Information Fusion 21 (2015) 145–158

Contents lists available at SciVerse ScienceDirect

Information Fusion

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

Providing SIEM systems with self-adaptation ☆

Guillermo Suarez-Tangil [a,*], Esther Palomar [a], Arturo Ribagorda [a], Ivan Sanz [b]

[a] Carlos III University of Madrid, Avda. Universidad, 30, 38911 Madrid, Spain
[b] Telefónica R & D, C/Don Ramon de la Cruz 84, 28006 Madrid, Spain

ARTICLE INFO

Article history:
Received 20 March 2012
Received in revised form 16 January 2013
Accepted 29 April 2013
Available online 16 May 2013

Keywords:
SIEM
Event correlation
Genetic programming
Artificial neural networks
Adaptive system

ABSTRACT

Security information and event management (SIEM) is considered to be a promising paradigm to reconcile traditional intrusion detection processes along with most recent advances on artificial intelligence techniques in providing automatic and self-adaptive systems. However, classic management-related flaws still persist, e.g. the fusion of large amounts of security events reported from many heterogeneous systems, whilst novel intriguing challenges arise specially when dealing with the adaptation to newly encountered and multi-step attacks. In this article, we provide SIEM correlation with self-adaptation capabilities to optimize and significantly reduce the intervention of operators. In particular, our enhanced correlation engine automatically learns and produces correlation rules based on the context for different types of multi-step attacks using genetic programming. The context is considered as the knowledge and reasoning, not only acquired by a human expert but also inferred by our system, which assist in the identification and fusion of events. In this regard, a number of artificial neural networks are trained to classify events according to the corresponding context established for the attack. Experimentation is conducted on a real deployment within OSSIM to validate our proposal.

1. Introduction

Primarily conceived to centralize all the security information generated by distributed data sources (named sensors) placed alongside a computer network, Security Information and Event Management (SIEM) systems focus on (a) normalizing sensory data (or collected events) in a common format, (b) providing a rapid access to reported events, (c) performing an efficient analysis of scattered events, and also (d) generating correlation alarms. These two latter functionalities (c) and (d) represent important success factors for SIEM, as they are concerned with the quality (or relevance) and not the quantity of the reported events to assist operators for taking the appropriate incident response decisions [1].

Currently, the role of information fusion in the computer security field is being considered as a compelling solution to enhance and automate (c), i.e., the analysis of security events [2]. In fact, fusion techniques can be effectively applied to intrusion detection since they are proven to efficiently deal with heterogeneity in three main aspects: when a number of different data sources are involved, when these involved sensors are located at different places, and when the information they process is represented possibly using different formats [3].

1.1. Motivation

Hence, SIEM systems should place equal effort on guaranteeing an appropriate security information fusion as on assuring an efficient event correlation analysis. However, despite their promising advantages, the following three intriguing challenges emerge.

First, current SIEM systems are highly dependent on the configuration of the multiple sensors deployed over the network. Multiple techniques have been proposed so far to combine data derived from disparate sources in such way that the inferred information and knowledge assist in the identification of attacks. Most of them explore data fusion [4], data mining [5] and artificial intelligence (AI) [6] algorithms in network monitoring and pattern recognition processes [7].

Secondly, existing correlation engines still require operators to invest a non-negligible effort to aggregate related alerts and also to select the most appropriate countermeasure [8]. Several techniques have used prerequisite and consequence of attacks for determining threat pattern sequences, most in the way of a series of alerts [9,10].

Finally, emerging trends in SIEM systems are paying special attention to the employment of automatic procedures for real time analysis of the security events. On one hand, few approaches have focused on the design of intelligent intrusion detection systems

☆ A Genetic-based Framework for an Adaptive Event Correlation.

* Corresponding author.
E-mail addresses: guillermo.suarez.tangil@uc3m.es (G. Suarez-Tangil), epalomar@inf.uc3m.es (E. Palomar), arturo@inf.uc3m.es (A. Ribagorda), isahe@tid.es (I. Sanz).

1566-2535/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.inffus.2013.04.009

(IDSs) capable of autonomous learning to incorporate new knowledge from previously suffered attacks [11]. On the other hand, by introducing self-learning and adaptation into event correlation, SIEM systems would be provided with an autonomous up-to-date knowledge on attacks specially focused on preventing zero-day attacks from any further damage [12]. Moreover, an intelligent correlation engine would be devoted to recognize related events from complex multi-step attacks. In this regard, it is still an open issue to automatically recognize the relevant events for a given attack. To encounter this challenge, we believe that a simple relevancy metric for attacks is mainly determined by their *contexts* (i.e., situational awareness [13]). Hence, in this work a context is considered as the prior known knowledge which is used to assign an interpretation to the sensory data, and resolve ambiguity upon the identification of multi-step attacks. In other words, determining the context of a given attack facilitates the extraction of the potential attack's representations, i.e., most in the way of a collection of events, from an holistic viewpoint.

1.2. Contribution

This article offers the following main findings contributing to any previous related work as follows:

- We present a self-adaptive SIEM system which optimizes the correlation process by applying AI techniques and includes the following functionalities:
  1. A context-based security information fusion subsystem, namely CONTEXTUAL, classifies all the events collected by the SIEM system by deploying artificial neural networks (ANNs—widely applied to classification processes [14,15]). A number of ANNs are trained to dynamically recognize and categorize events into attack scenarios being supervised by the prior known knowledge embodied in the context studied. More precisely, an ANN will establish the patterns containing type and number of events to represent a given multi-step attack. The classification provided by CONTEXTUAL will assist the correlation engine to chain context-related events.
  2. An enhanced event correlation subsystem, namely GENIAL, generates efficient correlation directives by introducing Genetic Programming (GP) into the SIEM correlation engine. GP is a machine learning technique inspired by biological evolution, and has already been applied to intrusion detection approaches such as those in [11,16,17]. Our GP implementation allows the correlation engine to automatically learn and produce correlation rules for different types of multi-step attacks so being able to relate events to the specific attack context and also making the incident response more efficient.
- Our system is designed to be self-adaptive mainly due to the fact that inferred correlation data can evolve and then be used to detect a certain type of attack.
- We carry out a real integration of our subsystems into `OSSIM` [18], namely the de facto standard Open Source SIEM. Our experimentation proves the accuracy of the correlation directives generated by our subsystems which successfully detect different attack scenarios, such as distributed denial of service (DDoS) attacks.

The remainder of the article is organized as follows. In Section 2, the main background found in the literature is outlined. We introduce a brief overview of our system together with its building blocks in Section 3. In Sections 4 and 5, the two subsystems are described in detail. In Section 6, our proposal is deployed and analyzed in a real networking environment, and finally, some conclusions are established in Section 7.

2. Related work

Several SIEM software products have been recently developed to provide essential intelligence in order to reach important associated properties such as flexibility, adaptability, pattern recognition and efficiency. For example, ArcSight [19], RSA enVision [20], Sensage [21], HP CLW [22], Novell IBM [23], LogLogic [24], netForensics [25], Bitacora [26], and OSSIM [18] are some of the multilayered security frameworks presented so far which establish their own architecture and deployment options (we refer the interested reader to [27] for an excellent evaluation of current SIEM products). However, in general, a SIEM system involves these four main disciplines: *detection*, *storage*, *processing*, and *correlation and intelligence*.

Reasonable intrusion detection practices rely on IDSs [28] and event log analysis systems [29,30]. However, these techniques are not enough to detect complex distributed attacks when used separately. For instance, depending on where the IDS is placed it will detect some behaviors and skip others. Two different approaches have been proposed to address it: (1) a centralized integration of the events reported as a whole [31,32] and (2) distributed multi-agents systems [33,34]. Interested readers can find an excellent survey on collaboration-based IDSs in [35]. Thus, different AI techniques have been applied to optimize intrusion detection aimed at dealing with persistent disadvantages such as (i) the volume of alerts per day generated from different sources, (ii) a high false-positive rates and (iii) the inefficiency to discover novel attack scenarios [36]. In particular, Expert Systems [37], Data Mining [38,39], Statistical Analysis [40], Neural Networks [6,7,15,41], Machine Learning [34,42], Genetic Algorithms [43,44], and Artificial Immune systems [45] are the most representative AI-based approaches.

Regarding storage and processing processes, early data aggregation schemes [8,46][47] and current data fusion techniques [48,49] are conceived as a palliative for the critical management of heterogeneous (in data and location) events generated in bulk.

On the other hand, event correlation has been extensively addressed in difference domains such as network fault diagnostic [50], sensor networks [51] and attack detection [10] by applying multiple strategies. Recently, a strong conception exists towards the effectiveness of the correlation process when considering a centralized strategy [52]. Basically, the correlation engine infers extra information from the events finding out connections between them [53]. Principal objectives range from reducing the large number of alerts reported to identify multi-step attack scenarios, and also to identify new attack signatures. This latter objective is perhaps the greatest challenge as attacks are continuously evolving. However, the intelligent extraction of correlation rules still deals with runtime overheads in terms of computational power and memory consumption imposed by the application of AI techniques. In particular, recent approaches to provide the correlation process with automatization are mainly based on the following techniques[1]: (1) *Similarity-based Clustering* organizes related events into attack groups according to a degree of similarity [39], (2) *Aggregation* based on *pre-defined attack* scenarios performs correlation by matching alerts up with patterns specified by operators or learned through training datasets [8], and (3) *Pre-requisites/Consequences syntaxis* is used to define the necessary conditions that must exist for a certain attack to be successful, as well as the conditions that may exist after a specific attack has occurred [9].

In this work we focus on optimizing event correlation from a bio-inspired viewpoint. More specifically, genetic algorithms (GA)

[1] We further elaborate on these techniques in Section 6 in order to discuss about our approach's goodness.

and genetic programming (GP) have already been applied to either generate intrusion detection rules for IDS [16] or to classify attacks [17,43]. Our proposal scrutinizes the combined use of neural networks and GP to optimize the event correlation process as well as presenting a holistic framework to efficiently apply such AI techniques. Furthermore, the experiments and evaluations conducted in this work are performed on a real network deployment, with real traffic. We also provide an informal comparison of the goodness of our approach with some other related works.

3. Overview of our system

In this section, we provide a brief summary of the main subsystems, namely CONTEXTUAL and GENIAL, and other important characteristics that our system holds.

3.1. System input: events

Our system is deployed on an open source SIEM system published under GLP license, namely `OSSIM`. As depicted in Fig. 1, our system receives sensory data or events from a number of sources deployed within a OSSIM-monitored environment. We assume that sensors are well-configured and report the encountered events to the SIEM system in real-time. It is also assumed in our model that events are reported containing a description of the problem that caused each security event. Traditionally, such events are compiled and stored in bulk by the SIEM system for a further analysis which in turn promises near real-time alerts to some extent. Our enhancement introduces the ability to consolidate such sensory data in an efficient and automatic manner.

3.2. Security information fusion: CONTEXTUAL subsystem

Events are classified by CONTEXTUAL, i.e., an ANN and context–based security information fusion subsystem, which uses a series of previously defined tags. Basically, such tags are previously extracted by an experienced human operator with a good comprehension of each context, i.e., the concrete type and number of events occurred when a certain attack is launched. More precisely, the *Neural Network Tags* database specifies a series of labels or keywords and it is used by the *Pattern Generation Module* to define the patterns of a given attack scenario or context. Extracted patterns are stored in the *Neural Network Knowledge Base*.

In order to such a classification occurs adaptably, a number of ANNs is deployed for each context defined. Received events which fit in with any extracted pattern will comprise the subset of events called *Positive Events*.

3.3. Event correlation: GENIAL subsystem

The aforementioned subset resulting from CONTEXTUAL contributes to produce an adaptive event correlation subsystem namely GENIAL which is based on GP. Thus, GENIAL receives the sensory data or events especially preprocessed and fused by CONTEXTUAL. In particular, the classification of positive and negative samples assists in the generation of a population of genetic individuals on which GENIAL executes a genetic algorithm. An individual chains together events based on their relevant attributes which strongly rely on the context being examined as well as on the characteristics of the networking environment. Note that individuals are candidate solutions in our genetic program and will represent correlation directives in our domain. For several rounds or genera-

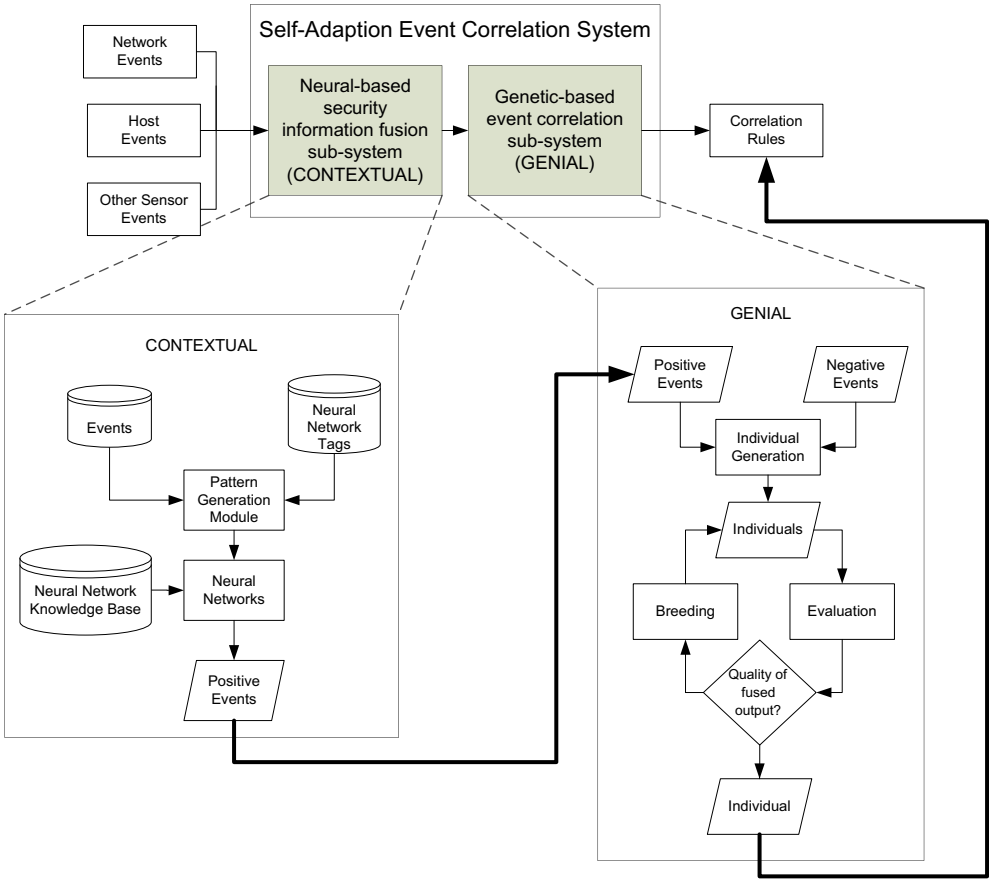

Fig. 1. Our system in a nutshell.

tions, evaluation and breeding procedures are therefore performed on a number of individuals towards producing the best individual.

3.4. System output: correlation directive

The best *Individual* produced during the execution of our genetic algorithm is selected and exported to the OSSIM correlation engine as a *Correlation Rule* using OSSIM's native syntax.

Since our proposal has been designed minimizing dependencies with the SIEM system, the only requirement to be satisfied is database availability. Therefore, our subsystems have access to the database where the SIEM system centralizes the reported events. In fact, our system uses sensory data as input and outputs SIEM-native event correlation rules targeting specific multi-step attacks. Note that experimentation focuses on analyzing complex distributed attacks such as DDoS and web-scans dissemination by using botnets, worms, trojans and virus, to name a few.

4. CONTEXTUAL: context-based security information fusion subsystem

In this section, we introduce CONTEXTUAL, an adaptive security information fusion subsystem based on ANNs which is devoted to classify the security events into different types of attacks. The context plays an essential role here since it represents the knowledge defined by the expert and used during the ANNs' configuration.

This subsystem consists of the following phases:

1. **Tag definition and context representation**. A data association process is performed by a static identification of groups of related events as described in Section 4.1.
2. **Prior knowledge injection and ANN deployments**. The training and deployment of a collection of ANNs (described in Section 4.2) optimize the classification of unknown events into the categories extracted during the phase above.

4.1. Tag definition and context representation

Current SIEM correlation engines are intent on recognizing malicious activity by identifying the existing associations amongst events. In this regard, the common practice is to employ reference numbers for defining types of events and the sensing sources which reported them. More precisely, sensors log the result of those associations in different formats but containing similar information extracted from the suspicious packet: source and destination's IP addresses and ports, protocols, timestamp, sensor's ID, or event ID defining the suspicious activity, to name a few. As a result, correlation rules highly depend on sensors' configuration and any change on this configuration leads to revisiting every rule. To encounter this problem, CONTEXTUAL is sensor-independent as received events are classified into a series of keywords, namely *tags*, which are extracted from the event logs towards an automatic recognition of event types.

Thus, on one hand, a database called *Neural Network Tags* is created by the human expert comprising the following:

Definition 1. A tag $t_j \in \mathcal{T}$ is a keyword which captures the nature of a given event description. Each event is therefore defined more precisely by its tag.

Thus, tags are extracted based on the experience of the human expert and according to (i) the characteristics of the environment, and also (ii) new incoming events in the SIEM system. For the sake of clarity, in our experimental testbed, any web-based intrusion pattern may be totally characterized as a particular set of tags which may contain any of the following: *web_access, web_attempt, web_attack, web_command, web_directory, web_password, web_crosssite, web_httpinspect, web_configuration, web_administration, web_xss, web_injection, web_overflow, web_scan, web_error, web_auth, web_dos, web_highseverity, web_response*, as shown in Table 1. Data mining tools can be applied to minimize the intervention of the expert in this stage.

On the other hand, a human expert is also needed to specify the concrete number and type of the events representing each context. In that regard, CONTEXTUAL generates a series of patterns defined as follows:

Definition 2. A pattern p defines an histogram of occurrences for each tag $t_j \in \mathcal{T}$ which all together determine a certain intrusive or misuse activity registered by a SIEM-monitored environment.

In other words, the frequency $f(t_j)$ of tag t_j in each histogram represents the number $n = f(t_j) = |\mathbf{E}_j|$ of the security events $\{E_1, E_2, \ldots, E_n\} \in \mathbf{E}_j$ tagged accordingly with t_j in pattern p, therefore,

Definition 3. A context representation C_i is defined by a total number m of patterns $p_m^i \in \mathcal{P}$ which consists of all the events that the attack i involves:

$$p_m^i = \{|\mathbf{E}_1|, |\mathbf{E}_2|, \ldots, |\mathbf{E}_j|\} \quad \forall\, \mathbf{E}_j \exists\, t_j \in \mathcal{T}$$

Table 2 represents an excerpt of a number of patterns in $\mathcal{P}$ which are generated in the phase below.

4.2. Prior knowledge injection and ANN deployments

CONTEXTUAL implements a series of ANNs which are defined to represent a particular attack so categorizing different types of security threats. In that regard, a human expert introduces the prior knowledge into the *Neural Network Knowledge Base* which keeps track of the identified patterns. Thus, such patterns attend to configure the ANNs, i.e., weights and neuron connections, which in turn indicate whether a collection of events (not necessarily in order) can be classified into a given context representation (see Fig. 2). ANNs are implemented using the Weka [54] library.

Table 1
A sample of tags extracted during this work experimentation.

Description	Tags
Web	Access, attempt, attack, command, directory, password, crossite, httpinspect, php, administration, script, scan, overflow, error, auth, DoS, highseverity, response, iis
Configuration	Conf, disclos, reveal
Sql	Sql, oracle, odbc
Injection	Injection
Virus	Virus, trojan, ware, spam, bot, infection, propagation, virexploit, irc, telnet, dtelnet, ftp, dftp, smb, dsmb, vir_request, vir_response, trojan_blacklist, ware_blacklist
Scan	Netscan, portscan
Crosssite	Crosssite, xss

Table 2
Initial set of patterns describing an attack scenario for a given context C_{web}. Each pattern p_m^{web} in each row represents the number of events with tag t_j and serves as input for each ANN. Here $f^m(t_{access}) = \{0, 0, 44, \ldots\} \forall m = \{1, 2, 3, \ldots\}$.

	$f(t_{access})$	$f(t_{attempt})$	$f(t_{attack})$	$f(t_{cmd})$	$f(t_{dir})$	$f(t_{xss})$	$f(t_{pwd})$	$f(t_{auth})$	$f(t_j)$	Out
p_1^{web}	0	0	0	0	0	0	0	0	0	...
p_2^{web}	0	0	0	0	0	0	0	0	...	No
p_3^{web}	44	31	13	0	0	1	0	...	7	Yes
p_4^{web}	463	0	0	1	462	0	...	462	0	Yes
p_5^{web}	50	19	8	0	0	...	0	8	20	Yes
p_6^{web}	50	30	14	0	...	4	0	3	4	Yes
p_7^{web}	563	0	0	...	0	563	563	0	0	Yes
p_8^{web}	64	49	...	0	0	1	0	4	0	Yes
p_9^{web}	653	...	20	0	0	0	0	0	0	No
p_m^{web}	...	6	649	0	0	1	0	0	1	No

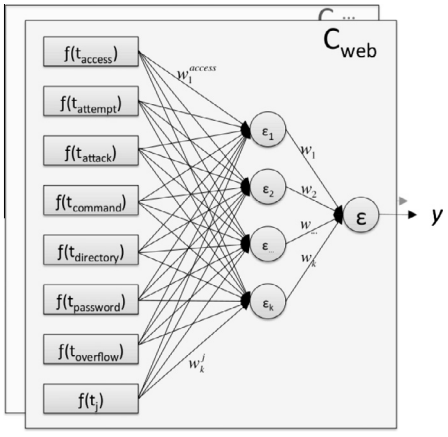

Fig. 2. Each ANN is a multilayer perceptron which maps a set of input patterns (*Knowledge Base*) with a number of incoming events. Let ε_k denotes a neuron and w_j the associated weight of each neuron connection. Function $f^m(t_j)$ returns the number of events tagged with $t_j \in \mathcal{T}$ for each tag encountered in C_i and for each pattern in $p_m^i \in \mathcal{P}$. Each ANN produces an appropriate output y (see Eq. (1)) for each C_i previously defined.

Hence, we deploy a number of multilayer feedforward ANNs which learn from a limited training set, i.e., the *Neural Network Knowledge Base* by executing the phases below:

1. *Pattern Generation Module.* Human expertise on a context is applied towards the specification of $\mathcal{P}$, building the model to relate a series of events to a concrete instance of a given multi-step attack C_i. Hence, a total number m of patterns $p_m^i \in \mathcal{P}$ is extracted and stored into the *Neural Network Knowledge Base* for each C_i.
   In order to create a training set for C_i as in Table 2, this process requires the identification of positive (the *yes*-taught set) as well as negative samples (the *no*-taught set). Samples are then chosen such that representing the attack scenario and its variations.
   Each ANN is trained offline to produce the output such that satisfying Eq. (1) for each $p_m^i \in \mathcal{P}$ of C_i:

$$y = \sigma_{p_m^i}(\varepsilon) = \begin{cases} yes & \text{if } \varepsilon \geqslant h \\ no & \text{if } \varepsilon < h \end{cases}$$
$$\text{where } \varepsilon = \sum_{i=1}^{k} w_i \cdot \sum_{j=1}^{|\mathcal{T}|} w_i^j \cdot f^m(t_j) \qquad (1)$$

where h denotes the threshold generated and adjusted by each ANN to achieved the expected classification for inputs, and $f^m(t_j)$ returns the total number of events tagged using t_j for a given pattern p_m^i of the context C_i. We run our experiments for a number of neurons which started small and kept increasing until satisfactory results were obtained.

Each ANN learns the best configuration (the reliability ratio is 100%) in order to satisfy the defined context representation. For example, receiving a number of 15 events tagged as *xss* and 12 *web_access* should match the *crosssite scripting* context C_{xss}. By contrast, considering a number of 30 events tagged as *web_error*, an 2 *web_auth* does not represent any threat.

2. *ANN Deployment.* Once trained, the ANNs are deployed in a real networking environment. Thus, the ANNs receive as inputs real-time events and classify them with the corresponding context label, e.g. `DDoS`. Moreover, new encountered patterns will be dynamically added on the *Neural Network Knowledge Base* which makes false positives to decrease rapidly.
   Participation of a human expert is also required at the end of this phase to carefully examine the knowledge base of *Positive Events*, $\mathcal{PE}$. This database consists of the events classified as any of the context representations analyzed by the ANNs deployed. In particular, the expert should inspect the patterns and reconfigure the ANNs, if needed, in terms of the following parameters: momentum, learning rate, and training time (see Weka [54] library for details on these parameters). This process represents the so-called event consolidation to a central database which is the output of CONTEXTUAL.

A note on the consolidation of the new encountered patterns. Preprocessed events in $\mathcal{PE}$ are correlated afterwards, most in the way of an information fusion system does. In our experimentation we test CONTEXTUAL output on both, the present correlation engine of `OSSIM` by default and also on the enhanced correlation subsystem using GP namely GENIAL (which is presented in the section below). It is interesting to note that CONTEXTUAL performance gets better results if GENIAL is executed straight after. By contrast, `OSSIM`'s correlator is not efficient interpreting CONTEXTUAL output even though this output is presented in the correct format.

5. GENIAL: GP-based event correlation subsystem

In this section, we apply GP to enhance the OSSIM's correlation engine aimed at automatically generating event correlation directives. A complete set of directives is an XML representation of security events as described further below. Basically, we implement GENIAL based on a Java-based Evolutionary Computation (ECJ) Research System which comprises the following phases (see also Algorithm 1):

1. **Preprocessing**. ANNs' outputs create a training set to guide a supervised learning of the behavior of a given context representation C_i and then assist correlation in automatically inferring the potential relationships amongst events. Important piece of correlation information is inferred in this stage by inspecting the events received.

2. **Representation of individuals**. An individual represents a correlation directive as a combination of rules connected via operator functions such as *AND* and *OR*. *Rules* are written as lists of conditions that should be satisfied between event log information (or characteristics of the events received) and a series of rule attributes described below.
3. **Initialization of the population**. A population consists of a forest of individuals (with a tree-based representation). The population is randomly initialized according to the inferred information extracted during the *Preprocessing* phase.
4. **Evaluation of individuals**. During a sequence of generations, individuals are evaluated using the training set created in the *Preprocessing* stage.
5. **Breeding**. By applying crossover and mutation operators on individuals, new individuals (offprints of the previous generation) are generated.

Subsections below describe these phases in detail.

5.1. Preprocessing

The purpose of preprocessing the events extracted from the ANNs is twofold. First, we need to construct a training set which supervises the evaluation of individuals as described in Section 5.4. Secondly, preprocessing assists us in inferring specific correlation information for the context representations (or simply context hereafter) identified.

On one hand, we have two different types of events, namely.

Definition 4. A positive context compiles the set of events that were classified as "positive" by CONTEXTUAL's set $\mathcal{PE}$.

Definition 5. A negative context compiles the set of events that were classified as "negative" by CONTEXTUAL and also were obtained randomly from other contexts.

Hence, context specifications, i.e. outcomes of each ANN, are all labeled as *Positive* and belong to the set $\mathcal{PE}$ in Algorithm 1 – line 1. On the contrary, unclassified events form the set of *Negative Contexts* denoted as $\mathcal{N}$ in Algorithm 1 – line 2. However, both collections will produce the training set which is generated by splitting them into several subsets of N-grams [55] called *Registers* (and denoted by R in Algorithm 1 – line 3). Thus, we have.

Definition 6. A register $r_i \in \mathcal{R}$ represents a sequence of k events, and are produced by different combinations between both *Positive*, $\mathcal{PE}$, and/or *Negative*, $\mathcal{N}$, contexts, i.e., $\mathcal{R} = \{r_1, \cdots, r_n\}$ $s.t. r_i \subset \mathcal{PE} \vee \mathcal{N}$.

A *Positive Register* compiles *Positive Contexts*, i.e., $r_i = \{E_1, \cdots, E_k\}, k > 0, s.t. \forall E_j \subset \mathcal{PE}$, whereas a *Negative Register* gathers together events from $\mathcal{N}$. Therefore, in this stage two files are generated to train a population of events, namely.

- A *positive training set* is formed by the collection of T_p positive registers.
- A *negative training set* consists of the total amount T_n of encountered negative registers.

On the other hand, by inspecting event characteristics from *positive contexts* in $\mathcal{PE}$ we can infer very useful information for the correlation process, especially for representing individuals in our domain. In particular, we have identified a limited number of *attributes* extracted from the received events which capture intrinsic event properties for the contexts performed in our evaluations. Hence, a number of rules will be constructed in the following stage of GENIAL which relate event characteristics to attributes so evaluating if a certain condition is satisfied, as shown in Table 3. From our conducted experiments, we define the following attributes of rules: *name*, *type*, $plugin_{id}$, $plugin_{sid}$, $addr_{from}$, $addr_{to}$, $port_{from}$, $port_{to}$, *protocol*, *reliability*, *timeout*, *occurrence*, *sticky*, *sticky_difference*, *condition*, *value*, *interval*, *absolute*, and *reliability_abs* (for further details on these attributes refer to [56,18]).

In this regard, events will be classified into categories according to the attributes in common. Thus, we can organize events in *categories* as.

Definition 7. A number of events are classified into the same category if values of their event characteristics match up with the following attributes: $addr_{from}$, $addr_{to}$, $port_{to}$, $plugin_{id}$, and $plugin_{sid}$.

Apart from these attributes, the existing timing relationships between any pair of consecutive events are also extracted from events' timestamps in $\mathcal{PE}$ to correlate temporal conditions (*timeout*).

5.2. Representation of the individual

In this phase, we deal with the representation of candidate solutions to our correlation problem which are represented by individuals in a population, i.e., in the form of correlation directives. A correlation directive is an XML data structure with the following schema:

- Directive → (Rule). *Directive* consists of a unique *Rule*. Attributes of directives are an *identifier*, a *name* and a *priority*.
- Rule → (Rules). *Rule* comprises at least one rule in *Rules*.
- Rules → (Rule+). Element *Rules* mandatorily has one or more occurrences of *Rule*. There are no specific attributes for this element.

GP defines individuals as trees where intermediate nodes (or simply nodes) have an operator function and every leaf node has an attribute of our model. In the experiments conducted, the operator for non terminal nodes executes operators like AND and OR, whilst attributes for terminal nodes consist of `OSSIM` rules. For example, put simply an individual such the expression: $rule_x$ *AND*

Table 3
Event characteristics used to identify correlations between positive-tagged contexts and attributes of a rule. We mark with a √ when the attribute is applicable, and with – if no correlation is achieved.

	Category							
	$addr_{from}$	$addr_{to}$	$plugin_{id}$	$plugin_{sid}$	$port_{to}$	$port_{from}$	*protocol*	*timeout*
Number of events per	√					–	–	–
	√	√	√	–	√	√	√	–
	–	–	√		–	–	–	–
Protocols per	–	–	√		–	–	–	–
Time range	–	–	–	–	–	–	–	√
Max slot time	–	–	–	–	–	–	–	√
Min slot time	–	–	–	–	–	–	–	√

($rule_y$ *OR* $rule_z$). However, individuals should meet the following constraints according to the aforementioned XML schema [18]: (i) nodes have at most two "sibling" nodes, (ii) nodes' children will be either leaves (i.e. rules) or AND operations, and (iii) the root of each individual cannot be an OR operation.

Now, to establish the best configuration for individuals in our domain, we have organized rule attributes into two different types, as follows:

- A number of rule attributes, which are referred as *Relevant* attributes, are applied to find equivalences between events.
  In particular, main relevant attributes for events are source and destination addresses ($addr_{from}$,$addr_{to}$), their corresponding ports ($port_{from}$,$port_{to}$), the sensor identifiers (i.e. $plugin_{id}$ and $plugin_{sid}$), and their *timestamp* when detected.
- A number of additional attributes, which are called *Advanced* attributes, are applied to the evaluation process.
  Attributes of this type are *priority*, *reliability*, *occurrence* and *timeout*.

Hence, *Relevant attributes* are those which identify sensory data from both the attacker and the victim viewpoints. However, *Advanced attributes* comprise information regarding the probability of success of a given attack and the importance of the asset within the organization environment. Therefore, *Advanced attributes* depend on the organization network, whereas *Relevant attributes* rely on the particular attack context.

5.3. Initialization of the population

ECJ development toolkit provides a tree building method, namely the class `HalfBuilder`, which implements a complete procedure to initialize a population of individuals. In this process, nodes are inserted into the tree according to a certain probability which determines how much the tree spreads itself.

Additionally, ECJ provides a friendly parameter file to specify all the aforementioned constraints. However, *Advanced attributes* intrinsically depend on the length and width of the generated trees, and also on the information of the *Relevant attributes* which is randomly filled out. Thus, we modify the ECJ initial population builder (as described in Algorithm 1 – lines 4–7) in order to revisit the generated tree by performing the following two phases:

1. A new method is added aimed at counting the total number of leaves which match up with $plugin_{id}$ and $plugin_{sid}$ values for each tree branch.
2. A new method is included which uniformly distributes the attributes *timeout*, *reliability* and the total number of *occurrences* stored in the *positive training set* along the tree branches.
3. An heuristic to aggregate events is also added based on the source and destination IP addresses and ports information contained within events. Basically, a tuple of events can be aggregated in a *N*:*M* topology manner, where *N* is the number of different sources and *M* represents destinations. Thus, we generate different rules according to the following types of interaction: (i) unidirectional (aggregating events to/from a given computer), (ii) bidirectional (putting together events exchanged between a given pair of computers), and (iii) multi-directional (aggregating in terms of a different attribute as there is no correlation between IP addresses). This heuristic facilitates population building and systematically categorizes correlations between events according to the way events are aggregated.

5.4. Evaluation of the individuals

The process of evaluating individuals is supervised by the training sets as follows. First, each individual is evaluated using both types of registers, i.e., *positive* and *negative*, previously generated (see details in Fig. 3). Basically, trees are visited by applying pre-order traversal which recursively visits each node on the left and right subtrees from the root. In particular, nodes are evaluated in the following terms:

- Leaf nodes are evaluated according to the events in each register. In this regard, event characteristics are compared with the *Relevant attributes* of the rule being evaluated. Additionally, we seek for potential correspondences between each rule and the number *occurrence* of events in chronological order.
- Nodes are evaluated by applying AND and OR functions to the values produced in the evaluation of their children.

Once individuals have been iterated, they will be classified into:

- A *positive individual* returns *true* as a positive register matches up; or
- A *negative individual* occurs when a negative register evaluated over the individual returns *false*.

Thus, we make the population evolve for a series of generations in which a *fitness* function is evaluated on every individual (Algorithm 1 – line 10). The fitness function is used for measuring and ranking every individual based on its quality to represent a candidate solution. Since our objective is to maximize the number of positive classifications over *positive registers* as well as to maximize the number of negative classifications over *negative registers*, we have defined the fitness function as

$$fitness = 1 - \frac{(P+N)\times P}{(T_p+T_n)\times T_p} \tag{2}$$

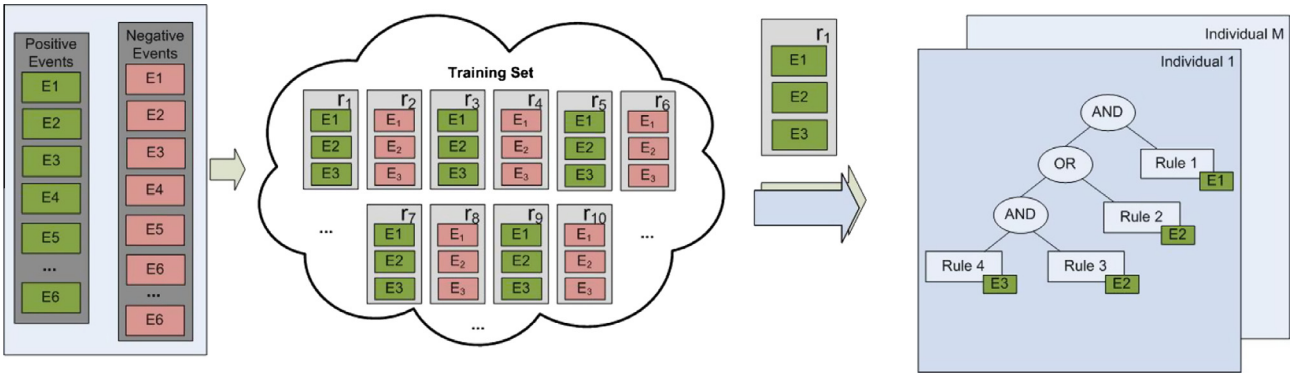

Fig. 3. Evaluation of register *r*1 over a given individual.

where *PN* and *N* are the number of positive and negative classifications respectively, and T_p and T_n are the total number of positive and negative registers evaluated. Fitness values can therefore take any nonnegative value between [0.0,1.0], being 0.0 the value for best individuals and 1.0 for the worst ones. Consequently, a selection of individuals is made during each generation based on their fitness, as shown in Algorithm 1 – line 11. To sum up, the higher the number of negative/positive classifications, the better the individuals' fitness is.

Algorithm 1. Proposed algorithm for GENIAL

1: $\mathcal{PE}$ ={Collection of *Positive Events*}
2: $\mathcal{N}$={Collection of Negative Events}
3: $\mathcal{R}$ ={r_1,r_2,r_3, . . . , r_n/$r_i \subset P \vee N$}
4: *Individuals* ← {Initialization of the population}
5: *Individuals* ← {Initialization of the *Relevant* attributes}
6: *Individuals* ← {Initialization of the *Advanced* attributes}
7: *directive* = *null*; $best_{individual}$ = *null*; $best_{fitness}$ = 1
8: **for all** *Generations* **do**
9: **for all** *i* in *Individuals***do**
10: *fitness* = *eval*(*i*,*R*) as in Eq. (2)
11: **if***fitness* is better than $best_{fitness}$**then**
12: $best_{fitness}$ = *fitness*
13: $best_{individual}$ = *i*
14: **end if**
15: **end for**
16: *Individuals* ← {Breed (Individuals) ⋃ $best_{individual}$}
17: **end for**
18: *directive* = ToOssimSintax ($best_{individual}$)
19: **return** *directive*

Important remarks. To define the appropriate fitness function is always hard and depends on the problem domain. For instance, such a function is generally determined by trial–and–error. We run our experiments considering different fitness functions which gave us worse results. For example, we have analyzed the function based on the risk assessment present in OSSIM by default:

$$Risk = \frac{(Asset \cdot Priority \cdot Reliability)}{25} \quad (3)$$

Function in Eq. (3) follows an heuristic algorithm which establishes a risk that the monitored asset might undergo for a specific thread (i.e. the reliability of the possible attack) and it is also based on the directive priority. This function has been proven to report false positives and to ignore relevant characteristics presented in several well-known datasets.

5.5. Breeding process

In this phase, the population is evolved by the recombination of individuals by applying selection, crossover and mutation operators. A tournament selection is performed so that the best individuals will survive in the next generation (Algorithm 1 – line 16). A new set of individuals are therefore generated by forcing mutation on the selected individuals.

We have modified the ECJ method called `MultiBreedingPipeline` to revisit the generated tree as advanced attributes are modified in each generation.

Finally, the best individual, i.e., the correlation directive which optimally fuses the events related to a specific context, is exported from GENIAL to OSSIM using its native syntax (Algorithm 1 – line 18).

6. Experimental evaluation of our system

We performed several experiments in order to evaluate our genetic-based framework for a self-adaptive event correlation.

6.1. Experimental testbed

A botnet (Fig. 4) has been implemented to disseminate and launch a number of attacks. Several zombies distribute malware from the master to all the bots. In particular, two types of DDoS are launched in our testbed: HTTPFlood and ICMPFlood. We create 30 zombies and 10,000 floods per zombie. Generated events are used by our framework to create the training set, as described in Section 5. Once the best individual is reached, it is parsed according to the correlation engine syntax, exported to the OSSIM, and re-evaluated afterwards. OSSIM is deployed into the testbed along with a number of both types, host and network-based, of intrusion detection agents such as `Snare` [57] and `Snort` [58].

6.1.1. DDoS attacks experiment: HTTPFlood and ICMPFlood

We analyze and compare correlation directives generated by OSSIM (versions 1.0.6 and 2.0.1) correlation engine with those produced by our framework for the detection of HTTPFlood and ICMPFlood attack contexts.

OSSIM correlation directives by default. On one hand, experiments show that `OSSIM v1.0.6` is proven to be non-efficient in detecting the HTTPFlood attack. In fact, sensors not only provide useless but also erroneous events as they report a "port scan" classification. The latter mistake is possibly derived from the web server while processing numerous bot's requests. More precisely, for each HTTP request, the server opens a new port, so behaving similar to a port scan. On the other hand, `OSSIM v2.0.1` enhances the configuration of `Snort` sensors triggering the following event: "*COMMUNITY SIP TCP/IP message flooding directed to SIP proxy*", and the following duplicated alert: "*Strange host behavior on SRC_IP*". However, OSSIM cannot produce a directive that correlates the reported sensory data and the web-server's log. Similarly, there is no alert reported when launching the ICMPFlood attack.

Directives produced by our framework. We train our genetic framework to detect the same attack. A total of 20,480 individuals were randomly generated and evaluated. The best individual showed a fitness of 0.250. This individual presented a number of 22 hits for the positive training set out of 27 (81%) and a number of 122 hits for the negative training set out of 122 (100%). After including the produced directive and executing the attack again, OSSIM was capable of successfully detecting the DDoS attack without human intervention. Additionally, the attack was detected within the 10 s soon after, whilst the DDoS alert showed the maximum rate of risk. Furthermore, only one alert was triggered whereas there exists more than one alert for the same attack in the default correlation. On the contrary, there is no alert triggered during normal operation of the web server.

We refer interested readers to Appendix A for further details on the experimentation results. In particular, Tables A.7 and A.8 show results for the correlation of HTTPFlood and ICMPFlood attacks respectively in the scenarios above.

6.1.2. Metasploit penetration experiments

Metasploit framework (MSF) [59] is an open-source tool-kit, which provides a framework to identify security issues, verify vulnerability mitigation, and manage expert-driven security assessments. We include an evaluation of our system considering the attacks occurred within our testbed when MSF is used to attack a vulnerable victim. A number of penetration tests based on MSF are presented and described in Appendix A.2 and Table A.9.

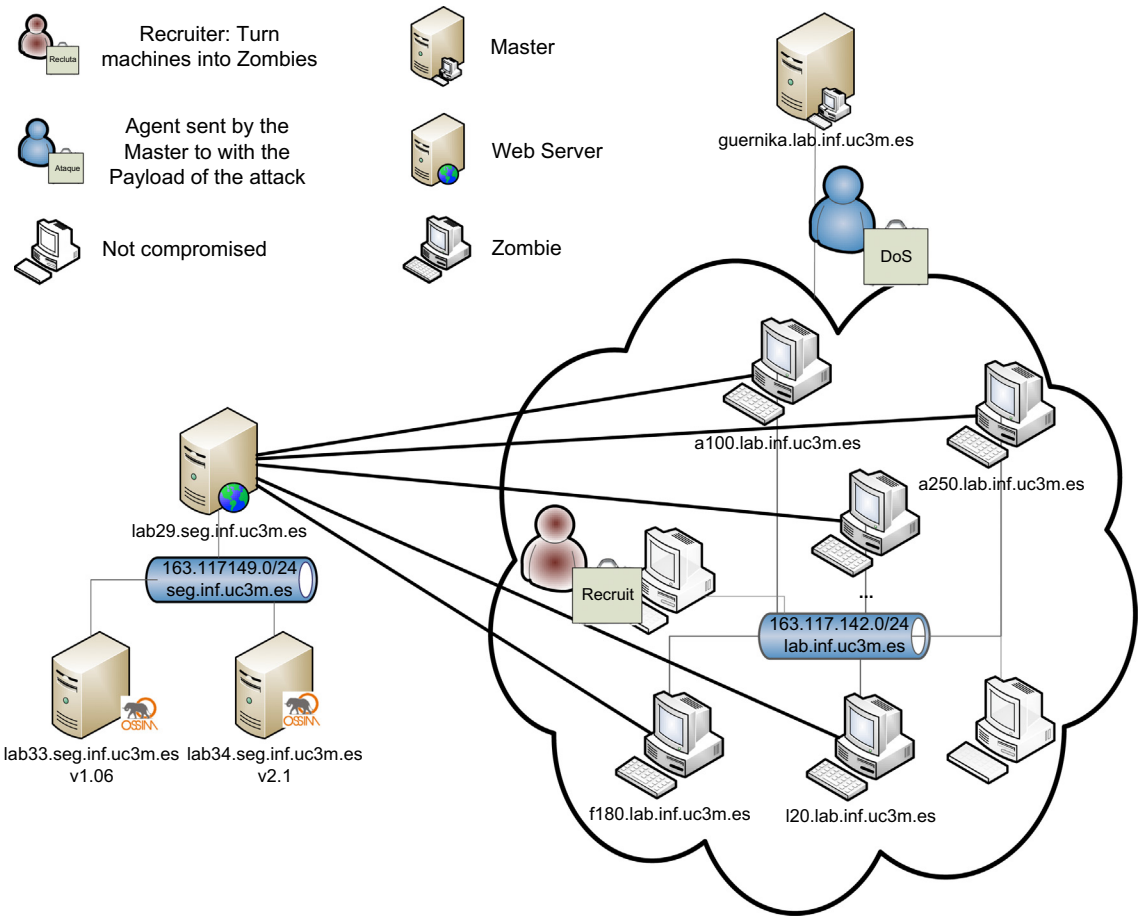

Fig. 4. Experimental testbed where a DDoS attack is launched against a monitored server by a botnet deployment. Events generated are exported and used as the training set.

OSSIM correlation directives by default. OSSIM configuration by default is not capable of processing incoming events from the attacks executed. Each sensor is configured to recognize specific network packets or log files from the exploits. Thus, sensors are reporting several de–correlated events for each exploit.

Directives produced by our framework. We train our genetic framework to detect the same MSF attacks. After inserting the resultant directives and executing the attacks again, the OSSIM system can successfully detect each of the exploits shown in Table 4. It is worthy of mentioning that attacks are detected while they are still occurring as well as no single false positive is detected.

Table 4
Evaluation of several complex-to-detect attacks using `Metasploit` penetration toolkit.

Exploit	Type	Run time (s)	Fitness
ms08_067_netapi	SMB exploit	10	9.76
amaya_bdo	Browser exploit	11	8.86
ms_06_57_webview_setslice	Browser exploit	8	8.76
ms06_067_keyframe	Browser exploit	9	8.71
adobe_getico	Browser exploit	13	9.30
msvidctl_mpgeg2	Browser exploit	51	7.95
ms10_002_aurora	Browser exploit	10	8.71

6.2. Real deployment

Our enhanced SIEM system is deployed in a large-scale network infrastructure in production. Both CONTEXTUAL and GENIAL subsystems have been integrated into an `OSSIM` instance located at a vantage point in the network. Additionally, various agents have been placed all around the intra-building local area network links as well as at end-user workstations and other system services such as web, proxy, and backup servers.

Results of the observations made during a period of a week are then commented and discussed. More specifically, a total of 15,000 bulky and complex flow of different events per second has been processed on average. Given the complexity of the network topology and considering the environment, it is crucial to tailor the monitoring system according to the (i) different security needs and capabilities, (ii) different assets and valuables to protect, and (iii) different threat scenarios against the networked systems. In this regard, OSSIM offers a number of open source tools to evaluate each of the assets deployed in the environment.

Analysis of the observations. To determine the efficiency and accuracy of our framework, two scenarios have been considered. First, a stand-alone `OSSIM` instance was used for security monitoring. Secondly, our framework was deployed in a separate instance in line with the first scenario. Measurement has been tackled during a period of 11 days. Fig. 5 shows the total number of correlation alerts reported for each scenario. During those days, observed events were manually inspected whereas reported alerts were classified into (i) new or zero-day attacks (ZDA), (ii) true positive (TP), and (iii) false negative (FN). ZDAs are those intrusions that were only discovered by our framework in the second scenario. TPs are those intrusions that triggered an alert, whereas FNs are those alarms that were triggered due to no intrusion in both scenarios. We discovered that, on one hand, our framework detected a number of ZDAs which were not detected by the stand-alone instance of `OSSIM`. Table 5 enumerates intrusions discovered and

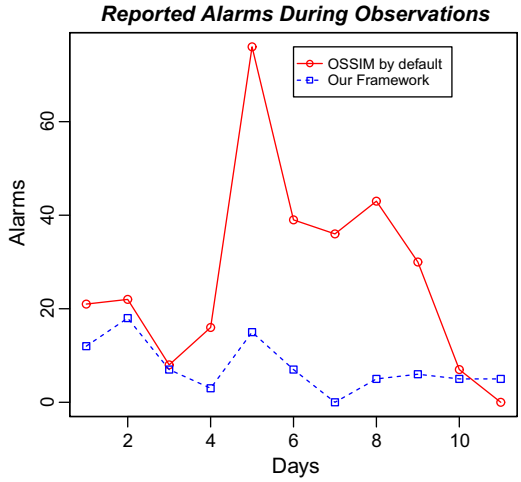

Fig. 5. Alerts observed in the two scenarios deployed: (i) stand-alone OSSIM instance, and (ii) our CONTEXTUAL and GENIAL framework integrated into OSSIM.

Table 5
Evaluation of several multi-step attacks reported by our framework in a real networking environment.

	Categories	Run time (s)	Fitness
Botnet anomaly	4	10	9.5
`Bredavi` trojan A	5	19	7.57
`Bredavi` trojan B	6	25	7.91
`Conficker` solaris	14	36	7.95
`Dell` remote access	17	42	7.95
Brute force SSHD	18	18	7.75
Port scan	24	123	7.94
Spyware	31	848	8.06
Telnet worm	56	85	8.09
Web scan	57	51	7.95

their evolving features as well. On the other hand, 53 out of 61 (86.89%) TP alerts were only reported by our framework. Whereas, the total number of FNs reported in both scenarios, just 30 out of 340 alarms (8.82%) were reported by our framework.

6.3. Discussion

In this section, we present a qualitative discussion on the goodness of our framework in terms of the observations obtained during the conducted empirical analysis. In that regard, we compare our approach with the most relevant related works presented so far. Table 6 summarizes our findings during this qualitative analysis mainly focuses on examining the level of human participation and adaptability of the approaches.

In particular, we find in [8] the least efficient framework in which neither adaptation nor intelligence are introduced into the aggregation and correlation processes. Thus, human involvement is high.

Automatic correlation approaches appear on the basis of prerequisites and consequences such as [9]. Although their experiments demonstrated the potential of their framework, it still requires an extensive and significant effort identifying both, prerequisites and consequences, for each possible scenario. By contrast, our proposal optimizes this task by allowing the expert to introduce his/her prior knowledge of the context (i.e., a generic/specific misuse activity) into the reported sensory data for an efficient classification and fusion of related events.

Zhu et al. [65] presented probabilistic alert correlation based on the extraction of attack strategies from a static training file. Their model applies machine-learning techniques but is not able to automatically relate events to the corresponding attack in real-time. Thus, similar algorithms such as [61–63,60] are not effective for real deployments.

Multi-step intrusions started to be addressed from different perspectives. For example, the approach proposed by Zhou et al. in [66] studied the logical relationships between events of multi-stage attacks aimed at automatically deriving what they called inference rules to define such relationship. Authors assume that events are previously classified into capabilities (a *knowledge base* that identifies variants of the same attack). Similarly, Sadoddin et al. [69] proposed the application of data mining techniques to perform frequency analysis and pattern structure extraction on a real-time framework. However, both approaches assume that the sensors do not report FN alarms, but considering that all extracted correlations will be effective. Based on our experiments, such assumptions have proven to be unrealistic and invalid for ZDA detection. Furthermore, a major drawback for such a frequency-based learning leads to new correlations can only be discovered if the underlying attack is launched repeatedly.

Human intervention is decreased as some sort of context-based knowledge is considered. For example, Zhou et al. [67] use a decentralized, multi-dimensional alert correlation algorithm which first integrates events locally at each sensor into eight different types of clusters with similar attack topology. The proposed clustering is based on a pre-defined set of patterns so unknown attack strategies could easily evade aggregation made locally. Furthermore, Joshua et al. [39] also presented a clustering method to reduce the sensory data reported. However, the classification method is limited as it is trained using static datasets. On the contrary, our approach includes dynamic data feedback which evolves according to the context representations occurred in the network.

Finally, Wang et al. [64] applied attack graphs (graphical representations of the existing interdependencies between vulnerabilities and connectivity in the network) to correlate and hypothesize events generated from both, known and unknown, attacks, and

Table 6
Qualitative comparison with respect to related works.

Approach	Human intervention	Automatic classification	Automatic correlation
Aggregation [8]	High	No	No
Pre-requisites/Consequences [9]	High	No	Yes
Scenarios generation [60]	High	Yes	Yes
Objective-based correlation [61]	High	No	Yes
Similarity-based correlation [62]	High	No	Yes
Attack graph correlation [63]	High	No	Yes
Attack graph correlation [64]	Low	No	Yes
Probabilistic correlation [65]	High	No	Yes
Inference-rules correlation [66]	High	No	Yes
Multi-dimensional correlation [67]	Medium	Yes	No
Similarity-based clustering [39]	Medium	Yes	No
Attack graph correlation [68]	Low	Yes	Yes
Our approach	Low	Yes	Yes

even to predict future alerts. From their experimentation, attack graphs present better performance than other machine learning-based approaches. Moreover, Ahmadinejad et al. [68] extended Wang et al.'s work towards self-adapting over unknown attacks. However, both proposals lack of robustness when unknown attacks present high dissimilarity to the current knowledge base. In fact, experiments conducted in [68] show correlations when partial graphs of the attacks (from DARPA 2000 dataset) are presented, whilst our framework is able to correlate ZDA attacks (i.e. when host vulnerabilities are not present in the model).

In summary, our approach differs from the previous, so overcoming others' common drawbacks as:

- Adaptation is considered by design. Our framework is self-adaptive in both processes, classifying sensory data based on attack contexts as well as generating correlation rules. Moreover, it has been proven that our system decreases the human intervention and prevents against ZDAs.
- Efficiency is considered by design. Our framework has been proven to be efficient and accurate when deployed in a real environment. In fact, our system presents efficiency in the way of true positive rate is increased whilst false positive rate decreases.
- Context-based knowledge is considered by design. AI-based approaches presented so far in security information fusion domains do not explicitly consider contextual information for correlation purposes, suffering from the classical problem of generality in AI [70].

7. Conclusion and future work

In this paper, a cutting-edge scheme tackling the design of a self-adaptive SIEM system is introduced. The novelty of our proposal falls on the adopted approach which can be summarized in the following terms. From the point of view of the optimized SIEM, the framework allows SIEM systems to automatically learn attack signatures based on contextual knowledge as well as to automatically produce optimum correlation directives. Two main subsystems are therefore introduced: a context-based event classifier based on ANNs, called CONTEXTUAL and, an enhanced SIEM correlation engine based on GP, called GENIAL. From the point of view of the human operator, our self-adaptive SIEM system considerably decreases the human supervision to some extent. Furthermore, we carried out a real integration of our subsystems into a `OSSIM`-monitored environment to evaluate the goodness and feasibility of our proposal.

Our future work is now focused on the following enhancements to the proposed system especially within CONTEXTUAL, which includes:

- The application of clustering techniques on the extraction of tags which will provide CONTEXTUAL with self-adaptation so eliminating the expert intervention in that matter, and also
- The application of filtering techniques during the compilation of relevant events, e.g. by introducing honeypots.

Table A.7
HTTP Flood.

OSSIM version	Num. zombis	Type of event	Num. events	Type of alert	Num. alerts	Risk
1.0.6	1	Spade: Non-live dest used	16	TCP Portscan against Web Server	5	5
		pam_unix: authentication successful	2			
		portscan: Open Port	396			
	5	Spade: Non-live dest used	1	TCP Portscan against Web Server	1	2
		pam_unix: authentication successful	1			
		portscan: Open Port	862			
	10	Spade: Non-live dest used	0	TCP Portscan against Web Server	1	5
		pam_unix: authentication successful	1			
		portscan: Open Port	526			
	30	Spade: Non-live dest used	2	TCP Portscan against Web Server	14	5
		pam_unix: authentication successful	11			
		portscan: Open Port	61			
2.1	1	snort: COMMUNITY SIP TCP/IP message flooding directed to SIP proxy	36	–	–	–
		rrd_threshold: ntop global upTo64Pkts	1			
		p0f: OS Same	1			
	5	snort: COMMUNITY SIP TCP/IP message flooding directed to SIP proxy	195	–	–	–
		rrd_threshold: ntop global upTo128Pkts	5			
	10	snort: COMMUNITY SIP TCP/IP message flooding directed to SIP proxy	462	Strange host behavior on SRCIP	2	0
		rrd_anomaly: ntop global				
		rrd_threshold: ntop global				
		p0f: New OS	6			
		p0f: OS Same	6			

◇ [Discovery]*Snort*: port scan detected.
◇ [Exploitation] *Snare*: Logon Fail-Unknown user name or bad password.
◇[Exploitation] *Snare*: The Windows Firewall has detected an application listening for incoming traffic.
◇ [Exploitation] *Snare*: Account Used for Logon by.
◇ [Exploitation] *Snort*: SMB protocol negotiation.
◇ [Exploitation] *Snort*: NTLMSSP session with unauthenticated user.
◇ [Payload–infection] *Snort*: Meterpreter payload infection.
◇ [Payload–infection] *Snare*: A new process has been created.

Acknowledgements.

This work is supported by CDTI, Ministerio de Industria, Turismo y Comercio of Spain in collaboration with Telefónica I + D; project SEGUR@ with reference CENIT-2007 2004.We would also like to thank the anonymous reviewers for their insightful comments which certainly helped us to improve our article.

Appendix A. Simulation results

A.1. DDoS experiment

In this appendix, results of the simulation conducted in our testbed are provided. In particular, two types of DDoS attacks were launched namely HTTP flood and ICMP flood. Tables A.7 and A.8 give details on the results of the evaluations on our experimental testbed.

A.2. Metasploit experiment

In this appendix, results of the penetration experiments conducted on our testbed are provided. In particular, two isolated computers are deployed, namely the *victim* and *attacker* computer. The victim is a vulnerable Windows XP configured with Snare Event Log Agent [57], and events are reported to the OSSIM v3 – located in the same network as the victim. On the other hand, the attacker is configured with MSF v3, and it is located in a different network than the victim. Victim's vulnerabilities are chosen based on their prevalence and exploit code availability to demonstrate the validity of our framework. The attacker uses selected ex-

Table A.8
ICMP flood.

OSSIM version	Num. zombis	Type of event	Num. events	Type of alert	Num. alerts	Risk
1.0.6	1	Spade: Non-live dest used	10	TCP Portscan against Web Server	2	5
		pam_unix: authentication successful	2			
		portscan: TCP Portscan	4			
		portscan: Open Port	1			
		Spade: Source used odd dest port	1			
	5	Spade: Non-live dest used	18	TCP Portscan against Web Server	7	5
		pam_unix: authentication successful	7			
		portscan: TCP Portscan	0			
		portscan: Open Port	16			
		Spade: Source used odd dest port	3			
	10	Spade: Non-live dest used	21	TCP Portscan against Web Server	13	5
		pam_unix: authentication successful	45			
		portscan: TCP Portscan	4			
		portscan: Open Port	19			
		portscan: TCP Portsweep	75			
2.1	1	snort: COMMUNITY SIP TCP/IP message flooding directed to SIP proxy	107	–	–	–
		rrd_threshold: ntop global	4			
		p0f: New OS	1			
		p0f: OS Same	2			
	5	snort: COMMUNITY SIP TCP/IP message flooding directed to SIP proxy	723	–	–	–
		rrd_threshold: ntop global	36			
		p0f: New OS	4			
		p0f: OS Same	6			
	10	snort: COMMUNITY SIP TCP/IP message flooding directed to SIP proxy	1701	–	–	–
		rrd_threshold: ntop global	82			
		p0f: New OS	6			
		p0f: OS Same	13			

Table A.9
Description of exploits used to test our framework.

Exploit	Description
ms08_067_netapi	Parsing flaw in the path canonicalization code of NetAPI32.dll through the Server Message Block (SMB). This vulnerability may allow an attacker to run arbitrary code on the victim's computer
amaya_bdo	A buffer-based overflow exploitation on the AMAYA W3C Web Browser could lead an attacker to perform a to run arbitrary code on the victim's computer
ms_06_57_webview setslice	Internet Explorer vulnerability based on WebViewFolderIcon ActiveX control flow that could allow an attacker to run arbitrary code on the victim's computer
ms06_067_keyframe	DirectAnimation ActiveX control flow that could allow an attacker to run arbitrary code through Internet Explorer on the victim's computer
adobe_getico	Stack buffer overflow exploitation in the Adobe Collaboration methods. This vulnerability may allow an attacker to run arbitrary code on the victim's computer
msvidctl_mpgeg2	Stack-based overflow exploitation on MPEG2 ActiveX control. This vulnerability may allow an attacker to run arbitrary code on the victim's computer
ms10_002_aurora	HTML Object Memory Corruption Vulnerability, which may allow an attacker to run arbitrary code through Internet Explorer on the victim's computer

ploits to compromise the victim. Table A.9 depicts the corresponding exploits.

For the sake of completeness, we include an excerpt of the events reported when running MSF attacks. In particular, `ms08_067_netapi` exploit comprises the following steps: (i) discovery phase, (ii) exploitation phase, and (iii) payload-injection, as follows:

Running *ms08_067_netapi* exploit, the following events are reported by the sensors:

References

[1] R. Gabriel, T. Hoppe, A. Pastwa, S. Sowa, Analyzing MALWARE log data to support security information and event management: Some research results, in: Proceedings of the 1st International Conference on Advances in Databases, Knowledge, and Data Applications, 2009, pp. 108–113.

[2] I. Corona, G. Giacinto, C. Mazzariello, F. Roli, C. Sansone, Information fusion for computer security: state of the art and open issues, Information Fusion 10 (4) (2009) 274–284.

[3] T. Bass, Intrusion detection systems and multisensor data fusion, Communications of ACM 43 (2000) 99–105.

[4] D. Hall, J. Llinas, An introduction to multisensor data fusion, Proceedings of the IEEE 85 (1997) 6–23.

[5] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Addison-Wesley Longman Publishing Co., Inc., 2006.

[6] I. Ahmad, A.B. Abdullah, A.S. Alghamdi, Artificial neural network approaches to intrusion detection: a review, in: Proceedings of the 8th Wseas International Conference on telecommunications and informatics, WSEAS, 2009, pp. 200–205.

[7] C. Zhang, J. Jiang, M. Kamel, Intrusion detection using hierarchical neural networks, Pattern Recognition Letters 26 (6) (2005) 779–791.

[8] H. Debar, A. Wespi, Aggregation and correlation of intrusion-detection alerts, in: Recent Advances in Intrusion Detection, Springer, 2001, pp. 85–103.

[9] P. Ning, Y. Cui, D.S. Reeves, Constructing attack scenarios through correlation of intrusion alerts, in: Proceedings of the 9th ACM Conference on Computer and Communications Security, ACM, 2002, pp. 245–254.

[10] T. Limmer, F. Dressler, Survey of Event Correlation Techniques for Attack Detection in Early Warning Systems, 2008.

[11] W. Lu, I. Traore, Detecting new forms of network intrusion using genetic programming, Computational Intelligence 20 (3) (2004) 475–494.

[12] W. Lee, S.J. Stolfo, K.W. Mok, Adaptive intrusion detection: a data mining approach, Artificial Intelligence Review 14 (2000) 533–567.

[13] R. Sommer, V. Paxson, Enhancing byte-level network intrusion detection signatures with context, in: Proceedings of the 10th ACM Conference on Computer and Communications Security, ACM, 2003, pp. 262–271.

[14] B. Ripley, Neural networks and related methods for classification, Journal of the Royal Statistical Society 56 (3) (1994) 409–456.

[15] J.Z. Lei, A. Ghorbani, Network intrusion detection using an improved competitive learning neural network, in: Proceedings of 2nd Annual Conference on Communication Networks and Services Research, IEEE Computer Society, 2004, pp. 190–197.

[16] C. Yin, S. Tian, H. Huang, J. He, Applying genetic programming to evolve learned rules for network anomaly detection, Advances in Natural Computation (2005) 323–331.

[17] K. Faraoun, A. Boukelif, Genetic programming approach for multi-category pattern classification applied to network intrusions detection, International Journal of Computational 3 (1) (2006) 79–90.

[18] OSSIM, Open Source Security Information Management <http://communities.alienvault.com/community>, visited March 2012.

[19] A. ESM, Enterprise Security Manager <http://www.arcsight.com/products/products-esm/>, visited March 2012.

[20] RSA, Envision <http://www.rsa.com/node.aspx?id=3170>, visited March 2012.

[21] SenSage, Sensage SIEM Solution <http://www.sensage.com/>, visited March 2012.

[22] H. CLW, Compliance Log Warehouse <http://h20338.www2.hp.com/NonStopComputing/cache/523873-0-0-0-121.html> (visited March 2012).

[23] N. Sentinel, Sentinel <http://www.novell.com/products/sentinel/> (visited March 2012).

[24] LogLogic, Log Management and Security Event Management <http://loglogic.com/> (visited March 2012).

[25] netForensics, nfx sim one <http://www.netforensics.com/products/security_information_management/SIM_One/> (visited March 2012).

[26] Bitacora, System of Centralization, Management and Exploitation of a Company's Events <http://bitacora.s21sec.com/> (visited March 2012).

[27] M. Nicolett, K.M. Kavanagh, Magic quadrant for security information and event management, Gartner RAS Core Research Note G 176034 (2010) 1.

[28] R. Bace, P. Mell, Intrusion detection systems, Nist Special Publication, NIST (2001).

[29] D. Casey, Turning log files into a security asset, Network Security 2 (2008) 4–7.

[30] W. Peng, C. Perng, T.L.H. Wang, Event summarization for system management, in: Proceedings of the 13th ACM SIGKDD Internatiuonal Conference on Knowledge Discovery and Data Mining, ACM Press, 2007, p. 1028.

[31] V. Paxson, Bro: a system for detecting network intruders in real-time, Computer networks 31 (23–24) (1999) 2435–2463.

[32] O. Depren, M. Topallar, E. Anarim, M. Ciliz, An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks, Expert Systems with Applications 29 (4) (2005) 713–722.

[33] E. Spafford, D. Zamboni, Intrusion detection using autonomous agents, Computer Networks 34 (4) (2000) 547–570.

[34] S.-C. Zhong, Q.-F. Song, X.-C. Cheng, Y. Zhang, A safe mobile agent system for distributed intrusion detection, in: Proceedings of the International Conference on Machine Learning and Cybernetics, vol. 4, 2003, pp. 2009–2014.

[35] C.V. Zhou, C. Leckie, S. Karunasekera, A survey of coordinated attacks and collaborative intrusion detection, Computers & Security 29 (1) (2010) 124–140.

[36] S.X. Wu, W. Banzhaf, The use of computational intelligence in intrusion detection systems: a review, Applied Soft Computing 10 (1) (2010) 1–35.

[37] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, C. Jalali, H. Javitz, A. Valdes, P. Neumann, T. Garvey, A real-time intrusion-detection expert system (ides), Project interim progress report, SRI International (1992).

[38] S. Brugger, Data Mining Methods for Network Intrusion Detection, Technique Report, UC davis.

[39] N.O. Joshua, Adaptive clustering method for reclassifying network intrusions, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 41, Springer Berlin Heidelberg, 2010.

[40] M. Bykova, S. Ostermann, B. Tjaden, Detecting network intrusions via a statistical analysis of network packet characteristics, in: Proceedings of the 33rd Southeastern Symposium on System Theory, 2001, pp. 309–314.

[41] M. Amini, R. Jalili, H.R. Shahriari, Rt-unnid: a practical solution to real-time network-based intrusion detection using unsupervised neural networks, Computers and Security 25 (6) (2006) 459–468.

[42] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, W.-Y. Lin, Intrusion detection by machine learning: a review, Expert Systems with Applications 36 (10) (2009) 11994–12000.

[43] G. Stein, B. Chen, A.S. Wu, K.A. Hua, Decision tree classifier for network intrusion detection with ga-based feature selection, in: Proceedings of the 43rd Annual Southeast Regional Conference, New York, USA, ACM Press, 2005, pp. 136–141.

[44] S. Owais, V. Snasel, P. Kromer, A. Abraham, Survey: using genetic algorithm approach in intrusion detection systems techniques, in: Proceedings of the 7th Computer Information Systems and Industrial Management Applications, IEEE, 2008, pp. 300–307.

[45] J. Kim, P. Bentley, Towards an artificial immune system for network intrusion detection: an investigation of clonal selection with a negative selection operator, in: Proceedings of the 2001 Congress on Evolutionary Computation, vol. 2, 2001, pp. 1244–252.

[46] H. Debar, D. Curry, B. Feinstein, Ietf rfc 4765 – the intrusion detection message exchange format <www.ietf.org/rfc/rfc4765.txt>, March 2007.

[47] C. Lonvick, Isoc rfc 3164 – the bsd syslog protocol <www.ietf.org/rfc/rfc4765.txt>, August 2007.

[48] S. Mathew, C. Shah, S. Upadhyaya, An alert fusion framework for situation awareness of coordinated multistage attacks, in: Proceedins of the International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems, IEEE Computer Society, 2005, pp. 95–104.

[49] Z. Li, Y. Chen, A. Beach, Towards scalable and robust distributed intrusion alert fusion with good load balancing, in: Proceedings of the 2006 SIGCOMM Workshop on Large-Scale Attack Defense, ACM, 2006, pp. 122–130.

[50] M. Sifalakis, M. Fry, D. Hutchison, Event detection and correlation for network environments, IEEE Journal on Selected Areas in Communications 28 (1) (2010) 60–69.

[51] S. Krishnamurthy, T. He, G. Zhou, J.A. Stankovic, S.H. Son, RESTORE: A real-time event correlation and storage service for sensor networks, in: Proceedings of the 3rd International Conference on Networked Sensing Systems (INSS), 2006, pp. 1–9.

[52] B. Morin, L. Mé, H. Debar, M. Ducassé, M2D2: a formal data model for IDS alert correlation, Proceedings of the 5th International Conference on Recent Advances in Intrusion Detection, Springer, 2002.

[53] J. Saraydaryan, V. Legrand, S. Ubéda, Modeling of information system correlated events time dependencies, in: Proceedings of the 8th International Conferernce on New Technologies in Distributed Systems (NOTERE), ACM, 2008, pp. 1–6.

[54] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. Witten, The WEKA data mining software: an update, ACM SIGKDD Explorations Newsletter 11 (1) (2009) 10–18.

[55] C. Marceau, Characterizing the behavior of a program using multiple-length *n*-grams, in: Proceedings of the 2000 Workshop on New Security Paradigms, NSPW '00, ACM, New York, NY, USA, 2000, pp. 101–110.

[56] G. Suarez-Tangil, E. Palomar, J.D. Fuentes, J. Blasco, A. Ribagorda, Automatic rule generation based on genetic programming for event correlation, in: Proceedings of the Computational Intelligence in Security for Information, Advances in Soft Computing, Springer, 2009, pp. 127–134.

[57] I. Alliance, Snare Event Log Agent <http://www.intersectalliance.com/projects/Snare/> (visited March 2012).

[58] M. Roesch, Snort – Lightweight Intrusion Detection for Networks, in: Proceedings of the 13th USENIX Conference on System Administration, USENIX Association, 1999, pp. 229–238.

[59] L. Metasploit, The Metasploit Framework <http://www.metasploit.com/> (visited March 2012).

[60] O. Dain, R. Cunningham, Fusing a heterogeneous alert stream into scenarios, in: Proceedings of the 2001 ACM CSS Workshop on Data Mining for Security Applications, vol. 13, Philadelphia, PA, 2001.
[61] F. Cuppens, F. Autrel, A. Miege, S. Benferhat, et al., Recognizing malicious intention in an intrusion detection process, in: Second International Conference on Hybrid Intelligent Systems, vol. 87, 2002, pp. 806–817.
[62] P. Ning, D. Xu, Learning attack strategies from intrusion alerts, in: Proceedings of the 10th ACM Conference on Computer and Communications Security, ACM, 2003, pp. 200–209.
[63] S. Noel, E. Robertson, S. Jajodia, Correlating intrusion events and building attack scenarios through attack graph distances, in: 20th Annual Computer Security Applications Conference, IEEE, 2004, pp. 350–359.
[64] L. Wang, A. Liu, S. Jajodia, Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts, Computer Communications 29 (15) (2006) 2917–2933.
[65] B. Zhu, A. Ghorbani, Alert correlation for extracting attack strategies, International Journal of Network Security 3 (3) (2005) 244–258.
[66] J. Zhou, M. Heckman, B. Reynolds, A. Carlson, M. Bishop, Modeling network intrusion detection alerts for correlation, ACM Transactions on Information and System Security (TISSEC) 10 (1) (2007) 4.
[67] C. Vincent Zhou, C. Leckie, S. Karunasekera, Decentralized multi-dimensional alert correlation for collaborative intrusion detection, Journal of Network and Computer Applications 32 (5) (2009) 1106–1123.
[68] S. Ahmadinejad, S. Jalili, M. Abadi, A hybrid model for correlating alerts of known and unknown attack scenarios and updating attack graphs, Computer Networks 55 (9) (2011) 2221–2240.
[69] R. Sadoddin, A. Ghorbani, An incremental frequent structure mining framework for real-time alert correlation, Computers & Security 28 (3) (2009) 153–173.
[70] J. McCarthy, Generality in artificial intelligence, Communications of the ACM 30 (12) (1987) 1030–1035.