Available online at www.sciencedirect.com

SCIENCE DIRECT®

Engineering Applications of Artificial Intelligence 19 (2006) 429–438

Engineering Applications of
ARTIFICIAL INTELLIGENCE

www.elsevier.com/locate/engappai

Alarm clustering for intrusion detection systems in computer networks

Roberto Perdisci*, Giorgio Giacinto, Fabio Roli

Department of Electrical and Electronic Engineering, University of Cagliari, Piazza D' Armi, 09123 Cagliari, Italy

Received 3 January 2006; accepted 3 January 2006
Available online 6 March 2006

Abstract

Until recently, network administrators manually arranged alarms produced by intrusion detection systems (IDS) to attain a high-level description of cyberattacks. As the number of alarms is increasingly growing, automatic tools for alarm clustering have been proposed to provide such a high-level description of the attack scenarios. In addition, it has been shown that effective threat analysis requires the *fusion* of different sources of information, such as different IDS. This paper proposes a new strategy to perform alarm clustering which produces unified descriptions of attacks from alarms produced by multiple IDS. In order to be effective, the proposed alarm clustering system takes into account two characteristics of IDS: (i) for a given attack, different sensors may produce a number of alarms reporting different attack descriptions; and (ii) a certain attack description may be produced by the IDS in response to different types of attack. Experimental results show that the high-level alarms produced by the alarm clustering module effectively summarize the attacks, drastically reducing the volume of alarms presented to the administrator. In addition, these high-level alarms can be used as the base to perform further higher-level threat analysis.

Keywords: Computer security; Intrusion detection; Alarm clustering

1. Introduction

The increasing number and value of services accessible through the Internet require an adequate protection against cyberattacks. Intrusion detection systems (IDS) are an essential component of a complete defense-in-depth architecture for computer network security. IDS collect and inspect audit data looking for evidence of intrusive behaviors. As soon as an intrusive event is detected, an alarm is raised giving the network administrator the opportunity to promptly react. The intrusion detection problem can be viewed as an instance of the generic signal-detection problem, whereby the attacks can be viewed as the signal to be detected, whereas normal system or network events can be viewed as the noise (Axelsson, 2000). Accordingly, IDS can be grouped into two broad categories, namely *misuse-based* and *anomaly-based* IDS. Misuse-based IDS base their decisions on signal characterization, whereas anomaly-based detectors base their decisions on noise characterization. In order to detect an attack, a misuse-based IDS must possess a description of the attack which can be matched to the attack manifestations (i.e., the signal). Such a description is often called *attack signature* and misuse-based IDS are also referred as *signature-based* IDS (Sy, 2005). Conversely, anomaly-based IDS rely on the assumption that attack manifestations are somehow distinguishable from the normal events (i.e., the noise). Therefore, for an anomaly-based IDS to detect an attack, it must possess a model of the normal events which can be compared to the attack manifestations. The attack can be detected if its manifestations deviate from the model of normal events. IDS can be further grouped into two categories with respect to the source of the audit data they analyze, namely *host-based* and *network-based* IDS. Host-based detectors collect audit data from operating system event logs, application logs, file system information, etc., whereas network-based detectors collect data from packets crossing a network segment.

At present, a number of commercial, open-source, and research IDS tools are available. Among them,

*Corresponding author. Tel.: +39 070 675 5776; fax: +39 070 675 5782.
E-mail addresses: roberto.perdisci@diee.unica.it (R. Perdisci), giacinto@diee.unica.it (G. Giacinto), roli@diee.unica.it (F. Roli).

0952-1976/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2006.01.003

signature-based network IDS are widely used in many organizations thanks to their ability to detect well-known patterns of intrusions while producing a low number of false alarms. Most signature-based network IDS analyze network traffic on a packet basis looking for packets that match the signature of known attacks. As soon as a signature is matched, an alarm is raised. Packet inspection provides a powerful source of fine-grain information related to suspicious activities in the protected network. Nevertheless, this fine-grain analysis causes IDS to produce a high number of alarms. The source of such a large number of alarms is motivated by the nature of some categories of attacks which send a large number of malicious packets. As signature-based IDS produce an alarm for each malicious packet, alarm flooding may occur. In addition, many attacks are performed as a sequence of steps. While each step can be easily detected by signature-based IDS, the valuable information for the network administrator relies on the aggregation of alarms related to the different steps, rather than on each single alarm. More importantly, the produced alarms are often imprecise and could report descriptions of the detected events that are either too specific or too generic. As each IDS implements different detection algorithms and signatures, the combination of complementary IDS is a promising technique that can be used to obtain a more precise and comprehensive view of suspicious network events (Bass, 2000). The use of multiple detection technologies can provide the following benefits: (i) for a given attack, different IDS may produce different outputs; (ii) for a given attack, only a limited number of IDS might be able to detect it; and (iii) the fusion of alarms raised by different IDS produces more comprehensive information about intrusion attempts than that attained using a single IDS technique. In order to gain an understanding of the intrusions against the protected network, a network administrator needs to correlate the alarms produced by different IDS to attain a high-level description of the threat. Obviously, it is infeasible for a network administrator to manually arrange the huge volume of alarms produced by multiple IDS.

Kruegel et al. (2005) presented a comprehensive view of the alarm correlation process. The proposed alarm correlation system consists of a sequence of components that transform IDS *elementary alarms* into high-level intrusion reports that are sent to the administrator. Alarm clustering is an essential part of the alarm correlation process. The aim of alarm clustering is to handle the elementary alarms produced by multiple IDS due to a certain attack, fusing them to produce an higher-level alarm message, called *meta-alarm*, that summarizes the characteristics of the detected attack and provides a reference to the related elementary alarm messages produced by the IDS. The obtained meta-alarms can be further processed by modules that perform attack scenario reconstruction and threat assessment.

This paper is a revised version of Giacinto et al. (2005), where we proposed a new *on-line* alarm clustering system. The proposed system consists of three main modules, namely an *alarm management interface* (AMI), an *alarm classifier* and an *alarm clustering* module. The AMI receives alarms from multiple IDS and translates them in a standard format. Then, the alarm classifier assigns a *class label* to the received alarms and sends them to the alarm clustering module, where the alarms are fused to obtain meta-alarms. In order to be effective, the proposed system takes into account two characteristics of IDS: (i) for a given attack, different sensors may produce a number of alarms reporting different attack descriptions; and (ii) a certain attack description may be produced by the IDS in response to different types of attack. Experimental results show that the high-level alarms produced by the proposed alarm clustering system effectively summarize the attacks, drastically reducing the volume of alarms presented to the administrator. Besides, the obtained information is suitable to be further processed by higher-level modules that perform scenario reconstruction and threat analysis.

The paper is organized as follows. A summary of the related works on alarm clustering and correlation is reported in Section 2. Section 3 presents the details of the proposed alarm clustering system. Experimental results attained by using commercial and open-source IDS are reported in Section 4. In particular, the structure of the meta-alarm is presented which can summarize a large number of elementary alarms. Conclusions are drawn in Section 5.

2. Related work

At present, only few products are available for alarm clustering and correlation, but their need is acknowledged by administrators which cannot cope with a very large number of alarms. For this reason, there is a huge effort to develop algorithms that effectively aggregate alarms. Without such tools, the logging function of IDS is likely to be turned off by administrators.

A heuristic/probabilistic approach to alarm correlation has been proposed in Valdes and Skinner (2001). Weighted distance functions are defined to aggregate alarms. An overall similarity index between alarms is obtained through a weighted sum of similarity indexes among features like the announced attack class, IP addresses, TCP/UDP source and destination ports, timestamps, etc. Expert systems have been also used to perform alarm correlation (Cuppens, 2001; Cuppens and Miege, 2002). Alarms are clustered according to suitable distance measures, and global alarms are produced. Distances among alarms are computed taking into account similarity between attack descriptions, source and target similarity, time similarity, etc.

A thorough threat analysis system has been proposed in the framework of the M-Correlator project (Porras et al., 2002). Alarms are filtered and clustered according to the knowledge of the network architecture and known vulnerabilities. For each alarm cluster, a relevance score is computed, which represents the likelihood for the detected attack to be successful. A different approach has been proposed in the framework of the SCYLLARUS system (Goldman et al., 2001). A Report Database stores the alarms produced by multiple IDS sensors, and a clustering preprocessor periodically queries the database to find clusters of alarms. As soon as a new cluster is detected, one or more hypotheses on the attack that generated the alarms are produced. These hypotheses are further evaluated by a distinct module that determines which of them can be deemed correct.

A heuristic clustering algorithm based on a variant of attribute-oriented induction has been recently proposed in Julish (2003), whereas Valeur et al. (2004) proposed a comprehensive view of the alarm correlation process. The proposed correlation system consists of a sequence of specialized modules. Each module accomplishes a sub-task of the entire process. The correlation process transforms elementary alarms produced by multiple IDS in high-level intrusion reports that are presented to the network administrator. Even though a number of works have been presented, research on alarm correlation is still immature and no standard performance evaluation strategy exists for alarm correlation systems. A testing methodology has been recently proposed by Haines et al. (2003) as part of the Cyber Panel Correlation Technology Validation (CTV) effort sponsored by DARPA.

With respect to the related work, in the present paper a novel on-line alarm-clustering algorithm is proposed. The objective is to achieve alarm volume reduction by fusing alarms produced by different sensors in consequence of a given attack. In particular, the main contribution is the introduction of a learning phase, which aims to extract the attack class(es) an attack description belongs to. This attack description classification process allows to cluster alarms seemingly produced by different attacks that actually belong to the same alarm thread.

3. The proposed alarm clustering system

In this section, we present our alarm clustering system designed to process the sequence of alarms produced by IDS, and then produce meta-alarms, i.e. summary descriptions of events obtained by aggregating correlated alarms produced by various IDS sensors. Such a summary information can be provided by the attack class the alarms refer to. In fact, while alarms refer to the particular action performed to attain a certain goal, attack classes are conventionally used to specify the goal of attacks. As the number of goals is very small compared to the number of different attacks, the attack class provides an effective high-level information (Kendall, 1999; Undercoffer et al., 2003). As an example, consider the following three attack classes: (i) *portscan*; (ii) *webscan*; and (iii) *denial of service* (*DoS*). A *portscan* attack is usually performed by sending a large number of TCP or UDP packets to different ports in order to spot whether a service is bound to a certain port or not. In a similar way a *webscan* attack is performed by sending a sequence of HTTP queries to a web server (the victim) looking for vulnerable web applications. *DoS* attacks, instead, are usually performed sending a large number of properly crafted packets to a victim host trying to exploit vulnerabilities which can cause the victim host or application to crash. Each meta-alarm is further described by a number of features needed to uniquely identify the event, such as start and stop time of the attack, source and destination IP addresses, source and destination TCP/UDP ports, etc. In addition, the identifiers of the aggregated alarm logs are reported to allow further inspection. The proposed system results particularly suitable in aggregating alarms produced by those kinds of attacks which cause the IDS to produce a high number of alarms. In the following, we will refer primarily to signature-based Network-IDS (NIDS) sensors, as it is the most widely used type of IDS sensors. Nevertheless, the reported discussion can be extended to other ID techniques. We will provide an overview of the architecture of the proposed alarm-clustering system first, then going into the details of each of the components.

3.1. System architecture

Fig. 1 depicts a schema of our alarm clustering system. The AMI receives alarms from multiple IDS sensors and performs data alignment by translating each alarm message toward a standard alarm message format. This is necessary because IDS from different vendors usually produce messages according to different formats. In our implementation, the AMI translate the alarm messages in intrusion detection message exchange format (IDMEF) because it has been proposed as standard format for alarm reporting by the IETF (Curry et al., 2004). The second block, i.e. the classification module, is designed to label an alarm message as belonging to one or more attack classes. The

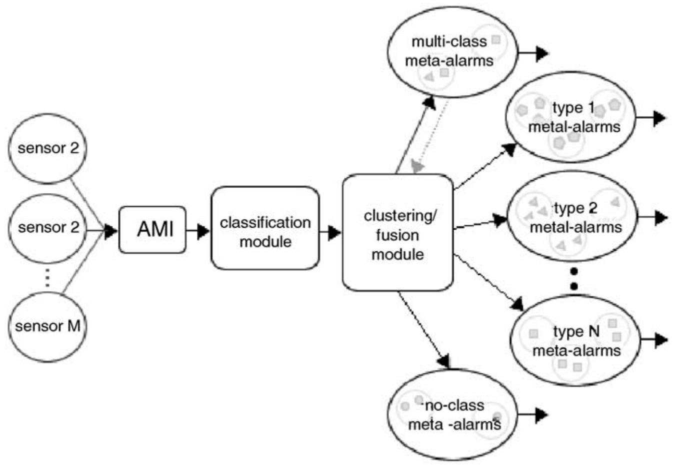

Fig. 1. Alarm clustering system.

classification module is motivated by two kinds of ambiguities: (i) for a given attack, different sensors may produce a number of alarms reporting different attack descriptions; and (ii) an attack description may be produced by the IDS in response to different attacks. As an example, when a *portscan* is carried out, an IDS may generate alarms related to *portscan* activities, as well as alarms related to a *DoS* attack to the services related to the scanned ports. The latter kind of alarms can be considered as *spurious* alarms, as no *DoS* attack has been actually performed. On the other hand, as users of the IDS, we have an ambiguity when such alarms are raised: are they related to an actual *DoS* attack, or are they spurious alarms produced by *portscan* activities? For this reason, we decided to label these alarms as belonging to a multiple-class, first. Then, the clustering process removes the ambiguity as soon as an alarm with a unique label is clustered with the multiple-class alarm. In fact, each cluster must contain alarms belonging to one attack class. We also used the "no-class" label for those alarms related to attacks for which a class has not been defined during classifier design. The details of the training procedure of the classification module will be given in Section 3.3. Classified alarms are sequentially sent to the clustering/fusion module, which applies a nearest-neighbor clustering algorithm (Jain et al., 1999). For each received alarm the clustering algorithm determines whether the alarm can be clustered, and thus fused, to one of the clusters or have to initialize a new meta-alarm (a new group of alarms). The whole system is designed to be used in a near real-time environment, i.e. IDS sensors send the alarms to the alarm reduction system as soon as they are produced. In such an environment meta-alarms that have not been involved in a clustering/fusion process for a time interval greater than a predefined timeout threshold will be removed from the system and sent in IDMEF format to the administrator.

3.2. Meta-alarm content

Before going into the details of the clustering algorithm, let us state which information we aim to extract from each cluster, and include in the related meta-alarm. A meta-alarm is characterized by the following features: a *classification name*, which is the common generalized class-name assigned to the clustered alarms by the classification module; the *create-time*, which is the time-stamp of the last meta-alarm update (the last fusion). Additional data include the start and stop time of the attack, the source IP addresses of the attack, the attack target[1] IP addresses, the involved source and target TCP/UDP ports. In addition, a reference to the log files of the IDS is reported so that further investigation on the aggregated elementary alarms can be carried out.

[1]We use *destination* and *target* as synonymous throughout the paper.

3.3. Classification module

An alarm class *C* consists of the set of attack descriptions provided by the sensors in response to attacks of type *C*. For example, the *portscan* alarm class consists of the set of attack descriptions reported in alarms obtained by simulating *portscan* attacks performed by using different techniques. We have already noticed that a given alarm can be raised by an IDS in response to different attacks. For example, a *portscan* may cause an IDS to produce a number of alarms that refer to *DoS* attacks in addition to the alarms related to the *portscan* activity. Such *DoS* alarms are confusing because they should be produced only in case of real *DoS* attacks. The role of the classification module is to assign each alarm to the attack class(es) that might have produced it. To this end, the classifier is designed by simulating a number of real attacks for each class of attacks. For example, if we consider attacks belonging to *portscan*, *webscan* and *DoS* classes, the designing process has to be performed in the following way:

(1) Simulate the most used *portscan* attacks with different techniques.
(2) Extract the pairs *sensor-name, alarm-message* from each alarm produced by step 1.
(3) Store the pairs *sensor-name, alarm-message* into a set called *portscan-descriptions*.
(4) Repeat steps 1–3 for *webscan* and *DoS* attacks, thus storing the pairs sensor-name, alarm-message into a *webscan-descriptions* set and a *dos-descriptions* set, respectively.

During the operational phase, when the classifier receives an alarm with description *Desc-1* produced by *Sensor-A*, the pair {*Sensor-A*, *Desc-1*} is compared to each pair contained into the *portscan-descriptions*, *webscan-descriptions* and *dos-descriptions* sets. The alarm is then labelled with the classes with matching pairs. For example, if the pair {*Sensor-A*, *Desc-1*} is found both into the *portscan-descriptions* set, and into the *dos-description* set, then the alarm will be labelled as belonging to both *portscan* and *DoS* classes. On the other hand, if the pair {*Sensor-A*, *Desc-1*} is not in any sets of descriptions, the alarm will be labelled as "no-class".

3.4. Clustering/fusion module

The clustering/fusion module is initialized by creating a number of empty sets, each one devoted to contain clusters related to one of the attack classes taken into account in the classifier design phase. In the case of the above example (see Section 3.3) the clustering/fusion block creates three sets: *PortscanMetaAlarmSet*, *WebscanMetaAlarmSet*, and *DoSMetaAlarmSet*. In addition, two other sets are created, namely the *MultiClassMetaAlarmSet* and the *NoClassMetaAlarmSet*. The *MultiClassMetaAlarmSet* set is devoted to temporarily contain meta-alarms obtained by clustering alarms labelled as belonging to multiple classes, whereas the *NoClassMetaAlarmSet* is devoted to contain meta-alarms obtained by clustering alarms that have not received a label from the classification module. It is worth recalling that alarm clustering is aimed at reducing the number of alarms produced by certain classes of attacks. Thus, a number of alarms are clearly not taken into account in the design phase.

Before going into the details of the clustering algorithm, some definitions have to be introduced:

Definition 1 (*Distance between pairs of features*). Let us denote the distance d between the i-th feature $A.f_i$ of an alarm A and the corresponding feature $M.f_i$ of a meta-alarm M as $d(A.f_i, M.f_i)$. Distance measures for various types of features such as the timestamp, target *IP*, target port, source *IP*, source port, etc., are defined in Section 3.5.

Definition 2 (*Correlation between an alarm and a meta-alarm*). An alarm A is assigned to the nearest cluster if the distance between A and the meta-alarm M associated with that cluster is below a predefined threshold. In particular, alarm A is assigned to the nearest cluster M if all the distances between the corresponding features are smaller than a set of predefined thresholds:

$$d(A.f_i, M.f_i) \leqslant t_i, \quad \forall i = 1, \ldots, v, \tag{1}$$

where v is the total number of considered features. The values of the thresholds $\{t_i\}_{i=1,\ldots,v}$ depend on the class M belongs to (e.g., $t_i = 0$, $\forall i = 1, \ldots, v$, if $M \in$ *NoClassMetaAlarmSet*). Besides they can be tuned with respect to the characteristics of the protected network. In the following, we will refer to an alarm A and a meta-alarm M satisfying Eq. (1) to be correlated.

Definition 3 (*Distance between an alarm and a meta-alarm*). If an alarm A and a meta-alarm M are correlated (i.e., they satisfy Eq.(1)), then their distance is computed as the time gap between the create-time of A and the create-time of the more recent alarm fused to M. Otherwise, the distance between A and M is set to $+\infty$.

Definition 4 (*Distance between an alarm and a set of meta-alarms*). The distance between an alarm A and a set of meta-alarms S is defined in the following way:

(1) If S does not contain any meta-alarm M correlated to A, then the distance is set to $+\infty$.
(2) If S contains k meta-alarms $M_1, M_2, \ldots, M_k$ correlated to A, the distance between A and S is computed as $\min_{i=1,\ldots,k}(\text{dist}(A, M_i))$.

In order to explain how the proposed clustering algorithm works, let us resort to an example. Let us suppose to be in a running state, and that each meta-alarm set contains a number of clusters. When a new alarm A is processed by the clustering module, three different cases may occur:

(a) *A has been labelled as belonging to none of the classes*: If A has been labelled as belonging to "no-class", then it will be inserted into the *NoClassMetaAlarmSet*, and it will be clustered with the nearest no-class meta-alarm. If the *NoClassMetaAlarmSet* contains no meta-alarm correlated to A, then A will initialize a new no-class meta-alarm that inherits A's features. An alarm A and a no-class meta-alarm M are considered correlated only if all A's and M's features (except the attack description) perfectly match (i.e., $t_i = 0$, $\forall i = 1, \ldots, v$ in Eq. (1)). In this case, there is a high probability that A and M are related to the same attack, even if the attack descriptions do not coincide.

(b) *A has been labelled as belonging to a unique class*: If the alarm A has been labelled, for example, as a *portscan* the following distances will be computed:

$$d_1 = \text{dist}(A, \mathit{PortscanMetaAlarmSet}),$$
$$d_2 = \text{dist}(A, \mathit{MultiClassMetaAlarmSet}),$$
$$d_3 = \text{dist}(A, \mathit{NoClassMetaAlarmSet}).$$

If $(d_1 = d_2 = d_3 = +\infty)$, then there is no meta-alarm correlated to A into the *Portscan*, *MultiClass*, and *NoClass* meta-alarm sets. In this case, A will be inserted into the *PortscanMetaAlarmSet* where it will initialize a new meta-alarm. If $d_1 = \min\{d_1, d_2, d_3\}$, A will be inserted into the *PortscanMetaAlarmSet*, and it will be fused to the nearest *portscan* meta-alarm that is correlated to A. Similarly, if d_2 or d_3 exhibit the minimum distance, A will be inserted, respectively, into the *MultiClassMetaAlarmSet* or the *NoClassMetaAlarmSet*, and it will be fused to the nearest correlated meta-alarm. In the case of $d_2 = \min\{d_1, d_2, d_3\}$, the resulting meta-alarm will be moved from the *MultiClassMetaAlarmSet* to the *PortscanMetaAlarmSet*, as the alarm A has a unique class label that can resolve the ambiguity of the correlated meta-alarm. In the case of $d_3 = \min\{d_1, d_2, d_3\}$, the class label given to A will not be further considered, and the resulting meta-alarm will have no class label. The reason for computing the distances d_2 and d_3 instead of immediately inserting A into *PortscanMetaAlarmSet* (given that A has been labelled as a *portscan* by the classification module) is justified by the following considerations: (1) Let us assume that alarm A is the n-th alarm caused by a *portscan* attack, and that the first $n-1$ alarms have been classified as belonging to multiple classes, *portscan* class included. By comparing A to the meta-alarms contained in the *MultiClassMetaAlarmSet*, it will be correctly fused to the correct sequence of alarms. (2) Given that a perfect matching is required among the features of the alarm A and those of a no-class meta-alarm M to be correlated, if $d_3 = \min\{d_1, d_2, d_3\}$, A and M are quite certainly related to the same attack even though A has been labelled as belonging to a certain class.

(c) *A has been labelled as belonging to multiple attack classes*: If alarm A has been labelled, e.g., as *portscan* and *DoS*, the following four distances will be computed:

$d_1 = \text{dist}(A, \textit{PortscanMetaAlarmSet})$,
$d_2 = \text{dist}(A, \textit{DosMetaAlarmSet})$,
$d_3 = \text{dist}(A, \textit{MultiClassMetaAlarmSet})$,
$d_4 = \text{dist}(A, \textit{NoClassMetaAlarmSet})$.

If $(d_1 = d_2 = d_3 = d_4 = +\infty)$, A will be inserted into the *MultiClassMetaAlarmSet*, and it will initialize a new meta-alarm. If one or more distances are not equal to $+\infty$, then A will be inserted into the nearest meta-alarm set, and it will be fused with the nearest meta-alarm in that set.

3.5. Distances among features

In this section we present the definition of the distances among features used by the clustering algorithm. Let A be an alarm and M a meta-alarm. Distances among IP addresses and port lists hold the same definitions, with respect to either target or source information (i.e. dist($A.sourceIP$, $M.sourceIP$) and dist($A.targetIP$, $M.targetIP$) hold the same definition, as well as distances among source or target port lists).

- $d(A.IP,\ M.IP)$: We consider only IPv4 addresses. The distance d between two IP addresses is defined as a *sub-network distance*. We consider the binary expression of $A.IP$ and $M.IP$, then we XOR the two binary strings. If we call n the number of consecutive zeros, counted starting from the left, in the resulting binary string, the distance d will be $d = 32 - n$. Consider the following example:

```
192.168.0.1 = 11000000.10101000.00000000.00000001
192.168.2.8 = 11000000.10101000.00000010.00001000
-------------------------------------------------
        XOR = 00000000.00000000.00000010.00001001
```

  It is easy to see that $n = 22$, thus $d = 10$.
- $d(A.portList,\ M.portList)$: The distance among $A.portList$ and $M.portList$ equals the number of port numbers present in $A.portList$ but not in $M.portList$. Consider, for example,

```
A.portList = {22,23,25,80,443,8080}
M.portList = {22,80,443,8080,8443}
```

  In this case $d = 2$.
- $time\_distance(A, M)$: The time distance t among an alarm A and a meta-alarm M is computed as the distance, in terms of milliseconds, among $A.createTime$ and $M.stopTime$.

4. Experiments

As mentioned in Section 2, at present no standard performance evaluation strategy exists for alarm correlation systems. Besides, no dataset explicitly designed for testing alarm clustering algorithms is publicly available. As a consequence, a direct comparison with results in the literature is rather difficult. Researchers usually evaluate their algorithms by performing some experiments on a typical network scenario, and assessing the effectiveness of the proposed techniques on such a scenario. We performed our experiments on two different sets of network traffic traces, using different open-source and commercial IDS. The first group of experiments were performed on the DARPA1999 dataset, whereas the second group of experiments were performed on network traces obtained from our academic network. We subdivided each group of experiments into three stages: (1) training of the classification module; (2) tuning of the thresholds involved in the clustering algorithm; and (3) performance tests. We report below the obtained results.

4.1. Results on DARPA1999 dataset

The DARPA1999 dataset was created by the MIT Lincoln Laboratory group to conduct a DARPA-sponsored comparative evaluation of different IDS (Lippmann et al., 2000). Even though the DARPA1999 dataset has been largely criticized (McHugh, 2000), it is the reference dataset in the evaluation of IDS performance. The dataset is made of five weeks of network traffic traces extracted from a simulated military department network. The traffic traces consist of normal background traffic generated by simulating normal user activities, and of several attacks grouped in four classes: (a) *Probe*; (b) User to Root (*U2R*); (c) Remote to Local (*R2L*); and (d) denial of service (*DoS*). The simulated network was divided into two segments, namely an *inside* network and an *outside* network. During our experiments we replicated some days of network traffic from both the *inside* and *outside* networks. The replicated traffic was monitored by three IDS. In particular, we used two Snort 2.2.0 sensors (Snort, Lightweight Intrusion Detection for Networks,) and one Symantec SNS 7161 sensor (Symantec Network Security 7100 Series,). The first Snort sensor was used to monitor the *outside* network traffic, whereas the second Snort sensor and the Symantec SNS sensor were both used to monitor the *inside* network traffic. We used an *out of the box* configuration for the

Snort sensors (i.e., the default set of activated signature and configuration parameters), whereas for the Symantec SNS sensor we activated all the available signatures. The alarm produced by the three IDS sensors were processed by our alarm clustering system. During the clustering process we considered three attack classes, i.e., *Probe*, *R2L*, and *DoS*. The *U2R* attacks were not considered because in general these attacks are local to a particular host and cannot be detected by network-based IDS. In addition to the three classes mentioned above, we also introduced a *suspicious-traffic* attack class, which allowed to cluster repetitive alarms that are not necessarily related to actual attacks (i.e., false alarms).

4.1.1. Training and tuning

The first three weeks of the DARPA1999 dataset contain network traffic that was created to be used as training set during the comparative IDS evaluation performed by the MIT Lincoln Laboratory group (Lippmann et al., 2000). In order to train our classification module and to tune the thresholds used by the on-line clustering algorithm, we used the traffic from Monday and Thursday of the second week as training set, because it contains a representative set of attack traces.

During the training stage, the attack classes *Probe*, *R2L* and *DoS* were considered one at a time. For each attack of a certain class in the training set, the pairs {*sensor-name*, *alarm-description*} extracted from the alarms produced by the IDS were stored according to the procedure described in Section 3.3. The *suspicious-traffic* class is somehow different because it aims at aggregating alarms not necessarily related to actual attacks. In order to train the classifier to recognize alarms belonging to the suspicious-traffic class, we manually extracted the alarm descriptions from repetitive alarms produced by the IDS due to the background traffic, and then clearly related to non-intrusive network traffic. The values of the thresholds used by the clustering algorithm (see Section 3.4) were estimated in two phases. During the first phase, initial values for the thresholds were chosen based on our knowledge of the attack characteristics. Then, in the second phase, we tuned the thresholds so that the clustering algorithm effectively aggregated all the correlated alarms produced by a given *training attack*. The notion of effectiveness may change according to the characteristic of the protected network, the needs of the network administrator, etc. Thus different tunings may fit different administrator's needs. The thresholds we obtained are reported in Table 1. The δt constant in the "Time" threshold column accounts for possible drifts among IDS sensors' clock. In our experiments, δt was set equal to 1 s.

4.1.2. Performance tests

The fourth and fifth weeks of the DARPA1999 dataset contain network traffic intended to be used as test set during the comparative IDS evaluation performed by the MIT Lincoln Laboratory (Lippmann et al., 2000). The traffic simulated for each day of the fourth and fifth weeks contains a number of attacks belonging to different classes. We used the traffic from Thursday of the fourth week and Thursday and Friday of the fifth week for performance tests. Table 2 reports a summary of the obtained results for the three considered days. The reported results represent, for each day of test, the total number of elementary alarms produced by the sensors, the total number of meta-alarms produced by our alarm clustering system, and the obtained alarm volume reduction with respect to the number of elementary alarms.

Tables 3 and 4 report the results obtained for Friday of the fifth week. In particular Table 3 reports the results obtained for each attack. The attacks are grouped with respect to the attack class. The "Training" column reports whether the attack was present in the training dataset. The "Cluster" column reports whether the elementary alarms were correctly clustered by our system by producing meta-alarms with the correct class-label, whereas the "Alarms" and "Meta-alarms" columns report the total number of alarms produced by the IDS and the number of meta-alarms produced by the alarm clustering algorithm, respectively. Note that even though the first and second rows refer to "Portsweep" attacks, the "Portsweep-1" attack was performed using a different technique with

Table 2
Results on three days of test traffic of the DARPA1999 dataset

Day	Number of alarms	Number of meta-alarms	Alarm reduction (%)
Thursday, fourth week	9245	1821	80.3
Thursday, fifth week	31,038	15,163	51.1
Friday, fifth week	12,257	4609	62.4

Table 1
Values of the thresholds used during the experiments on the DARPA1999 dataset

Meta-alarm class	SourceIP	TargetIP	SourcePort	TargetPort	Time (s)
Probe	0	0	$+\infty$	$+\infty$	$480+\delta t$
R2L	0	0	$+\infty$	0	$180+\delta t$
DoS	$+\infty$	0	$+\infty$	$+\infty$	$180+\delta t$
Suspicious	8	0	$+\infty$	$+\infty$	$600+\delta t$
No-class	0	0	0	0	$0+\delta t$

Table 3
Alarm clustering results for Friday of the fifth week

Attack class	Attack	Training	Cluster	Alarms	Meta-alarms
Probe	Portsweep-1	Yes	Not correct	30	30
	Portsweep-2	Yes	Correct	19	1
	Queso	No	Not complete	2	1(1)
R2L	Xsnoop	No	Not correct	2	2
	Guest	No	Not correct	5	5
	Sendmail	No	Not correct	1	1
DoS	CrashIIS-1	Yes	Correct	3	1
	Back	Yes	Correct	560	1
	CrashIIS-2	Yes	Correct	8	1
	CrashIIS-3	Yes	Correct	5	1
	Land	Yes	Correct	1	1
Suspicious-traffic	—	Yes	Correct	7060	2

Table 4
Summary performance results for Friday of the fifth week

Alarms		Meta-alarms	
Snort-outside	3712	Pobe	1
Snort-inside	3764	R2L	0
Symantec SNS	4781	DoS	4
		Suspicious-traffic	2
		Not classified	4482
		Others	51
Total alarms	12,257	Total Meta-alarms	4540

respect to the "Portsweep" attacks contained in the training set. The 30 alarms produced by the IDS contained a description never seen during training, therefore the alarm clustering system was not able to correctly group them and it produced, instead, one meta-alarm for each alarm message. On the other hand, for "Portsweep-2" the system correctly produced one meta-alarm by grouping all the 19 elementary alarms generated by the IDS. For the "Queso" attack the clustering system produced one meta-alarm containing only one out of two alarms produced by the IDS. The *R2L* attacks were not present in the training dataset and the related alarms were not grouped correctly, whereas the alarms related to *DoS* attacks were all correctly grouped. The last row reports the number of alarms related to the background traffic. Our system correctly grouped those alarms into two suspicious-traffic meta-alarms. It is worth noting that the alarms that were not correctly clustered caused the generation of meta-alarms containing just one alarm. These meta-alarms are imprecise and useless, given that the aim of the alarm clustering system is to group multiple alarms related to the same alarm. Nevertheless, it is easy to filter these meta-alarms by applying a simple *cluster validation* algorithm which considers incorrect all the clusters (i.e., meta-alarms) containing just one elementary alarm.

Table 4 reports a summary of the alarms produced by each IDS and by our alarm clustering system for Friday of the fifth week. The "Alarm" column reports the number of alarms generated by each IDS and the total number of alarms. The "Meta-alarm" column reports, for each class of attacks (i.e., *Probe*, *R2L*, *DoS* and *Suspicious-traffic*), only the correct meta-alarms that contained more than one alarm. The "Not classified" row reports the number of meta-alarms that do not have a class label, whereas the row "Others" reports the meta-alarms containing only one alarm and labelled either correctly or incorrectly. As mentioned before, the meta-alarms reported in the row "Others" can be easily filtered and further processed. During the post-processing phase, the original elementary alarms in the meta-alarms can be retrieved in order to be sent to the administrator as they are, whereas the meta-alarms can be disregarded. It is easy to see that the post-processing phase does not modify the volume reduction performances of the clustering system, given that each disregarded meta-alarm contains exactly one elementary alarm.

4.2. Results on a live network

The proposed alarm clustering strategy has been also tested on a live network containing dozens of hosts, some of them chosen as victims. The traffic of the considered network was made up of normal background traffic, i.e., the normal activity of the users of the network, and by a number of simulated attacks. Three IDS have been used to monitor the network traffic: Snort 2.1.0 (Snort, Lightweight Intrusion Detection for Networks), Prelude-NIDS 0.8.6 (Prelude Intrusion Detection System), and ISS Real Secure Network Sensor 7.0 (ISS, Inc. RealSecure intrusion detection system). The training and tuning stages was carried out by executing attack simulations in an isolated network made up of three hosts, two victims (a Linux host and a Win2k host), and an attacker host. The performed experiments were related to three attack classes, i.e., *portscan*, *webscan*, and *DoS*, as they usually produce a large number of alarms. In addition to these classes, we have also introduced a *suspicious-traffic* attack class, which allows to cluster alarms related to patterns of live traffic classified as suspicious by the sensors.

4.2.1. Training and tuning

The classification module was designed using a number of tools available on the Internet. In particular, we used *nmap*, *wups*, *ipeye*, etc. as *portscan* tools; *nikto*, *babelweb*, *whisker*, *winCGIscan*, etc. as *webscan* tools; *teardrop*, *jolt* (aka ping of death), *synflood*, *saihyousen*, etc. as *DoS* attacks. During this phase, for each attack, the pairs {*sensor-name*, *alarm-description*} were stored according to the procedure described in Section 3.3. As far as the suspicious-traffic class is concerned, it is worth noting that the pairs {*sensor-name*, *alarm-description*} are difficult to produce by traffic simulations. For this reason, we manually extracted the alarm descriptions directly from the corresponding sets of rules of each available sensor (often called *bad traffic* or *suspicious-traffic*). Again, the values of the thresholds used by the clustering algorithm (see Section 3.4) were estimated in two phases. In the first phase, an initial value for the thresholds was chosen by heuristics based on attack characteristics. Then, in the second phase, attack simulations were performed in the isolated network to suitably tune the thresholds in order to effectively cluster all the correlated alarms produced by a given attack. The obtained values are reported in Table 5. Again, δt was set equal to 1 s.

Table 5
Values of the thresholds used during the experiments on *live network* traffic

Meta-alarm class	SourceIP	TargetIP	SourcePort	TargetPort	Time (s)
Portscan	0	0	$+\infty$	$+\infty$	$480+\delta t$
Webscan	0	0	$+\infty$	0	$120+\delta t$
DoS	$+\infty$	0	$+\infty$	$+\infty$	$120+\delta t$
Suspicious	8	0	$+\infty$	$+\infty$	$120+\delta t$
No-class	0	0	0	0	$0+\delta t$

4.2.2. Performance tests

A large number of attacks have been executed in the selected live network to test the feasibility of the designed system to correctly cluster attacks. Results showed that the proposed technique produced not only meta-alarms related to the simulated attacks, but also meta-alarms related to the suspicious-traffic. As the design phase was carried out in an isolated network, these results show the feasibility of the proposed approach. Table 6 reports the details of the most significant results.

Portscan: When *portscan* have been performed, the clustering algorithm successfully produced a meta-alarm for every *portscan* activity. As an example, consecutive SYN and Xmas *portscans* have been performed from one source towards a victim, producing a total of 3074 alarms from the three considered IDS (see the first column in Table 6). During these attacks, the sensors also produced alarms related to malicious activities in the background traffic. The clustering algorithm correctly produced 1 meta-alarm related to the *portscan*, and 15 meta-alarms related to other activities. The meta-alarm related to *portscan* activities is correctly labelled as *portscan*, and contains the list of scanned ports, the source and target hosts, the start and stop times of the attack, and references to the alarms that originated the meta-alarm. Investigating the meta-alarms related to background traffic we found that most of them contained alarms classified as *suspicious-traffic*.

Webscans: Similar results have been also attained with *webscans*. In some cases, long attacks originated more than one meta-alarm, because of time gaps among groups of alarms. Nevertheless, this issue can be resolved by a post-processing situation refinement module that is aimed at finding relationships among meta-alarms. As an example, Webscan2 (nikto) originated 19,164 alarms that were clustered into 37 clusters. The size of the first two clusters was equal to 7464 and 11,631 alarms, respectively. It is worth noting that 69 alarms produced by the ISS Real Secure sensor generated 35 meta-alarms. These alarms were related to attack responses produced by the webserver that were not correctly recognized by Real Secure.

Table 6
Experimental results on live network

	Portscan	Webscan1	Webscan2	DoS
Alarms from snort	1058	94	7143	71
Alarms from prelude	1314	93	6601	8828
Alarms from ISS real secure	728	63	5586	186
Total Number of alarms	3100	250	19,330	9085
Attack-related alarms	3074	244	19,164	9028
Alarms produced by background traffic	26	6	166	57
Meta-alarms from simulated attacks	1	1	37	4
Meta-alarms from background traffic	15	3	85	33
Total number of Meta-alarms	16	4	122	37

DoS: Four *DoS* attacks have been performed against the same host. The proposed alarm clustering was able to correctly produce 4 meta-alarms corresponding to the different attacks carried out and 33 meta-alarms corresponding to alarms related to suspicious background-traffic.

5. Conclusions

In this paper, we proposed a novel on-line alarm-clustering system whose main objective is the reduction of the volume of alarms produced by today's IDS sensors. The clustering system has been devised to work in near real time. Experiments performed in different attack scenarios on a live network showed that the proposed algorithm effectively groups alarms related to the same attack, even though IDS produced alarms whose descriptions were erroneously referred to different types of attacks. The produced meta-alarms provide the system administrator with a concise high-level description of the attack. In addition, the produced meta-alarms can be used as the starting point for the development of modules for situation refinement and threat analysis.

References

Axelsson, S., 2000. A preliminary attempt to apply detection and estimation theory to intrusion detection. Technical Report, Department of Computer Engineering, Chalmers University of Technology, Sweden, March.

Bass, T., 2000. Intrusion detection systems and multisensor data fusion. Communications of the ACM 43 (4), 99–105.

Cuppens, F., 2001. Managing alerts in a multi-intrusion detection environment. Proceedings of the 17th Computer Security Applications Conference, ACSAC 2001, pp. 22–31.

Cuppens, F., Miege, A., 2002. Alert correlation in a cooperative intrusion detection framework. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 202–215.

Curry, D., Debar, H., Feinstein, B., 2004. The intrusion detection message exchange format ⟨http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-11.txt⟩.

Giacinto, G., Perdisci, R., Roli, F., 2005. Alarm clustering for intrusion detection systems in computer networks. In: Perner, P., Imiya, A. (Eds.), Machine Learning and Data Mining in Pattern Recognition, MLDM 2005. Springer, Berlin, pp. 184–193.

Goldman, R.P., Heimerdinger, W., Harp, S.A., Geib, C.W., Thomas, V., Carter, R.L., 2001. Information modeling for intrusion report aggregation. In: Proceedings of the DARPA Information Survivability Conference Exposition II, DISCEX 2001, vol. 1, pp. 329–342.

Haines, J., Ryder, D.K., Tinnel, L., Taylor, S., 2003. Validation of sensor alert correlators. IEEE Security Privacy, January–February 2003, pp. 46–56.

ISS, Inc. RealSecure intrusion detection system ⟨http://www.iss.net⟩.

Jain, A.K., Murty, M.N., Flynn, P.J., 1999. Data clustering: a review. ACM Computing Surveys 31 (3), 264–323.

Julish, K., 2003. Clustering intrusion detection alarms to support root cause analysis. ACM Transactions on Information and System Security 6 (4), 443–471.

Kendall, K., 1999. A database of computer attacks for the evaluation of intrusion detection systems. Master's Thesis, MIT.

Kruegel, C., Valeur, F., Vigna, G., 2005. Intrusion detection and correlation. Advances in Information Security Series, vol. 14. Springer, Berlin.

Lippmann, R., Haines, J.W., Fried, D.J., Korba, J., Das, K., 2000. The 1999 DARPA off-line intrusion detection evaluation. Computer Networks 34 (4), 579–595 (Special issue on recent advances in intrusion detection systems).

McHugh, J., 2000. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. ACM Transaction on Information and System Security 3 (4), 262–294.

Porras, P.A., Fong, M.W., Valdes, A., 2002. A mission-impact-based approach to INFOSEC alarm correlation. In: Wespi, A., Vigna, G., Deri, L. (Eds.), Fifth International Symposium on Recent Advances in Intrusion Detection RAID 2002. Lecture Notes in Computer Science, vol. 2516. Springer, Berlin, pp. 95–114.

Prelude Intrusion Detection System ⟨http://www.prelude-ids.org⟩.

Snort, Lightweight Intrusion Detection for Networks ⟨http://www.snort.org⟩.

Sy, B., 2005. Signature-based approach for intrusion detection. In: Perner, P., Imiya, A. (Eds.), Machine Learning and Data Mining in Pattern Recognition, MLDM 2005. Springer, Berlin, pp. 526–536.

Symantec Network Security 7100 Series ⟨http://enterprisesecurity.symantec.com⟩.

Undercoffer, J., Joshi, A., Pinkston, J., 2003. Modeling computer attacks: an ontology for intrusion detection. In: Vigna, G., Jonsson, E., Kruegel, C. (Eds.), Sixth International Symposium on Recent Advances in Intrusion Detection, RAID 2003. Lecture Notes in Computer Science, vol. 2820. Springer, Berlin, pp. 113–135.

Valdes, A., Skinner, K., 2001. Probabilistic alert correlation. In: Jonsson, E., Valdes, A., Almgren, M. (Eds.), Fourth International Symposium on Recent Advances in Intrusion Detection, RAID 2001. Lecture Notes in Computer Science, vol. 3224. Springer, Berlin, pp. 54–68.

Valeur, F., Vigna, G., Kruegel, C., Kemmerer, R.A., 2004. Comprehensive approach to intrusion detection alert correlation. IEEE Transactions on Dependable and Secure Computing 1 (3), 146–169.