Computer Networks 57 (2013) 1289–1317

Contents lists available at SciVerse ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Survey Paper

A model-based survey of alert correlation techniques

Saeed Salah, Gabriel Maciá-Fernández *, Jesús E. Díaz-Verdejo

Dept. Signal Theory, Telematics and Communications, University of Granada, C/Periodista Daniel Saucedo Aranda s/n, 18071 Granada, Spain

ARTICLE INFO

Article history:
Received 2 May 2012
Received in revised form 5 September 2012
Accepted 10 October 2012
Available online 23 January 2013

Keywords:
Alert correlation
Network management systems
Fault localization
Intrusion detection systems
SCADA systems

ABSTRACT

As telecommunication networks evolve rapidly in terms of scalability, complexity, and heterogeneity, the efficiency of fault localization procedures and the accuracy in the detection of anomalous behaviors are becoming important factors that largely influence the decision making process in large management companies. For this reason, telecommunication companies are doing a big effort investing in new technologies and projects aimed at finding efficient management solutions. One of the challenging issues for network and system management operators is that of dealing with the huge amount of alerts generated by the managed systems and networks. In order to discover anomalous behaviors and speed up fault localization processes, alert correlation is one of the most popular resources. Although many different alert correlation techniques have been investigated, it is still an active research field. In this paper, a survey of the state of the art in alert correlation techniques is presented. Unlike other authors, we consider that the correlation process is a common problem for different fields in the industry. Thus, we focus on showing the broad influence of this problem. Additionally, we suggest an alert correlation architecture capable of modeling current and prospective proposals. Finally, we also review some of the most important commercial products currently available.

1. Introduction

Over the past few years, decision makers in large management companies have realized the urgent need to enhance service level agreements (SLAs) with their customers, in order to increase their profits. This need has emerged in parallel with the evolution, in terms of scalability and complexity, of telecommunication networks, and the severity of the services that they offer. As a consequence, network and system operators have developed methods and techniques that make them able to manage, monitor and follow-up this evolving complexity in an efficient way. Thus, we have seen an increasing number of commercial products designed with the ability to monitor and collect information about the network and sending it to the operators as alerts. These alerts provide valuable information if they are used in an effective way, thus supplying network operators with a comprehensive knowledge about their monitored networks.

Actually, a medium-sized telecommunication network operations management centre may eventually receive a huge amount of alerts per day. In these scenarios, it could become a very difficult task for network and system operators that of rapidly discovering the root causes of a problem. Fundamental questions that network operators should deal with are: which alerts can be filtered out, how alerts could be grouped and correlated, and how existent alerts should be prioritized. To answer these questions, different alert correlation techniques, algorithms and models coming from different design approaches have been proposed in industrial and research communities, each one having its own advantages and disadvantages. Despite of this, there are no definite solutions until this moment, mainly because the alert correlation process is inherently a complex task, composed of many interrelated

* Corresponding author. Tel.: +34 958242305; fax: +34 958 240831.
E-mail addresses: sasalah@science.alquds.edu (S. Salah), gmacia@ugr.es (G. Maciá-Fernández), jedv@ugr.es (J.E. Díaz-Verdejo).

1389-1286/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comnet.2012.10.022

sub-processes that are expected to collaborate in a coherent manner to do their job.

In this paper, we aim to achieve the following four contributions. First, we provide a comprehensive state of the art in alert correlation techniques based on a new proposed taxonomy, providing a global insight to the different efforts made in this field from different research and industry communities within the last few years. Second, unlike many surveys that are restricted to the network security field, we also describe the efforts coming from the network management field (NMS) and the processes control field (SCADA systems) in the industry. Third, we propose a comprehensive architectural correlation model which is intended to survey all the stages, techniques and methodologies suggested in the state of the art. Finally, we review the most important alert correlation tools currently available and analyze their alignment with both the state of the art and the proposed architectural model.

The paper is structured as follows. Some concepts related to the correlation process and the scope of our discussion are presented in Section 2. In Section 3, we review the different sources of information considered by different authors in the correlation process. An alert correlation taxonomy is presented in Section 4. Then, an architectural model is discussed in Section 5. In Section 6 we make a comparison of existing commercial products that implement certain correlation techniques. Some related work is reviewed in Section 7. Finally, we conclude the paper in Section 8.

2. Concepts and scope

The term "correlation" has been used in many diverse applications and domains such as science, engineering, biomedical, business and others. Correlation is defined as "*an action to carry back relations with each other*" [1] or "*a measure of the relation between two or more variables*" [2]. It is useful because it can indicate a predictive relationship that can be exploited in practice. Alert correlation is one variant of correlation. It is a widely accepted technology for dealing with the management information coming from large and complex communication networks. Alert correlation is defined by Jacobson and Weissmann in [3] as "*conceptual interpretation of multiple alarms such that new meanings are assigned to them*". Gardner and Harle in [4] defined it as the "*interpretation of multiple alarms so as to increase the semantic information content associated with a reduced set of messages*". In these contexts, alerts, also referred to in the literature as alarms or simply events, are short messages with a specific textual format defined by vendors, and generated as an external manifestation of a potential failure or a disorder occurring in an element of the managed network or system. Typically, such alerts contain information regarding the device issuing them and the event itself, i.e., the creation and reception time, a description of the fault, the severity of the alert, etc. Besides, alerts may provide information with different levels of detail: specific data regarding the status of the devices and their configurations, or higher level details, with aggregated information gathered from several alerts.

The need for alert correlation comes from many reasons. First, alerts are generated in response to the detection of malicious activities or faults. Therefore, they typically contain descriptive information about the problems and their symptoms. However, they do not usually include explicit information about the root causes of a problem. Second, network operators in large management companies may receive hundreds of alarms per day. Most of them can be generated in response to non-relevant events. Therefore, it is necessary for those network operators to filter out irrelevant alarms and focus on the most crucial ones. Last but not least, in collaborative systems, the diversity and heterogeneity of networking elements poses a significant challenge to network and system operators, due to the difficulty of converging alerts coming from multiple data sources in order to develop coherent management strategies. The leveraging technique that alleviates the above problems and translates alerts into more understandable and thus exploitable information is alert correlation.

It is also interesting to consider the scope of application of alert correlation techniques. Indeed, they are mainly associated and extensively used in three different application domains: (i) network management, (ii) network and system security and (iii) industrial processes control (SCADA systems).

A Network Management System (NMS) is used by network operators for carrying out management tasks such as configuration management, which aims at configuring the devices settings and functions; fault management, which deals with faults, their effects and the solutions; performance management, which monitors the network status and some performance issues, and others, like security and accountability management. For the interaction between network and system equipments and the NMS, network management protocols are used. The Simple Network management Protocol (SNMP) [5] is the dominant one. Here, agents are processes running on managed devices that collect information and send it as alerts (*traps*) to the manager, where further processing is done before showing the information to the network operator. Alert correlation is mainly used here to help operators in real-time diagnosis and to speed up the fault localization process.

Network and system security is another important application field of alert correlation. Here, it is used for building a consolidated security picture of the whole system. Alerts are typically generated by security elements such as NIDS, HIDS, firewalls, antiviruses and others, in response to discovering malicious or simply anomalous activities. Since the detection sensitivity of these security elements is variable, they could generate a massive amount of alerts, where some of them could really correspond to normal events that are mistakenly considered as attacks (false positives). Alert correlation helps security experts to verify the validity of those alerts, and to detect complex and multistep attack scenarios.

Alert correlation has also been used in industrial sectors to advance the automation of processes control (SCADA systems). In most manufacturing systems, there are a lot of switches, sensors and actuators that could generate a

huge amount of alerts in response to process disturbances or failures. Existing alert management systems are really improved with the help of alert correlation techniques to help in the task of detecting root causes for problems.

3. Sources of information

The alerts generated by monitoring systems are obviously the basic source of information which should be present in any alert correlation system. Yet, there exists many other sources of information that can greatly contribute to a correlation analysis. In this sense, different authors have proposed the use of a wide variety of sources of information in order to achieve the goals of alert correlation effectively and accurately. In this section, we review the most relevant data sources which have been proposed for the correlation process.

3.1. Alerts database

This database contains the alerts generated by the different equipments and detection systems in the monitored environment. Alerts are the basic source of information for any correlation technique. They can either be generated by network management agents via management protocols such as traps (generated by SNMP), event reports (generated by the Common Management Information Protocol (CMIP) [6]), or via Intrusion Detection Systems (IDS) such as Snort [7] and GrIDS [8], or even by SCADA monitoring systems.

3.2. Topology information

The main purpose of the topology information is to provide an accurate representation of the monitored network as a set of links and nodes. The representation of the location of nodes, and the direction and connectivity of links are of particular relevance. The network topology information contains extensive details of network and equipment structure such as switches, routers, and servers; configuration parameters such as IP addresses and their matchings to names, subnets, virtual LANs; and host information such as OS type, open services. This information is typically gathered by entities and stored in a database. Jinqiao et al. [9] and Chyssler et al. [10] use topology agents to collect such dynamic network topology information, storing it in a database.

3.3. Vulnerabilities database

This source of information has been mainly considered for its use in intrusion detection environments. It stores all well-known exploits and system vulnerability information, usually with the corresponding security solutions. It is built by collecting the configuration information of the monitored resources, such as operating systems or network application services potentially susceptible to be exploited by attackers. As an example, the project implemented in [11] proposed the use of Common Vulnerability and Exposures, CVE, which is a well-known vulnerability database, free for public use. CVE is a dictionary of publicly known information about security vulnerabilities and exposures. Jinqiao et al. [9] used CVE, bugtraq and CERT vulnerability identifications for categorizing and sorting vulnerabilities in its collaborative intrusion detection system TRINETR. For each vulnerability reference, there is an associated rule to specify the corresponding evaluation process or action.

3.4. Trouble ticketing system (TTS) information

Ticketing systems are workflow tools extensively used by companies for tracking processes and evaluating their functioning. In several contexts, they have been introduced to assist in speeding up the fault recovery process and adding more advanced functions to maintain the networks [12,13]. Here, ticketing systems store tickets, which are usually generated by network operators, based on the perception of certain problems. Many of the records in the ticket database contain information related to problems generated by events identified as network failures. This makes it possible to integrate tickets in the process of alert correlation, as they implicitly could act as expert information provided to an expert system. The incorporation of this new information into an alert correlation system would permit to alleviate network operators from decisions, as well as it would allow to improve, speed up and prioritize the diagnosis of problems. Lewis and Dero [14] described some research trends pointing to an extension of the TTS framework to provide advanced functions in fault localization and alert correlation. Costa et al. [15] introduced an intelligent alarm management system for alarm correlation by adapting association rule algorithms and using trouble-ticket information to get feedback from the event correlation results.

3.5. Ontology database

Ontologies provide powerful constructs and constitute useful tools to deal with such diverse knowledge as that coming from alerts. They include machine interpretable definitions and formal specification of the concepts and relationship that can exist between entities within a domain. As an example of application in our field of study, Li and Tian [16] proposed an intrusion alert correlation system based on ontology knowledge base, and introduced modules for reducing redundant alerts to attack actions, using IDMEF [17] and CVE standards. According to alerts information and attacks knowledge, they use the reasoning power of ontologies to infer the correlation of alerts.

3.6. Cases database

A case is the description of a known problem, its associated alerts and the solutions. This source of information is mainly used in the analysis of event and error messages. Each case is described with two different elements: situation, and solution. The situation describes the context of the case. It consists of a set of patterns that can be matched with a list of alerts. The solution gives reasons about why the error has occurred and suggests how to solve it. E.g.,

Holub et al. [18] proposed a run-time correlation engine to analyze log data which uses a cases database to provide a mechanism for matching known problems in large volumes of data.

3.7. Knowledge representation

An additional source of information is related to the use of rules or models that somehow represent the relationships among alerts. These rules or models can be explicitly set by experts or inferred from the analysis of the alerts by means of learning procedures, usually from labeled samples (supervised learning). This way, a knowledge representacion allows to incorporate human knowledge in the alert correlation procedure or somehow mimic this knowledge. One of the most common ways of expressing this information is through the use of rules for an expert system. Most experts are capable of expressing their knowledge in the form of rules for problem solving. The database includes a set of facts used to match against the *IF* (condition) parts of rules stored in the knowledge-base. Other methods are related to the pattern learning field, in which the knowledge is usually represented by a model (e.g., neural networks, Markov models, Bayesian networks). As an example of its use in the security field, Kabiri and Ghorbani [19] proposed an intrusion alert correlation system using a rule-based inference engine to derive the correlation between alerts (using inference engine and a working memory that constitutes an Expert System). The inference engine was implemented using a scenario-based knowledge base, and the extraction of attack scenarios was performed by a security expert, before being stored in a knowledge-base to become operational.

4. Taxonomy of alert correlation techniques

After a thorough review of the literature, we have found several efforts focused on providing taxonomies for alert correlation techniques. Most of them have adopted a classification criterion which is based only on the correlation method used [20–23]. For this reason, we consider that they are narrowed on a small picture with limited scope.

In this paper, we suggest a classification for the existing alert correlation techniques as that depicted in Fig. 1. It tries to provide a global view of the alert correlation problem, taking into account additional aspects and not only the correlation methods, i.e., the number of data sources, the application field of the correlation techniques, and the architectural design of the system. In what follows, we describe the scope of all these aspects.

4.1. Number of data sources

Alert correlation techniques can be classified based on the number of used data sources. They can either accept the data from only one data source, or multiple data sources.

4.1.1. Single data source

Single data source techniques are those in which data comes from a single type of source of information from the list described in Section 3. Note that this does not mean that data should arrive from a single node. For example, an alerts database itself is considered as one source of information despite alerts may be coming from various nodes with different formats and natures. Most of the commercial tools which are listed in the tables in Section 5 use alerts databases (Column 4) as the only source of information.

Single source correlation techniques are usually built in for specific purposes and applications. Although their main advantage is their simplicity, they do not achieve optimal results from the correlation and they are not the best solution for collaborative monitoring systems.

4.1.2. Multiple data sources

Most of the existing correlation techniques are collaborative, which means that they depend on more than one source of information, in order to provide a more precise and coherent view about the monitored network.

Obviously, the cost of obtaining better results when multiple data sources are used is a higher complexity in alert correlation systems, mainly due to the heterogeneity of the different inputs. Moreover, they need extra amount of resources when compared with single data source techniques.

Many significative examples of these systems appear in the literature. Zhuang et al. [24] proposed an alert correlation model for detecting large distributed attacks such as DDoS attacks considering three type of information sources, namely alerts database, topology and vulnerability information. Chang et al. [25] proposed a multi-source security fusion system architecture, called MS2IFS. Besides alerts, which are generated from more than one monitoring equipment such as Snort NIDS, Ossec HIDS, Nessus, they use a vulnerability database. Hu et al. [26] also proposed an alert correlation system for hybrid networks by analyzing alarms generated by three types of network equipments in which they also used the topology information to locate the root cause of network problems. The correlation architecture which was presented by Chyssler and others in [10] also used two data sources: one is composed of the alerts themselves, generated by three different security devices: Snort IDS, the Samhain file integrity checker, and Syslog; the other is the network topological information.

4.2. Type of application

Typically, existing alert correlation techniques are implemented towards one application. Despite their potential use in many other fields, we have detected three main fields of application where these techniques have been proposed and evaluated: network management systems, IT security, and process control in manufacturing systems (SCADA systems).

4.2.1. Network management system (NMS)

Network Management Systems, as previously described, aim at allowing operators to monitor and configure a communication network. Monitoring systems

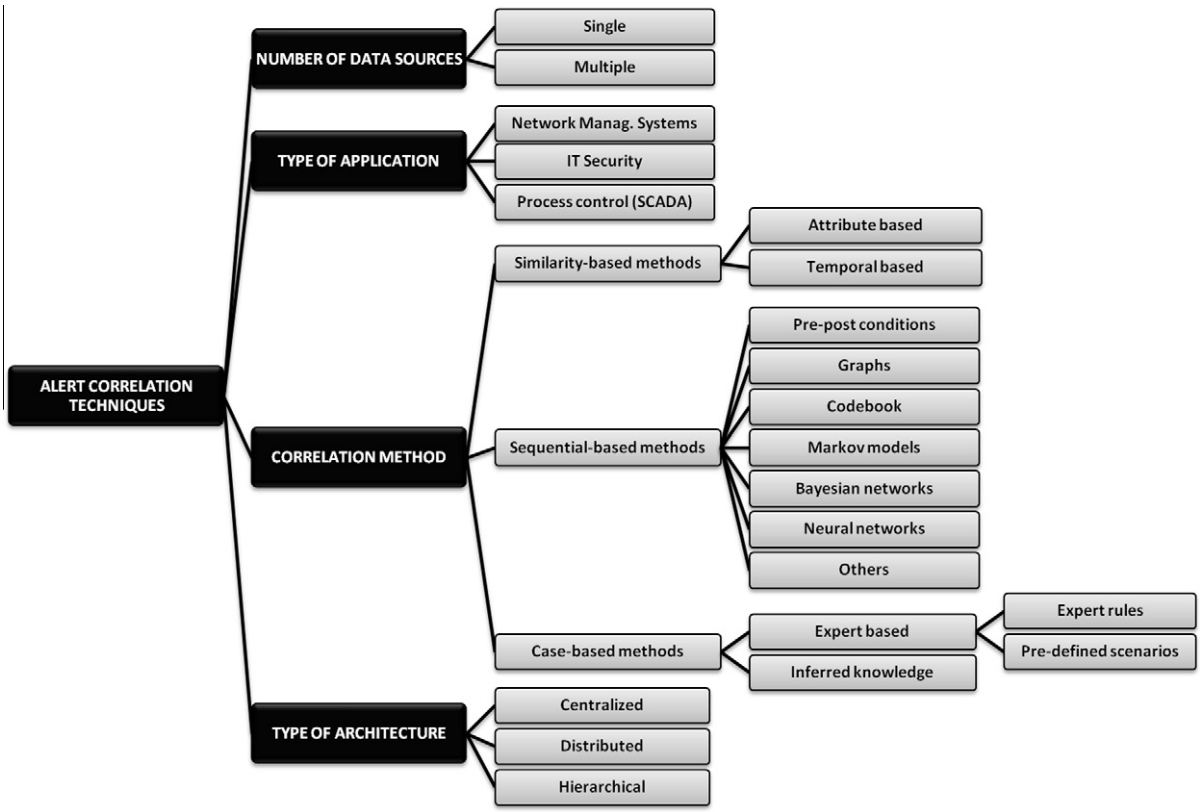

Fig. 1. Taxonomy of alert correlation techniques.

generate alerts for warning operators about problems in the network.

In NMS, alert correlation techniques are placed among the most important fault localization techniques. They have been extensively used by the research community to group and correlate alerts which have the same root causes [15,18,27,28].

4.2.2. IT security

The most prolific field in which alert correlation techniques have been studied is the systems and network security field, with special emphasis on the development of efficient intrusion detection systems.

The main objective of alert correlation techniques in this field is to produce attack reports that capture a coherent view of the activity on the network or systems without losing security-relevant information [9,29–31].

4.2.3. Process control (SCADA systems)

Since manufacturing applications increase in complexity and scale, SCADA control systems should incorporate efficient mechanisms to identify the root cause of problems or processes disturbances. This is essential for not delaying the decision making process.

Accordingly, a number of research projects have been carried out to advance the automation of manufacturing process control. In these projects, alert correlation techniques have also played an important role. As an example, Chen and Lee [32] proposed an alert correlation technique for the purpose of assisting operators' decision making in manufacturing systems. The technique uses autonomous data mining method to search historical alarm logs to find the causal relationship, and a time-shift similarity based clustering method, used to carry out the correlation pattern search.

4.3. Correlation method

In the last years, researchers and vendors have proposed many correlation approaches for alert reduction and correlation in a joint effort with networking experts. Nevertheless, alert correlation is a complex multistep transformation process, and the bulk of the current proposals operate only on partial aspects of the correlation process with different correlation methods. Here, we make a classification of the different techniques proposed in the field of alert correlation based on the correlation method used. Instead of focusing on the mathematical tools or mechanisms used for the correlation, we put our attention on the strategy followed by the different authors to correlate alerts. Thus, three major categories have been identified: similarity-based, sequential-based and case-based methods. For every of these categories, a description is provided next, along with a survey of relevant research contributions. Table 1 shows a summary of these contributions.

4.3.1. Similarity based methods

Similarity-based techniques aim at reducing the total number of alerts by clustering and aggregating them using their similarities. Each generated alert has several associated attributes or fields such as: source and destination IP addresses, source and destination port numbers, protocols, alert description, and timestamp information. The main assumption here is that similar alerts tend to have same root causes or similar effects on the monitored system. How to define similarity measures is a critical performance issue for such kind of techniques. To answer this question, several similarity measures have been proposed by many researchers, some of them are listed in the subsections below. The aim is to define a suitable similarity function for each attribute, as attributes may have different weights and effects on the correlation process.

Techniques that belong to this category exhibit many advantages. First, they are usually implemented with lightweight algorithms of less complexity than those in other categories, mainly because these algorithms are based on simple logical comparisons. Second, this category has proven its effectiveness in reducing the total number of alerts, which is an essential step in the correlation process, given the usually large number of alerts reported to network operators. However, these techniques also have some weaknesses. The most important is that they simply work on attributes level and cannot detect causal relationships between alerts, in order to discover root causes for the problems.

Similarity-based correlation techniques can be grouped into two categories: those based on attributes similarities and those based on temporal information. Next, we describe the most common proposals followed by different authors in these groups.

4.3.1.1. Attribute based. Attribute-based correlation techniques correlate alerts by using the similarity between some of their attributes or features. Here, several different features have been used, like source and destination IPs, timestamps, ports, kind of service, and users. A similarity measure is typically calculated by computing certain metrics, such as Euclidean, Mahalanobis, Minkowski and/or Manhattan distance functions. The resulting scores, when compared with threshold values, determine if these alerts are to be correlated or not. Choosing the suitable distance measure will increase the overall performance of the correlation process, as two alerts might be close or far depending on the distance function considered.

A lot of contributions regarding these techniques have appeared in the literature, as they are one of the most

Table 1
Summary of examples of alert correlation techniques.

Alert correlation techniques			
Classification			References for contributions
Type of application	NMS		Costa et al. [15]; Holub et al. [18]; Klinger et al. [28]
	IT Security		Jinqiao et al. [9]; Alserhani et al.[29]; Qin [30]; Valeur et al.[31]
	Process control (SCADA systems)		Chen and Lee [32]
Number of data sources	Single		As illustrated in tables (Column 4) in Section 6
	Multiple		Chyssler et al. [10]; Zhuang et al. [24]; Chang et al. [25]; Hu et al. [26]
Correlation method	Similarity-based methods	Attribute	Zhuang et al. [24]; Valdes and Skinner [33]; Lee et al. [34]; Siraj and Vaughn [35]; Julish in [36]; Julish et al.[37]; Debar and Wespi [38]; Cuppens [39]
		Temporal	Hossein et al. [41]; Ma et al. [42]; Qin and Lee [43]; Morin and Debar [44]
	Sequential-based methods	Pre/Post conditions	Alserhani et al. [29]; Zhaowen et al. [45]; Ning et al. [46]; Templeton and levitt [47]; Xiao et al. [48]
		Graphs	Gruschke [27]; Roschke et al. [49]; Wang et al. [50]; Li and Lifang [51]
		Codebook	Kilger et al. [28]
		Markov models	Ourston et al. [53]; Farhadi et al. [54]; Zan et al. [55]; Zhicai and Yongxiang [56]
		Bayesian networks	Qin [30]; Steinder and Sethi [57]; Marchetti et al. [58]; Harahap et al. [59]
		Neural networks	Zhu et al. [60]
		Others	Al-Mamory and Zhang [61]
	Case-based methods	Expert based	Expert rules: Cronk and others [62]; Kar-Wing et al. [63]; Jector and et al. [64]
			Pre-defined scenarios: Cuppens and Ortalo [65]; Kemmer and Vinga [66]; Eckmann et al. [67]; Liu et al. [68]; Cheung et al. [69]
		Inferred knowledge	Katipally et al. [75]; Sadoddin and Ghorabani [76]
Type of architecture	Centralized		Jinqiao et al. [9]
	Distributed		Mohamed and Basir [77]; Khatoun et al. [78]
	Hierarchical		Tian et al. [79]

widely deployed. Indeed, there exist a lot of variations in the nature of the applied technique, despite the use of a given similarity metric for the correlation. Some relevant examples of these variations are described next.

Valdes and Skinner [33] proposed a probabilistic method to correlate alerts based on a mathematical framework that is able to find the minimum similarity specification to fuse alerts from multiple sensors. The method considers appropriate attributes contained in alert reports as features for a multivariate matching algorithm. Only those features that overlap are considered for the overall similarity calculation. For every matching feature, they defined an appropriate similarity function with range zero (mismatch) to one (perfect match), and the overall similarity is calculated using a predefined equation. Alerts are correlated with a high degree of attribute similarity, generating a meta-alert if there is a match. Depending on the situation, they incorporate the expectation of match values (which are used to compute a weighted average of similarity over the overlapping features), as well as a minimum match specification that unconditionally rejects a match if any feature fails to match at the minimum specified value. For each new alert, they compute similarity for existing meta-alerts, and merge the newly created alert with the best matching meta-alert, as long as the match passes a threshold value. They reported a reduction of one-half to two-thirds in alert volume in a live environment.

A similar approach is used in Lee et al. [34]. However, unlike in [33], they used a similarity metric based on the Euclidean distance to calculate the similarity value between two alerts and make clustering. They applied this technique to detect DDoS attacks.

Siraj and Vaughn [35] assigned different similarity scores at different level of attribute abstractions. They considered the similarity notion in terms of category/class/type matching. For example, suppose that two alerts A and B have the same IP addresses; the matching will be at the type level and given a score of 4. But, if they have different IP addresses the matching is checked with higher level, in this case at subnet level, if they have the same subnet the matching score will be 3, and so on. Therefore, different types of alert clusters are generated according to different combinations of the features and feature similarities.

Zhuang and others [24] used three types of similarity mechanisms to decide whether two alerts need to be aggregated or not. These mechanisms are alphabetical, bit-by-bit, and max value comparisons, which were applied on three different attributes, i.e., alert identifier, IP addresses and port similarities, respectively.

Julisch [36] proposed the principle of dissimilarity measure instead of similarity. They defined a dissimilarity function, which takes two alerts as input, and return a numerical value that indicates how adequately these alerts can be modeled by a single generalization alert. Here, dissimilarity is inversely related to similarity, i.e., if the numerical value is very small this means that the two alerts have higher correlation and they can be modeled by a generalization alert.

The same author with others [37] tried to use a wide variety of attribute types, including numerical, categorical, time, and free-text attributes to aggregate alerts into clusters. To do that they used a variant of the classic Attribute Oriented Induction (AOI) technique as a conceptual clustering tool to discover root causes.

Debar and Wespi [38] used another method based on predefined situations to form groups of alerts by creating a small number of relationships that are exposed to the operator instead of exposing the raw alerts. To do that, they proposed an Aggregation and Correlation Algorithm (ACC), which processes the incoming alerts by extracting common information such as source and target host, and tries to find a match for these attributes in previous observations. Once the situations have been updated, the appropriate alarms are generated, if necessary, and the alerts are stored in a database.

Unlike others, instead of using mathematical formulas for calculating the similarity measures, Cuppens [39] defines the similarity relationship using an expert system approach in which each similarity requirement is specified using expert rules. Here, four categories of rules were defined in order to specify in which case alerts of the following attributes are similar: classification, time, source, and target.

4.3.1.2. Temporal based. Some similarity-based techniques use temporal time constraints to find the relationships between alerts in order to perform alert aggregation and correlation [40]. Temporal relationships between alerts provide valuable information that could be used in the alert correlation process .The idea behind temporal alert correlation techniques is to recognize that alerts caused by the same fault are likely to be observed within a short time after the fault occurrence. The simplest method of temporal correlation relies on time-windows, where only alerts occurring within a time-window are to be correlated.

The main advantage of the temporal alert correlation is to reduce the number of alerts generated by the management nodes and to convert them into high-level alerts. The strength of the temporal relationship between two alerts is labeled as *strong* if their time intervals have relatively stable values, or *loose* if the time interval is not precise. However, these approaches are usually deterministic, which limits their applicability.

Hossein et al. [41] used several time windows in order to avoid comparing new alerts with the whole set of received alerts. After that, they applied a probability estimation function to calculate a threshold value to make the correlation. Two alerts are correlated if their temporal similarity is higher than that predefined threshold. Correlated alerts are then signaled as hyper-alerts.

Ma et al. [42] applied the same mechanism as in [41]. Nevertheless, they used user-defined time periods or different kinds of time horizons to cluster and correlate alerts. This allows them to predict automatically the upcoming next step of multistage attacks in real time, by discovering sequential patterns over this predefined time window.

Instead of using time windows, Qin and Lee [43] applied a time series-based statistical analysis method to determine whether two alerts are correlated or not. They used the Granger Causality Test (GCT). The intuition of Granger

Causality is that if an event x is the cause of another event y, then the event x should precede the event y.

Morin and Debar [44] used another mechanism called the chronicle formalism to fuse alerts. A chronicle is defined as a set of events, linked together by time constraints, whose occurrence may depend on the context. The available time information allows ordering and the specification of time spans between two occurrences of events. If several identical events occur at the same time, only one of them is considered.

4.3.2. Sequential-based methods

Here, alerts are correlated by using causality relationships among them. *Pre-conditions* are defined as the necessary requirements that must exist for the attack to be successful, and the *consequences* of the attack are defined as the effects that appear after a specific attack has occurred. This relationship is mainly represented as a logical formula using combinations of predicates of logical operators such as AND/OR connectives.

At the first glance, we observed that the bulk of the work done in this category was restricted to security field. This is because it is useful to model and analyze complex attack scenarios from the sequence of individual events or steps that are a part of the same attack.

One of the main advantages of sequential-based methods is that they are scalable; they can potentially uncover the causal relationship between alerts, and are not restricted to known attack scenarios. Furthermore, the correlation results are easy to understand and directly reflect the possible attack scenarios. However, the correlation results may contain a large number of false correlations, this being for two possible reasons: either the logical predicates are not well configured or the quality of the sensors alerts is not adequate.

Sequential correlation can be subdivided into several major categories, depending on how they represent the modeled scenarios: Pre/Post conditions, graphs, codebook, markov models, Bayesian networks, neural networks, and other techniques. We analyze them in detail in what follows.

4.3.2.1. Pre/Post conditions. In this category, the correlation process tries to find causal relationships among alerts through their pre and post conditions. The assumption here is that older alerts prepare for the later ones. If post conditions of an alert satisfy the pre-conditions of another alert, they are correlated. As an example, suppose we have an attack against sadmind, a remote administration tool. A scanning attack may discover UDP services vulnerable to certain buffer overflow attacks. Then the predicate UDPVulnerableToBOF (VictimIP, VictimPort) can be used to represent the attacker's discovery (i.e., the consequence of the attack) that the host having the IP address VictimIP runs a sadmind service at UDP port VictimPort and that the service is vulnerable to the bu?er over?ow attack.

Many proposals have been suggested here. We cite some of them: Zhaowen et al. [45] proposed RIAC, a real time alert correlation model to analyze and discover attack scenarios behind alerts. The assumption here states that the component attacks are usually not isolated, but related at different stages of the attacks, with the early ones preparing for the later ones. By using logical predicates, they introduce the notion of hyper-alerts to represent the prerequisite and the consequence of each type of alert. Each hyper-alert is a tuple (*fact, prerequisite, consequence*), where *fact* is the set of alerts attribute names, and *prerequisite* and *consequence* are two different sets, each one consisting of a logical combination of predicates expressed as mathematical conditions on variables contained in the set *fact*.

Ning et al. [46] also published a similar work. They presented TIAA, a toolkit for constructing attack scenarios by using predicates as the basic constructs to represent the prerequisites and (possible) consequences of attacks. Based on the prerequisites and consequences of different types of attacks, the proposed method correlates alerts by partially matching the consequences of some prior alerts with the prerequisites of some later ones.

Whereas TIAA allows partial satisfaction of prerequisites, JIGSAW [47] requires that all capabilities be satisfied. JIGSAW is a multistage correlation system. It uses capabilities and concepts to formulate the attack conditions. Capabilities are used to describe the information that the attacker must know to perform a certain attack, while concepts are used to model fragments of complex attacks.

Xiao et al.[48] proposed an alert correlation approach for alert fusion. It has two phases. First, using a fuzzy clustering algorithm, some alert subsets are created. Second, the method of correlating alerts based on prerequisites and consequences of attacks is adapted to be applied to these subsets.

Finally, Alserhani et al. [29] developed a rule based correlation language MARS, a Multi-stage Attack Recognition System. Unlike others, they add another two parameters for modeling attack consequences, i.e., vulnerability and extensional consequences. MARS is mainly based on the phenomena of "cause and effect". It has two main components: online and offline. The main purpose of the online component is to receive raw alerts and generate hyper-alerts. Then, multi-stage attack recognition is applied to correlate hyper-alerts based on rules provided by the offline component.

4.3.2.2. Graphs. Graph-based correlation techniques collect the sequential information of alerts by mapping them into graphs. The relationships between alerts can be represented as a directed acyclic graph where the set of nodes represent alerts and the edges connecting those nodes represent the temporal relationship of the connected alerts (nodes).

Alert correlation graphs have several advantages. First, graphs are quite easy to generate from whatever management models, especially from object–oriented system models with relations or associations between objects. Second, the operations permitted on graphs can be implemented in a robust manner, i.e., adding or deleting objects and dependencies are easy tasks. Third, graphs are naturally manageable in a distributed manner. Objects and dependencies can be added or deleted by different administrators independently.

However, the efficiency and accuracy of these techniques depend on a priori specification of how a failure

condition or alarm in one monitored system is related to failure conditions or alarms in other systems, and this requires an accurate knowledge of current dependencies among abstract and physical system components.

Gruschke [27] proposed an event correlation approach for fault localization in NMS based on dependency graphs. It consists of two components, nodes or objects, which reflect the managed objects in the system, and edges, that collect the functional dependences between the managed objects. Two objects are correlated if a failure in one of them causes a failure in the other. Their algorithm works as follows. First, each received event or alert is mapped to its corresponding object, which is signaled as faulty in the dependency graph. Second, a breadth–first search process is started from the initial dependent objects through the whole dependency graph looking for objects from which all (or many) initial objects depend on. These common dependent objects are forwarded as a condensed event. The algorithm assigns one of two states to the objects of the dependency graph: faulty and correct.

Instead of using breadth–first search process like in [27], Roschke et al. [49] used a Floyd–Warshall algorithm to find all the shortest paths in an attack graph to identify multiple attack scenarios. In this graph, each node represents a single attack step in a sequence. Each step may require a number of previous attack steps before it can be executed, represented by incoming edges and, on the other hand, may lead to several possible next steps, denoted by outgoing edges capable of creating only explicit correlations and identifying multiple attack scenarios.

Wang et al. [50] proposed a correlation approach for security alerts based on a Queue Graph (QG), which has the ability to hypothesize missing alerts and to predict future alerts. It only keeps in memory the latest alert matching for well-known exploits (host-bound vulnerabilities). The correlation between a new alert and those in-memory alerts is explicitly recorded, whereas the correlation with other alerts is implicitly represented using the temporal order between alerts.

Li and Lifang [51] used the concept of bipartite graph to represent the probability of dependency among events. They proposed a fault localization technique based on this representation, with the ability to reduce the fault localization time by using an Incremental Hypothesis Updating (IHU) algorithm.

4.3.2.3. Codebook. Codebook techniques encode the relationship between network faults and their symptoms by creating a matrix of problem codes that represent the dependency between observable symptoms and the underlying problems [52]. All alerts are first grouped into alert vectors. Then, they are matched to problem signatures and stored in a so-called codebook. The codebook is basically a matrix representation; events/alerts are represented as rows, and the symptoms of the problems as columns. The matrix contains binary digits (either 0 or 1). The value of 1 at a position in the matrix indicates a cause-effect relationship between a problem and a symptom. In other words, a one in the matrix denotes the appearance of a particular symptom, and a zero denotes that the symptom has not been observed. Distinction among problems is measured by the Hamming distance between their codes.

Codebook correlation techniques are efficient in the detection of network problems in terms of speed and accuracy, because they are performed only once to detect the root causes. However, they are not suitable for dynamic networks, because any change in the network topology requires regenerating the codebook, which is a time consuming process. Furthermore, they mainly depend on expert knowledge to construct the codebook matrix.

Klinger et al. [28] described a novel approach to event correlation in networks based on these coding techniques. According to their claims, their approach tries to solve some performance issues that faced existing codebook techniques while dealing with high rates of symptom losses and false alarms. In order to do that, they reduced the size of the codebook to contain a smaller set of symptoms capable of accomplishing a desired level of distinction among problems. The reduction of the codebook is based on a generalization of the Hamming distance. They define two metrics to calculate the distance. One is used for deterministic correlation, where the codebook matrix contains either 0 or 1, and the other for probabilistic correlation where the codebook matrix contains weights in the range [0,1], representing the probability of having a relationship between symptoms and problems.

4.3.2.4. Markov models. A Markov model is a stochastic production model composed of discrete states and a matrix of state transition probabilities. The events in this model are assumed to follow the Markov property, by which the next state only depends on the current state and not on the sequence of events that preceded it. In addition, every state has a vector of observable symbol probabilities. In the definition of a Markov model, the transition probabilities among states and the initial states probabilities should be defined. These could be statically defined, although they are typically trained from a dataset. Once that a model is defined and their associated probabilities obtained by training the model, a sequence of events can be evaluated, thus obtaining a probability. This probability is formerly compared to a threshold value to decide if a correlation is present or not. Hidden Markov Models (HMMs) are an important variant. Here, the sequence of states is not observable.

Most of the implemented alert correlation techniques of this category are focused on the security field and concentrate on HMM. Ourston et al. [53] claimed that HMMs are particularly useful and well-suited to address the multi-step attack problem through its prerequisites when there is an order for the actions constituting the attack (that is, for the case where one action must precede or follow another action in order to be effective).

In general, Markov-based techniques are especially well suited to address problems with a sequential nature. Yet, the main drawback of these models is the amount of data needed for suitably training them and their dependence on tuning parameters, e.g., the detection threshold for deciding if an alert should be correlated or not.

Farhadi et al. [54] proposed an alert correlation system for intrusion detection that consists of two major

components. First, they introduced an Attack Scenario Extraction Algorithm (ASEA), which mines the stream of alerts and extracts the current attack scenario. The algorithm has the ability to combine both prior knowledge as well as statistical relationships. Second, they proposed a HMM-based correlation method to predict the next attack class of the intruder.

Zan et al. [55] used Hidden Markov Models to represent typical attack scenarios, and designed a complete framework named HMM-AIP. It is composed of an online tracking and prediction module and an offline model-training module. They also presented a novel and effective tracking and predicting attack intention algorithm.

Zhicai and Yongxiang [56] proposed a Hidden Markov Model for detecting attacks. They firstly classify the warning events into different types. Then, the sequences of warning event types from different network monitors are correlated and their inherent relationship is mined to detect the type of network attacks and to forecast their threat severity.

4.3.2.5. Bayesian networks. Bayesian networks models (BNs), also known as belief networks (or Bayes nets for short), are one of the most powerful probabilistic graphical (GMs). These graphical structures are used to represent knowledge about an uncertain domain. Bayesian networks are mainly specified by two components: (*i*) A graphical component composed of a directed acyclic graph (DAG) where vertices represent events and edges represent relations between events and (*ii*) a numerical component consisting of a quantification of different links in the DAG by a conditional probability distribution of each node in the context of its parents. In particular, each node in the graph represents a random variable, while the edges between the nodes represent probabilistic dependencies among the corresponding random variables. A Bayesian network consists of several parameters, i.e., prior probability of parent node's states and a set of conditional probability tables (CPTs) associated with child nodes. CPT encodes the prior knowledge between child node and its parent node.

In the alert correlation problem, the probabilistic relationships among a large number of alerts are represented in order to work out a probabilistic inference from them. Given certain symptoms (received as alerts), a Bayesian network can be used to compute the probability that a specific problem have been happened.

Bayesian networks give several advantages when applied to solve the alert correlation problem. First, the speed of correlation is high. Second, they can incorporate prior knowledge and expertise by populating the CPTs. Third, they are also convenient to introduce partial evidence and find the probability of unobserved variables. Fourth, they are also capable of being adapted to new evidence and knowledge by updates through network propagation. Finally, the correlation output is a probability, rather than a binary result from a logical combination. However, this method needs a large numbers of training events for obtaining the prior probabilities and the correlation relies on experts' knowledge. Furthermore, a probabilistic inference in a Bayesian network is NP-hard, i.e., efficient solutions for large networks are difficult to implement in practice.

Steinder and Sethi [57] applied Bayesian reasoning techniques to identify multiple simultaneous faults. Their system has the ability to deal with false positive, lost, and spurious symptoms in complex communication systems. To isolate multiple faults, they applied two Bayesian inference algorithms that calculate belief-updating and most-probable-explanation queries in singly connected belief networks to perform fault localization in belief networks with loops. To deal with false positive and spurious symptoms, they proposed a heuristic that applies the belief-updating algorithm to perform event-driven diagnosis, and based on the results of belief updating, they calculated the explanation hypothesis.

Qin [30] proposed a probabilistic correlation model based on Bayesian mechanisms to correlate alerts and identify the related alerts if they conform to these three properties: (*i*) they have a cause-effect relationship; (*ii*) they have a sequential relationship, what implies a time constraint between a causal alert and an effect alert; and (*iii*) there exists a high statistical one-way dependence from the effect alert to the causal alert.

Marchetti et al. [58] proposed a pseudo-Bayesian correlation algorithm, which aims to highlight correlations among intrusion alerts that belong to the same multistep attack scenario. The algorithm consists of two steps. First, a pseudo-Bayesian probability is used to determine the likelihood of any two alerts for being correlated, which is converted into a weighted graph in which the nodes represent alerts, and the vertexes express the likelihood of two connected alerts to be correlated. Second, a dynamic threshold algorithm is used to prune the correlation graph by removing the vertexes having a relatively low weight.

Harahap et al. [59] proposed a failure prediction method to solve the network problem in network management systems (NMS) by making a prediction of failure based on network-data behavior. The prediction is represented by a conditional probability generated by the Bayesian network.

4.3.2.6. Neural networks. An artificial neural network (ANN) is composed of a number of interconnected processing elements (neurons) working jointly to solve specific problems. The neurons are interconnected to each other's according to a model inspired by the neural system existing in the human brain. Each neuron is considered as a simple autonomous processing unit, provided with local memory and unidirectional channels for the communication with other neurons. They are typically used to model complex relationships or to find patterns in data where non-linear dependency exists between inputs and outputs.

The most important issue in ANN is the learning phase, which can be accomplished by continuously adjusting the inter-neuron connection strengths (weights) until the overall network yields the desired results for the observations in the training set. There are two training methods, unsupervised training, where hidden neurons find an optimum operating point by themselves, without external in?uence, or supervised training, which requires that the network be given sample input and output patterns to

learn. The learning process is based on the iteration over the training set until a satisfactory optimum operating point or a predefined threshold is reached.

There are several advantages in using an ANN based approach. First, ANN can be made tolerant against noise in the input. Second, they have better properties than other techniques to generalize the results. This means that a trained network could classify data from the same class as the learning data that it has never seen before. Third, ANNs can acquire knowledge straight from the data without the need for a human expert to build up sets of domain rules and facts. Fourth, once trained, ANNs can be very fast, accurate and have high precision for near real-time applications. And finally, while feeding data during the learning phase, ANNs may use a type of dimensionality reduction that allows to input large amounts of information without efficiency bottlenecks. Yet, they share some weaknesses. The training process to tune its weights may take long sessions. Moreover, there are not particular rules to guide the selection of the number of layers and the number of neurons in each layer; hence, a trial and error process should be performed during the training period until the network finally stabilizes.

ANN has been used to solve the alert correlation problem in several approaches. As an example, in [60], Zhu and others proposed an alert correlation technique to discover attack strategies. The proposed approach is based on two different approaches, namely, Multilayer Perceptron (MLP) and Support Vector Machine (SVM). In addition, a correlation system based on a knowledge representation scheme, called Alert Correlation Matrix (ACM), is used to store correlation strengths of any two types of alerts. ACM is updated in the training process, and the information (correlation strength) is then used for extracting high level attack strategies.

4.3.2.7. Others. Besides the major correlation methods cited above, we have found some research efforts suggesting alternative methods like data mining, context-free grammars, fuzzy logic and others. For example, Al-Mamory and Zhang [61] proposed an alert post-processing and correlation method for the detection of multi-step intrusions. They called it Alerts Parser. In this method, alerts are treated as tokens, and a modified version of the Left-to-right (LR) parser algorithm is used to generate parse trees representing the scenario in the alerts. An attribute context-free grammar (ACF-grammar) is used for representing the multi-step attacks.

4.3.3. Case-based methods

Case-based correlation methods rely on the existence of a knowledge-base system used to represent well-defined scenarios. From this information, mining methods looking for specific patterns are designed. Many correlation techniques of this type have been implemented, most of them trying to correlate alerts based on known scenario templates. These templates are expressed either by human intervention using expert rules or by correlation languages, or inferred by using machine learning or data mining techniques.

When a problem is successfully solved, the solution (or its parts) is stored in a knowledge base, called case base. When a new case is raised, the system searches the cases database for the most similar cases having the same symptoms. The main two questions here are: which are the key attributes of a case? And which attributes will be used to index and access a case? To answer these questions, several case matching algorithms have been implemented such as: Nearest neighbor, Inductive, and Knowledge-based indexing. When a matching case if found, its associated solution is retrieved and used to suggest solutions to the current problem. If it is successfully solved, this solution or certain parts from it that are likely to be useful in the future are stored. When an attempt to solve a problem fails, then the reason for the failure is identified and 'remembered' in order to avoid a recurrence of such a mistake. Therefore, case-based methods use to keep updating the database with the new observed scenarios through some kind of inference mechanism or expert intervention.

Case-based correlation techniques are efficient for solving well-known problems specifying a complete action plan or previously observed scenarios. Therefore, these approaches can help network experts to discover all possible scenarios including potential solutions. However, it is not easy sometimes to exhaustively list all scenario templates and build a database containing a comprehensive set of problems solutions. In addition, time inefficiency may make them unusable in real-time alarm correlation.

The existing solutions can be grouped in two main categories: expert based and inferred knowledge.

4.3.3.1. Expert based. Expert based techniques build the knowledge database by human intervention. This knowledge is formulated either by using expert rules or predefined scenarios. They tend to imitate knowledge of a human, which may be either resulting from experience, or from understanding the system behavior from its principles. One of the main difficulties of this approach is the scalability, because updating the knowledge-base according to the evolution of a system is a problem which has to be taken into account when components are subject to frequent changes (topological or functional).

There are two main possibilities to build the database in this approach, which are explained in what follows.

4.3.3.2. Expert rules. Expert rules or rule-based systems are one of the most dominant categories among alert correlation techniques. They have been introduced by many researchers and mostly applied in various commercial correlation systems. This approach develops the knowledge as conditional, `if-then` rules. These sets of rules are matched to events when they come in. Each rule consists of two expressions, which are well-formed formulas of predicate calculus linked by an implication connective (=>). The left side of each rule contains a prerequisite which must be satisfied, so that the rule is applicable. The right side describes the action to be executed if the rule is applied. There are two types of rule matching, i.e., exact and partial matching. In exact rule matching, the whole left hand side of the rule must be matched before determining which action should be triggered, while in

partial matching, the action is determined if some, but not all, of these conditions are fulfilled.

Rule-based approaches are appropriate for systems whose configuration is rarely altered. Moreover, they are simpler, modularized, and easy to maintain when applied to small systems. However, they have some weaknesses. First, the high cost of implementation and adaptation to changes make it difficult to apply these strategies to large systems (with potentially a large amount of alerts). Second, they are inefficient in dealing with inaccurate information or unseen problems.

Cronk et al. [62] divided a rule-based system into three levels: (*i*) control level, as an inference engine that determines how the rules are applied from the knowledge base to solve a given problem; *(ii)* knowledge level, which is a database repository for all knowledge about the system in the form of declarative knowledge; and *(iii)* a data level, which is a global database which contains the data about problems being dealt with. Rules are expressed in the form of `IF condition THEN action`.

Wing et al. [63] proposed a new method to organize the system rules by distinguishing between core and customized knowledge. According to their judge, the customized knowledge allows to accurately isolate a fault from the selected group of system entities. The correlation rules are organized as composite event definitions, as another work suggested by Jector et al. [64]. In this approach, unlike others, the distinction is made between primitive events, i.e., alarms and composite events.

4.3.3.3. Pre-defined scenarios. Pre-defined scenarios methods are similar to rule-based methods. Both acquire manual knowledge. However, they use two different strategies on how to represent this knowledge. As we mentioned above, rule-based approaches use expert generic rules to represent this knowledge. Here, a specific language is used to implement well-defined scenarios. A huge number of correlation languages have been proposed, especially in the security field, related to the specification of attack scenarios. To build these attack sequences, a straightforward way is to first predefine some attack scenario templates. This approach starts with the hypothesis that alerts belonging to one problem have similar attribute values (e.g., source IP address). If various alerts contribute to the construction of a predefined scenario, they should be correlated.

The advantage of this method is that the correlation result is easy to understand and can help security officers to discover all scenarios variants. However, it is not easy sometimes to exhaustively list all attack sequence templates and consequently they fail to be generic. Another limitation of these methods is that novel attack patterns or obfuscation methods created by attackers may cause that the corresponding attack scenarios are not recognized.

Cuppens and Ortalo [65] presented an attack description language called LAMBDA, used to describe with logical expressions the effects and conditions of an attack starting from the variable state of a victim system. In LAMBDA, an attack is specified using five fields: (*i*) Attack pre-condition: a logical condition that specifies the conditions to be fulfilled for the success of the attack. (*ii*) Attack post-condition: a logical condition that specifies the effect of the attack when it succeeds. (*iii*) Attack scenario: the combination of events that the intruder performs when executing an attack. (*iv*) Detection scenario: the combination of events which are necessary to detect an occurrence of the attack. (*v*) Verification scenario: A combination of events to be launched to check if the attack has succeeded. In LAMBDA, several attack specifications can be merged automatically by comparing pre and post conditions. This enables tracing the progress of an attack within a system. Using an attack history, the system can estimate what actions are to be performed by the attacker.

Unlike in [65], where five fields are used to specify an attack, in the State Transition Analysis Technique (STAT) proposed by Kemmer and Vinga [66], an attack has an initial state and at least one ending state. States are characterized by means of assertions, which are predicates on some system security aspects. Attacks modeled using STAT techniques are represented using the STATL language, which was proposed by the same authors in [67]. STATL is an extensible state/transition based attack description language designed to support intrusion detection scenarios. This language allows describing computer penetrations as sequences of actions that an attacker performs to compromise a computer system. The high-level alert patterns and alert correlation rules are organized as expert knowledge.

Liu et al. [68] proposed an alert correlation model based in the use of finite automata for the specification of the scenarios. In this model, they generated three kinds of high-level view of attacks: process-critical scenarios, attacker-critical scenarios and victim-critical scenarios. In process-critical scenarios a non-deterministic finite automata is used to model the intrusion process that takes place between one attacker and one victim. In attacker-critical scenario, the scenario rebuilds the intrusion process that an attacker implemented towards the whole target network. And victim-critical scenarios rebuild the intrusion actions implemented towards a specific system. According to their judge, the model can generate scenarios that are much more directly-perceived.

Cheung et al. [69] proposed a Correlated Attack Modeling Language (CAML). It aims at modeling multistep attack scenarios by representing them as trees. Each scenario is subsequently divided into sub-goals or modules. A module specification consists of three sections, namely, activity, pre-condition, and post condition. To support event-driven inferences, the activity section is used to specify a list of events needed to trigger the module. After that, it identifies logical steps (sub-goals) in attack scenarios and specifies some relationships among these steps: temporal, attribute values, and prerequisites. Each module is linked to others by using pre-post conditions to recognize attack scenarios.

4.3.3.4. Inferred knowledge. Expert knowledge-based systems can be built by using inference methods with machine learning algorithms. Here, explicit symbolic classification rules are automatically constructed from some training cases. The classification rule learning task can be defined as follows: Given a set of training examples (alerts or meta-alerts for which the classification is known), find a set of classification rules that can be used for prediction or

classification of new instances, i.e., new incoming alerts or meta-alerts.

The main advantage of these methods is that there are no assumptions about the model that will be used in the correlation process, as it is really learned from the training instances. Yet, it is really difficult to find representative datasets for the training. Furthermore, a big issue is to make the models capable of generalizing the results for events not observed in the training dataset. Finally, the main drawback of these methods is the computational load implicit in the process, which usually makes them unsuitable for real time systems.

Some contributions have been done in this line, like the work done by Smith and others [70]. In this work, they suggested an alert correlation system based on unsupervised machine learning algorithms. It is implemented in two stages. First, alerts are grouped together such that each group forms one step of an attack. Second, the groups created at the first stage are combined in such a way that each combination of groups contains alerts for a complete attack process.

Some other researchers have used data mining techniques to automate the process of finding meaningful activities and interesting features from training datasets and build the knowledge base [71–74]. Data mining is a set of techniques and tools used for the non-trivial process of extracting and presenting implicit knowledge. Specifically, Katipally et al. [75] used data mining techniques to generate association rules and build predefined attack scenarios, which are used for predicting multistage attacks.

Sadoddin and Ghorabani [76] proposed a framework for real time alert correlation that incorporates two techniques: one for aggregating alerts into structured patterns, and other for incremental mining of frequent structured patterns. In the proposed framework they use time-sensitive statistical analysis to find the relationships between alerts. These are maintained in an efficient data structure and updated incrementally to reflect the latest trends of patterns.

4.4. Type of architecture

Alert correlation techniques can also be classified based on the type of architecture they use. Different architectures have been proposed in the literature to enable the effective aggregation and correlation of alerts. We classify these architectures as centralized, distributed and hierarchical.

4.4.1. Centralized

In centralized alert correlation approaches, the data collection is done locally by the different network agents and then reported as alerts to a central management server where the correlation analysis is done. Correlation algorithms in this architecture are simpler, easier to implement and can correlate overall alerts quickly. Yet, their scalability is limited and their main drawback is that there exists a single point of failure. [9].

4.4.2. Distributed

During the past few years, many researchers have concluded that the alert correlation process should be carried out in a distributed fashion. According to their claims, this need for a distributed alert correlation architecture, or completely distributed architecture as mentioned in the literature, emerged from the fact that current and perspective communication networks increase in their size, complexity, speed, and the level of heterogeneity. For this reason, using central correlation architectures for processing large volumes of correlation information would be computationally prohibiting and infeasible. Therefore, distributed alert correlation techniques that would allow the management correlation agents to reach the solution collectively are necessary.

In these systems, alerts or high-level meta-alerts are exchanged, aggregated, and correlated in a completely cooperative and distributed fashion. All agents are equally weighted, and there are no hierarchic ranks among them. Besides data collection, a partial correlation is done locally by every agent. All agents keep communicating to each other's using some form of distributed protocols, e.g., peer-to-peer protocols (P2P) or others. Information from that partial correlation made at certain agents could be used by others for optimizing their own correlation. To do that, typically a central correlation unit is randomly selected amongst all agents. Each agent has the chance to be selected as a central unit. When the central unit has collapsed, another alert correlation unit can substitute it.

This architecture enhances prominently the scalability and the fault tolerance, when compared with centralized architecture, since the correlation process is distributed amongst several correlation agents. However, there are some issues that need to be solved. First, it consumes more bandwidth due to the information sharing. Second, there is no coherent and consolidated view of the whole system, because the computations are distributed among several entities. Third, load balancing is considered an important issue in this type of architecture, because some management correlation agents may be overloaded in comparison with others. Finally, the requirements and complexity of hardware and implementation are higher.

Mohamed and Basir [77] proposed a distributed alarm correlation and fault identification approach. They divided the network topology into disjoint management domains, and each management domain is assigned to a dedicated intelligent agent, which is responsible of monitoring and collecting alarms within its management domain. All agents use majority vote rule to determine the root cause of network malfunctioning.

Khatoun et al. [78] proposed a decentralized alert correlation technique to detect DDoS attacks based on P2P architecture, which correlates alerts produced by various intrusion detection systems and provides a high-level view of attempted intrusions. Each IDS supervises its sub-network. Each of these IDSs is a peer inserted in a P2P system that conform the global collaborative IDS. A Distributed Hash Table (DHT) is used to efficiently route resource information about the potential victims, and to share data about possible attacks among peers.

4.4.3. Hierarchical

Hierarchical architectures are also called hierarchical distributed architectures because they embed a form of

distributed architecture in its design. Here, the correlation process is performed in a hierarchical and a cumulative way. Unlike in a distributed architecture, the management agents in the hierarchical model are distributed across different levels. The management agents are located and partitioned in multiple groups, according to different features such as geography, administrative control, and others. Within each group there is a horizontal communication among peer agents, and vertical communication among agents in different levels. The output of their correlation results are passed upward where higher level dedicated correlation units are found. They correlate alerts from both their own level and their children nodes. Then, the correlated alerts are passed upward to a higher level for sharing and further analysis. This process continues until reaching the root, where a central correlation unit collects all the correlation views which were done in lower levels to build a global correlation picture.

Hierarchical architectures are somehow a form of distributed architectures, so they share the same advantages with the latter, regarding scalability and fault tolerance. Yet, they outperform the distributed architectures in terms of coordination and communication costs, especially when very large systems are being managed. Furthermore, their deployment is simpler. On the other hand, in these architectures, the correlation units of the higher levels in the hierarchy still limit the scalability of the correlation system, and their failure can stop the function of their whole sub-tree.

Tian et al. [79] proposed a hierarchical alert correlation algorithm for intrusion alerts. It consists of three stages. First, IDS sensor data are aggregated. Second, some local correlation units correlate alerts and build the local correlation graph. Third, the centralized alert correlation unit constructs the global correlation graph via the local correlation results.

4.5. Comparative study of alert correlation techniques

As shown before, researchers and vendors have proposed and used many different design paradigms for implementing alert correlation techniques. To our knowledge, there is still no comprehensive comparative study of alert correlation techniques except some efforts done by few researchers. However, these works only consider typically one application area, e.g., security field, and cover only a subset of the literature related to their work. According to our opinion, there are some reasons for that: first, as a consequence of using a wide diversity of methods, a comparative analysis of alert correlation techniques becomes a difficult, tedious and sometimes error prone task. This is because these correlation methods have their own philosophy of dealing with the alert correlation problem. They have their own capabilities, strengths and weaknesses. As a consequence, some correlation techniques perform well in some situations and others outperforms in other situation. Second, researchers use different methods for validating their ideas. There is no standard performance strategy or benchmarks for evaluating these techniques. Third, the evaluations are applied on different datasets; mostly build ad hoc for the considered method. At the time of writing this paper, we did not find a publicly available dataset explicitly designed for testing alert correlation algorithms. Finally, many different authors use the same terminology to refer to different alert correlation operations. Therefore, some authors talk about alarm correlation when referring to the clustering and fusion process, while others call correlation to the process of creating new scenarios. This issue makes the task of building a comparative study a complex and possibly infeasible task. For this reason, the few comparative study proposals that we have found have either limited the study to a number of papers or are biased to a specific application. In the following, we describe these efforts.

Yusof et al. [80] suggested six capability criteria for evaluating alert correlation techniques. Their work focused on the security field and, specifically, on Intrusion Detection Systems (IDSs). The proposed capabilities are: alert reduction, alert clustering, identification of multistep attacks, reduction of false alerts, detection of known attacks and detection of unknown attacks. They first classify alert correlation techniques into four groups: (*i*) Similarity-based, (*ii*) Pre-defined Attack Scenarios, (*iii*) Pre-requisites and consequences of individual attacks, and (*iv*) Statistical Causality. Then, they made a relationship with the capability measures. Their conclusions were: First, similarity-based alert correlation techniques have the ability to do alert reduction, alert clustering, and detection of known attacks; whereas they failed to reduce false positives and detect multi-step and unknown attacks. Second, predefined attack scenarios correlation techniques have the same results as similarity-based techniques. Third, prerequisites and consequences of individual attack correlation techniques have the ability to do alert reduction, alert clustering, false positive reduction, and detection of multi-step and known attacks. Finally, statistical causality techniques are the only category that has the ability to detect known and unknown attack scenarios, besides alert reduction and clustering. However, they failed to reduce false positives and detect multi-step attacks. In summary, the overall conclusion that they extracted from this analysis is that further improvements should be done on the process of detecting known, unknown and multi-step attacks, as these capability criteria shall overcome large number of false alert problem. Table 2 below summarizes this analysis.

Siraj et al. [81] suggested a comparative study covering only the most representative work in alert correlation area related to the security field. Since the correlation process is a multi-step complex task that consists of many operations, they decided to choose five of them to make the comparison. Their selected operations are: normalization, verification, aggregation, correlation, and attack scenario analysis. Their conclusions were: First, statistical and probability based techniques suggested in the discussed papers cover all the operations and, for this reason, they are considered as the top category of alert correlation techniques. Second, statistical and some of rule based techniques cover four operations and are thus considered as the next top most categories. Others such as case-based, probabilistic, and some rule based techniques only cover three operations and are then considered as the third top most category. The category that lies at the end of the ranking

Table 2
Alert correlation technique versus proposed capability criteria (capable = √, incapable = X) (taken from Yusof et al. [80]).

Technique name	Alert reduction	Alert clustering	Multi-step attack	Reduction false positives	Detection known attacks	Detection unknown attacks
Similarity-based	√	√	x	x	√	x
Pre-defined attack scenarios	√	√	x	x	√	x
Pre-requisite and consequences of individual attacks	√	√	√	x	√	x
Statistical Causality	√	√	x	x	√	√

Table 3
Comparative analysis of existing alert correlation techniques (capable = √, incapable = X) (taken from Siraj et al. [81]).

Techniques	Operations				
	Normalization	Verification	Aggregation	Correlation	Attack scenario analysis
Rule-based (case 1)	√	x	√	√	x
Rule-based (case 2)	x	√	√	√	√
Rule-based (case 3)	√	√	√	√	x
Rule-based (case 4)	x	√	√	√	x
Rule-based (case 5)	x	x	√	√	x
Model-based	x	x	√	√	x
Statistical-based	√	√	√	√	x
Probabilistic-based	√	x	x	√	√
Probabilistic-based & Case-based	x	x	√	√	√
Statistical and probability-based	√	√	√	√	√

is the model-based techniques. Table 3 below summarizes these conclusions in more detail.

Sadoddin et al. [22] classified the alert correlation techniques in Intrusion Detection Systems into three categories. They are knowledge-base methods (which includes rule-based and scenario-based techniques), statistical-based and temporal-based techniques. They concluded that the last two approaches are capable of correlating alerts related to unknown attacks, while scenario-based and rule-based correlation methods are capable to detect known attack scenarios, as they are solely dependent on predefined attack scenarios and rules in the knowledge-base of the system, respectively. Knowledge-based correlation methods have higher accuracy than statistical and temporal techniques, which are at the same time very time consuming. In addition, statistical and temporal correlation methods fail to discover causality relation between noisy alerts or when there are deliberate delays planned by the attacker among the low-level alerts.

Abouabdalla et al. [82] summarized several research efforts which were done in the false positive reduction task within the Intrusion Detection Systems (IDS) area. They concluded that not all researchers deal with the false positives reduction issue in the same way. Some of them work on the IDS level and tries to improve the detection efficiency and, as a consequence, this lead to a reduction of the false alerts and an increment of the detection accuracy at the sensor level. Others work at a higher level than the IDS and study other important data like the actual behavior of network traffic and firewall and router logs. They concluded that, with all the benefits obtained from the proposed methods, there is still not a perfect method and some weak points still remain. For instance, when similarity-based techniques are used to remove false positives, the analyst does not discover the actual reasons of IDSs having triggered these alerts. Therefore, this will be only one step further to reduce the false positives alerts. On the other hand, scenario-based techniques are also inefficient in reducing false positives, mainly because this process is enforced to be done in real time so that the response will be more effective. At last, they observed that different authors use the same false positive reduction terminology to refer to different concepts. Some authors refer to it as data mining and clustering, while others mentioned false positive reduction to the process of correlation. As a consequence, some form of standardization is needed to clarify false positive reduction terminology.

To conclude this section, we observe that, up to now, the alert correlation process is still an active area of research in both NMS and security fields. Despite the big amount of efforts done in this field, there is not a clear agreement between researchers and vendors on how to formulate efficient solutions and what the performance criteria that determine their effectiveness are. This is the reason why it is not possible to find any generic architectures and benchmarks for evaluating them.

From the above comparative studies, we can summarize the following results.

1. *Similarity-based techniques*: Most techniques belonging to this category share some characteristics:
   (a) They are simple, i.e., most of similarity-based algorithms use simple mathematical functions to calculate distances between two alerts based on their features. Therefore, they have higher performance in terms of processing speed, and give results faster than others.
   (b) They are generic. All of these algorithms can be applied and implemented in a wide variety of tools. As we will describe later on in Section 6, we have check the existence of several tools and systems that utilize and apply them in their designs.

(c) These techniques are performing well when applied to some correlation operations such as alert reduction, clustering, aggregation and pattern matching, mainly because they operate on the attribute level and will provide valuable results.
(d) They are scalable and can give good results regardless the size of the dataset, as they do not rely on prior knowledge.
(e) The detection accuracy is low. When applied to the NMS field for discovering root causes or to the security field to discover attack scenarios, they did not give a high detection rate. As a consequence, many researchers use them as the first step of the correlation process before applying their own methods.
(f) They can detect very simple known attacks, but usually they fail to detect false positives, multi-step and unknown attack scenarios.

2. *Sequential-based techniques:* Most techniques belonging to this category share these characteristics:
(a) The majority of these techniques use complex correlation algorithms to discover root causes or attack scenarios.
(b) Sometimes, the algorithms are generic, because they work on data online. Typically, they are not biased to specific scenarios or problems.
(c) They are scalable and can operate with unseen problems.
(d) The detection accuracy is high. They can detect known problems accurately and precisely.
(e) These techniques perform better in security than in NMS, except codebook based ones, as they are mainly designed to discover root causes in NMS.
(f) They can detect false positives and unknown attacks.

3. *Case-based techniques*: Most techniques belonging to this category share these characteristics.
(a) They perform well in static environments, where the system behaviors and topological information remain unchanged; as they build the correlation based on previous cases and predefined scenarios.
(b) They have higher accuracy for detecting well-known problems and have the ability to specify a complete action plan.
(c) The scalability is a big issue, because it is inversely proportional to the accuracy. They do not provide answers for new problems.
(d) They do not perform well when applied to real time systems, since in these types of systems the response should be very fast. These techniques need considerable time to retrieve the most similar case, especially when the database is large.
(e) They fail to reduce false positives and detect multi-step and unknown attacks.

4.6. Alert correlation challenges

From the above studies about alert correlation techniques, we conclude with some challenges and research issues that still remain open. In our opinion, the alert correlation problem currently presents three main challenges:

1. Assessment. There is a lack of standard strategies for evaluating the performance of alert correlation techniques. Indeed, it is necessary to dispose of standard datasets and validation methods such as benchmarks that make researchers able to do this task efficiently and accurately using some performance criterions.
2. Architecture and scalability. In spite of the complexity and heterogeneity of the communication networks, we surprisingly find only very few proposals utilizing distributed architectures to make a distributed alert correlation. A large number of proposed solutions for alert correlation problem are based on a centralized architecture, which leads to scalability problems. There is a need of works in this direction.
3. Detection. Existing alert correlation techniques do not deal with false positives and unseen problems efficiently. As a consequence, and according to our analysis, these two issues are the top two limitations of all existing alert correlation techniques. Therefore, new proposals in this field should find new strategies for dealing with this problem.

5. Correlation process model

Many different authors have described in their works the process of alert correlation. While some of them normally focus on the different stages of this process [18,23,25,31,48], others are interested on the different methods used for the correlation itself. We claim that previous works does not completely aim at providing a comprehensive enough view of the whole process of alert correlation, mainly because they present a description of this process as an introduction for describing a specific approach and, in many cases, their point of view is biased towards the techniques that they propose. Here, we propose a correlation model which is intended to survey all the stages, techniques and methodologies suggested in the state of the art.

The proposed model is composed of four modules, as shown in Fig. 2:

- *Alert preprocessing module*: It accepts raw alerts and converts them into a unified data format understandable by other modules.
- *Alert reduction module*: This module is intended to filter and validate alerts.
- *Alert correlation module*: It aggregates similar alerts and converts them into a higher level view.
- *Alert prioritization module*: It analyzes the severity of alerts on the system and provides a classification methodology.

The correlation model proposed in this paper has two main features. First, it tries to generalize the model by selecting the most important correlation modules generally accepted by the research community in several applications.

Note that the scope of the correlation model proposed here is generic, which means that it tries to cover all aspects of alert correlation. Nevertheless, it implies that it

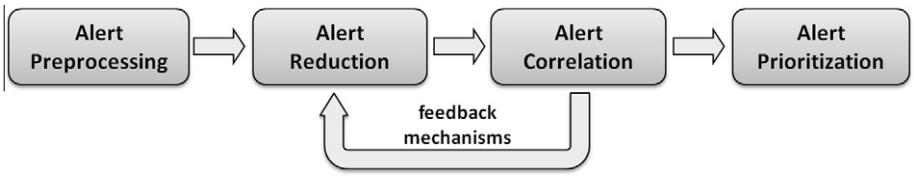

Fig. 2. Proposed correlation process model.

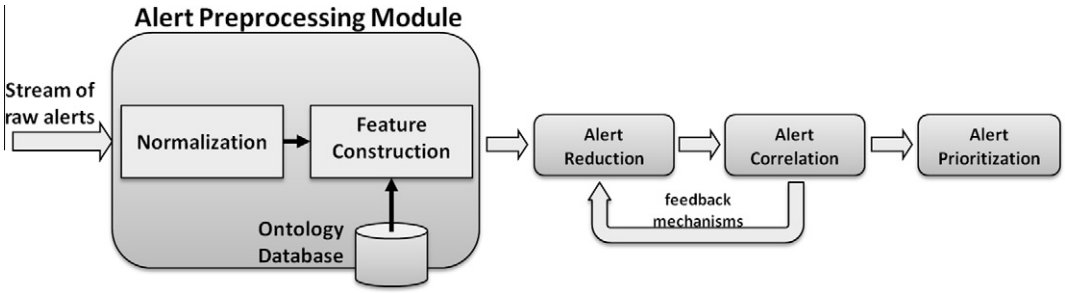

Fig. 3. Architecture of the alert preprocessing module.

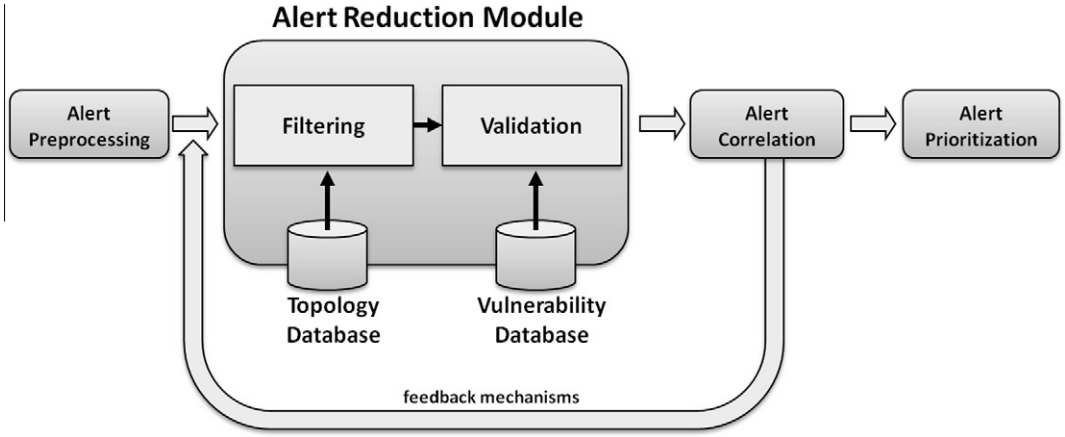

Fig. 4. Architecture of the alert reduction module.

is not mandatory for any existing implementation to consider all of them.

The second feature of our model is that the alert correlation process is not represented as a linear process, but it is iterative. Iterations are established between the alert reduction and the alert correlation modules, as detailed later on. This way, we consider that it is necessary to implement a feedback mechanism between these modules, aimed at refining their outputs.

The feedback mechanism is leveraged to enhance the alert correlation process in discovering root causes or malicious activities efficiently, due to the fact that, in many real problems, it is quite complicated to identify the causes of the network problem in a single iteration.

In order to effectively detect these types of problems, an especial type of alert is defined: *hyper-alert*. They are alerts resulting from a first aggregation and correlation process. Hyper-alerts are generated as outputs from the alert correlation module, and are supposed to be returned and fed back as inputs to the alert reduction module, where the correlation process repeats itself again.

This iterative nature for the model is well-suited for complex or hierarchical scenarios, in which multiple filtering and aggregation phases are required. This way, hyper-alerts can be considered as new alerts and filtered and aggregated consequently to produce new hyper-alerts that can possibly feed the system in new iterations. This mechanism can ease the procedures, simplify the required aggregation techniques and, subsequently, improve the results.

The convenience for this iterative process can be clarified by analyzing two examples taken from the security field. As a first case, consider a Distributed Denial of Service attack (DDoS) being targeted to a network segment or

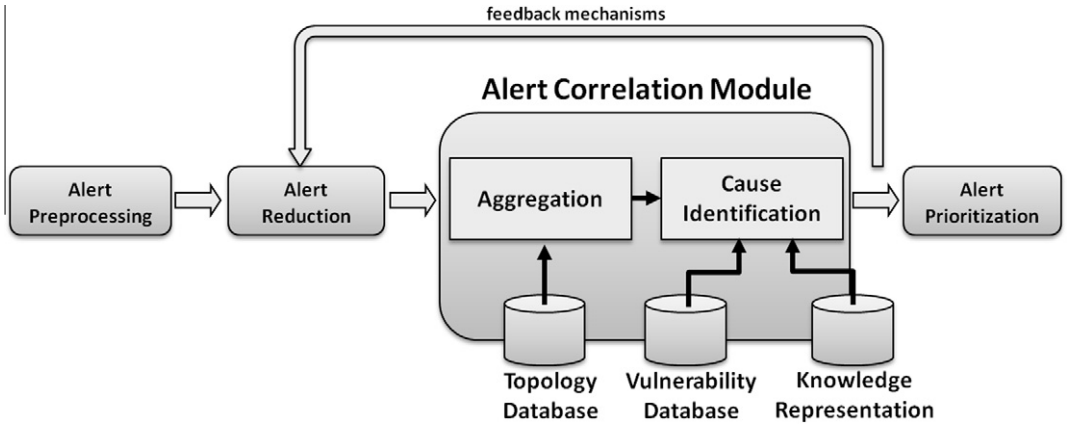

Fig. 5. Architecture of the alert correlation module.

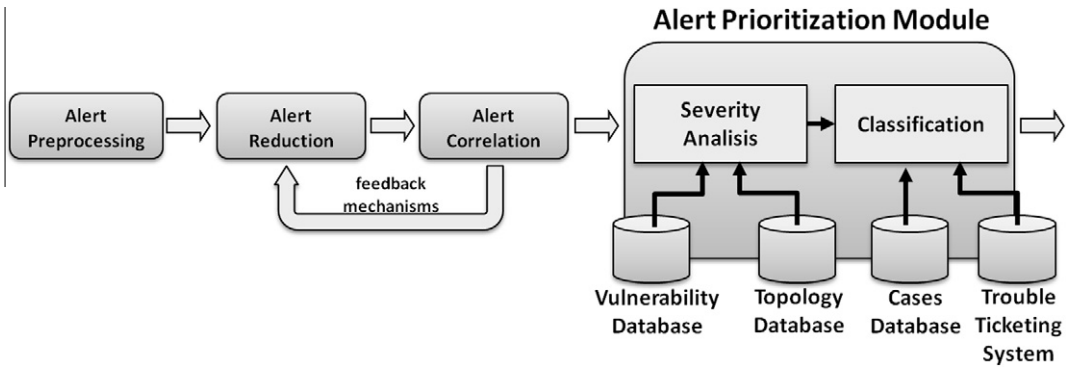

Fig. 6. Architecture of the alert prioritization module.

server. Here, many alerts may be generated at different border routers or links warning about an increase in the traffic. To ease the detection of this attack and to provide some insights of its impact, many steps of filtering and aggregation for the alerts can be considered, each of them targeted to a different dimension. For example, a first step would be targeted at generating hyper-alerts for the routers and links in a subnet level and then joining them at an operator level.

A second case to consider is a low-rate scan attack. This kind of attacks is characterized by the scanning of ports of multiple targets at a very low rate for each of the targets. That is, each host is scanned for a single port with a high period of scanning, while there are a lot of scans in a relatively short period of time, but targeted to many different hosts. In this scenario, one of the detection approaches should go through two different steps of filtering and aggregation for the generated alerts. One of the steps should focus on the generation of hyper-alerts for one of the dimensions of the problem, i.e., aggregation of many connection attempts from a single computer or subnet to many different hosts being monitored. For this, probably a previous filtering is advisable. After these hyper-alerts are generated, a second filtering and aggregation step should focus on the other dimension, i.e., the existence of multiple events of this type across a given period.

In the next subsections, we give a detailed description of these described modules and their internal phases. Furthermore, as a survey, we provide several significative research efforts proposed by different authors related to these phases.

5.1. Alert preprocessing

Currently, in order to provide a better network monitoring and give a global view of intrusion activities, most of the organizations use collaborative and heterogeneous monitoring systems from different vendors. Examples of such monitoring systems are network management systems (NMS), host intrusion detection system (HIDS), network intrusion detection system (NIDS), firewalls, antivirus systems, etc. These systems detect abnormality conditions of monitored networks by using different detection methods and, consequently, they generate alerts with different data formats. The alert preprocessing module implements the necessary processes that are used to clean and transform all raw alerts into a unified and integrated format in order to be understood by other modules.

Recently, Davis and Clark [83] presented a review of the state of the art on data preprocessing techniques used by anomaly-based network intrusion detection systems (NIDS). They divided the techniques into different groups

according to the network traffic features used for detection, e.g., packet header anomaly detection, protocol anomaly detection, content anomaly detection, etc. They concluded that these techniques are valid for NMS, but they are insufficient for the security field, since attackers use other methods such as web-based attacks and crafted application data which may need more complicated preprocessing operations.

As shown in Fig. 3 the preprocessing module can be mainly divided into two phases: normalization and feature construction, which are discussed below.

5.1.1. Normalization

A normalization phase is first executed to convert heterogeneous alerts from multiple sources into a standard format which is acceptable by other correlation modules.

Holub et al. [18] used Generic Log Adapters (GLAs) to convert the events from various logs to Common Base Event (CBE) format before using them in their proposed correlation engine for system monitoring and testing. GLA provides a command-line interface to convert application log files to the CBE format.

Another relevant example widely used in NIDS and implemented by the Internet Engineering Task Force (IETF) in cooperation with the intrusion detection working group (IDWG) is the Intrusion Detection Message Exchange Format (IDMEF) [15]. It is a kind of reporting language that uses an object-oriented representation to model the alert data generated by intrusion detection systems. Each alert is translated into a vector of attributes with the following contents: {`Alert ID`, `Sensor ID`, `Detection time`, `Source IP Address`, `SourcePort`, `Destination IP address`, `Destination port`, `Service Protocol`, `Alert Type`}. One of the main goals of the IDMEF model is to be able to express relationships between alerts, and exchange procedures for sharing information of interest to intrusion detection and response systems and to the management systems that may need to interact with them. A Document Type Definition (DTD) has been proposed to describe the IDMEF model data format, with an implementation that uses the Extensible Markup Language (XML).

5.1.2. Feature construction

Since some of the monitoring systems may omit some of the fields expected in the normalized format of the alerts (i.e., start time, end time), the feature construction phase aims to supply, as accurately as possible, the missing attributes in alerts and also to create additional features using some time-based statistical measures, which would have better discriminative ability than the initial features set.

This issue has been treated in many research contributions. To solve it, some authors have suggested the use of ontology databases consisting of attribute-value pairs for different type of nodes which provide the global information necessary for the feature construction process. For example, IDMEF [17] has strong features that make it able to do this function properly. For example, there are three different time classes defined in the IDMEF standard: `CreateTime` (the time at which the alert is created), `DetectTime` (the time at which the events producing an alert are detected), and the `AnalyzerTime` (the time at which the alert is sent by the IDS to the correlation system). By using different approximation functions, the missing time attributes are substituted, provided that network equipment clocks are synchronized by, for example, using Network Time Protocol (NTP) [84].

5.2. Alert reduction

As previously discussed, network monitoring and detection systems may overwhelm network operators when triggering a huge number of alerts, especially if 99% of them are redundant or there is a considerable number of false positives (normal events being predicted as abnormal) [85]. Being able to filter out and to validate a high percentage of those uninteresting alerts increases the accuracy of the correlation process and the efficiency as well.

As shown in Fig. 4, the alert reduction module is composed of two main phases: *Filtering*, where redundant, duplicated and uninteresting alerts are removed; and *validation*, aimed at verifying whether an alert is true or a false positive.

5.2.1. Filtering

The preprocessed alerts are fed as an inputs to the filtering phase, in which uninteresting and redundant alerts are filtered out. In the literature, we can find two types of filters: predefined and inferred filters. Predefined filters are really rules assigned by experts and updated manually. As an example, one of the components of the correlation engine proposed in [44] is a powerful filtering scheme called composite filtering, defined as a set of basic and composite filters organized in a logical tree. Filters are applied in depth-first search (DFS) order where a descendant filter is considered to be a refinement of the ascendant filter.

On the other hand, inferred filters are learned by some kind of inference method. They adapt the rules periodically or on demand by using filtering algorithms that have the ability to use some topology network information which is mainly stored in a topology database. Lin et al. [86] proposed an inferred alarm filtering system capable of classifying alarms with high confidence. Chyssler et al. [10] proposed an automatic filter detector based on Naïve Bayesian (NB) learning, which is designed to filter syslog records by using a trained classifier that looks at the words contained in the alerts.

5.2.2. Validation

Alert validation is one of the most relevant and sensitive tasks in the correlation process. The main function of the validation phase is to verify the validity of each alert according to its effect on the overall monitored system. In order to actively and accurately distinguish true alerts from false positives, the validation process uses several sources of information and tries to find the logical connections between them. To do that, it extensively makes a deep comparison between all the available sources of information, and then calculates the value of the correlation between alerts and the monitored system. Here, the main sources of information are the alerts themselves,

which contain useful information about the operating systems, network services and others, and a *vulnerabilities database*, which stores known exploits and system vulnerabilities information, together with the corresponding security solutions.

The validation process is performed by using passive or active techniques. Passive techniques carry out the checking of the validity of an alert by using prior information stored in a vulnerabilities database. The advantage of these techniques is that it is not necessary to perform additional network data collection, thus not interfering with the normal operation of the network. Their main disadvantage is that it is not easy to promptly update the base; there is a potential difference between the state stored in it and the actual monitored status of the network.

Active techniques update the vulnerabilities database automatically by using several real time scanning tools that actively monitor the whole network and update the status information stored in this database. The database contains updated information that gives a correct view about the current status of the network. These techniques have still some drawbacks when compared to static techniques: they may generate some extra alerts, and they also consume more bandwidth and network resources. Furthermore, the scanning process could make some services crash.

Note that this phase normally involves extensive analysis and processing, implying high load procedures. Thus, for performance issues, the *validation* phase is located after the *filtering* phase.

Some authors like Xiao et al. [48] adopted a compromise solution that collect together the advantages of both passive and active techniques. Firstly, they collect the configuration information of network resources, and then they scan periodically the whole network to update the database.

5.3. Alert correlation

The alert correlation module is at the heart of the whole correlation process. It receives alerts from the alert reduction module and tries to find out the logical relationships between them, in order to give a higher level and coherent view about the status of the monitored network and discover root causes. In Fig. 5 we show a possible structure of the alert correlation module. Here, the process is divided into two phases; the *aggregation* phase, in which alerts that share the same root cause are merged, and the *cause identification* phase, in which a higher level processing is handled to generate *hyper-alerts*, i.e., alerts which have higher level and richer information about the root causes of a problem.

5.3.1. Aggregation

As said, the target of the alert aggregation phase is to merge multiple alerts that share a same root cause. The aggregation phase receives alerts from the alert reduction module and tries to compare them with previously existent or aggregated alerts. The meaning of the new aggregated alerts is an induced generalization of those used for the aggregation.

Alerts in a given group are supposed to be similar to each other and dissimilar to each of other groups. Sometimes, it is necessary for the aggregation process to obtain topological information, which is mainly stored as mentioned above in a topology database.

As explained in Section 4.3, a huge number of proposed clustering techniques for aggregation have been contributed. These techniques may need to use additional information, e.g., a topology database.

5.3.2. Cause identification

The cause identification phase receives the aggregated alerts generated by the aggregation process and tries to discover and recognize the logical relationships between them. Although this phase may not have a significant effect in reducing the number of alerts, the main role of the cause identification phase is to convert these aggregated alerts into hyper-alerts, which have more meaningful and semantic contents when compared with the previous ones. In order to correlate aggregated alerts and produce *hyper-alerts*, the *cause identification* phase may use several sources of information like a cases *database* or knowledge representation (see Section 3).

The output of the alert correlation module has two main destinations: one is the input to the next module, i.e., *alert prioritization* for further processing; the other is entering in a cyclic feedback process and enters again in the alert reduction module for filtering and validation again. As previously justified, this cyclic process enriches the correlation process with extra information.

5.4. Alert prioritization

The last module of the proposed correlation model is called alert prioritization. The purpose of the alert prioritization module is to analyze *hyper-alerts* received from the alert correlation module and to classify them based on their severity. As shown in Fig. 6, the alert prioritization module can be divided into two main phases: severity analysis and classification.

5.4.1. Severity analysis

The purpose of the *severity analysis* phase is to determine the importance of the network alerts or hyper-alerts. The severity analysis is a complicated task that needs several sources of information to determine the criticality of a particular alert in the overall system. Certain sources of information, like cases, vulnerability, and network topology databases might be useful to provide information about the causes, their descriptions, their dependencies to the operating systems, hardware and software platforms. The severity analysis results depend on the nature of the services and the network being monitored. As an example, a mission impact intrusion report correlation system, M-Correlator, was implemented by Porras and others [87], focused on intrusion attacks. Here, the system is used to classify alerts based on severity and take appropriate actions to deal with each one of the alert classes.

Table 4
Survey of network management tools.

Product name	Manufacturer	Input information	Alignment with Section 3 (Info. sources)	License	Version	Correlation techniques	Alignment with Section 5 (Correlation model)	Homepage
Pandora FMS	Ártica Soluciones Tecnológicas Ltd.	SNMP traps,Syslog	3.1 Alerts database	Comm.	4.0.1	Filtering, Aggregation, Validation, Alignment with Section 4.3 "Temporal based similarity"	5.2.A. Filtering 5.2.B. Validation 5.3.A. Aggregation	http://pandorafms.org/
Osmius	Peopleware SL	SNMP traps,Syslog,	3.1 Alerts database	Comm.	11.01.0	Filtering, Data mining, Aggregation, Cause identification, Alignment with Section 4.3 "Attribute based similarity"	5.2.A. Filtering 5.3.A. Aggregation 5.3.B. Cause identification	http://www.osmius.com/
HP Openview	HP	SNMP traps,Syslog	3.1. Alerts database 3.2. Topology information	Comm.	2011	Filtering, Verification, Aggregation, Alignment with Section 4.3 "Expert rules"	5.2.A. Filtering 5.2.B. Validation 5.3.A. Aggregation	http://www.hp.com
Avaya VPFM	Avaya Inc.	SMTP traps, Syslogs	3.1. Alerts database 3.2. Topology information	Comm.	7.5	Filtering, Aggregation, Cause identification	5.2.A. Filtering 5.3.A. Aggregation 5.3.B. Cause identification	http://www.avaya.com
NagiosXI	Worldwide Nagios community	SNMP traps,Syslog,	3.1 Alerts database	Comm.	2011R1.9	Filtering, Data mining, Pattern matching, Aggregation	5.2.A. Filtering 5.3.A. Aggregation	http://www.nagios.org
IBM Tivoli Enterprise Consol (TEC)	IBM	SNMP traps, Syslogs	3.1. Alerts database 3.2. Topology information	Comm.	39	Normalization, Filtering, Aggregation, Cause identification, Prioritization,Alignment with Section 4.3 "Expert rules"	5.2.A. Filtering 5.2.B. Validation 5.3.A. Aggregation 5.3.B. Cause identification 5.4.B. Classification	http://www.ibm.com
OpenNMS	The Open NMS Group	SNMP traps, Syslog,	3.1 Alerts database	Open source GPLv2	1.8.16	Feature construction, Filtering, Validation, Alignment with Section 4.3 "Expert rules"	5.1.B. Feature construction 5.2.A. Filtering 5.2.B. Validation	http://opennms.org
AggreGate Network Manager	Tibbo Technology Inc.	SNMP traps, Syslogs	3.1. Alerts database 3.2. Topology information	Comm.	4.50.01	Normalization, Filtering, Clustering, Aggregation lignment with Section 4.3 "Expert rules"	5.1.B. Feature construction 5.2.A. Filtering	http://aggregate.tibbo.com

(*continued on next page*)

Table 4 (*continued*)

Product name	Manufacturer	Input information	Alignment with Section 3 (Info. sources)	License	Version	Correlation techniques	Alignment with Section 5 (Correlation model)	Homepage
AccelOps	AccelOps, Inc.	SNMP traps, Syslogs	3.1 Alerts database	Comm.	2010	Normalization, Filtering, Clustering, Validation, Aggregation, Prioritization Alignment with Section 4.3 "Expert rules	5.3.A. Aggregation 5.1.A. Normalization 5.2.A. Filtering 5.2.B. Validation 5.3.A. Aggregation 5.4.B. Classification	http://www.accelops.com/
SolarWinds Orion Application Performance Monitor	SolarWinds	SNMP traps, Syslogs	3.1. Alerts database 3.2. Topology information	Comm.	2011	Filtering, Verification, ggregation	5.2.A. Filtering 5.2.B. Validation 5.3.A. Aggregation	http://www.solarwinds.com
up.time	Uptime Software	SNMP traps, Syslogs	3.1 Alerts database	Comm.	7	Filtering, Validation	5.2.A. Filtering 5.3.A. Aggregation	http://www.uptimesoftware.com
NetCrunch	AdRem Software, Inc.	SNMP traps, Syslogs,	3.1. Alerts database 3.2. Topology information	Comm.	6	Normalization, Event suppression, Aggregation, Alignment with Section 4.3 "Expert rules	5.1.A. Normalization 5.2.A. Filtering 5.3.A. Aggregation	http://www.adremsoft.com
VMware vCenter Operation Management Suite	Vmware, Inc.	SNMP traps, Syslogs,	3.1 Alerts database	Comm.	5.4.1	Filtering, Validation, Cause identification	5.2.A. Filtering 5.2.B. Validation 5.3.B. Cause identification	http://www.vmware.com
Verax NMS	Verax Systems Corp.	SNMP traps, Syslogs,	3.1 Alerts database	Comm.	1.9.0	Filtering, Aggregation lignment with Section 4.3 "Expert rules	5.2.A. Filtering 5.3.A. Aggregation	http://www.veraxsystems.com

Table 5
Survey of security tools.

Product name	Manufacturer	Input information	Alignment with Section 3 (Info. sources)	License	Version	Correlation techniques	Alignment with Section 5 (Correlation model)	Homepage
Snort	Sourcefire, Inc.	Snort alerts	3.1. Alerts database	Open source GPL	2.9.2.2	Filtering, Aggregation, Prioritization,Alignment with Section 4.3 "Expert rules"	5.2.A. Filtering 5.3.A. Aggregation 5.4.B. Classification	http://www.snort.org/
BrO	BrO team	Snort alerts, Syslogs	3.1. Alerts database	Open source BSD	2.0Beta	Filtering,Pattern matching, Alignment with Section 4.3 "Temporal based similarity, Dependency graphs"	5.2.A. Filtering	http://www.bro-ids.org/
Bitacora	S21sec	Syslogs, IDS alerts	3.1. Alerts database 3.3. Vulnerabilities database	Comm.	5	Normalization, Filtering, Validation, Real-time correlation	5.1.A. Normalization 5.2.A. Filtering 5.2.B. Validation	http://www.s21sec.com/
OSSEC HIDS	Trend Micro	SMTP traps, IDS alerts,Syslogs	3.1. Alerts database	Open sourceGPLv3	2.6	Filtering, Verification, Severity analysis, Cause identification, Alignment with Section 4.3 "Expert rules"	5.2.A. Filtering 5.2.B. Validation 5.3.B. Cause identification 5.4.A. Severity analysis	http://www.ossec.net/
OSSIM	AlienVault	Snort alerts	3.1. Alerts database	Open source GPL	3.1	Normalization,Filtering, Alignment with Section 4.3 "Expert rules"	5.1.A. Normalization 5.2.A. Filtering	http://www.ossim.net
Prelude SIEM	PreludeIDS Technologies	SMTP traps, Syslogs	3.1. Alerts database	Open source GPL	1	Normalization, Filtering Aggregation, Alignment with Section 4.3 "Expert rules"	5.1.A. Normalization 5.2.A. Filtering 5.3.A. Aggregation	http://www.prelude-ids.com
ACID	CERT	Snort alerts, Syslogs	3.1. Alerts database	Open source GPL	0.0.9.6b23	Filtering, Validation,Clustering, Prioritization, Alignment with Section 4.3 "Attribute based similarity"	5.2.A. Filtering 5.2.B. Validation 5.3.A. Aggregation 5.4.B. Classification	http://acidlab.sourceforge.net/
neuSECURE	GuardedNet Inc.	SNMP traps, Syslogs, IDS alerts	3.1. Alerts database	Comm.	3	Normalization, Validation, Aggregation Prioritization, Multi-variant correlation	5.1.A. Normalization 5.2.B. Validation 5.3.A. Aggregation 5.4.B. Classification	http://www.guarded.net/
SecurityCenter	Tenable Network security	IDS alerts, Syslogs	3.1. Alerts database 3.2. Topology information	Comm.	4.4	Filtering, Validation, Aggregation, Cause identification, Severity analysis, Prioritization	5.2.A. Filtering 5.2.B. Validation	http://www.tenable.com/

(*continued on next page*)

Table 5 (*continued*)

Product name	Manufacturer	Input information	Alignment with Section 3 (Info. sources)	License	Version	Correlation techniques	Alignment with Section 5 (Correlation model)	Homepage
			3.4. TTS information				5.3.A. Aggregation 5.3.B. Cause identification 5.4.A. Severity analysis 5.4.B. Classification	
Net Forensics Console	netForensics	IDS alerts, Syslogs	3.1. Alerts database	Comm.	1.2f	Normalization, Filtering Aggregation, Alignment with Section 4.3 "Expert rules"	5.1.A. Normalization 5.2.A. Filtering 5.3.A. Aggregation	http://www.netforensics.com/
ArcSight ETRM Platform	Arcsight Inc.HP Company	Syslogs, Alerts	3.1. Alerts database 3.2. Topology information 3.3. Vulnerability database	Comm.	2011	Aggregation, Prioritization, Alert correlation	5.3.A. Aggregation 5.4.B. Classification	http://www.arcsight.com/
Virtuoso	ConostixInc.S.A.	Syslogs, IDS alerts	3.1. Alerts database	Comm.	Hardware LogCollector LC-360	Normalization, Filtering, Real-time correlation	5.1.A. Normalization 5.2.A. Filtering	http://www.conostix.com/product-virtuoso.html

5.4.2. Classification

In this final phase, *hyper-alerts* are classified based on their severity measurements and a relevance metric is now calculated The output is sent to the network expert as a report containing alerts in an ascending order according to their relevance.

In [87], the classification is done based on assigning a relevance score for each alert, which is produced through a comparison of the alert target's known topology against the vulnerability requirements of the incident type. Next, a priority calculation is performed per alert to indicate, first, the degree to which the alert is targeted at critical assets; second, the amount of interest the user has registered for this alert type. Last, an overall incident rank is assigned to each alert, which brings together the priority of the alert with the likelihood of success. Zomlot et al. [88] presented an approach to classify intrusion analysis using an extended Dempster–Shafer theory. The proposed algorithm captured sensor quality that corresponds to the intuitive interpretation, and designed an algorithm for calculating confidence values for hypotheses on an alert correlation graph. According to their judge, the proposed Dempster Shafer application can correctly combine non-independent evidences commonly found in correlated IDS alerts.

Alsubhi and others [89] proposed two techniques, one for alert rescoring and prioritization which is based on fuzzy logic inference mechanism, and another which is used for rescoring alerts to show the early steps of the attackers. Wallin et al. [90] proposed a neural network-based approach for alarm filtering and prioritization by using the knowledge gained from alarm flow properties and trouble ticketing information. Jiang et al. [91] proposed a novel peer review mechanism based on rule based systems to rank the importance of alerts resulting in the top ranked alerts being more likely to be true positives. After comparing a metric value against a threshold to generate alerts, the algorithm compares the value with the equivalent thresholds from many other rules to determine the importance of alerts.The output of the classification phase is sent to the network operator as a report which usually contains the causes in descending order from higher to lower severities.

6. Existing alert correlation tools

In this section, we describe some of alert correlation systems currently in use. Our intention here is not to provide a complete list of existing tools, but to make a comparison that allows to check which of the previously described techniques are used and which ones are not.

We classify the correlation tools based on their application field as discussed above. Some network management tools are listed in Table 4. Some of the most important security tools are listed in Tables 5 and 6 collects some of the SCADA monitoring systems. In these tables, the "Input information" column indicates the types of the input data. "Alignment with Section 3" column matchs "Input information" contents with Section 3. "License" column indicates if the tool is commercial ("Comm.") or open source with GPL license. The "Version" column provides the tool's version evaluated here. The specific correlation operations

Table 6
Survey of SCADA tools.

Product name	Manufacturer	Input information	Alignment with Section 3 (Info. Sources)	License	Version	Correlation techniques	Alignment with Section 5 (Correlation model)	Homepage
SIMATIC WinCC	Siemens	Alarms	3.1. Alerts database	Comm.	7	Feature construction, Filtering, Aggregation, Alignment with Section 4.3 "Attribute based similarity"	5.1.B. Feature construction 5.2.A. Filtering 5.3.A. Aggregation	http://www.siemens.com/
U.C.ME™-OPC	Control-See	Alarms	3.1. Alerts database	Comm.	2012	Filtering, Validation, Aggregation, Severity analysis, Classification, Alignment with Section 4.3 "Attribute based similarity"	5.2.A. Filtering 5.2.B. Validation 5.3.A. Aggregation 5.4.A. Severity analysis 5.4.B. Classification	http://www.controlsee.com/
Shopfloor-Online	Lighthouse Systems	Alarms	3.1. Alerts database	Comm.	2009	Aggregation, Classification, Alignment with Section 4.3 "Attribute based similarity"	5.3.A. Aggregation 5.4.B. Classification	http://www.lighthousesystems.com
Plant Triage	ExperTune	Alarms	3.1. Alerts database	Comm.	9	Filtering, Clustering, Prioritization, Cause Identification, Alignment with Section 4.3 "Sequential based methods-Graphs"	5.2.A. Filtering 5.3.A. Aggregation 5.3.B. Cause identification 5.4.B. Classification	http://www.expertune.com
WINLOG pro	SIELCO SISTEMI	Alarms	3.1. Alerts database	Comm.	2.07.11	Normalization, Prioritization	5.2.A. Filtering 5.4.B. Classification	http://www.sielcosistemi.com
AggreGate SCADA	Tibbo Technology Inc.	Alerts	3.1. Alerts database	Comm.	40.50.01	Normalization, Filtering, Clustering, Alignment with Section 4.3 "Attribute based similarity"	5.1.A. Normalization 5.2.A. Filtering 5.3.A. Aggregation	http://www.aggregate.tibbo.com
IGSS	7-Technologies A/S	Alarms	3.1. Alerts database	Comm.	9	Feature construction, Filtering. Alignment with Section 4.3 "Expert rules"	5.1.B. Feature construction 5.2.A. Filtering	http://www.igss.com
Argos	Centro de Innovación Tecn. del Aluminio	Alarms	3.1. Alerts database	Open source GPLv3	0.6.670	Normalization, Aggregation	5.1.A. Normalization 5.3.A. Aggregation	http://www.cintal.com.ve
Mango M2M2	Serotonin Software Tech.	Alarms	3.1. Alerts database	Comm.	1.13.0	Filtering, Prioritization	5.2.A. Filtering 5.4.B. Classification	http://www.mango.serotoninsoftware.com/
IntegraXor	EcavaSdn. Bhd	Alarms	3.1. Alerts database	Comm.	3.7.1	Filtering, Grouping, Cause identification	5.2.A. Filtering 5.3.A. Aggregation 5.3.B. Cause identification	http://www.integraxor.com/
MESbox SCADA	OridinalSotware Inc.	Alarms	3.1. Alerts database	Comm.	2012	Filtering, Aggregation, Severity analysis	5.2.A. Filtering 5.3.A. Aggregation 5.4.A. Severity analysis	http://www.ordinal.fr

and techniques of each tool are listed in the column "Correlation techniques"; "Alignment with Section 5" column matchs "correlation techniques" contents with Section 5 and, finally, the webpage for the tool is listed in column "Homepage".

6.1. Network management tools

Table 4 summarizes the most relevant information for some of the most widely deployed network management tools. In particular, we focus in the types of inputs and correlation techniques used. Some remarks related to Table 4 worth mentioning. First, the majority of these tools are implemented by important private companies like HP, IBM and others. Therefore, most of them are commercially distributed. Second, most of them were implemented to deal with a single type of information sources (Section 3), which are alerts with different formats. Third, a large number of these tools implement an expert rules correlation engine. Besides, they use very simple versions of correlation operations such as alert reduction, clustering, and aggregation. We think that the main reason for that is because these tools were not initially designed to deal with alert correlation specifically, that is, they were implemented for system monitoring and diagnostics purposes. Some of them have implemented other correlation engines, but they have proprietary patents which prevent us from clearly understand which kind of correlation approaches were implemented. But, generally speaking, and according to the data sheets, it seems they mostly use rule-based correlation approaches. Fourth, we notice that there is a big gap between the sophisticated correlation techniques which are presented in the research community and revised in Section 4 above, with those implemented in these tools. In our opinion, this big gap resulted from the fact that the industrial and research communities have different algorithms design goals. The research community proposed algorithms to prove some research ideas, so the algorithms may be very complex in terms of processing time and resource consumption, whereas private companies goal is to build scalable and efficient tools with lightweight algorithms.

Furthermore, it seems that they implemented correlation techniques that will save implementation time and cost, in order to enter the market quickly. Thus, most of the commercial tools simply handle only one type of information sources, as illustrated in Column 4, and this can be related to the lack of complex correlation techniques.

6.2. Security tools

Similar observations resulted from Table 5, which lists a number of security tools. First, some of these tools are open source like Snort and Bro; others are commercial like Bitacora and Net Forensics Console. Second, the majority of these tools were designed to accept snort alerts and syslog messages as input formats. Therefore, they were implemented for specific purposes. Third, like most NMS tools, most of the listed security tools have implemented simple correlation operations like filtering, pattern matching, validation and others. Some of them like Snort and OSSIM have implemented more complex mechanisms as rule based correlation techniques. Fourth, the security tools present the same issue as NMS tools regarding the design gap between the complexities of the correlation techniques which were suggested by the research community compared with what was implemented in these tools.

6.3. SCADA tools

With regard to Table 6, which contains some of the current SCADA monitoring systems, we also have some observations. First, unlike network management and security tools, we observed that alert correlation in SCADA systems still needs much work to be done in both industrial and research sectors. Second, the majority of these tools are commercial with free-trial versions. Third, they were implemented for various industrial applications which accept various alarms formats. Fourth, after a thorough review of the data sheets of these products we found that most of them mainly use attribute based similarity for doing several operations like filtering, agregation and prioritization to discover root causes.

7. Related work

In a thorough review of the literature for alert correlation we have found a wide number of research efforts in the form of research projects and surveys aimed to describe in a clear way the nature of the alert correlation process. In 2003, Pouget and Dacier [1] made a comprehensive review of the state of the art of alert correlation techniques and existing tools. They mainly focused on techniques used within the intrusion detection domain, providing a high level description of them.They divided the various research efforts into several groups such as rule-based, attack scenario method, and context reasoning approaches. They also grouped the existing tools into three categories: log analysis tools, management consoles and testbed research tools. One year later, Steinder and Sethi [20] presented a comprehensive overiew of existing fault localization techniques in communication systems by classifiying them according to three main categories: artificial intelligence techniques, model traversing techniques and fault propagation models.They also discussed the challenges facing the fault localization process in complex communication systems. Unlike Pouget and Dacier, they focused only on techniques used within the network management field. In the same year, Valeur and others [31] proposed a comprehensive real time alert correlation approach for intrusion detection, which consists of many components such as: normalization, preprocessing, alert fusion, thread reconstruction and others. They presented a detailed explanation of the alert correlation process by applying a tool to a number of well known intrusion detection data sets in order to identify how each component contributes to the overall goal of the correlation tool. Also in 2004, Zurutuza and Uribeetexeberria [21] presented a survey of alert correlation techniques for intrusion detection. The survey was based on three main components: preprocessing, alarm analysis and alarm correlation. In 2006, Sadoddin and Ghorbani [22] presented a survey of alert correlation techniques

based on a proposed framework, which consists of sixcomponents.These components include normalization, aggregation, correlation, false alert reduction, attack strategies analysis and normalization. Their main focus was on the techniques proposed only for the intrusion detection field. In 2008, Zang et al. [23] presented a general summary of alert correlation and fusion technologies. They classified the techniques based on a proposed model, which has five components: normalization, aggregation, verification, and correlation and attack scenarios. In 2010, Elshoush and Osman [92] presented an state of the art of alert correlation algorithms in CIDS. They made a classification based on system architectures, and showed that the current research revealed the need for artificial inteligence and fuzzy logic techniques to satisfy the growing demand of reliable, intelligent, and flexible IDSs.

Previous contributions surveying alert correlation techniques have some limitations. First, they are biased to a specific application domain, normally management or security, being the security field the more frequent. Second, the existing alert correlation models proposed in these cited works do not take into account all the sources of information used by different authors (as discussed in Section 3). Third, in spite of the complexity of the correlation process, most of the proposed models conceptualize it as a sequence of interrelated processes linked to each other linearly. Each correlation component does its task locally and forwards the results to the next one, without knowing what other components have done. We suggest that this linearity does not fit in the correlation process and, accordingly, we present a new model which considers feedback mechanisms.

This paper has four main contributions as a difference from other reviews and surveys. First, the paper presents an updated review of the state of the art on alert correlation techniques based on a proposed taxonomy. Second, it covers three different applications: network management, security, and process control (SCADA systems). Third, a new correlation model is suggested, where complex feedback processes are considered. And finally, a comparison of existing commercial products is presented.

8. Conclusions

In this paper, we have reviewed the research efforts carried out in the alert correlation field. We have provided a classification of them based on the type of applications, number of data sources, correlation method and type of architecture. We have shown that it is possible to classify correlation techniques as belonging to different applications: security, network management and process control (SCADA systems). Despite the big amount of efforts done in the alert correlation field, we observe that this is still an active research area in both, especially in NMS and security applications. Yet, considerable work should still be done in the SCADA systems area, where only few efforts have been contributed.

Regarding the architecture and scalability of most of the current solutions, a large number of them are based on centralized architectures, which limits their scalability. For this reason, there is a need to investigate on distributed architectures for dealing with the alert correlation problem.

In addition, we have found that there is no clear agreement between researchers and vendors about how to measure the efficiency and accuracy of the proposed solutions. As a consequence, there are no standard benchmarks for evaluating and comparing them.

We have also presented in this paper a correlation process model. After having analyzed the limitations of the different solutions proposed in the literature, we claim that the alert correlation problem should be modeled as a cyclic process with feedback mechanisms. However, none of the current proposals in this field consider these mechanisms.

As a final contribution, we made a review of some available correlation tools, intended for the cited three different applications: NMS, security and SCADA systems. We have noticed that there is a big gap between the complexities behind the correlation techniques suggested by the research community and those really implemented in these available tools.

Acknowledgment

This work has been partially supported by Spanish MICINN through project TEC2011-22579.

References

[1] F. Pouget, M. Dacier, Alert Correlation: Review of the State of the Art, Research Report RR-03-093, Institut EURECOM, 2003. <http://www.eurecom.fr/en/research/networking-and-security-department/publications>.
[2] Electronic Statistics Textbook, Statsoft. Inc., 2011. <http://www.statsoft.com/textbook/>.
[3] G. Jakobson, M. Weissman, Alarm correlation, Journal of IEEE Network 7 (1993) 51–59.
[4] R.D. Gardner, D.A. Harle, Methods and systems for alarm correlation, in: Proc. of the Global Telecommunications Conf. (GLOBECOM '96), vol. 1, 1996, pp. 136–140.
[5] J.D. Case, K. McCloghrie, M.T. Rose, S. Waldbusser, Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2), IETF Network Working Group, 1996. <http://www.ietf.org/rfc/rfc1905.txt>.
[6] ISO, Information Processing Systems-OSI, ISO Standard 9596-1: Common Management Information Protocol, Part 1: Specification, 1998.
[7] http://www.snort.org/.
[8] S. Staniford-chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagl, K. Levitt, C. Wee, R. Yip, D. Zerkle, GrIDS – a graph based intrusion detection system for large networks, in: Proc. of the 19th National Information Systems Security Conf., 1996.
[9] J. Yu, Y.V.R. Reddy, S. Selliah, S. Kankanahalli, S. Reddy, V. Bharadwaj, TRINETR: An intrusion detection alert management systems, in: Proc. of 13th IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE), 2004, pp. 235–240.
[10] T. Chyssler, S. Nadjm-Tehrani, S. Burschka, K. Burbeck, Alarm reduction and correlation in defense of IP networks, in: Proc. of 13th IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE), 2004, pp. 229–234.
[11] The MITRE Corporation, Common Vulnerabilities and Exposures. <http://cve.mitre.org/>.
[12] G. Dreo, Design Issues for an Extended Trouble Ticket System, International report (available from the author), University of Munich, 1992.
[13] R. Valta, G. Dreo, Using an integrated trouble ticket system to support fault diagnostics (in German), HMD, Theorie und Design Wirtschaftsinformatik 171 (1993) 45–59.
[14] L. Lewis, G. Dreo, Extending trouble ticket systems to fault diagnostics, Journal of IEEE Network 7 (1993) 44–51.

[15] R. Costa, N. Cachulo, P. Cortez, An intelligent alarm management system for large-scale telecommunication companies, in: Proc. of the 14th Portuguese Conf. on Artificial Intelligence (EPIA'09), 2009, pp. 386–399.
[16] W. Li, S. Tian, Preprocessor of intrusion alerts correlation based on ontology, in: Proc. of the Int. Conf. on Communications and Mobile Computing (CMC '09), vol. 3, 2009, pp. 460–464.
[17] http://www.ietf.org/rfc/rfc4765.txt.
[18] V. Holub, T. Parsons, P. O'Sullivan, J. Murphy, Run-time correlation engine for system monitoring and testing, in: Proc. of the 6th IEEE Int. Conf. on Autonomic Computing (ICAC-INDST '09), 2009, pp. 43–44.
[19] P. Kabiri, A. Ghorbani, A rule-based temporal alert correlation system, International Journal of Network Security 5 (2007) 66–72.
[20] M. Steinder, A. Sethi, A survey of fault localization techniques in computer networks, Journal of Science of Computer Programming 53 (2004) 165–194.
[21] U. Zurutuza, R. Uribeetxeberria, Intrusion detection alarm correlation: a survey, in: Proc. of the Int. Conf. on Telecommunications and Computer Networks (IADATs), 2004, pp. 1–3.
[22] R. Sadoddin, A. Ghorbani, Alert correlation survey: framework and techniques, in: Proc. of the Int. Conf. on Privacy, Security and Trust (PST '06), 2006, pp. 6–15.
[23] T. Zang, X. Yun, Y. Zhang, A survey of alert fusion techniques incident, in: Proc. of the Int. Conf. on Web-Age Information Management (WAIM '08), 2008, pp. 475–481.
[24] X. Zhuang, D. Xiao, X. Liu, Y. Zhang, Applying data fusion in collaborative alerts correlation, in: Proc. of the Int. Symposium of Computer Science and Computational Technology (ISCSCT '08), vol. 2, 2008, pp. 124–127.
[25] J. Chang, J. Yu, Y. Pei, MS2IFS: a multiple source-based security information fusion system, in: Proc. Of the Int. Conf. on Communications and Intelligence Information Security (ICCIIS'10), 2010, pp. 215–219.
[26] J. Hu, H. Chen, T. Liu, H. Tseng, D. Lin, C. Yang, C.E. Yeh, Implementation of alarm correlation system for hybrid networks based upon the perfSONAR framework, in: Proc. of the Int. Conf. on Advanced Information Networking and Applications Workshops (WAINA'10), 2010, pp. 893–898.
[27] B. Gruschke, Integrated event management: event correlation using dependency graphs, in: Proc. of the 9th Int. Workshop on Distributed Systems Operation & Management (DSOM '98), 1998, pp. 130–141.
[28] S. Klinger, S. Yemini, Y. Yemini, D. Ohsie, S. Stolfo, A coding approach to event correlation, in: Proc. of the 4th Int. Symposium on Integrated Network Management, 1996, pp. 266–277.
[29] F. Alserhani, M. Akhlaq, I. Awan, A. Cullen, MARS: multi-stage attack recognition system, in: Proc. of the 24th IEEE Int. Conf. on Advanced Information Networking and Applications (IFIP/IEEE), 2010, pp. 753–759.
[30] Xinzhou Qin, A Probabilistic-Based Framework for INFOSEC Alert Correlation, Phd thesis, College of Computing, George Institute of Technology, 2005.
[31] F. Valeur, G. Vigna, C. Kruegel, R.A. Kemmerer, A comprehensive approach to intrusion detection alert correlation, dependable and secure computing, Journal of IEEE Transactions 1 (2004) 146–169.
[32] Y. Chen, J. Lee, Autonomous mining for alarm correlation patterns based on time-shift similarity clustering in manufacturing system, in: Proc. of the Int. Conf. on Prognostics and Health Management (PHM'11), 2011, pp. 1–8.
[33] V. Alfonso, S. Keith, Probabilistic alert correlation, in: Proc. of the 4th Int. Symposium on Recent Advances in Intrusion Detection, 2001, pp. 54–68.
[34] K. Lee, J. Kim, K. Kwon, Y. Han, S. Kim, DDoS attack detection method using cluster analysis, Journal of Expert Systems Applications 34 (2008) 1659–1665.
[35] A. Siraj, R.B. Vaughn, Multi-level alert clustering for intrusion detection sensor data, in: Proc. of the Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS), 2005, pp. 748–753.
[36] K. Julisch, Clustering intrusion detection alarms to support root cause analysis, Journal of ACM Transaction on Information System Security 6 (2003) 443–471.
[37] K. Julisch, M. Dacier, Mining intrusion detection alarms for actionable knowledge, in: Proc. of the 8th ACM Int. Conf. on Knowledge Discovery and Data Mining, 2002, pp. 366–375.
[38] H. Debar, A. Wespi, Aggregation and correlation of intrusion–detection alerts, in: Proc. of the Int. Symposium on Recent Advances in Intrusion Detection (RAID'02), 2002, pp. 85–103.
[39] F. Cuppens, Managing alerts in a multi-intrusion detection environment, in: Proc. of the 17th Annual Conf. on Computer Security Applications, 2001, pp. 22–31.
[40] G. Jakobson, M. Weissman, Real-time telecommunication network management: extending event correlation with temporal constraints, in: Proc. of the 4th Int. Symposium on Integrated Network Management (IEEE/IFIP), 1995, pp. 290–301.
[41] S.H. Ahmadinejad, S. Jalili, Alert correlation using correlation probability estimation and time windows, in: Proc. of the Int. Conf. on Computer Technology and Development (ICCTD'09), vol. 2, 2009, pp. 170–175.
[42] J. Ma, Z. Li, W. Li, Real-time alert stream clustering and correlation for discovering strategies, in: Proc. of the Int. Conf. on Fuzzy Systems and Knowledge Discovery (FSKD'08), vol. 4, 2008, pp. 379–384.
[43] X. Qin, W. Lee, Statistical causality analysis of infosec alert data, in: Proc. of the 6th Int. Symposium on Recent Advances in Intrusion Detection (RAID'03), 2003, pp. 73–93.
[44] B. Morin, H. Debar, Correlation of intrusion symptoms: an application of chronicles, in: Proc. of the 6th Int. Conf. on Recent Advances in Intrusion Detection (RAID'03), 2003, pp. 94–112.
[45] L. Zhaowen, L. Shan, M. Yan, Real-time intrusion alert correlation system based on prerequisites and consequence, in: Proc. of the Int. Conf. in Wireless Communications Networking and Mobile Computing (WiCOM'10), 2010, pp. 1–5.
[46] P. Ning, Y. Cui, D. Reeves, D. Xu, Techniques and tools for analyzing intrusion alerts, Journal of ACM Transactions on Information and System Security 7 (2004) 274–318.
[47] S. Templeton, K. Levitt, A requires/provides model for computer attacks, in: Proc. of the Int. Workshop on New Security Paradigms (NSPW), 2000, pp. 31–38.
[48] S. Xiao, Y. Zhang, X. Liu, J. Gao, Alert Fusion Based on Cluster and Correlation Analysis, in: Proc. of the Int. Conf. on Convergence and Hybrid Information Technology (ICHIT'08),(2008) 163-68.
[49] S. Roschke, F. Cheng, C. Meinel, A new alert correlation algorithm based on attack graph, in: Proc. of the 4th Int. Conf. on Computational Intelligence in Security for Information Systems (CISIS'11), 2011, pp. 58-67.
[50] L. Wang, A. Liu, S. Jajodia, Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts, Journal of Computer Communications 29 (2006) 2917–2933.
[51] L. Li, F. Lifang, Fast fault localization for internet services based on bipartite graph, International Journal of Digital Content Technology and its Applications 5 (2011) 8–16.
[52] S.A. Yemini, S. Klinger, E. Mozes, Y. Yemini, D. Ohsie, High speed and robust event correlation, Journal of IEEE Communications 34 (1996) 82–90.
[53] D. Ourston, S. Matzner, W. Stump, B. Hopkins, Applications of hidden Markov models to detecting multi-stage network attacks, in: Proc. of the 36th Annual Hawaii Int. Conf. on System Sciences (HICSS'03), vol. 9, 2003, pp. 334–344.
[54] H. Farhadi, M. AmirHaeri, M. Khansari, Alert correlation and prediction using data mining and HMM, International Journal of Information Security 3 (2011) 1–25.
[55] Z. Xin, G. Feng, H. Jiuqiang, S. Yu, A hidden Markov model based framework for tracking and predicting of attack intention, in: Proc. of the Int. Conf. on Multimedia Information Networking and Security (MINES'09), vol. 2, 2009, pp. 498–450.
[56] S. Zhicai, X. Yongxiang, A novel hidden Markov model for detecting complicate network attacks, in: Proc. of the Int. Conf. on Wireless Communications, Networking and Information Security (WCNIS), 2010, pp. 312–315.
[57] M. Steinder, A.S. Sethi, Probabilistic fault localization in communication systems using belief networks, Journal of IEEE/ACM Networking Transactions 12 (2004) 809–822.
[58] M. Marchetti, M. Colajanni, F. Manganiello, Identification of correlated network intrusion alerts, in: Proc. of the 3rd Int. Workshop on Cyberspace Safety and Security (CSS), 2011, pp. 15–20.
[59] E. Harahap, W. Sakamoto, H. Nishi, Failure prediction method for network management system by using Bayesian network and shared database, in: Proc. of the 8th Asia-Pacific Symposium Information and Telecommunication Technologies (APSITT), 2010, pp. 1–6.
[60] B. Zhu, A. Ghorbani, Alert correlation for extracting attack strategies, International Journal of Network Security 3 (2006) 244–258.
[61] S. Al-Mamory, H. Zhang, IDS alerts correlation using grammar-based approach, Journal in Computer Virology 5 (2008) 271–282.
[62] R.N. Cronk, P.H. Callahan, L. Bernstein, Rule-based expert system for network management and operations: an introduction, Journal of IEEE Networks 2 (1988) 7–21.

[63] K.W. Edward, A network diagnostic expert system for acculink multiplexers based on a general network diagnostic scheme, in: Proc. of the 3rd Int. Symposium on Integrated Network Management with Participation of the IEEE Communications Society (CNOM), C-12, 1993, pp. 659–669.
[64] G. Liu, A.K. Mok, E.J. Yang, Composite events for network event correlation, in: Proc. of the 6th Int. Symposium on Integrated Network Management, Distributed Management for the Networked Millennium (IFIP/IEEE), 1999, pp. 247–260.
[65] F. Cuppens, R. Ortalo, LAMBDA: a language to model a database for detection of attacks, in: Proc. of the 3rd Int. Workshop on Recent Advances in Intrusion Detection (RAID '00), 1907/2000, 2000, pp. 197–216.
[66] R.A. Kemmer, G. Vigna, A Model-Based Real Time Intrusion Detection System for Large Scale Heterogeneous Networks, Technical report, 2003. <http://www.stormingmedia.us/42/4280/A428024.html>.
[67] S. Eckmann, G. Vigna, R.A. Kemmerer, STATL: an attack language for state-based intrusion detection, Journal of Computer Security 10 (2002) 71–104.
[68] L. Liu, K.F. Zheng, Y.X. Yang, An intrusion alert correlation approach based on finite automata, in: Proc. of the Int. Conf. on Communications and Intelligence Information Security (ICCIIS), 2010, pp. 80–83.
[69] S. Cheung, U. Lindqvist, M.W. Fong, Modeling multistep cyber attacks for scenario recognition, in: Proc. of the Conf. on DARPA Information Survivability and Exposition (DISCEX III), 2003, pp. 284–292.
[70] R. Smith, N. Japkowicz, M. Dondo, P. Mason, Using unsupervised learning for network alert correlation, in: Proc. of the 21th Canadian Conf. on Artificial Intelligence, 2008, pp. 308–319.
[71] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, in: Proc. of the 20th Int. Conf. on Very Large Databases, 1994, pp. 487–499.
[72] R. Agrawal, R. Srikant, Mining sequential patterns, in: Proc. of the 11th Int. Conf. on Data Engineering, 1995, pp. 3–14.
[73] R. Srikant, R. Agrawal, Mining sequential patterns: Generalizations and performance improvements, in: Proc. of the 5th Int. Conf. on Extending Database Technology (EDBT, 96), 1996, pp. 1–17.
[74] H. Mannila, H. Toivonen, A.I. Verkamo, Discovering frequent episodes in event sequences, Journal of Data Mining and Knowledge Discovery 1 (1995) 210–215.
[75] R. Katipally, W. Gasior, X. Cui, L. Yang, Multistage attack detection system for network administrators using data mining, in: Proc. of the 6th Annual Workshop on Cyber Security and Information Intelligence Research (CSIIRW '10), 2010, pp. 1–4.
[76] R. Sadoddin, A.A. Ghorbani, Real-time alert correlation using stream data mining techniques, in: Proc. of the 20th Int. Conf. on Innovative Applications of Artificial Intelligence, 3, 2008, pp. 1731–1737.
[77] A.A. Mohamed, O. Basir, Fusion based approach for distributed alarm correlation in computer networks, in: Proc. of the 2nd Int. Conf. on Communication Software and Networks (ICCSN '10), 2010, pp. 318–324.
[78] R. Khatoun, G. Doyen, D.R. Saad, A. Serhrouchni, Decentralized alerts correlation approach for DDoS intrusion detection, in: Proc. of the Int. Conf. on New Technologies, Mobility and Security (NTMS '08), 2008, pp. 1–5.
[79] T. Donghai, C. Hu, Q. Yang, J. Wang, Hierarchical distributed alert correlation model, in: Proc. of the 5th Int. Conf. on Information Assurance and Security(IAS'09), 2009, pp. 765–768.
[80] R. Yusof, S. Selamat, S. Sahib, Intrusion alert correlation technique analysis for heterogeneous log, International Journal of Computer Science and Network Security 8 (2008).
[81] M. Sirag, S. Hashim, Network intrusion alert correlation challenges and techniques, Journal of Teknologi Maklumat 20 (2008) 12–36.
[82] O. Abouabdalla, H. El-Taj, A. Manasrah, S. Ramadass, False positives reduction in intrusion detection system: a survey, in: Proc. of the 2nd Int. Conf. in Broadband network & Multimedia Technology, 2009, pp. 463–466.
[83] J.J. Davis, J.C. Andrew, Data preprocessing for anomaly based network intrusion detection: a review, Journal of Computers and Security 30 (2011) 353–375.
[84] L.D. Mills, Network Time Protocol (Version 3), Specification, Implementation and Analysis, technical report, Network Working Group, 1992. <http://tools.ietf.org/html/rfc1305>.
[85] G. Giacinto, R. Perdisci, F. Roli, Alarm clustering for intrusion detection systems in computer networks, Journal of Engineering Applications of Artificial Intelligence19 (2006) 429–438.
[86] H.S. Lin, H.K. Pao, C.H. Mao, Adaptive alarm filtering by causal correlation consideration in intrusion detection, in: Proc. of the 1st Int. Symposium on Intelligent Decision Technologies (IDT'09), 2009, pp. 437-447.
[87] P.A. Porras, M.W. Fong, A. Valdes, A mission-impact-based approach to INFOSEC alarm correlation, in: Proc. of the 5th Int. Symposium on Recent Advances in Intrusion Detection (RAID), 2002, pp. 95–114.
[88] L. Zomlot, S. Sundaramurthy, K. Luo, X. Ou, S.R. Rajagopalan, Prioritizing intrusion analysis using Dempster–Shafer theory, in: Proc. of the 4th ACM workshop on Security and Artificial Intelligence, 2011, pp. 59–70.
[89] K. Alsubhi, E. Al-Shaer, R. Boutaba, Alert prioritization in intrusion detection systems, in: Proc. of the Int. Symposium on Network Operations and Management, 2008, pp. 33–40.
[90] S. Wallin, V. Leijon, L. Landen, Statistical analysis and prioritization of alarms in mobile networks, International Journal of Business Intelligence and Data Mining (IJBIDM) 4 (2009) 4–21.
[91] G. Jiang, H. Chen, K. Yoshihira, A. Saxena, Ranking the importance of alerts for problem determination in large computer systems, Journal of Cluster Computing 14 (2011) 213–227.
[92] H.T. Elshoush, I.M. Osman, Reducing false positives through fuzzy alert correlation in collaborative intelligent intrusion detection systems – a review, in: Proc. of the Int. Conf. on Fuzzy Systems (FUZZ), 2010, pp. 1–8.

Saeed Salah received the B.S. degree in Electrical Engineering from Al-Najah National University in 2003 and M.S. degree in Computer Science from Al-Quds University in 2009. He then worked at the same department as an instructor. He is currently a PhD student at the Department of Signal Theory, Telematics and Communications at Granada University, Spain. His research interests include the network management and configurations, network architecture, network security and routing protocols.

Gabriel Maciá-Fernández is an Associate Professor at the Department of Signal Theory, Telematics and Communications of the University of Granada (Spain). He received a MS in Telecommunications Engineering from the University of Seville, Spain, and the Ph.D. in Telecommunications Engineering from the University of Granada. In the period 1999–2005, he worked as a specialist consultant at "Vodafone España". His research interests are focused on computer and network security, with special focus on intrusion detection, reliable protocol design, network information leakage and denial of service.

Jesús E. Díaz-Verdejo is Associate Professor in the Department of Signal Theory, Telematics and Communications of the University of Granada (Spain). He received his B.Sc. in Physics (Electronics speciality) from the University of Granada in 1989 and held a Ph.D. grant from Spanish Government. Since 1990 he is a lecturer at this University. In 1995 he obtained a Ph.D. degree in Physics. His initial research interest was related with speech technologies, especially automatic speech recognition. He is currently working in computer networks, mainly in computer and network security, although he has developed some work in telematics applications and e-learning systems.