph-based techniques are widely used for reasoning about multi-step attacks. Likewise, several visualization tools have been proposed in recent years in order to enhance incident detection; the use of graphical models makes reconstructing incidents a more intuitive and sometimes unconscious process.

Our model aims to support incident response and offers more comprehensive and accurate root cause analysis than ordinary security tools and procedures; nonetheless, when

discarding time constraints which are used for monitoring and analysis, it may also help performing internal corporate or digital forensic investigation, following the occurrence of serious computer incidents.

The research presents some limitations as well. First, although the scenario cases we chose for the design phase are based on real circumstances and up-to-date threats, the synthetic nature of our attack scenarios reduces the complexity of the events to some extent. Despite the scarcity of real and exhaustive datasets which accurately outline consistent multi-threat scenarios is a prohibitive condition, the relevance of detecting such “artificial” incidents is also confirmed by the latest trends shown by recent security reports.

Second, in order to build our model, we consider only a subset – quite wide though – of scientific literature in the field. Most commercial products available in the market do not provide sufficient details for their application within our designed solution. In addition, several times it is not clear what procedures are put in practice by IT security operators when reconstructing multiple threat events. Thus, the sources of business needs for the design of our artifact are mainly represented by scientific literature and industry reports, while efforts in identifying and interviewing different stakeholders may provide further requirements which are missing in this research.

The lack of public ad-hoc datasets for performance evaluation of correlation methods represent a significant deficiency for empirical comparisons of the alternative solutions to be implemented. Moreover, producing a valid dataset in short time is unfeasible and requires a large commitment in terms of time and resources; hence, in the research we limit to present some fundamental performance assessments as they are reported by the relative authors, rejecting any possible criticism about the validity of each testing dataset.

Finally, for various reasons we do not discuss the implementation of the concerned techniques in full. Since most of the techniques require codifying expert knowledge, and massive commitments towards labeling of events are expected in order to achieve effectiveness, the time constraint for this research did not allow us to put any solution in place. Moreover, each study supplies a different degree of knowledge for practical deployment.

8. CONCLUSIONS

In this thesis, we proposed a model for correlating event data with the goal of detecting sophisticated computer security incidents and help incident management. To design the model, we went through different research stages, including reviewing a large collection of scientific papers and industry reports, inquiring of security experts about common practices and business requirements, making up significant attack scenario cases, combining all the collected information in a novel configuration of existing techniques, suggesting alternatives for the implementation. This final chapter summarizes the main contributions of this thesis research and proposes directions for future researches.

8.1. MAIN CONTRIBUTIONS

We recognized the following contributions:

1. The research inquires about the detection of advanced cyber-security incidents by employing multiple sources of security data. In particular, the focus is on attacks which are executed in multiple stages by malicious users and target large organizations. This choice is due to their need of defending high-value assets and the set of complex issues that such organizations have to face compared with smaller organizations.

2. The research investigates on the relation between incidents and events which are observable within today's IT infrastructures of large organizations, thus providing an insight into effective security incident detection based on monitoring and intelligent analysis.

3. The research identifies and analyzes novel taxonomies and alert correlation methods, which are the most influential in the scientific community, showing their strengths and limitations along the design and the implementation planning phases.

4. The research offers an abstraction of an incident detection system, when multiple data sources and detection engines are employed.

5. The research decomposes the problem into components, which address the initial challenges and represent the phases of a comprehensive correlation process, and maps them to the above-mentioned generic detection system architecture.

6. The research offers alternatives and suggestions for the implementation of effective techniques and procedures to detect sophisticated threats.

7. The research allows a classification of novel and influential event correlation techniques.

This general model represents the link between problem and solution, and requires adequate evaluation for its utilization. However, it can simply be adapted to the specific needs and IT infrastructures of various companies. Moreover, small variations allow employing this model for different purposes such as monitoring of industrial processes or physical security incidents. Finally, the model can be seen as an important advance in the development of an intelligent security protection which improves incident response and support decision making processes in the management of today's organizations.

8.2. SUGGESTIONS FOR FUTURE RESEARCH

The limitations and reflections of the previous chapter offer some opportunities for further research, which are discussed in this section.

It is widely accepted that management of large organizations should invest in new projects and adopt novel technologies to defend their IT assets against advanced cyber-attack techniques. However, given that highly-sophisticated attacks such as APTs or digital espionage activities have emerged only in the last few years, their impacts are not clear yet; hence, investigation towards long-run impacts on large firm's market value, changes in reputation of victims of cyber-espionage, and their loss of competitive advantage can offer important indicators for supporting future investments in cyber-security.

By implementing the proposed techniques and setting up a test environment, one would be able to better define the operational requirements of this model. Another opportunity is offered by the correlation of common security data with new sources of data and meta-data, such as reputation systems, web browser information, ticketing systems as well as how to structure meta-data effectively. Moreover, adaptation of the model towards collaborative incident detection, which involves multiple incident response teams, requires further investigation.

The production of datasets including real and heterogeneous data of multi-threat scenarios, and a valid evaluation dataset for correlation systems is essential to provide proactive and cutting-edge security solutions. As the model is implemented, comparisons with commercial product may be valuable to create industry benchmarks. Also, efforts in identifying and

interviewing different stakeholders may provide business requirements and challenges which might be missing in this research.

The graphical means proposed for investigating multi-stage threats at a high-level of abstraction push research towards advanced visualization techniques and practices, which are widely considered very helpful. For example, one may consider of integrating incident reports with colors and other constructs to illustrate the likelihood of certain events within a multi-step attack scenario. Hence, they will provide a more intuitive interface which facilitates the incident analysis and speeds up incident response.

Finally, understanding incidents by looking at events from multiple perspectives suggests updating current security policies, and offers the opportunity to research models for a strategic deployment of security sensors and other data sources.

BIBLIOGRAPHY

1. *Information Technology and Productivity: A Review of the Literature.* **Brynjolfsson, Erik and Yang, Shinkyu.** 1996, Advances in Computers 43, pp. 179-214.

2. *Addressing Dynamic Issues in Information Security Management.* **Abbas, Haider, et al.** 2011, Information Management & Computer Security, Vol.19, pp. 5-24.

3. **World Economic Forum.** *Global Risks 2012.* Cologny : Risk Response Network, 2012.

4. **PwC.** Cybercrime: Protecting against the growing threat. *Events & Trends Vol. 256.* March 2012, pp. 6-18.

5. *Security mistakes in information system deployment projects.* **Sommestad, Teodor, et al.** 2011, Information Management & Computer Security, Vol. 19 No.2, pp. 80-94.

6. *An online adaptive approach to alert correlation.* **Ren, Hanli, Stakhanova, Natalia and Ghorbani, Ali A.** Bonn : Springer-Verlag, 2010. Proceedings of the 7th international conference on Detection of intrusions and malware, and vulnerability assessment. pp. 153-172.

7. *Security information management as an outsourced service.* **Debar, Hervé and Viinikka, Jouni.** 2006, Information Management & Computer Security, Vol. 14 No.5, pp. 417-435.

8. **SecurityWeek.** Worldwide IT Security Spending to Top $60 Billion in 2012, Says Gartner. *securityweek.com.* [Online] September 13, 2012. http://www.securityweek.com/worldwide-it-security-spending-top-60-billion-2012-says-gartner.

9. **IDC.** Security Products: Market Analysis. *IDC #231292, Volume: 1* . Framingham, MA, USA : s.n., November 2011.

10. *Alert Correlation for Extracting Attack Strategies.* **Zhu, Bin and Ghorbani, Ali A.** 2006, International Journal of Network Security, Vol.3, No.3, pp. 244–258.

11. **GTISC and GTRI.** *Emerging cyber threats report 2012.* Atlanta : Georgia Tech Cyber Security Summit, 2011.

12. *Temporal and Spatial Distributed Event* . **Jiang, Guofei and Cybenko, George.** Boston : s.n., 2004. American Control Conference. pp. 996-1001.

13. *M2D2: a formal data model for IDS alert correlation.* **Morin, Benjamin, et al.** Berlin : Springer-Verlag, 2002. RAID'02 Proceedings of the 5th international conference on Recent advances in intrusion detection . pp. 115-137.

14. *A Comprehensive Approach to Intrusion Detection Alert Correlation.* **Valeur, Fredrik, et al.** 2004, IEEE Trans. Dependable Secur. Comput. 1, 3, pp. 146-169.

15. *A model-based survey of alert correlation techniques*. **Salah, Saeed, Maciá-Fernández, Gabriel and Díaz-Verdejo, Jesús E.** 2013, Computer Networks, Volume 57, Issue 5, pp. 1289-1317.

16. *Asynchronous Alert Correlation in Multi-agent Intrusion Detection Systems.* **Gorodetsky, Vladimir, Karsaev, Oleg and Samoil, Vladimir.** St. Petersburg : Springer, 2005. Computer Network Security, Third International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security.

17. **Dua, Sumeet and Du, Xian.** *Data Mining And Machine Learning In Cybersecurity.* s.l. : Auerbach Publications, 2011.

18. **Yu, Dong and Frincke, Deborah.** A Novel Framework for Alert Correlation and Understanding. [book auth.] Markus Jakobsson, Moti Yung and Jianying Zhou. *Applied Cryptography and Network Security.* Yellow Mountain : Springer, 2004, pp. 452-466.

19. **Olsson, Fredrik.** *Intrusion Management.* s.l. : Växjö University, Faculty of Mathematics/Science/Technology, School of Mathematics and Systems Engineering, 2006.

20. **Hevner, Alan R., et al.** Design Science in Information Systems Research. *MIS Quarterly 28, 1.* March 2004, pp. 75-105.

21. **Baumgart, Matthias, et al.** Research Methods in Informatics and its Applications: Design-Oriented Research. *TU Munich.* [Online] April 27, 2007. http://www14.informatik.tu-muenchen.de/personen/baumgart/download/public/presentation_DR.pdf.

22. **Vaishnavi, Vijay and Kuechler, William.** Design Science Research in Information Systems. *Design science research in information systems and technology.* [Online] September 30, 2011. http://desrist.org/desrist.

23. *Applying Design Research Method to IT Performance Management: Forming a New Solution.* **Ardakan, Mohammad Abooyee and Mohajeri, Kaveh.** 2009, Journal of Applied Sciences, 9, pp. 1227-1237.

24. *On the use of different statistical tests for alert correlation: short paper.* **Maggi, Federico and Zanero, Stefano.** Gold Goast, Australia : Springer-Verlag, 2007. Proceedings of the 10th international conference on Recent advances in intrusion detection. pp. 167-177.

25. **Burton, James, et al.** *Cisco Security Professional's Guide to Secure Intrusion Detection Systems.* s.l. : Syngress-Elsevier, 2003.

26. **Cichonski, Paul, et al.** *SP 800-61 Rev. 2. Computer Security Incident Handling Guide.* Gaithersburg, MD : National Institute of Standards and Technology, 2012.

27. **Verizon RISK Team.** *Data Breach Investigation Report.* s.l. : Verizon Enterprise, 2012.

28. **Fox-IT.** *Black Tulip* . Delft : s.n., 2012.

29. **Alperovitch, Dmitri.** *Revealed: Operation Shady RAT.* Santa Clara, CA : McAfee, 2011.

30. **Information Warfare Monitor.** *Tracking GhostNet.* 2009.

31. **Scarfone, Karen A. and Mell, Peter M.** *SP 800-94 Rev.1. Guide to Intrusion Detection and Prevention Systems (IDPS) (draft).* s.l. : National Institute of Standards and Technology, 2012.

32. **Petersen, Chris.** *An Introduction to Network and Host Based Intrusion Detection.* Boulder : LogRhythm, Inc., 2006.

33. *An analysis framework to aid in designing advanced persistent threat detection systems.* **de Vries, Johannes A., et al.** 2012.

34. **Limmer, Tobias and Dressler, Falko.** *Survey of Event Correlation Techniques for Attack Detection in Early Warning Systems.* Erlangen : University of Erlangen, Dept. of Computer Science 7, 2008.

35. **Debar, Hervé, Curry, Dave and Feinstein, Ben.** The Intrusion Detection Message Exchange Format. *http://tools.ietf.org.* [Online] March 2007. http://tools.ietf.org/html/rfc4765.

36. *Intrusion Detection Systems & Multisensor Data Fusion: Creating Cyberspace Situational Awareness.* **Bass, Tim.** 2000, Communication of the ACM, Vol. 43(4) , pp. 99-105.

37. *Intrusion Alert Correlation Technique Analysis for Heterogeneous Log.* **Yusof, Robiah, Selamat, Siti Rahayu and Sahib, Shahrin.** 2008, International Journal of Computer Science and Network Security, VOL.8 No.9, pp. 132-138.

38. *Evaluation of the diagnostic capabilities of commercial intrusion detection systems.* **Debar, Hervé and Morin, Benjamin.** Berlin : Springer-Verlag, 2002. Proceedings of the 5th international conference on Recent advances in intrusion detection. pp. 177-198.

39. *Using unsupervised learning for network alert correlation.* **Smith, Reuben, et al.** Windsor : Springer-Verlag, 2008. Proceedings of the Canadian Society for computational studies of intelligence, 21st conference on Advances in artificial intelligence. pp. 308-319.

40. *A program-based anomaly intrusion detection scheme using multiple detection engines and fuzzy inference.* **Hoang, Xuan Dau, Hu, Jiankun and Bertok, Peter.** London, UK : Academic Press Ltd., 2009, J. Netw. Comput. Appl. 32, 6, pp. 1219-1228.

41. *Probabilistic Alert Correlation.* **Valdes, Alfonso and Skinner, Keith.** Davis, CA, USA : Springer, 2001. In Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection. pp. 54-68.

42. *Integrating IDS alert correlation and OS-Level dependency tracking.* **Zhai, Yan, Ning, Peng and Xu, Jun.** San Diego : Springer-Verlag, 2006. Proceedings of the 4th IEEE international conference on Intelligence and Security Informatics. pp. 272-284.

43. *STATL: an attack language for state-based intrusion detection.* **Eckmann, Steven T., Vigna, Giovanni and Kemmerer, Richard A.** 2002, Journal of Computer Security, 10, 1-2, pp. 71-103.

44. *LAMBDA: A Language to Model a Database for Detection of Attacks.* **Cuppens, Frédéric and Ortalo, Rodolphe.** London : Springer-Verlag, 2000. RAID '00 Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection. pp. 197-216.

45. **Qin, Xinzhou.** *A probabilistic-based framework for INFOSEC alert correlation.* s.l. : Georgia Institute of Technology, 2005.

46. *Managing Alerts in a Multi-Intrusion Detection Environment.* **Cuppens, Frédéric.** Washington, DC : IEEE Computer Society, 2001. Proceedings of the 17th Annual Computer Security Applications Conference. p. 22.

47. *Alert correlation survey: framework and techniques.* **Sadoddin, Reza and Ghorbani, Ali.** Markham, Ontario, Canada : ACM, 2006. Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services. pp. 37:1--37:10.

48. **Wood, Mark and Erlinger, Michael.** *Intrusion Detection Message Exchange Requirements.* Claremont : IETF, 2007.

49. *A Mission-Impact-Based Approach to INFOSEC Alarm Correlation.* **Porras, Phillip A., Fong, Martin W. and Valdes, Alfonso.** Zurich : Springer-Verlag, 2002. Proceedings of the 5th international conference on Recent advances in intrusion detection. pp. 95-114.

50. *Alert Fusion for a Computer Host Based Intrusion Detection System.* **Feng, Chuan, et al.** s.l. : IEEE Computer Society, 2007. Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems. pp. 433-440.

51. *Fusing a Heterogeneous Alert Stream into Scenarios.* **Dain, Oliver and Cunningham, Robert K.** Philadelphia : s.n., 2001. In Proceedings of the 2001 ACM workshop on Data Mining for Security Applications. pp. 1-13.

52. *Alert Correlation in a Cooperative Intrusion Detection Framework.* **Cuppens, Frédéric and Miège, Alexandre.** Berkeley : IEEE Computer Society, 2002. IEEE Symposium on Security and Privacy. pp. 202-215.

53. *Statistical Causality Analysis of INFOSEC Alert Data.* **Qin, Xinzhou and Lee, Wenke.** 2003. Proceedings of The 6th International Symposium on Recent Advances in Intrusion Detection. pp. 73-93.

54. *Alarm clustering for intrusion detection systems in computer networks.* **Perdisci, Roberto and Giacinto, Giorgio Roli, Fabio.** 2006, Eng. Appl. Artif. Intell. vol. 19, 4, pp. 429-438.

55. *A data mining analysis of RTID alarms.* **Manganaris, Stefanos, et al.** 2000, Computer Netw. vol. 34, 4, pp. 571-577.

56. *Hybrid Intrusion Detection with Weighted Signature Generation over Anomalous Internet Episodes.* **Hwang, Kai, et al.** 2007, IEEE Trans. Dependable Secur. Comput. vol. 4, 1, pp. 41-55.

57. *Using Adaptive Alert Classification to Reduce False Positives in Intrusion Detection.* **Pietraszek, Tadeusz.** s.l. : Springer, 2004. roceedings of the Symposium on Recent Advances in Intrusion Detection. pp. 102-124.

58. *Information fusion for anomaly detection with the dendritic cell algorithm.* **Greensmith, Julie, Aickelin, Uwe and Tedesco, Gianni.** 2010, Inf. Fusion vol. 11, 1, pp. 21-34.

59. *A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers.* **Toosi, Adel Nadjaran and Kahani, Mohsen.** 2007, Comput. Commun. vol. 30, 10, pp. 2201-2212.

60. *Correlation of Intrusion Symptoms: an Application of Chronicles.* **Morin, Benjamin and Debar, Hervé.** 2003. Proceedings of the 6th International Conference on Recent Advances in Intrusion Detection. pp. 94-112.

61. *Clustering intrusion detection alarms to support root cause analysis.* **Julisch, Klaus.** 2003, ACM Trans. Inf. Syst. Secur. vol. 6, 4, pp. 443-471.

62. *Mining intrusion detection alarms for actionable knowledge.* **Julisch, Klaus and Dacier, Marc.** Edmonton : ACM, 2002. Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 366-375.

63. **Gula, Ron.** *Correlating IDS Alerts with Vulnerability Information.* Columbia, MD : Tenable Network Security, Inc., 2011.

64. *Aggregation and Correlation of Intrusion-Detection Alerts.* **Debar, Hervé and Wespi, Andreas.** London : Springer-Verlag, 2001. RAID '00 Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection . pp. 85-103.

65. *Attack Plan Recognition and Prediction Using Causal Networks.* **Qin, Xinzhou and Lee, Wenke.** Washington : IEEE Computer Society, 2004. ACSAC '04 Proceedings of the 20th Annual Computer Security Applications . pp. 370-379.

66. *Building Attack Scenarios through Integration of Complementary Alert Correlation Methods.* **Ning, Peng, et al.** San Diego : s.n., 2004. IN PROCEEDINGS OF THE 11TH ANNUAL NETWORK AND DISTRIBUTED SYSTEM SECURITY SYMPOSIUM. pp. 97-111.

67. *Assessing Attack Threat by the Probability of Following Attacks.* **Li, Zhi-tang, et al.** 2007. International Conference on Networking, Architecture, and Storage. pp. 91-100.

68. **Danyliw, Roman and Meijer, Jan.** http://www.ietf.org/rfc/rfc5070.txt. *The Internet Engineering Task Force (IETF).* [Online] December 2007. http://www.ietf.org/.

69. *Alert Confidence Fusion in Intrusion Detection Systems with Extended Dempster-Shafer Theory.* **Yu, Dong and Frincke, Deborah.** Kennesaw, Georgia : ACM, 2005. Proceedings of the 43rd annual Southeast regional conference - Volume 2. pp. 142-147.

70. *BotHunter: detecting malware infection through IDS-driven dialog correlation.* **Gu, Guofei, et al.** Boston, MA : USENIX Association, 2007. Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium.

71. **Javitz, Harold S. and Valdes, Alfonso.** *The NIDES Statistical Component: Description and Justification.* 1993.

72. *Fusion of multiple classifiers for intrusion detection in computer networks.* **Giacinto, Giorgio, Roli, Fabio and Didaci, Luca.** 2003, Pattern Recogn. Lett., pp. 1795-1803.

73. *Constructing Attack Scenarios through Correlation of Intrusion Alerts.* **Ning, Peng, Cui, Yun and Reeves, Douglas S.** Washington, DC : ACM, 2002. Proceedings of the 9th ACM conference on Computer and communications security. pp. 245 - 254 .

74. *Techniques and tools for analyzing intrusion alerts.* **Ning, Ping, et al.** 2004, ACM Transactions on Information and System Security 7, 2, pp. 274-318.

75. *CAMP: Content-Agnostic Malware Protection.* **Rajab, Moheeb Abu, et al.** 2013. Prroceedings of 20th Annual Network & Distributed System Security Symposium.

76. *Fast effective rule induction.* **Cohen, W. W.** Lake Tahoe, California : Morgan Kaufmann, 1995. In Machine Learning: Proceedings of the Twelfth International Conference.

77. **Jang, J. R.** ANFIS: adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics.* 1993, Vol. 23, 3, pp. 665-685.

78. **Sugeno, Michio.** *Industrial applications of fuzzy control.* s.l. : Elsevier Science Ltd., 1985.

79. *An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller.* **Mamdani, E. H. and Assilian, S.** 1, 1975, International Journal of Man-Machine Studies, Vol. 7, pp. 1-13.

80. **Shafer, Glenn.** *A Mathematical Theory of Evidence.* s.l. : Princeton University Press, 1976.

81. *Learning attack strategies from intrusion alerts.* **Ning, Peng and Xu, Dingbang.** Washington D.C. : ACM, 2003. Proceedings of the 10th ACM conference on Computer and communications security. pp. 200-209.

APPENDIX A

DATA NORMALIZATION

The Unified Modeling Language (UML) diagram of the “Alert” class (and its aggregate classes) is depicted in Fig. 25 IDMEF Alert class.

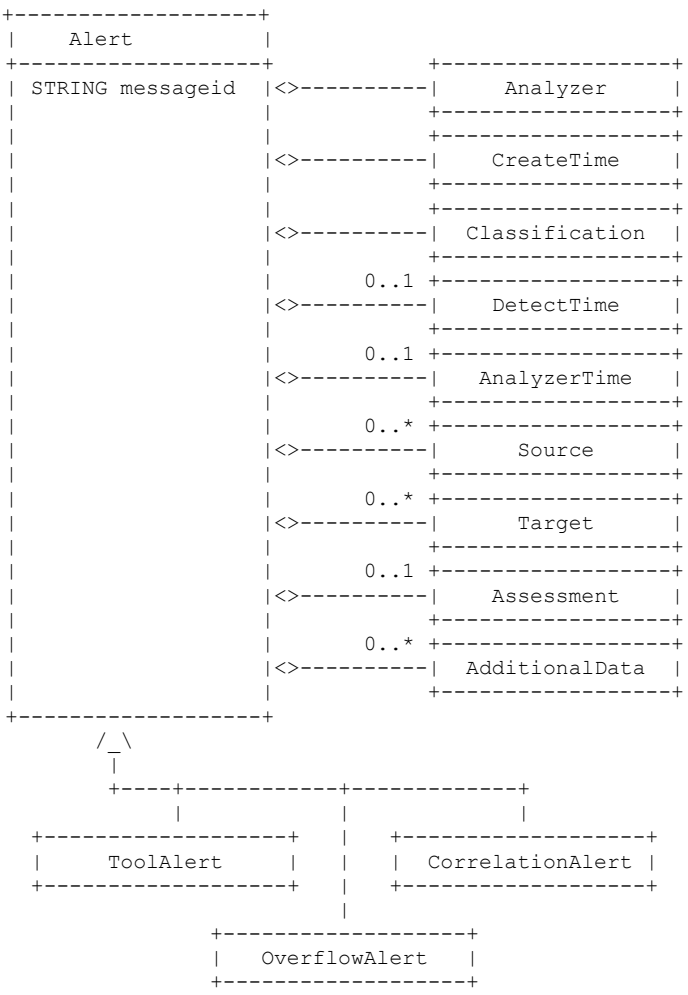

FIG. 25 IDMEF ALERT CLASS

In addition, given the continuous evolution of detection and correlation techniques, AdditionalData class allows for extension of the IDMEF model. For the sake of clarity, we report here an example from the IDMEF memorandum; it represents a port scan (meta-) alert from multiple detection systems (i.e., various “analyzerid”):

```
<?xml version="1.0" encoding="UTF-8"?>
<idmef:IDMEF-Message version="1.0" xmlns:idmef="http://iana.org/idmef">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="bc-corr-01">
```

```
        <idmef:Node category="dns">
          <idmef:name>correlator01.example.com</idmef:name>
        </idmef:Node>
      </idmef:Analyzer>
      <idmef:CreateTime ntpstamp="0xbc72423b.0x00000000">
        2000-03-09T15:31:07Z
      </idmef:CreateTime>
      <idmef:Source ident="a1">
        <idmef:Node ident="a1-1">
          <idmef:Address ident="a1-2" category="ipv4-addr">
            <idmef:address>192.0.2.200</idmef:address>
          </idmef:Address>
        </idmef:Node>
      </idmef:Source>
      <idmef:Target ident="a2">
        <idmef:Node ident="a2-1" category="dns">
          <idmef:name>www.example.com</idmef:name>
          <idmef:Address ident="a2-2" category="ipv4-addr">
            <idmef:address>192.0.2.50</idmef:address>
          </idmef:Address>
        </idmef:Node>
        <idmef:Service ident="a2-3">
          <idmef:portlist>5-25,37,42,43,53,69-119,123-514
          </idmef:portlist>
        </idmef:Service>
      </idmef:Target>
      <idmef:Classification text="Portscan">
        <idmef:Reference origin="vendor-specific">
          <idmef:name>portscan</idmef:name>
          <idmef:url>http://www.vendor.com/portscan</idmef:url>
        </idmef:Reference>
      </idmef:Classification>
      <idmef:CorrelationAlert>
        <idmef:name>multiple ports in short time</idmef:name>
        <idmef:alertident>123456781</idmef:alertident>
        <idmef:alertident>123456782</idmef:alertident>
        <idmef:alertident>123456783</idmef:alertident>
        <idmef:alertident>123456784</idmef:alertident>
        <idmef:alertident>123456785</idmef:alertident>
        <idmef:alertident>123456786</idmef:alertident>
        <idmef:alertident analyzerid="a1b2c3d4">987654321
        </idmef:alertident>
        <idmef:alertident analyzerid="a1b2c3d4">987654322
        </idmef:alertident>
      </idmef:CorrelationAlert>
    </idmef:Alert>
  </idmef:IDMEF-Message>
```

It is noteworthy to highlight that the alerts, which contribute to the global alert in example, are reported as a list of related alert identifiers; the optional attributes “analyzerid“ of “alertident” indicates the diverse sources of security events.

APPENDIX B

DATA PRIORITIZATION

An ‘incident handling fact base’ partly overcomes this issue given that it includes historical information about previous attacks, services, hardware, vulnerabilities, OS dependencies of alert types, and so forth. Another database contains topological information (e.g., network assets, IP to hostname mappings, active services, etc.) which may be retrieved by automatic scheduling *nmap*. Moreover, the network topology requires to be checked for vulnerability dependencies [49].

The incident handling fact base must contain, at least, the following fields:

- Incident identification code;
- Incident description;
- OS types, versions, and hardware type required for the incident to succeed;
- Ports, services, and applications required for a ‘successful’ incident;
- References (e.g., Bugtraq ID and URL).

A relevance score is computed for each alert, based on the likelihood of successful incident, ranging from 0 to 255. In other words, for each alert, the prioritization module should look for known dependencies in the incident handling fact base and check them against configuration of the targeted system. 0 represents no match at all, while values close to 255 show that most of the required dependencies match the topology of the target.

It is also relevant that security analysts define the *mission profile*, in order to perform a mission-impact analysis:

- critical assets and services;
- unreliable users and administrative accounts;
- high-relevance classes of incidents.

The rank is defined as the assessment of the events with respect to:

- mission profile
- likelihood of success

Relevance, priority and rank – i.e., low, medium and high – can be mathematically calculated using belief propagation on a Bayesian network. Probabilities P(criticality|priority) are elements of adaptive Conditional Probability Tables (CPTs) which

encode the mathematical relationship between evidence and intermediate node values of the incident rank tree in Fig. 26, which represent the beliefs of outcome, priority and relevance scores.

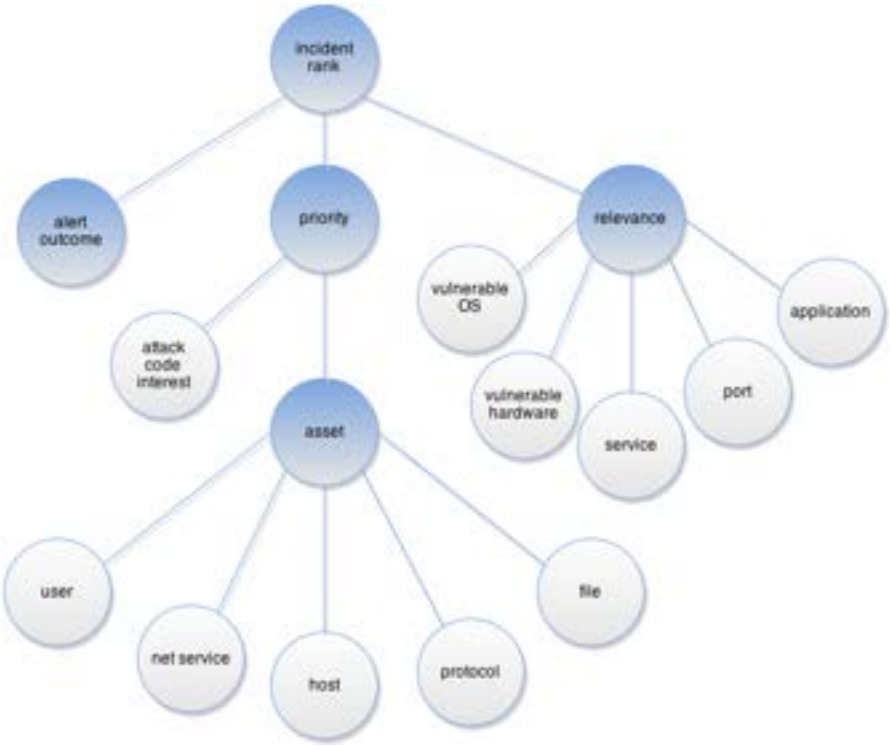

FIG. 26 INCIDENT RANK TREE [49]

APPENDIX C

FALSE ALERT REDUCTION

CORRELATION OF SIGNATURE- AND ANOMALY-BASED DETECTION

An early mixed technique has been proposed to reduce the volume of reported events by filtering out redundancies and increasing confidence about the security endangerment of a system. While the signature-based detection is used to detect a large range of well-known threats, a statistical module aims at detecting anomalies comparing long to short-term behaviors of each user. The following equation represents the degree of anomaly *D* in the short term behaviors [71]:

$$D = (S_1^{\,2} + S_2^{\,2} + \ldots + S_n^{\,2}) / n$$

where S_i ($0<i<n+1$) are the degrees of anomaly for each selected feature, n is the total number of measures. Hence, large D represents anomalous behavior. Historical data related to D can be used to adjust the threshold value for the declaration of false positives.

Uncertain patterns resulting from signature-based detection may be input to an episode mining engine [56] which produces frequent episode rules. The episode mining engine will consists of episode rules mining and pruning. The rules are the compared to episode rules produced with normal traffic; highly mismatching episodes are labeled as anomalous, become inputs of an attack signature database, and are used to update the rule-based detection engines with newly generated signatures.

Alternatively, frequent episode rules are employed to extract normal patterns of events which satisfy the requirements of support in the association rule and confidence, as well as events which may be false positives; alert patterns which are consistent with a frequent episode rule are considered false positives. In order to generate rules, cleansed network traffic is adopted for the learning process. The algorithm for anomaly detection is the following [55]:

```
LOOP
        B = read_next_burst_of_alarms
        IF (∃s ∈ F : s = B) THEN
            check_rules(R), report_frequent(B)
        ELSE
            M = most_specific_supported_itemsets(B, F)
            D = B − ∪_{m_i ∈ M} m_i
            check_rules(R), report_infrequent(B, M, D)
        END
END
```

Let us consider having a set of alerts B, frequent item sets F, and high-confidence rules R. Initially, the alerts are checked against a frequent item set; if they do not match, the most supported item set M is identified. M contains the alerts which occur frequently within the alert burst B, but they are not known to occur simultaneously. D contains the alerts that occur less frequently than the minimum support threshold, which is set according to the frequency of known innocuous alerts.

META-DATA CORRELATION

Correlation of event-based data with meta-data is employed to achieve false positive reduction [63; 57; 42]. This is due to the fact that for the same type of attack, the threat can vary in different environments.

The strength of this technique is that network and host sensors can perform faster, alert data is cleansed, thus providing quality information; the disadvantage is that this system warns only if attacks are directed towards specific vulnerabilities, ignoring all the other malicious activities. Gula distinguishes three approaches to correlate IDS alerts with vulnerabilities [63]:

- Persistent correlation occurs within the triggering mechanism of a detection system. Network vulnerability assessment scans are performed daily and vulnerability checks are maintained in a database. When alerts are related to any of the vulnerabilities, a ‘high quality’ alert is raised. Hence, the model links several alert types with the vulnerability checks and is capable to query the vulnerability database rapidly, so that it can keep pace with the event flow. The system is able to filter out alerts which do not target any known vulnerability.

  Therefore, using heterogeneous detection systems it is possible to detect activities which relate to different vulnerabilities. On the other hand, not every technology can correlate with vulnerability checks directly.

- Near-time approach does not maintain a vulnerability database; the network is inspected actively as security events occur – i.e., OS, application, vulnerability, in sequence – and events that are not correlated to vulnerabilities are ignored.

  However, despite the advantage of not storing and managing a large database with vulnerability information, the lack of it translates into a slower correlation and gives the chance to malicious users to perform many spoofed attacks, while the correlation engine looks for much more detailed vulnerability information.

- Real-time correlation system assumes that IDSs may derive real-time vulnerability information with signatures to perform correlation and automatically discard irrelevant vulnerability checks. However, information about a number of vulnerabilities cannot be gathered passively but requires an active vulnerability assessment. Such a real-time model requires speed, information reliability, and accuracy to provide valuable alerts.

Any case, correlating vulnerabilities provide information with high level of confidence towards the occurring threat. Hence, the technique enables automated and better decisions, such as tuning firewall and IDS rules, for instance.

Moreover, since vulnerability assessments can also produce false positives or negatives, it is advisable to periodically perform a manual check concerning the quality of correlation, so to avoid situation in which actual threats are not identified, or normal traffic is considered malicious.

MULTIPLE CONFIDENCE CORRELATION

Classification algorithms are also employed for reducing the amount of false alerts [59; 57]. Let assume to train the system with true and false labels, a rule learner such as RIPPER [76], which is efficient with noisy datasets, is capable of building rule-sets by discriminating between the two classes of alerts. The system also produces training examples from experts' feedback, which verify the correctness of the classification, and continuously update the classifier in order to improve classification; such incremental learning is executed in batches and can be controlled by a threshold parameter. However, the weakness of this technique is that the volume of training sets increases infinitely.

Generating ordered rule sets – i.e., rules for only one class – has been preferred to unordered – i.e., for both true and false alerts – rules, since the former are smaller and easier to interpret; a rule is made up of a comparison between attribute values and a class label. In order to optimize misclassification cost, training sets are weighted through a cost matrix, which encloses the misclassification costs for each class pair. Therefore, the system is able to attach a confidence value, which is the likelihood of a correct classification.

Neuro-fuzzy classifiers – so called ANFIS, adaptive neuro-fuzzy inference system [77] – can be trained with specific alert classes [59] – e.g., normal, probe, DoS[31], U2R[32] and R2L[33].

[31] Denial of Service attack
[32] Remote user-to-root attack
[33] Remote user-to-local attack

ANFIS may produce fuzzy rules without using expert knowledge. A method of subtractive clustering determines the number of rules and the initial membership functions.

A fuzzy inference module takes the outputs of the neuro-fuzzy classifiers and determines whether those data patterns are normal or malicious. When the pattern is intrusive, the 'normal' classifier determines the attack class.

The neuro-adaptive (supervised) learning is employed to overcome the difficulty, in some situations, to choose the membership functions by only observing the data; the associated parameters can be chosen by looking at the variation of input and output data.

ANFISs implement the following reasoning mechanism. Let us consider a fuzzy inference system with the first order of Sugeno Fuzzy Model [78], with input x, y and output z; a fuzzy rule-set with two fuzzy *if-then* rules is:

if x is A1 and y is B1 then f1 = p1x + q1y + r1;

if x is A2 and y is B2 then f2 = p2x + q2y + r2;

where A, B are input fuzzy sets, z=f(x,y) is a zero- or first-order polynomial function.

The structure of the FIS implements a Mamdani Fuzzy Model [79]; let us consider this model with inputs x, y and output z; a fuzzy rule-set with two fuzzy *if-then* rules is:

if x is A1 and y is B1 then z is C1;

if x is A2 and y is B2 then z is C2;

where A, B are input fuzzy sets, C is the fuzzy output set.

The two models are equal in the way they fuzzyfy the inputs and apply the fuzzy operator; however Sugeno output membership function is either linear or constant.

Genetic operators – i.e., mutation, crossover, selection – are employed to optimize the inference engine by selecting, from the current population, individuals who evolve towards the optimal solution.

Apart from fuzzy inference systems, fusion of multiple classifiers has also been proposed by employing [72]:

- majority voting rule
- average of classifiers' outputs
- Naïve Bayesian rule

- Decision trees rule
- Dynamic Classifier Selection (DCS)

The first two items are fixed rules, which assume that diverse classifiers make uncorrelated errors even though they should present pair-wise output correlation. In Naïve Bayes, a confusion matrix on the training set is employed to weight each classifier. The dynamic classifier selection aims at automatically selecting, for each new pattern, the most accurate classifier.

There is evidence showing that DCS performs better with data presenting pairwise output correlation and different accuracies. Known attacks are effectively classified by DCS and decision trees; the fusion rule of the latter also exhibits a better performance on unknown attacks than methods based on single classifiers.

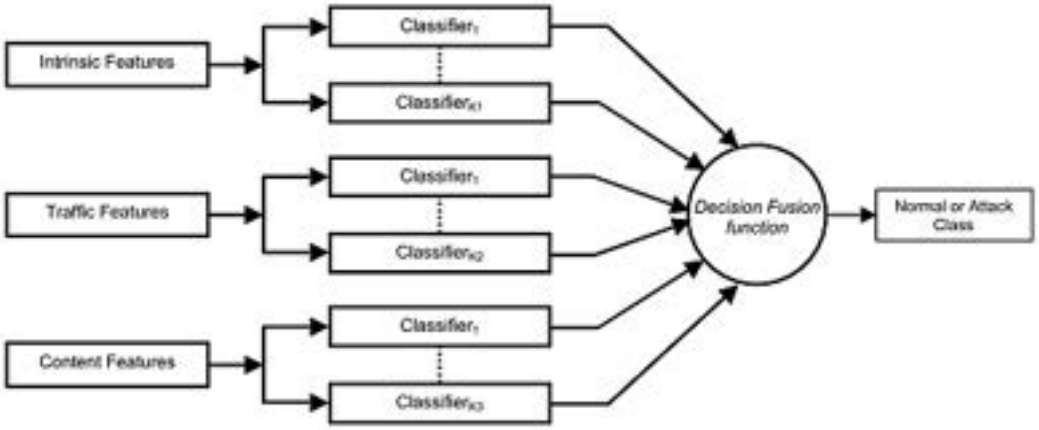

FIG. 27 MULTIPLE CLASSIFIER SYSTEM [72]

The architecture of the Multiple Classifier System is shown in Fig. 27. It is clear from the figure, that three types of feature have been suggested as input for the classification:

- *content features*, i.e., payload information, such as OS errors;
- *intrinsic features*, i.e., general traffic information;
- *traffic features*, i.e., comparison with previous similar occurrences.

Similar to a classification algorithm, Dendritic Cell Algorithm [58] is a data filtering method modeled upon the human autoimmune system. The algorithm has been hitherto employed for the detection of ping-based port scans – i.e., SYN scans – but can be possibly used with other time-dependent datasets.

Dendritic cells are capable of activating adaptive immune responses by correlating external signals, processing antigens and show them on their own surface, to the other immune cells. DCA model's inputs include pathogen associated molecular patterns (PAMP), safe signals, danger signals, and inflammatory cytokines; inputs provide a description of an antigen's context.

First, a population of immature dendritic cells (iDC) is created to get antigens and signals, and convert them into co-stimulatory molecules (csm), semi-mature dendritic cell cytokines (smDC), and mature dendritic cell cytokines (mDC). The first one monitors a value of maturation of DC; essentially, the DC is mature if exceed a specific threshold value. Mature cells are used for anomaly detection, and maturation basically correlates input data and signals to safe or danger contexts. Table 4 DCA input signals shows how the input signals are modeled in intrusion detection.

Signal	Biological Property	Abstract Property	Computational Example
PAMP	Indicator of microbial presence	Signature of likely anomaly	Error messages per second
Danger Signals	Indicator of tissue damage	High levels indicate "potential" anomaly	Netowrk packets per second
Safe Signals	Indicator of healthy tissue	High levels indicate normally functioning system	size of network packets
Inflammation	Indicating general tissue distress	Multiplies all other input signals	User physically absent

TABLE 4 DCA INPUT SIGNALS [58]

Authors provide evidence on the robustness of DCA when PAMP and danger signals are misrepresented; however, safe signal must be chosen accurately to achieve a satisfying performance.

Every mature cell transforms signals to output $o_p(m)$ according to the following equation [58]:

$$o_p(m) = \frac{\sum_t \sum_{j \neq 3} W_{tjp} s_{tj}(m)}{\sum_t \sum_{j \neq 3} |W_{tjp}|} \qquad \forall p$$

where W_{ijp} are transforming weights for a signal type i, category j, in the cell's signal matrix s(m). A parameter controls the mature context antigen value (MCAV) of a given antigen type, which determines its degree of anomaly, according to the following equation:

MCAV(antigen_type) = mature_count/antigen_count

Therefore, MCAV is a value between 0 and 1, representing the probability that an antigen is a danger signal. Sensitivity analysis, performed by the authors of the research, shows that a good performance is achieved at high frequency of signal sampling. However, it is difficult to compare this technique with other machine learning or rule-based techniques for classification, given that in general the latter cannot correlate such signals.

Dempster-Shafer theory [80] allows taking evidence from multiple sources and obtaining degrees of belief from subjective probabilities, thus modeling uncertainty and confidence conflict between different information sources. Therefore, the component for confidence aggregation returns an overall confidence value. Dempster's rule [69] considers various weights of the confidence values, given by the various sources. For example, events originating from a signature-based detection engine are supposed to have higher confidence score than suspicious events reported from anomaly-based sensors.

Let us assume that two observers O_1, O_2 believe hypothesis H is true with different confidence values $m_1(H)$, and $m_2(H)$, the following equation represents the overall confidence on hypothesis H according to a Weighted D-S combining rule [80]:

$$m_{12}(H) = \frac{\sum_{B \cap C = H} \left[m_1(B) \right]^{w_1} \left[m_2(C) \right]^{w_2}}{\sum_{B \cap C \neq \phi} \left[m_1(B) \right]^{w_1} \left[m_2(C) \right]^{w_2}}$$

Where w_i are weights for the values of confidence; they can be computed in many ways such as according to a Maximum Entropy principle, Minimum Mean Square Error, and so on.

APPENDIX D

DATA AGGREGATION

TEMPORAL-BASED

Temporal based aggregation is used to group together events which belong to a certain time pattern – so-called *temporal fusion* [50]. Temporal logic formalism can also be utilized for modeling threat scenarios through so-called *chronicles* [60], which are series of events (malicious or harmless) compelled to time and context, providing a procedure to model temporal patterns and examine the evolution of the monitored system. Time information allows users to specify time intervals between events.

Each new alert is checked against predefined chronicles; if the attack pattern matches with any of them, the chronicle's state is modified; otherwise new chronicles are built up. Let us consider the following example of chronicle for scans [60]:

```
chronicle scan[?source, ?target]
{
event(alarm[sid_1,?source,?target], t1)
occurs((1,+oo),alarm[sid_2,?source,?target], (t1+1,t2))
noevent(alarm[sid_3,?source,?target], (t1,t2))
event(alarm[sid_3,?source,?target], t2+1)
t1<t2

when recognized {
emit event(alarm[scan, ?source, ?target], t2);
}
}
```

The example relies on the fact that we have alerts showing the beginning of the scan, several others during the scan, and an alert when the scan is supposed to end – respectively sid_1, sid_2, and sid_3, with same source and target. The chronicle starts with the first event, which instantiates t1, while the last event instantiates t2; between t1 and t2+1, infinite port scan alerts may occur. Moreover, a constraint that sid_3 does not occur more than once within a chronicle is also required.

This technique, which is based on the knowledge of specific scenarios, seems to be capable of reducing the high number of alerts as well as false positives, due to the analysis of contextual events. Chronicles have analogies with attack scenario modeling languages, such as LAMBDA [44]; however, these are not based on time and often rely on straightforward definition of the scenarios, impeding flexibility in understanding attacker's intentions when

arbitrary actions are performed. Hence, chronicles do not explicitly represent attack scenarios but is meant to model attack patterns in which many events as well as other contextual information are involved.

FEATURE SIMILARITY-BASED

It has been widely recommended to cluster events together when they have similar features [18; 49; 51; 41] [53] [52]. M-Correlator [49], for example, proposes to group together alerts with regards to the following matching features:

- ports and IP addresses, i.e., same network session;
- process ID or user ID, i.e., host activity;
- alert type, i.e., attack class defined for previous alert instances in an Incident Handling Fact Base (see Data Normalization).

Naïve aggregation [51] suggests clustering events which share the same source IP addresses. Although this technique is very effective in reducing data volume for simple attacks – i.e., performed through simple scripts by unskilled computer fanatics –, it makes detection of advanced attacks nearly impossible.

Security experts can also define a rule set to cluster alerts based on similarity of features related to [46]:

- attack type, i.e., defined in a classification field;
- timestamp, i.e., the actual time of occurrence of the event;
- event source and target, i.e., for most network attacks we can compare services and nodes, for other attacks – such as TCP scan – it is enough to compare nodes.

A meta-alert must be updated as soon as new alerts are added; for this purpose it has been proposed to utilize a maximum delay time for cluster stability. In other words, new alerts cannot be inserted into an existing meta-alert after a certain time-span. In addition, if the last meta-alert's update occurred in a remote past, the relative cluster can be reported to security officers, and further ignored [54]. The temporal threshold, however, is predetermined and depends on the detected class of attack.

It is noteworthy to say that alerts within the same meta-alert might have some differences, thus leading to a local information conflict. Hence, it may be helpful defining some integrity constraints to check during the aggregation; for instance, checking that "this attack type has

always a single source" allow detecting conflicts and potentially reporting additional information [46], such as a second source for the meta-alert at hand.

Possible ambiguity of an alert that belongs to multiple attack classes can be solved by taking into account the class of the following alert with unique label. Nonetheless, if alerts have no class specified, they can be clustered in a "no-class" meta-alert, and then manually labeled [54].

A different technique [53; 6] with regards to the selected features considers grouping alerts with identical attributes but different timestamps, although they should belong to the same (predetermined) temporal window. By using a sliding time window, alerts belonging to the same window are clustered into meta-alerts if they also have similar values of the feature set. Depending on the type of attack that one aims at detecting and the computational performance, a threshold parameter is employed to modify the width of the time window.

Clustering algorithms based on learning such as Nearest Neighbor (NN) have also been proposed [54] with the goal of classifying items and assigning them to the nearest cluster; the distance, in other words, between the item (e.g., alert) and the cluster (e.g., meta-alert) must be lower than a given threshold:

$$dist(A.feat_i , M.feat_i) \leq thres_i \quad \forall i = 1, .., v$$

where A is an alert to be clustered, M is a meta-alert, $feat_i$ is the i-th feature, v is the total number of features. The threshold is chosen based on the attack class and the characteristics of the network.

If $dist(A,M_i) \leq threshold \quad \forall i = 1, .., k$ then the we should consider $min_{i=1,\ldots,k}\ dist(A,M_i)$; in other words, when an alert can be clustered with several meta-alert, we will always choose the association that minimizes the distance. Some of the distances among features proposed for NN clustering include:

- distance between IP addresses;
- distance between port;
- time span (i.e., in milliseconds) between alert's creation time and last meta-alert update.

The algorithm is able to effectively group events belonging to the same attack and evidence has shown some robustness to erroneous attack classification by IDSs.

Similarly to the aforementioned technique, distance between feature values within the relative *generalization hierarchy* [61] [62] can explain event feature similarity. Generalization hierarchies are structures used to hierarchically order features, from the most general to the most specific values; generalization is achieved by analyzing the number of different values for every feature. The following pictures depict examples of generalization hierarchies (or taxonomies).

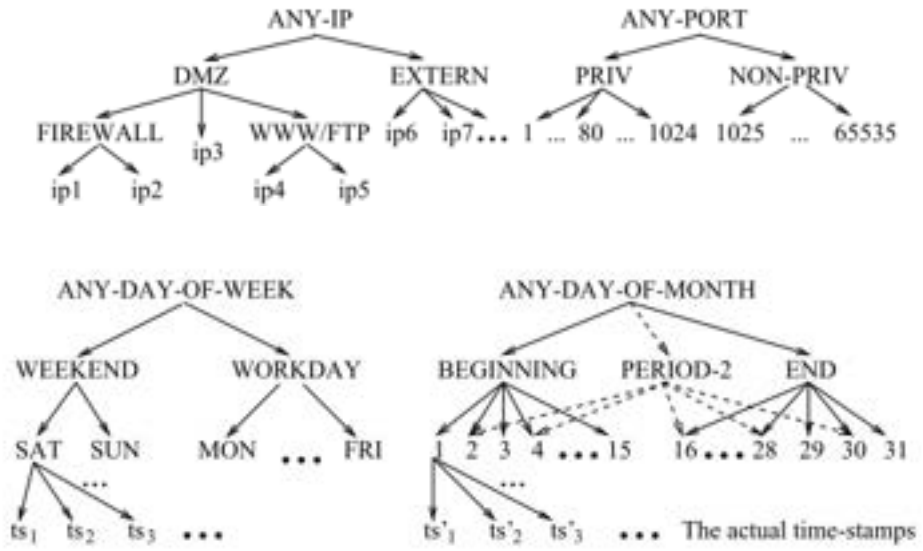

FIG. 28 GENERALIZATION HIERARCHIES FOR IP ADDRESSES (LEFT UP), PORT NUMBERS (RIGHT UP), TIME (BOTTOM) [61, 62]

Given a desired minimum size for clusters *n*, the algorithm automatically group events which are not dissimilar. The dissimilarity between alerts is the sum of each feature dissimilarity *D*; the latter is defined as the longest path between feature values in the corresponding hierarchy:

$$D_i = max\ dist(A.feat_i, B.feat_i)\ \forall i = 1, .., n$$

where A, B are two alerts, $feat_i$ is the *i-th* feature, *n* is the total number of features.

Given that identifying root-causes in large clusters usually translates into valuable results, security analysts can take advantage of presetting a minimum size for clusters. Moreover, it mitigates the risk of grouping alerts which are similar only by chance. However, the approach does not take into account the fact that various features can have diverse significance, and various alerts might produce clusters of different size. In addition, there is a risk that threats – such as stealthy attacks – may trigger a number of alerts smaller than minimum size *n*, and not produce any meta-alert.

THREAT SCORE-BASED

The approaches in this category combine data fusion with prioritization towards the criticality of the features. Consider a subject-verb-object model to describe events in such a way that the subject is the origin of the action, the verb is the action itself, and the object is something or someone involved in the action executed by the subject. It is possible to define a *threat chart* [50] as in Table 5, where different features can have a different threat score within one occurring event.

Event	Subject	Verb	Object
1	Feature 1 Feature 2 Feature 3 … Feature n	Action	Feature 1 Feature 2 Feature 3 … Feature n
…	…	…	…

TABLE 5 THREAT CHART [50]

For example, the score of the subject P_x can be computed through the following equation [50]:

$$P_x = \sum_{1}^{n} F_x$$

where n is the number of features, and F_x is the score of a single feature.

The following equation defines the final threat score [50]:

$$E_x = S_x * I_s + O_x * I_o + I_v$$

where Ex is the final score, Sx score of the subject, Ox score of the object, Is security index of the subject, Io security index of the Object, and Iv is the security index of the Verb. Security experts set the security indexes; that is to say, a threat level is assigned to each involved process.

APPENDIX E

CORRELATION OF META-ALERTS

TEMPORAL-BASED

Time series analysis can be employed to identify causal relationships among events modeled in time series variables. A time series is a set of ordered and finite values on a time axis, related to a variable of interest. Let us consider to model meta-alerts in time series variables, based on alerts per time unit; the assumption is that if event A causes event B, then A occurs before B. Granger Causality Test [53] is utilized to determine statistical causality between each vector autoregression (VAR) model variable, whose evolution is explained through an equation based on its own lags and those of other model variables.

The prior knowledge – in our case – is a set of meta-alerts which may affect each other across different periods in time. Autoregressive (AR) estimates the variable *y* at given time *t* based on its previous values; autoregressive moving average (ARMA) estimates the variable y at given time t based on previous values of variable x as well as its own previous values. The Null Hypothesis to be tested is the following:

$$H_0: \beta_i = 0,\ i=1, 2, \ldots, p$$

In other words, *x* does not affect *y*, up to a delay *p*. The null hypothesis is rejected if a Granger Causality Index *g* is bigger than a critical value:

$$g = \frac{(R_0 - R_1)/p}{R_1/(T - 2p - 1)} \sim F(p, T - 2p - 1)$$

With F(a,b) Fisher's distribution. The residuals of AR model are computed as follows:

$$R_0 = \sum_{k=1}^{T} e_0^2(k)$$

The equation resulting in the residuals of ARMA model is the following:

$$R_1 = \sum_{k=1}^{T} e_1^2(k)$$

where T is the difference between the size of the time series variable and the lag length. Essentially, if ARMA gives a better estimation than AR, then x Granger-causes y. The main strength of the technique is that no prior knowledge about attacks is required.

PROBABILITY-BASED

An early approach to causality analysis between clusters of alerts involve the construction by security experts of a static matrix [41] which encodes the probabilistic similarity measure between attack classes; essentially, analysts are required to assess the probability which a given type of alert will follow another type.

Each new alert is compared to existing meta-alerts in order to identify common features; a similarity function returns, for each feature, a value between 0 and 1, where the latter represents two identical items. The overall similarity is defined as follows [41]:

$$SIM(X,Y) = \frac{\sum_j E_j SIM(X_j, Y_j)}{\sum_j E_j}$$

where X is the new alert, Y is the candidate meta-alert, j is the feature index, E is the expectation of similarity.

The expectation of similarity weights the average of similarities of features, and is a value between 0 and 1; 0 if, for each feature, the similarity value is less than a predefined minimum similarity. Minimum matching criteria for any feature must be satisfied – regardless of the general similarity – in order for alerts to be merged. Similarity expectation represents the prior expectation that a feature match if two alerts are correlated. Each new alert may be embedded in the most similar meta-alert or start a new meta-alert if the similarity does not reach the minimum match threshold. Generic features that are utilized by the correlation engine include the following:

- Sensor information (e.g., location, name, etc.)
- Meta-alert
- Incident class
- List of source and target IP addresses
- List of ports
- Source and target user IDs
- Timestamp

If correlation is performed within an individual sensor, high minimum expectation similarity are applied on attack class, IP addresses, ports, and the sensor identifier; whereas, by loosening it on the features related to the sensor, and increasing the minimum expectation on

features such as attack class, source and target, the technique has been seen capable to correlate meta-alerts from heterogeneous sources.

Bayesian networks are able to retrieve causal relationships and relevant features among two meta-alerts, with no knowledge of predefined attack scenarios. The technique [6] results in the probability of the occurrence of alerts in meta-alert B – i.e., $P(B|A[f_j])$ – given the prior occurrence of alerts of meta-alert A. Therefore one should have knowledge about:

- prior probability of alerts in meta-alert B P(B);
- probability of alerts in meta-alert A with a certain feature value $P(A[f_j] = dom(B,f_j))$. $Dom(B,f_j)$ is the range of values that the j-th feature for alerts in meta-alert B;
- probability of alerts in meta-alert B with a certain feature value $P(B \wedge A[f_j] = dom(B, f_j))$.

Given a threshold value t, the technique also helps determining:

- features with negative influence, i.e., $P(B|A[f_j] = dom(B, f_j)) < P(B)$, feature f_j values of A decreases the occurrence probability of B;
- features with positive influence, i.e., $P(B) < P(B|A[f_j] = dom(B,f_j)) < t$, feature f_j values of A increases the occurrence probability of B;
- features with critical influence, i.e., $P (B|A[f_j] = dom(B,f_j)) > t$, feature f_j values of A considerably increases the occurrence probability of B;
- irrelevant features, i.e., $P (B|A[f_j] = dom(B,f_j)) = P(B)$, feature f_j does not affect the occurrence probability of alerts in B.

PREREQUISITE AND CONSEQUENCE-BASED

Meta-alert correlation can be based on pre-requisites and consequences of attacks. Apart from other attributes, each meta-alert can be specified as logical predicates by its pre-requisites and consequences. This information must be known and specified in advance by using state description languages, such as LAMBDA [44]. By using this language, attacks are set of events which can be specified by:

- pre-condition and post-condition of an attack;
- events which are performed by the attacker (i.e., “attack scenario”);
- events which allow the detection of the attack (i.e., “detection scenario”);

- conditions proving the attack succeeded (i.e., “verification scenario”).

The following code is an example of TCP scan in LAMBDA [52]:

```
<?xml version="1.0" encoding="UTF-8"?>
<attack attackid="MIR-0074">
<name>tcpscan</name>
<pre>use_soft(Source_address,tcpscan),
          use_service(Target_address,Target_service),
         service_type(Target_service,tcp)
</pre>
<post>
   knows(Source_user,use_service(Target_address,Target_service))
</post>
<scenario>Action</scenario>
<cond_scenario>script(Action,’tcpscan $Target_address’)
</cond_scenario>
<detection>Alert</detection>
<cond_detection>alert(Alert),
       source(Alert,Source),
       source_node(Source,Source_node),
       address(Source_node,Source_address),
       source_user(Source,Source_user),
       target(Alert,Target),
       target_node(Target,Target_node),
       target_service(Target,Target_service),
       classification(Alert,"MIR-0074")
</cond_detection>
</attack>
```

If there is a link between the post-condition of an attack A and the pre-condition of attack B, one can correlate an occurrence of A with an occurrence of B; if there is a link between the post-condition of A and the post-condition of B, one can assume they have the same objective, thus it is possible to correlate them. In this way, one can build a set of correlation rules which are used to correlate meta-alerts [52].

Correlation can be performed by matching consequences of an alert with prerequisites of another one, taking into account temporal characteristics [73]; if there is at least one consequence of meta-alert x which occur before the first occurrence of one of the prerequisites of meta-alert y, then we may state that x prepares for y. The following SQL statement epitomizes the correlation process:

```
SELECT DISTINCT c.MetaAlertID, p.MetaAlertID
FROM PrereqSet p, ExpandedConseqSet c
WHERE p.EncodedPredicate = c.EncodedPredicate
AND c.end time < p.begin time
```

It is noteworthy to clarify that an attack may occur despite the occurrence of earlier attacks that are encoded as its prerequisites; and vice versa, an earlier attack may exist even if it does not cause attacks which are specified as its consequences.

Given that the technique offers a further specification of the events, it is potentially able to reduce the negative impact of false positives – i.e., they are most likely to appear accidental and not correlated to others – as well as false negatives, as the approach is robust even if a few intrusions have not been detected; however, the technique will not adequately correlate two meta-alerts if the upstream detection engines is not able to detect critical attacks, or there is no defined relationship between them. Hence, the authors also suggested employing it together with other correlation techniques in order to improve its effectiveness; in addition, this technique requires security experts to encode quality additional information in advance, which may be an expensive and unreasonable task.

APPENDIX F

CORRELATION/AGGREGATION OF INTRUSION ALERTS

SCENARIO-BASED

These techniques rely on the knowledge of specific attack scenarios in order to determine causality between alerts and group them based on predefined relationships, thus reducing the number of suspicious events to be further examined by security analysts.

Attack scenarios can be described with consequence rules, which contain the ordered relationships between attack classes [64]. Back and forward reasoning is employed to detect duplicates and consequences of alerts from multiple detection sensors; duplicates refer to alerts with matching features from different detection sensors, whereas consequences are sets of ordered and correlated security events – within a specified time window – which describe attack scenarios. Duplicates and consequences – defined in a configuration file – are identified to logically link alerts together. The duplicate definition is represented by the following fields:

- class of the initial alert;
- class of the duplicate alert, i.e., the alert candidate to be duplicate;
- list of alert attributes, i.e., attributes should match for considering two alerts duplicates;
- updated severity level, i.e., the severity level is updated by any found duplicate.

First, each intrusion alert is checked against previous alerts in order to find those with similar ‘class of duplicate alert’, originating from different sensors (e.g., by looking at ‘analyzerid’ in IDMEF message). Then the algorithm looks for the early alert which match the ‘class of the initial alert’ and the attribute conditions; if it is found, the current alert is linked to the above mentioned initial alert, and the severity level of the duplicate definition is updated.

The consequence mechanism involves a set of connected alerts and requires a fixed order for the occurrence of them, within a temporal window. The consequence definition is represented by the following terms:

- class of the initial alert;
- source (probe) of the initial alert;
- class of the consequence alert;

- source (probe) of the consequence alert;
- severity level, i.e., predetermined severity level of the consequence;
- wait period, i.e., timer.

Forward reasoning is used to check whether an arriving intrusion alert is the consequence of previous alerts. If the consequence does not occur during the wait period, an internal alert with predefined severity level is created and linkages are discarded; or else the consequence link is marked as occurring.

Finally, events emerging from the correlation process are clustered into data structure called *situations* based on the following features:

- source;
- target;
- attack class.

Situations can be seen as *quasi*-meta-alerts; when the threshold value for the severity level of a situation is exceeded, the meta-alert is created. Depending on the features which are selected for aggregation, the authors distinguish seven possible situations, ordered from more to less specific:

- Source, target, and class of the alerts are similar (e.g., an attacker attacking a web server).
- Source and target of the alerts are similar (e.g., an attacker attacks various services on a machine).
- Target and class of the alerts are similar (e.g., distributed attack of a single target).
- Source and class of the alerts are similar (e.g., attacker launching name server attacks on DNS servers).
- Source of the alerts are similar (e.g., single attacker, multiple targets).
- Target of the alerts are similar (e.g., distributed attacks).
- Class of the alerts is similar.

The more specific a situation is, the higher the priority – among other situations – for the production of a meta-alert.

Techniques based on machine learning employ manually-labeled alerts to learn scenarios; labels specify the scenario which alerts belong to. A heuristic technique [51] compares each new intrusion alert with the most recent alerts in a scenario. Scenarios can be considered as

meta-alerts which contain ordered sequences of correlated alerts. The probability P that two alerts having different class should belong to the same scenario is computed as the following product:

$$P = L * T * S$$

Where L is the probability that alert of a generic class A is followed by an alert of a generic class B, T is a sigmoid function which accounts for the time span between the two alerts, S accounts for the source IP addresses; P, L, T, and S are all values in the range [0,1]. This technique is based on the assumption that alerts belonging to a given scenario occur close in time and involve the same range of source IP addresses; in addition, the authors believe that some sequences of alert types occur more often than others.

Analogously, decision trees, radial base function network (RBFN), and multilayer perceptrons (MLP) have been utilized to build alert classifiers, thus clustering alerts into scenarios. The features that have been utilized are summarized as follows [51]:

- class of the arriving alert;
- class of the 3 most recent alerts in the scenario;
- time span between the new alert and the most recent in the scenario;
- maximum number *r* of bits in a subnet mask;
- previous alerts in a scenario are same class of the new alert (e.g., same tools or attacks repeated);
- destination IP address (e.g., attack to a single target);
- maximum value of r when comparing two source IP address (e.g., spoofed attacks);
- maximum value of r when comparing source IP address of a new alert to the destination IP of all alerts in a scenario (e.g., a compromised host is used as source of attacks).

Each alert may be attached to a scenario (i.e., meta-alert) which has the highest probability, or starts a new one if the probability estimation does not exceed a predefined threshold value. Probabilities of false alarms may also be assigned to scenarios in order to decrease the rate of false positives and to increase the sensor sensitivity [51]. There is evidence showing that decision trees, on equal training data, have the lowest error rate than MLP and RBFN.

Extensive expert knowledge is required for attack scenario specification, as well as manual alert labeling for training. The weaknesses of these techniques are that the models resulting

from it may over-fit the training data; moreover, they are not able to correlate previously unseen scenarios.

PREREQUISITE AND CONSEQUENCE-BASED

Likewise meta-alert correlation, techniques based on prerequisites and consequences of alerts can also be employed to correlate and cluster intrusion alerts.

Neural Networks (i.e., Multi-layer Perceptrons and Support Vector Machines using a sigmoid function) employ supervised machine learning to compute the probability that two alerts are correlated [10]. Prerequisites and consequences of alerts are encoded, in terms of correlation weight and average time span between two alerts, in a correlation matrix, which is incrementally updated during training. Hence, one is able to detect variations in known attack patterns and discover new scenarios. An example of correlation matrix for 3 alerts (i.e., a1, a2, a3) is depicted in Table 6.

	a1	a2	a3
a1	C(a1,a1)	C(a1,a2)	C(a1,a3)
a2	C(a2,a1)	C(a2,a2)	C(a2,a3)
a3	C(a3,a1)	C(a3,a2)	C(a3,a3)

TABLE 6 EXAMPLE OF CORRELATION MATRIX [10]

For two arbitrary alerts ai and aj with i≠j, C(ai,aj) and C(aj,ai) are different values, which depend on the order of occurrence of the two alerts; for instance, C(ai,aj) implies that ai occurs before aj. In addition, if C(ai,aj) is much bigger than C(aj,ai) then aj is consequence of ai; whereas, if C(ai,aj) and C(aj,ai) are similar values, one can conclude that there is no correlation between the two alerts. The correlation strength is calculated according to the following equation [10]:

$$C(ai, aj) = \sum_{k=1}^{n} p_{i,j}(k)$$

where p(k) is the probability of the k-th correlation, n represents the number of times that ai and aj have been directly correlated.

The following six features have been suggested by the authors for the correlation of two intrusion alerts:

- similarity between source IP addresses (i.e., *sim(ip1, ip2) = n/32*, where n is the number of identical bits, 32 is the number of bits of an IPv4 address);

- similarity between target IP addresses (i.e., *sim(ip1, ip2) = n/32)*;
- similarity of target port numbers;
- similarity between source IP address of the new alert and the target IP address of previous alert;
- backward correlation strength between two alerts (i.e., $\Pi_b = C(a_i, a_j) / \sum_{k=1}^{n} C(a_k, a_j)$);
- frequency of correlation of two alerts.

Selected features form the input vector for a feed-forward neural network model – e.g., authors have employed one hidden layer with seven neurons – known as multilayer perceptron (MLP). For the training stage, MLP uses back-propagation, which occurs by modifying the weights after each piece of data is processed, according to the comparison between the expected result and the error in the output.

The training consists of assigning the desirable correlation probabilities to some training patterns. When all the features match the training pattern will be assigned label 1, opposed to label 0 when all the features mismatch; other labels are assigned by comparing one example with those that were previously labeled. The output of the MLP is the correlation probability for two alerts, which is a value between 0 and 1.

Correlation can also be considered as a binary classification problem. Hence, support vector machine (SVM) can be employed to state whether or not two alerts are correlated. Unlike MLP, SVM training patterns are assigned class labels 1 and -1 – i.e., correlated and not correlated, respectively – and not their probabilities [10]. The output of the SVM is not in the form of probability but represents the distance between the input vector and the hyperplane which divides the two classes optimally. However, by fitting a sigmoid that maps SVM output to posterior, a good mapping from SVM to probability is obtained.

The authors utilize two thresholds: correlation threshold and correlation sensitivity; the former is a value between 0 and 1 (e.g., 0.5) which determines if two alerts are correlated or independent, the latter basically determines the number of correlations of a new alert to previous ones.

Although SVM is generally very robust for classification and non-linear regression with large number of inputs, and makes quicker and easier labeling process and training, evidence shows that MLP produces more accurate outputs (i.e., correlation probabilities). In order to improve accuracy, adequate training examples should be chosen for the SVM. Nonetheless, the training of MLP is slower and it may over-fit the training data. The big

advantage of using these techniques is that it does not require a huge amount of rules in order to find causality between alerts.

Causal relationships between alerts can also be inferred through statistical approaches such as Bayesian Networks (BN) [45; 6; 42]. The technique can be used to automatically retrieve casual relationships among a large amount of alerts, as well as features which are influential for the degree of relevance of two alerts, according to a prior correlation probability.

In essence, a Bayesian network is made up of a directed acyclic graph (DAG), which defines variables, and the probabilities of each variable (e.g., alerts, system attributes, etc.). Occurrence of 'child' alerts depends on the state of 'parent' nodes. Prior knowledge is represented by prior probability of root's states and conditional probabilities of child nodes. The following probabilistic inference can be propagated from parent nodes to child nodes:

$$P\,(child|parent) = P\,(child \wedge parent)/P\,(parent)$$

As the Bayesian model also computes the occurrence probability of alerts, it can be used to check alert behavior variations and update correlation probabilities real-time, thus updating causal relationships between meta-alerts. Statistical relationships between alerts in each time slots are used to fill in *Conditional Probability Tables* (CPT), which encode the prior probabilities. Table 7 shows an example of *CPT* where A, B, and C are alerts, P1, …, P8 are probability values; the value 1 corresponds to the occurrence of the alert, 0 the non-occurrence.

Each alert can be assigned to time slots and be represented by a binary value 1 or 0, either it occurs or not in the considered temporal window [45]. Conditional probabilities, which can be derived from the statistical relationships between alerts within each time slot, show the strength of the relationship between an alert and another one that has caused it.

	AB=00	**AB=01**	**AB=10**	**AB=11**
C=0	P1	P2	P3	P4
C=1	P5	P6	P7	P8

TABLE 7 EXAMPLE OF CONDITIONAL PROBABILITY TABLE [45]

Each non-root node in the causal network relates to a CPT, showing the strength of the relationship with the parent nodes. Probability threshold can be tuned to get better results with respect to the detection rate or false positive rate.

Alternatively, let us assume to represent alerts and system attributes (i.e., security state of the system) as nodes in a directed acyclic graph [42]; edges connect attributes to alerts or to other attributes. In the first case, the system attribute is a prerequisite of an attack; the second case indicates that one attribute implies another. Alerts are considered true – and can be added to the graph – only if the precondition attribute is true.

In other words, the probability P(a) that alert *a* is true, is the prior confidence of *a* when the prerequisites are satisfied, otherwise P(a) = 0. Hence, each graph node is associated with a conditional probability table which specifies the relationship between a child and parent nodes. Given n alerts belonging to an attack type T, the aggregated prior confidence $Pr_a(T)$ is computed as follows [42]:

$$Pr_a(T) = 1 - \prod_{i=1}^{n}(1 - Pr(Type_{a_i}))$$

where a_i are alerts to be merged into meta-alerts, $Type_{ai}$ represent the attack class of the *i-th* alert. Clearly, high confidence in state-based data translates into certainty about alerts. Hence, it is crucial to monitor the system regularly and strike a balance between system scanning frequency and resource consumption.

APPENDIX G

MULTI-STEP INCIDENT CORRELATION

Different stages in multi-step cyber incidents can be reconstructed in various ways. Reasoning about sophisticated threats may also require hypothesizing about events which have been discarded or not detected by the deployed security sensors [65] [52; 66].

SIMILARITY-BASED

One naïve technique [41] consists of measuring the similarity between source and targets of event reports. In other words, similarity of source and target of the correlated meta-alerts are measured in order to reconstruct different stages in a multi-step attack. Let us evoke the concept of expectation of similarity of section 0, the technique has been able to reconstruct multistep attacks by relaxing the minimum expectation on attack class features, although it is considered inadequate for attack strategy analysis and intrusion prediction.

STATE DESCRIPTION-BASED

Likewise attack scenario patterns lead to sequences of state transitions, security incidents such as bot infection can be identified through communication sequences [70]. As a matter of fact, a subset of potential predetermined transactions can be observed during bot infections; these potential transactions shape a generic infection dialog model as follows:

E1: Inbound scan
E2: Inbound exploit
E3: Outbound binary acquisition
E4: Outbound C&C communication
E5: Outbound infection scanning

In order to solve the problem of missing events, one can set up minimum and sufficient conditions for bot infection declaration (e.g., one meta-alert showing outbound host coordination and one meta-alert of exploit propagation; at least two distinct outbound bot dialog meta-alerts, and so forth). Sequences of meta-alerts are tracked and stored in a dynamic data structure (e.g., network dialog correlation matrix in Fig. 29) for each internal host in the relative dialog transaction column (E1, …, E5); ‘old’ meta-alerts will be removed from the matrix by utilizing a temporal-based pruning algorithm. The bot infection dialog model may require periodical updates, depending on the evolution of malware which is involved in the infection.

Int. Host	Timer	E1 ◷	E2	E3	E4	E5
192.168.12.1	◷	$A_a \ldots A_b$				
192.168.10.45	◕		$A_c \ldots A_d$		$A_e \ldots A_f$	
192.168.10.66	◕		A_g			
192.168.12.46	◕				$A_h \ldots A_i$	$A_j \ldots A_k$
:						
192.168.11.123	◷ ◕	A_l	$A_m \ldots A_n$	A_o		

FIG. 29 EXAMPLE OF NETWORK DIALOG CORRELATION MATRIX [70]

State description languages (e.g., LAMBDA) also allow the abduction of intermediate alerts from scenarios, even though they have not been detected, thus extracting correlation rules online and building up whole – and still unknown – attack scenarios. Essentially, if an arbitrary alert is the known consequence of an event that should have raised another alert too – which however is not received by the correlation engine – the missing event is hypothesized [52].

GRAPH-BASED

Apart from state description or dialog modeling, abduction of virtual events can be obtained by graphical means. In other words attack scenario trees can be drawn with system state as nodes, and patterns of observed events which enable predetermined state transitions as edges [50; 67] (see Fig. 30 Example of attack scenario tree); or even report goals and sub-goals of attacks (i.e., leaves are malicious actions, internal nodes are sub-goals, and the root is the final goal of the attack, see Fig. 31).

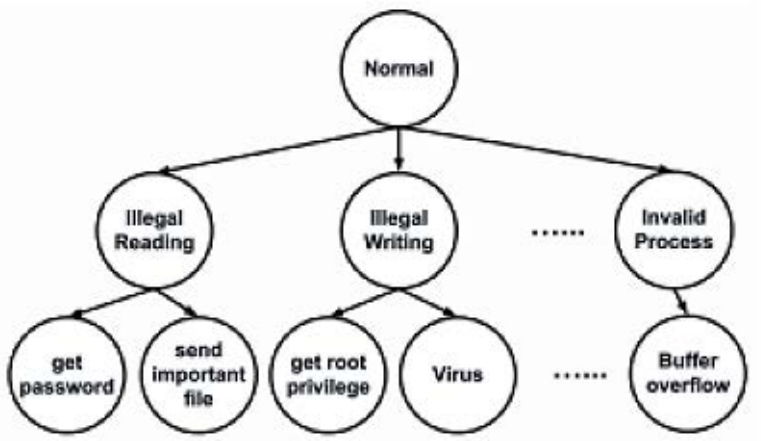

FIG. 30 EXAMPLE OF ATTACK SCENARIO TREE [50]

By combining complementary techniques to correlate alerts independently, we are able to create a set of correlation graphs, which in general might be disconnected because of missing alerts. Graphs that contain alerts with similar features are further analyzed by experts, and virtual missing alerts are included in order to satisfy the causal relationships [65; 66]. For

instance, one can use prior knowledge about attacks and timestamps in order to form hypotheses on causality between events in two different graphs. Therefore, the technique is not effective when none of the attacks in multi-stage incidents is known.

Prior knowledge about attacks can be stored in causal networks converted from scenario trees and employed to correlate isolated scenarios [65]; for example, if a scenario is sub-goal of another and occurs after it in time, individual scenarios can be combined. Thus, the technique allows collecting information which may be useful for further investigation; however, it requires beforehand definition – by security experts – of an updateable archive of known attack strategies in order to apply automatic correlation and predict multi-step incidents using the Bayes model. Moreover, by applying this technique one assumes to have understood the actual goal of the attacker; nonetheless, that task is very challenging when many attackers participate in the same attack.

Attack strategy graphs can also be depicted as directed acyclic graph (DAG) in which nodes are attacks, edges shape the order of the attacks, and some satisfying conditions for the attack execution are defined on nodes and edges. Similarity between attack patterns are explained in terms of the cost of converting a scenario into another. In other words, assuming that size of attack graphs is such as to make analysis feasible in reasonable time, similarity between sequences of events can be seen as an error tolerant graph/subgraph isomorphism problem; error tolerance handles noise and distortion in the input graphs. Hence, similarity between two graphs is the least cost pattern of editing operations, which turns a graph into another (e.g., by inserting nodes, replacing edges, and so on) [81].

An arbitrary attack graph can be considered as a quadruple of nodes, edges, nodes labeling, and edges labeling – i.e., G = (N, E, T, C). Given two graphs G_1 and G_2, a bijective function f is defined a graph isomorphism from G_1 to G_2 if [81]:

- for all $n_1 \in N_1, T_1(n_1) = T_2(f(n_1))$;
- for all $e_1 = (n_1, n_1') \in E_1$, there exists $e_2 = (f(n_1), f(n_1')) \in E_2$ such that $C(e_1) = C(e_2)$, and for all $e_2 = (n_2, n_2') \in E_2$, there exists $e_1 = (f^{-1}(n_2), f^{-1}(n_2')) \in E_1$ such that $C(e_2) = C(e_1)$.

An arbitrary subgraph of G is a graph $G_s = (N_s, E_s, T_s, C_s)$ with $Ns \subseteq N$, $E_s = E \cap (N_s \times N_s)$, $T_s(n_s) = T(n_s)$ for each node of G_s, and $C_s(e_s) = C(e_s)$ for each edge of G_s. Given two graphs G_1 and G_2, an injective function f is defined a subgraph isomorphism from G_1 to G_2 if:

$$\exists\ G_{2s} \text{ subgraph of } G_2 : f \text{ is a graph isomorphism from } G_1 \text{ to } G_2$$

Graph isomorphism problem relates to the similarity between two attack strategies, while subgraph isomorphism problem aims at identifying strategies which are components of another strategy. Graph edit distance determines the similarity of two graphs as the least cost sequence of edit operations to transform one graph into another one. Let assume having N_n node edit operations to transform a graphs G_a into G_b, with n_a and n_b nodes respectively, the similarity between them results from the following equation [81]:

$$Sim(G_a, G_b) = 1 - \frac{N_N}{n_a + n_b}$$

The model in which a DAG has attributes and alerts as nodes [42] (i.e., a system state leads either to alerts or other security states) may contain graph inconsistencies due to missed events, or unpredicted variations in the attributes; security analysts can ground their hypotheses about missing events on such inconsistencies.

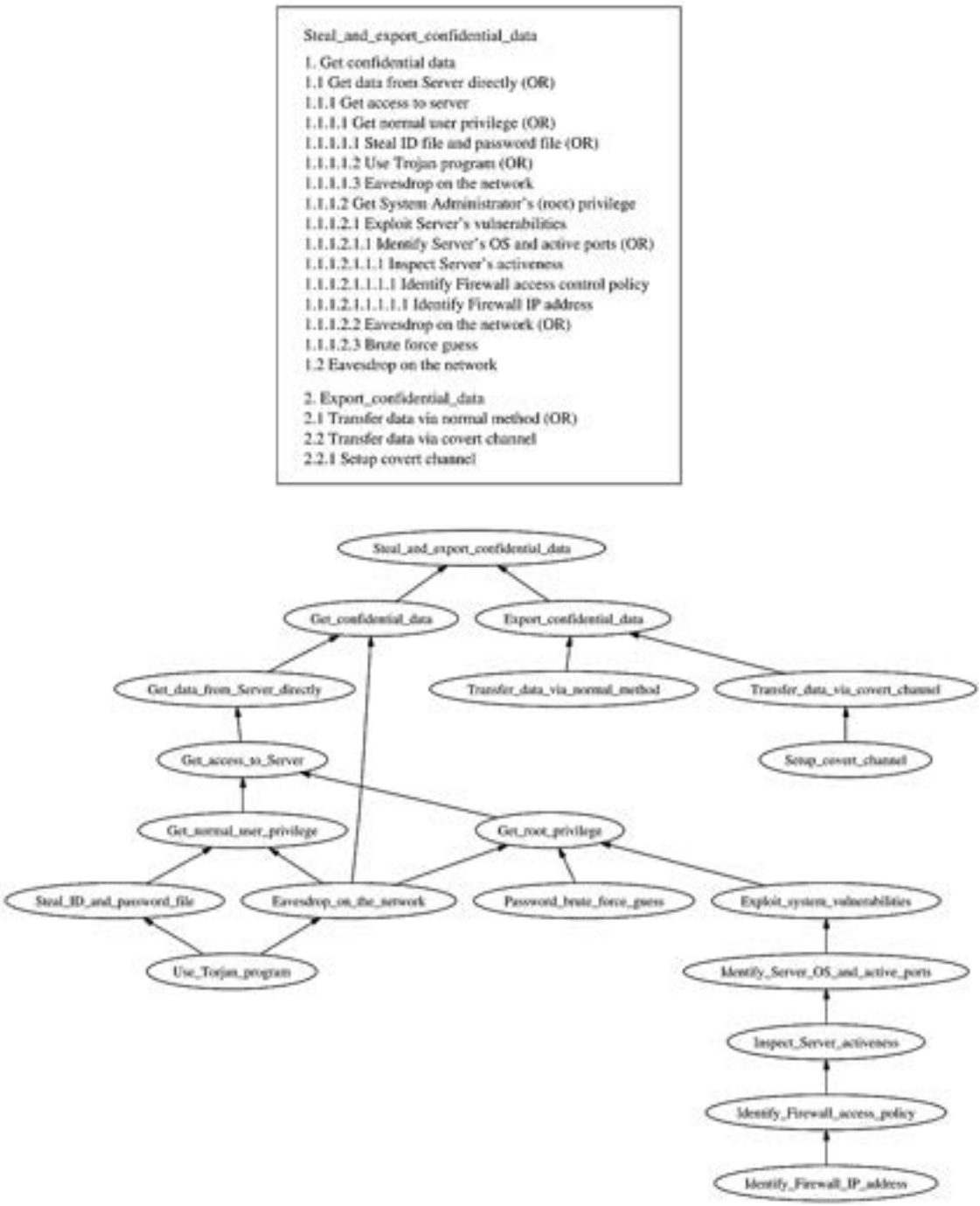

FIG. 31 EXAMPLE OF ATTACK TREE (TOP) AND CAUSAL NETWORK (BOTTOM) [42]

In order to predict incidents, alert trees can be transformed into Bayesian networks by attaching a correlation probability to each edge [6] (see Fig. 32) – i.e., probability that alert of arbitrary type A leads to alert of type B has been widely and effectively used for reconstructing high-level attack scenarios [6; 10] – or by computing goals and sub-goals probabilities [65], derived from the parent nodes' state, and then apply inference to the Bayesian network based on satisfied sub-goals and observed events (see Fig. 31).

Alert Type Pair	Correlation Probability	Selected Features
<Port_Scan, Buffer_Overflow>	75%	DesIP,DesPort
<Buffer_Overflow, FTP_User>	90%	SrcIP,DesIP
<Buffer_Overflow. FTP_Pass>	90%	SrcIP,DesIP

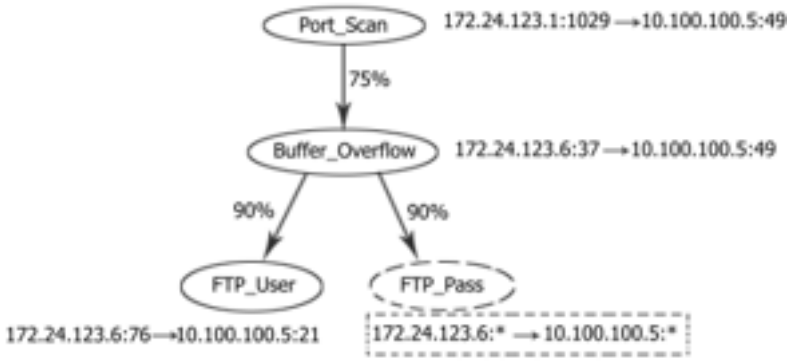

FIG. 32 EXAMPLE OF ATTACK GRAPH WITH CORRELATION PROBABILITIES [6]

By applying a data mining algorithm, it is also possible to derive scenarios and their *probability of having following attacks* (PFAs), by taking into account as input a set of candidate scenarios and a minimum support value; then, computed probabilities can be attached to attack graphs and used to rank attack patterns [67]. Fig. 33 shows an example of attack graph, in which S_i are system security states, e_j are exploits that lead to state changes, the decimal quantities represent the PFAs.

This technique presents some limitations due to the fact that the PFA of some attacks cannot be retrieved, as well as the resulting attack graphs may be incomplete. However, the technique is capable to identify novel attack scenarios and to predict evidence for probable following attacks.

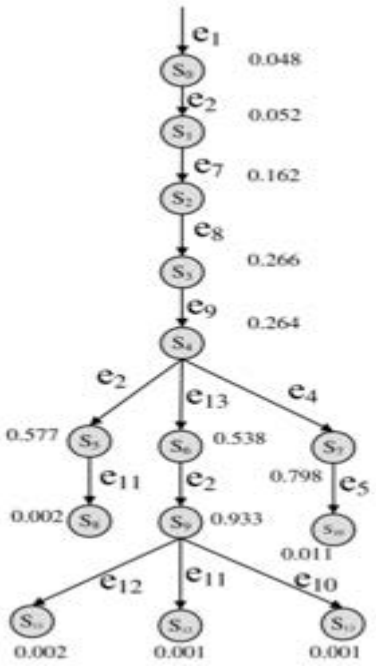

FIG. 33 EXAMPLE OF ATTACK GRAPH WITH PROBABILITY OF HAVING FOLLOWING ATTACK [67]