Computer Communications 32 (2009) 419–430

Contents lists available at ScienceDirect

Computer Communications

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

Intrusion detection alarms reduction using root cause analysis and clustering

Safaa O. Al-Mamory *, Hongli Zhang

School of Computer Science and Technology, Harbin Institute of Technology, 150001, China

A R T I C L E   I N F O

Article history:
Received 5 April 2008
Received in revised form 8 November 2008
Accepted 11 November 2008
Available online 21 November 2008

Keywords:
Network security
Intrusion detection system
False positive
Root causes
Alarms clustering

A B S T R A C T

As soon as the Intrusion Detection System (IDS) detects any suspicious activity, it will generate several alarms referring to as security breaches. Unfortunately, the triggered alarms usually are accompanied with huge number of false positives. In this paper, we use root cause analysis to discover the root causes making the IDS triggers these false alarms; most of these root causes are not attacks. Removing the root causes enhances alarms quality in the future. The root cause instigates the IDS to trigger alarms that almost always have similar features. These similar alarms can be clustered together; consequently, we have designed a new clustering technique to group IDS alarms and to produce clusters. Then, each cluster is modeled by a generalized alarm. The generalized alarms related to root causes are converted (by the security analyst) to filters in order to reduce future alarms' load. The suggested system is a semi-automated system helping the security analyst in specifying the root causes behind these false alarms and in writing accurate filtering rules. The proposed clustering method was verified with three different datasets, and the averaged reduction ratio was about 74% of the total alarms. Application of the new technique to alarms log greatly helps the security analyst in identifying the root causes; and then reduces the alarm load in the future.

1. Introduction

The frequency of computer intrusions has increased rapidly in the past several years. Intrusion Detection Systems (IDSs) are an essential component of a complete defense-in-depth architecture for computer network security. They collect and inspect packets, looking for evidence of intrusive behaviors. As soon as an intrusive event is detected, an alarm is raised giving the security analyst an opportunity to react promptly. Unfortunately, they provide unmanageable amount of alarms. Inspecting thousands of alarms per day [1] is unfeasible, especially if 99% of them are false positives [2].

Several methods to deal with alarms are available, but main objectives of these investigations are: to reduce the amount of false alarms [3], to study the cause of these false positives [2,4], to recognize high-level attack scenarii [5], to provide a coherent response to attacks, and finally to understand the relationship among different alerts [6]. In this chapter, we focus on the methods that study the root causes of the false alarms. Our method makes use of alarms history to refine the future alarms' quality. The root cause (or the configuration problem) instigates the IDS to trigger alarms that almost always have similar features. These similar alarms can be clustered together; the proposed algorithm does extract a pattern (or a *generalized alarm*) from each cluster. The extracted patterns help the security analyst in specifying the root causes behind these false alarms and in writing accurate filtering rules. The framework of the proposed system can be seen in Fig. 1.

Our approach focuses on identifying the root causes for large groups of alarms, which typically correspond to problems in the computing infrastructure leading to generate many false positives. From another point of view, for small networks, it is easy to well configure all the equipments in the network and there is no need for our technique. However, our method will be useful in the large networks because it is a hard task to configure all the equipments well.

We have developed a new approximation algorithm to cluster the alarms; these clusters are modeled by generalized alarms. These generalized alarms, which are related to root causes, are used to filter the alarms. The proposed algorithm generates a generalized alarm for each cluster. In more details, by representing each cluster by its center, the distance between every new alarm and all clusters centers will be computed. Then, the alarm is assigned to the nearest cluster. A new cluster is created if the distance is more than a threshold. Finally, we separately generalize the alarms of each cluster. The contribution of this paper is developing new data mining method which has new features' generalization technique to avoid overgeneralization, and has good distance measure between alarms' features values. Applying of our method to alarms log greatly helps the analyst in identifying the root causes; and then reduces the alarm load in the future.

* Corresponding author.
E-mail addresses: safaa_vb@yahoo.com (S.O. Al-Mamory), zhl@pact518.hit.edu.cn (H. Zhang).

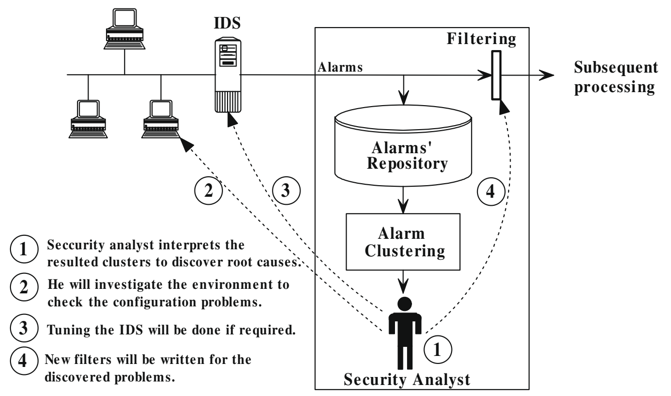

IDS
Filtering
Alarms
Subsequent processing
Alarms' Repository
Alarm Clustering
Security Analyst
1 Seccurity analyst interprets the resulted clusters to discover root causes.
2 He will investigate the environment to check the configuration problems.
3 Tuning the IDS will be done if required.
4 New filters will be written for the discovered problems.

Fig. 1. Semi-automated cluster processing.

This paper is organized as follows: Section 2 reviews related works. Section 3 describes some basic concepts. In Section 4, we propose alarms' clustering algorithm. Section 5 presents the empirical results. The discussion is presented in Sections 6 and 7 concludes this paper.

2. Related works

In the last decade, researchers have designed several systems dealing with the problem of overwhelming the security analyst with alarms. A survey on the work of alarms processing techniques is given in [7].

Several techniques have been introduced recently to correlate IDS alarms. The first class of approaches uses clustering to build attack scenarii. Siraj et al. [8] proposed a clustering technique to reconstruct attack scenarii. Probabilistic alert correlation finds similarity between alerts that match closely, if not exactly [9]. According to Valdes et al., probabilistic alert correlation correlates attacks over time, over multiple attempts, and from multiple sensors. Alert correlation tasks consist of [9]: identifying alert threads, identifying incidents by clustering/correlating threaded alerts with meta-alerts (i.e. scenarii), and clustering/correlating meta-alerts with meta-alerts. Dain et al. [5] used an alert clustering scheme which fuses the alerts into scenarii using an algorithm that is probabilistic in nature. In this system, scenarii are developed as they occur, i.e., whenever a new alert is received it is compared with current existing scenarii and then assigned to the scenario that yields highest probability score. Our method differs from these methods in that it is not used for building attack scenarii.

Another class of methods uses clustering to reduce the volume of alarms presented to the security analyst. Perdisci et al. [3] used clustering to introduce a concise view about attacks and to reduce the volume of alarms. Julisch [2,4] proposed a new method in which alarm clustering is performed by grouping together alarms whose root causes are generally similar. A generalized alarm for a specific alarm cluster represents a pattern that all of the alarms in the cluster must match in order to belong to that cluster. Our method can be considered as a variation of Julisch's work; however, we have designed a new data mining technique, which is different from these clustering methods, to reduce alarms volume.

The third class of methods uses artificial neural networks to extract attack strategies from IDS alarms. Zhu et al. [10] used multilayer perceptron and support vector machine to find the causal relationship between any two alarms. Smith et al. [11] has employed three methods to recognize multi-step attacks, which are an autoassociator neural network, EM algorithm, and SOM neural network. These systems can be considered as a complementary to our system.

The Adaptive Learner for Alert Classification (ALAC) [12] is an adaptive alert classifier based on the feedback of an intrusion detection analyst and machine-learning technique. The classification of IDS alerts is a difficult machine-learning problem. ALAC was designed to operate in two modes: a recommender mode, in which all alerts are labelled and passed to the analyst, and an agent mode, in which some alerts are processed automatically.

3. Preliminaries

This section presents basic knowledge that are necessary for the proposed system. Section 3.1 describes the using of root cause analysis in alarms processing with some representative examples. Section 3.2 presents some data mining principals and the requirements which should be satisfied by the proposed method.

3.1. Root cause and root cause analysis

Root cause analysis, in the past, was used to identify the most basic factors that play a role to an incident. In other words, root cause analysis is simply a tool designed to help incident investigators describe what happened during a particular incident, to determine how it happened, and to understand why it happened. The basic idea behind that is to learn from past failures and avoid similar incidents in the future [13]. Root cause analysis adopts the investigation techniques that identify root causes, i.e. the reasons why an incident occurred. A useful definition is the one used by Paradies et al. [14], that is

Definition 1. A *root cause* is the most basic cause that can be reasonably identified and that management has control to fix. *Root cause analysis* is the task of identifying root causes.

IDS alarm handling using root cause analysis was introduced by Julisch [4]. He argued that few root causes are liable for generating huge number of false alarms. Most environments have predominant root causes and few of these root causes are responsible for generating 90% of all alarms [2]. In addition, these root causes are persistent unless some one removes them [4]. As a consequence, we have designed a new data mining technique which permits the security analyst to understand why alarms are being triggered and how they should be handled in the future. In the sequel of this subsection, we will present some representative examples to illustrate root causes (which manifest themselves in IDS alarms' groups) and alarm patterns:

1. The brute force attack against an FTP server to guess a password will cause the IDS to trigger an alarm for each trial (this is the default setting for Snort [15] rules). IDS generates huge "*FTP Login Failure*" alarm messages for this event.
2. A broken router may copy the port specification into flag field causing to generate bogus flag contents. In this case, IDS thinks that the router do scan and will generate "*FULL XMAS scan*" alarms.
3. External infected network with Slapper worm causes the IDS to trigger "*Slapper*" alarms.
4. The management console periodically does a vulnerability scan to discover the existence of new vulnerabilities; this check is achieved by a normal scan. Any IDS in this situation should trigger "*Host Scan*" alarms if there are no rules to discard these alarms.
5. Because many attacks use telnet to control Trojan programs, it is useful to trigger an alarm whenever a telnet session is established. However, in the large networks with an enabled telnet service, there will be countless "*Telnet Access*" alarms.

It should be noted that the alarms of the first root cause originate from a non-privileged source port of the attacker machine. Furthermore, all of these alarms are targeted at source ports 21. In addition, these alarms always have "FTP Login Failure" as alarm type. Therefore, the alarm pattern induced by the first root cause can be represented as shown in the first row of Table 1. Similarly, the second row of the table shows that the second root cause induces the alarm pattern "FULL XMAS scan" alarms being triggered from any port of the router against any port of the internal network. Equivalently, the remaining rows of Table 1 show the alarm patterns that the other root causes induce. The Seq. (sequence) column of the table refers to the item numbers in the above enumeration of root causes; the entry "Non-priv." denotes the set {1024,..., 65,535} of non-privileged ports, and the entry "Privileged" stands for the set of privileged ports below 1024.

In the rest of this paper, we will call each row in Table 1 as generalized alarm. Similar to ordinary alarms, some of the generalized alarms' attribute values may be elementary values. In addition to these elementary values, the generalized alarms may have other generalized attributes' values. By generalized attribute value such as "Privileged" or "External network", we mean a concept name that represents a set of elementary attribute values. The generalized alarms are capable of capturing and representing the main features of alarm groups [4].

3.2. Data mining technique requirements

Data mining is the analysis of (often large) observational datasets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner. The relationships and summaries derived through a data mining exercise are often referred to as models or patterns. Examples include linear equations, rules, clusters, graphs, tree structures, and recurrent patterns in time series [16].

It is convenient to categorize data mining into types of tasks, corresponding to different objectives for the person who is analyzing the data. These tasks are as follows: First, *exploratory data analysis*, the goal here is simply to explore the data without any clear ideas of what we are looking for. Second, *descriptive modelling* which tries to describe all of the data (or the process generating the data). Third, *predictive modelling (classification and regression)* task aims to build a model that will permit the value of one variable to be predicted from the known values of other variables. Fourth, *discovering patterns and rules* task look for patterns and rules embedded in the data. Finally, *retrieval by content* task in which the user has a pattern of interest and wishes to find similar patterns in the dataset [16]. Our method belongs to the fourth class mentioned above.

For the data mining technique to be effective, there are some requirements that should be satisfied. We present these requirements, as supposed in [4], which should be fulfilled by the proposed technique:

- *Scalability*: Scalability of pattern and rule discovery algorithms is obviously an important issue [16]. Because of the fact that more than a million alarms are triggered by IDSs per month [4], so the data mining algorithms should be scale predictably (e.g., linearly) as the number of alarms and/or the number of variables grow. For example, naive implementation of a decision tree algorithm will exhibit a dramatic slowdown in run-time performance once records become large enough that the algorithm needs to frequently access data on disk [16].
- *Noise tolerance*: Huge software packages are distributed electronically, but the very success of the Internet makes some bugs invisible. These invisible bugs make the intrusion detection alarms very noisy [17]. In addition, to evade the detection by the IDS, the attacker flood a network with noise traffic to attack the victim with little or no intervention from the IDS.
- *Multiple feature types*: Intrusion detection alarms can contain numerical features (e.g. count and flags parameters), categorical features (e.g. IP addresses and port numbers), time features, etc [4]. The efficient data mining techniques should support all of these feature types.
- *Ease of use*: The easier the technique is the more suitable one because the security analyst is not a data mining expert. The data mining approach should require few knowledge from the security analyst and should have parameters that is easy to set.
- *Interpretability*: The generated patterns should be highly interpretable to be useful; otherwise they will be useless. For example, one of the factors that made the naive Bayes model popular in the machine-learning literature is the interpretability [16].

4. The alarms clustering algorithm

This section describes the proposed algorithm in details. Section 4.1 shows the generalization hierarchies as background knowledge. Section 4.2 proposes the distance measure which has been suggested to compute the distance between alarms. Section 4.3.2 presents our algorithm and Section 4.4 focuses on finding techniques to setting the control parameter.

4.1. Background knowledge representation

The availability of certain background knowledge, such as generalization hierarchies (hierarchies for short), does improve the efficiency of a discovery process and also expresses user's preference for guided generalization, which may lead to an efficient and desirable generalization process [18]. A hierarchy defines a sequence of mappings from a set of concepts to their higher-level correspondences [19].

The hierarchies play an important role on the results. A hierarchy should be defined for each alarm' feature. As we have seen in Section 3.2, we have different alarm's features types, like numerical and categorical features. Fortunately, there are several techniques to build hierarchies [20]. Han et al. [21] designed algorithms for building numerical hierarchies automatically depending on the datasets. Moreover, algorithms are suggested to build the hierarchies for the categorical features [22]. Our algorithm assumes that meaningful hierarchies have been defined for all alarm's features (features for short). Some hierarchies can be seen in Fig. 2 [2].

Table 1
A sample of alarm patterns derived from IDS alarm log.

Seq.	Src-IP	Src-Port	Dst-IP	Dst-Port	Alert type
1	Attacker	Non-priv	FTP Server	21	FTP Login Failure
2	Router	Any	Internal network	Any	FULL XMAS scan
3	External network	Non-priv	Internal network	80	Slapper
4	Management console	Privileged	Internal network	Any-Port	Host Scan
5	External network	Non-priv	ip_1	23	Telnet Access

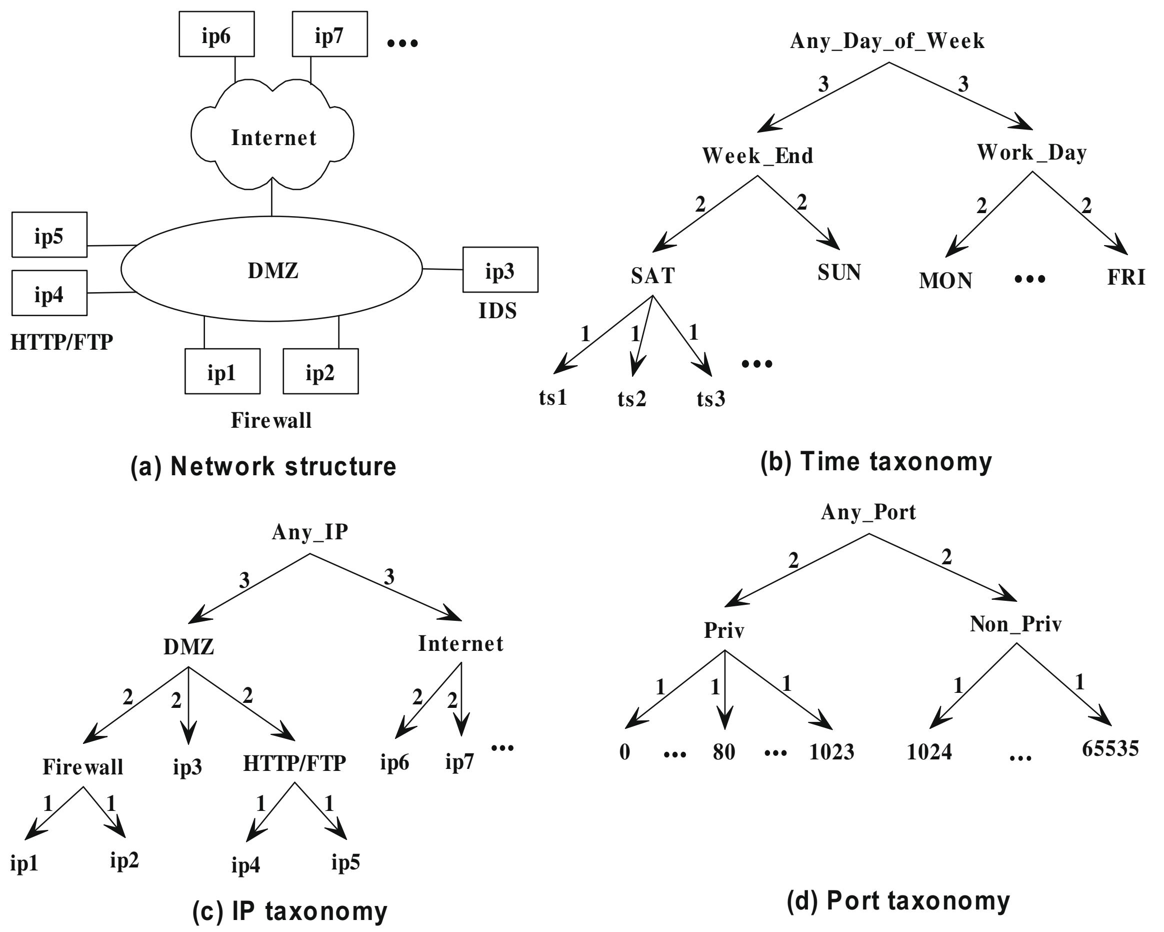


Fig. 2. Network structure and sample generalization hierarchies for IP address, port, and time features.

In spite of the fact that the above methods are useful, we have used an interesting methodology which is suggested by Pietraszek [23]. For IP address hierarchy, he labeled IP addresses according to their role (Firewall, HTTPServer, etc.), then grouped them according to their network location (Internet, DMZ, Subnet1, etc.) with final top-level generalized address AnyIP. If these classification hierarchies are not known, the IP addresses can be generalized according to the hierarchies in the addressing structure, for example, class C followed by class B which in turn followed by class A and finally AnyIP [23].

Furthermore, for other attributes like port hierarchies, Pietraszek [23] has classified the well-known ports (i.e. privileged ports, 0–1023) depending on their function. For example, HttpPorts can be 80, 443, 8080, or 9090; MailPort can be 25, 110, 143, 993, or 995; ChatPort can be 194, 258, 531, or 994. The non-privileged ports (1024–65,535) and privileged ports are generalized to a top-level category of Any-Port.

We should update generalization hierarchy for the IP address every time the network structure had changed. Methods to refine dynamically the generalization hierarchies are available [21]. Execution of our algorithm in short time periods enables us to be more responsive to changes in the network structure. However, some root causes could not be detected in short time periods. On the other hand, the long time period for our algorithm execution is inconsistent with the change in network structure, but enabling us to detect rare patterns occurring over long periods.

We have adopted edge numbering (as can be seen in Fig. 2b–d) because we want the cost of clustering the divergent features values to be higher than the convergent features values. The edges weights values in the trees of Fig. 2 are chosen consecutively with a condition, which is when we go up in the tree then the values will increase. In other words, let D be a depth of the tree and $L(Node)$ is the level of a *Node* in the tree. We will then number the edges of the trees depending on $D–L(Node)$ relation. We adopted the linear increasing of edges weights because we experimentally got good results with it; however, the logarithmic increasing of weights may also be promising.

4.2. Distance measures

The IDS triggers alarms to report presumed security violations. Let A be an alarm having many features $\{f_1, f_2, \ldots, f_n\}$ such that space of any feature is $Dom(f_i)$ (i.e. possible values). Then, the ith feature of alarm A is denoted as $A[f_i]$. In this paper, the considered alarm's features are source IP, destination IP, source port, destination port, time stamp and alarm type. It should be noted that we used IP address referring to both source IP and Destination IP features. We make the same assumption for port features.

For a given feature f_i, a set of values can be grouped as a generalized concept. For example, the subset of IP addresses, ip_1 and ip_2, can be generalized to *FIREWALL* as shown in Fig. 2c. Same thing is assumed with port feature, where values 20, 21, 80 are generalized to *PRIV* as can be seen in Fig. 2d. $General(f_i)$ is the set of generalized values of f_i. As a result, a $BigDom(f_i)$ is $Dom(f_i) \cup General(f_i)$, where $Dom(f_i) \cap General(f_i) = \emptyset$. The values of any feature f_i of the generalized alarm (g) belong to

$BigDom(f_i)$. The $BigDom(f_i)$ can be represented by a tree (τ_i) which is a hierarchy as shown in Fig. 2b–d. All hierarchies are represented by trees; they can also be represented by a directed acyclic graphs. The generalized alarms represent the final result of our algorithm. Let G be the table of generalized alarms and T be the table of alarms.

Definition 2. For any given two elements α and $\beta \in \tau_i$, α is an ancestor of β if a path from the root of τ_i to β goes through α; it is denoted by $\alpha \rightarrow \beta$. A node $\mu \in \tau_i$ is a common ancestor of α and β if it is an ancestor of both α and β. Furthermore, the node μ is called the nearest common ancestor of nodes α and β, $NCA(\alpha, \beta)$, if μ is a common ancestor of α and β and is the nearest to α and β among their common ancestors. Any node in τ_i is an ancestor of itself.

To illustrate the concepts in Definition 2, as shown in Fig. 2c, the $NCA(ip_1,ip_3)$ is DMZ and the $NCA(ip_5,DMZ)$ is DMZ. In addition, $NCA(ip_6, ip_6)$ is ip_6 because any node is the ancestor of itself.

Julisch [2] has used a measure to compute the distance between alarms' feature values; we will denote this measure by M_A. Using M_A, the distance between α and β is the length of the shortest path in τ_i that connects α and β; all edge weights in Fig. 2 are assumed to be ones.

In this paper, we will use different measure to compute the distances between features values, named as M_B. The distance using M_B, $dist(\alpha, \beta)$, is the weighted sum of the shortest path in τ_i (using the sum of edges weights in τ_i) which connects α and β. The edges weights, when using M_B, are assumed to be as in Fig. 2. As a result, the distance is computed using Eq. (1).

$$dist(\alpha,\beta) = \begin{cases} dist(\alpha,\beta) & \text{if } \alpha \rightarrow \beta, \\ dist(\alpha,p) + dist(\beta,p), & otherwise, \end{cases} \tag{1}$$

where p is the $NCA(\alpha, \beta)$. To illustrate distance computing, using M_A, the $dist(ip_1, ip_3) = 3$, $dist(ip_5, DMZ) = 2$ and $dist(ip_6, ip_6) = 0$. When using M_B, the $dist(ip_1, ip_3) = 5$, $dist(ip_5, DMZ) = 3$ and $dist(ip_6, ip_6) = 0$.

The weights of the edges will be consecutive. We mean by these weights, the higher the value of L(nearest common ancestor) is, the distance is lesser. This will make the generalization to be more accurate. For demonstration, see Fig. 2c, ip_5 is nearer to ip_1 than *Internet*. In other words, $dist(ip_5, ip_1) < dist(ip_5,\ Internet)$. The measure M_B distinguishes between $dist(ip_5, ip_1) = 6$ and $dist(ip_5, Internet) = 9$, while M_A measure does not distinguish this, e.g., $dist(ip_5, ip_1) = dist(ip_5, Internet) = 4$. As a consequence, M_B is more meaningful than M_A.

After mentioning the distance between features values, let $Dist(x,y)$ be the distance between alarm x and alarm y, where x and $y \in T$. So, $Dist(x,y)$ is $\sum dist(x[f_i], y[f_i])$ $\forall i$: $1 \leqslant i \leqslant n$.

4.3. The proposed algorithm

This section describes an algorithm for solving alarm clustering problem. We begin in Section 4.3.1 by presenting the classic attribute-oriented induction algorithm because our algorithm is a variation of attribute-oriented induction algorithm. Section 4.3.2 describes the suggested algorithm.

4.3.1. The attribute-oriented induction algorithm

The attribute-oriented induction (AOI) [24] is a set-oriented database mining method which generalizes the task-relevant subset of data attribute-by-attribute; it is a well-known conceptual clustering algorithm [18,19,25]. AOI produces condensed tables from large relational databases, which makes it an efficient method. AOI makes use of background knowledge, i.e. generalization hierarchies, to extract several types of rules: characteristic rules, discriminant rules, cluster description rules, and multiple-level association rules. The pseudo-code of AOI is presented in Algorithm 1.

As described by Algorithm 1, the AOI algorithm composes of three main steps: (i) select a feature F_i from the table; Julisch [4] has proposed a heuristic method for this selection; (ii) generalize each value of the selected feature F_i to its parent; this generalization depends on the generalization hierarchy of feature F_i and the threshold which has associated with feature F_i; (iii) merge the identical rows and retain information about merged rows (i.e. count). The output of AOI is a set of generalized rows.

```
   Input: A relational database DB composed of $n$ features; Generalization
          hierarchies $\{\tau_1, \ldots, \tau_n\}$; A generalization threshold $t$ for each
          feature $F$;
   Output: A cluster description rules;
 1 Begin
 2 while number of rows is higher than desirable do
 3     choose some feature $F_i$;
       /* perform generalization step                              */
 4     foreach row $r \in DB$ do
 5         generalize $r[F_i]$ to its parent using $\tau_i$;
 6     end
       /* perform merge step                                        */
 7     foreach identical rows $r$, $\acute{r} \in DB$ do
 8         $r$[count]= $r$[count]+ $\acute{r}$[count] and delete $\acute{r}$ from DB;
 9     end
10 end
11 End
```

Algorithm 1: The pseudo-code of attribute-oriented induction algorithm

Alarm clustering is not an easy task because the IDS generate huge number of alarms every day and these alarms are noisy. This requires a clustering algorithm which is scalable and noise tolerance. AOI by itself is not suitable for alarm clustering because it is too sensitive to noise [4]. This motivates us to design a new clustering algorithm which is scalable and noise tolerance. The next section presents the suggested algorithm.

4.3.2. Our algorithm description

The alarm clustering problem is NP-Complete as proved by Julisch [26]. We have developed an approximation algorithm to find a set of clusters; each cluster has maximum similarity among its alarms. The proposed algorithm is different from Julisch's work in four trends. First, we used a different distance measure between features values as we have seen in Section 4.2. Second, we uses a different stop condition. Third, any generalization does not happen unless there is need. And finally, our algorithm has a *NeighbOring_thresholD* (NOD for short) control parameter that controls the distances between cluster's alarms.

The main idea of the proposed algorithm is to find clusters whose distances among their alarms are less than or equal to NOD, then find out a generalized alarm for each cluster. This process will continue until no alarms are found in T. Hereafter, the identical generalized alarms would be merged. More formally, our system works in the following steps:

- The set of alarms contains several clusters $\{\mathscr{C}_1, \mathscr{C}_2, \ldots, \mathscr{C}_m\}$ representing patterns for attacks, root causes, and noise.
- Every cluster $\mathscr{C}_i$, $1 \leqslant i \leqslant m$, contains alarms whose distances between them and the center of cluster $\mathscr{C}_i \leqslant$ NOD.
- Extract the generalized alarm for every cluster $\mathscr{C}_i$, $1 \leqslant i \leqslant m$, by finding separately the nearest common ancestor for each feature. Merge the identical generalized alarms.
- Forward these generalized alarms to security analyst in order to extract root causes and to write filters for them.

```
   Input: An alarm clustering problem (T, NOD, $\tau_1, \ldots, \tau_n$)
   Output: Set of generalized alarms G
1  Begin
2  Set C and G to Empty;
3  foreach alarm A ∈ T do
4      $k$ = number of discovered clusters;
5      distance = Dist($A$,$C_i$) where Dist($A$,$C_i$) < Dist($A$,$C_k$) ∀ $k$;
6      if distance ≤ NOD then
7          increment $G_i$[count];
           /* update generalized alarm's features values */
8          foreach $f_j$ in $G_i$ do
9              $f_j$ = NCA($f_j$, $A[f_j]$) using $\tau_j$;
10         end
11     else
12         C = C+$A$ and consider $A$ as a new cluster center;
13         G = G+$A$ and set the new generalized alarm count to 1;
14     end
15 end
   /* merge identical generalized alarms */
16 foreach identical alarms $g$, $\acute{g}$ ∈ $G$ do
17     $g$[count]= $g$[count]+ $\acute{g}$[count] and delete $\acute{g}$ from G;
18 end
19 return $G$
20 End
```

Algorithm 2: The pseudo-code of the proposed algorithm

For more detail, Algorithm 2 shows the pseudo-code of the proposed algorithm. Clusters centers are stored in table C while table G contains the generalized alarms for all clusters' center; at the beginning, line 2 sets these two tables to empty. It should be noted that for each cluster's center in C there is a generalized alarm in G having the same index. An important feature of the proposed algorithm is that it passes over T once which makes it very fast as shown in line 3. Line 5 finds the minimum distance between the new alarm and all clusters' centers in C. If this distance is ⩽NOD, then the generalized alarm's count (g[count]) of matching cluster's center is incremented (line 7). Lines 8 and 9 will check if there is necessity to generalize *g*. Fig. 3 shows how generalization happens. If the distance >NOD holds (line 12), the new alarm will be considered as a new cluster's center and will be appended to C. In addition, It is appended to G and set its count to 1 (line 13). Lines 8 and 9 generate many generalized and identical alarms that are merged in lines 16 and 17. The count field of the generalized alarm represents the number of merged alarms. Line 19 will output all generalized alarms in G.

The proposed algorithm tries to scan *T* to find the nearest neighbors of a given alarm (using the NOD parameter) and then generalizes them. It produces generalized alarms, most of which are root causes. However, if the features of the resulting generalized alarms are excessively abstracted, then the analyst can not identify the root causes of false alarms. As a result, the proposed algorithm minimizes the generalization steps trying to mitigate the influence of generalization by taking the nearest common ancestor as shown in Fig. 3.

Our algorithm is a variation of AOI algorithm. The generalization used by our method is different from the one used in AOI algorithm. AOI algorithm generalizes one concept to its parent in the hierarchy whereas our generalization is done by finding the nearest common ancestor of two concepts in a hierarchy. Furthermore, AOI algorithm generalizes all values of a given feature in each pass over the alarms table while our algorithm generalizes different features in each alarm depending on the NOD value. Our algorithm makes use of the generalization concept of AOI algorithm and the neighboring concept.

In this paper, the generalization is performed by finding nearest common ancestor for identical features of any two alarms in the same cluster. This occurs when we compute the distance between the new alarm and all clusters' centers. If the minimum distance ⩽NOD holds, then nearest common ancestor between identical features of the new alarm and the generalized alarm of the matching cluster will be computed. Put it in another way, assume that we want to find the generalized alarm for a given cluster. Then, for all alarms which belong to a cluster, the nearest common ancestor will be computed for each pair of identical features, e.g., $g[srcIP] = NCA(A_1)[srcIP], NCA(A_2[srcIP], NCA(\ldots))$. The resulting generalized alarm will be stored in table *G*, which contains the generalized alarms.

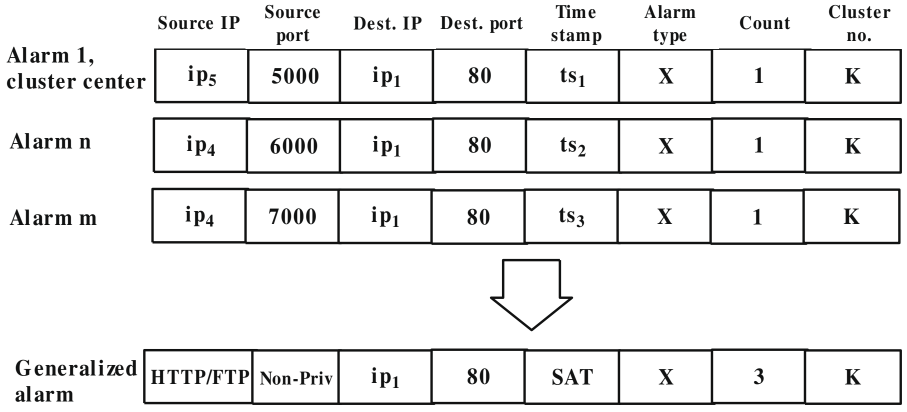
4.4. Choosing the best NOD parameter value

Most of the clustering algorithms have control parameters [27]. The proposed algorithm also has a control parameter, which is NOD. It should be adjusted carefully by the analyst. If the selected NOD value is too big, then the real root causes will be lost due to overgeneralization; the real root causes will spread over many clusters if the selected NOD value is small. The number of resulting clusters is inversely proportion to the NOD value.

In this section, we will present two methods to find the best value for the NOD parameter. The first method (method A) finds a range of values which the NOD value belongs to; the second method (method B) finds the best value for the NOD parameter. Moreover, these two methods can be used together to set the NOD parameter where the method A estimates the possible values and method B finds the best one.

4.4.1. NOD setting using heuristic (method A)

The value of the NOD parameter can be estimated depending on the generalization hierarchies (the input), as stated in the following definition:



	Source IP	Source port	Dest. IP	Dest. port	Time stamp	Alarm type	Count	Cluster no.
Alarm 1, cluster center	ip_5	5000	ip_1	80	ts_1	X	1	K
Alarm n	ip_4	6000	ip_1	80	ts_2	X	1	K
Alarm m	ip_4	7000	ip_1	80	ts_3	X	1	K
Generalized alarm	HTTP/FTP	Non-Priv	ip_1	80	SAT	X	3	K

Fig. 3. An example of the generalization process.

Definition 3. A longest path of (τ_i), $\mathrm{LP}(\tau_i)$, in an edge weighted tree is a path such that the sum of edges weights on it is nonnegative and as large as possible. It always starts and ends in leaf nodes. In the same way, a shortest path of (τ_i), $\mathrm{SP}(\tau_i)$, between any two leaves is a path such that the sum of edges weights is the smallest in the tree.

To demonstrate Definition 3, the longest path of port hierarchy as shown in Fig. 1d is six (e.g. between port 1 and port 1025) while the shortest path is two (e.g. between port 1 and port 80).

The value of NOD mainly depends on hierarchies. In other words, the distance between any two alarms is the sum of distances between the identical features which can be obtained from hierarchies. As a result, the upper and lower limits of NOD value can be estimated depending on hierarchies. For a given hierarchy (τ_i), if we consider the value of NOD as the longest path, then we will obtain too abstracted feature values. Therefore, it is better to assume a moderate value for NOD which not leads to overgeneralization. Let the value ranging from $\mathrm{SP}(\tau_i)/2$ to $\mathrm{LP}(\tau_i)/2$ be moderate range for any hierarchy. Thus, the upper and lower limits of NOD can be calculated from Eqs. (2) and (3), respectively.

$$UpperLimit = \sum_{i=1}^{n} \frac{LP(\tau_i)}{2}, \tag{2}$$

$$LowerLimit = \sum_{i=1}^{n} \frac{SP(\tau_i)}{2}. \tag{3}$$

Some hierarchies can be provided by knowledge engineers or domain experts, which are reasonable even for large databases since a concept hierarchy registers only the discrete feature values or ranges of numerical values for a feature, which is, in general, not very large [13]. In other words, the size of hierarchies is rational even for large databases. As a consequence, the ranges between the upper and lower limits of NOD values would be reasonable.

4.4.2. NOD setting using cluster validity (method B)

The best value for the NOD parameter can be determined depending on validation of the clustering results. We can run the clustering algorithm repetitively with different NOD parameter values and compare the results against a well-defined validity index. The validity index is a formula that measures goodness in a quantitative manner [4]. For any dataset, the best partitioning will maximize (or minimize) the validity index. More formally, for a given dataset, we:

- Perform n clustering runs $\mathscr{R}_{c,i}$, where $i \in [1, n]$. Each run is with different NOD_i value.
- Compute validity index value ($IDX_{\mathscr{R}c,i}$) for each $\mathscr{R}_{c,i}$.
- Choose the NOD_i value that is associated with the maximum (or minimum) $IDX_{\mathscr{R}c,i}$ value.

In the sequel, we will present the clustering validity and the main available validity indices. In other words, the validity indices that can be used with crisp clustering are listed. We will state the validity index that will be used to find the best NOD value.

Cluster validity process is used to evaluate the results of a clustering algorithm. Three approaches to investigate cluster validity are exist [28]: external criteria, internal criteria, and relative criteria. The two first approaches are based on statistical tests and their major drawback is their high computational cost. Moreover, the indices related to these approaches aim at measuring the degree to which a dataset confirms an a-priori specified scheme. On the other hand, the third approach aims at finding the best clustering scheme that a clustering algorithm can be defined under certain assumptions and parameters [29]. The more suitable criteria for NOD value estimation is the relative criteria. There are two criteria proposed for clustering evaluation and selection of an optimal clustering scheme [30]: compactness and separation. Compactness means that the members of each cluster should be as close to each other as possible while separation means that the clusters themselves should be widely spaced.

Several validity indices have been proposed in the literature for each of the above approaches [31–33,28,34]. There are many validity indices for the relative criteria, among them are: the Silhouette [35], the Dunn [36], the Davies–Bouldin [37], and the SD [29] indices. We have selected SD index because its time complexity is $O(n)$ [31]. In the sequel of this subsection, we will state this index.

The SD validity index is defined based on the concepts of the average scattering for clusters and total separation between clusters. The average scattering for clusters [31] is defined by Eq. (4):

$$Scat(c) = \frac{1}{c} \sum_{i=1}^{c} \frac{\|\sigma(v_i)\|}{\|\sigma(X)\|}, \tag{4}$$

where c is the number of clusters, v_i is the center of cluster i, $\sigma(v_i)$ is the variance of cluster i, and $\sigma(X)$ is the variance of a dataset. The definition of total separation between clusters is given by Eq. (5) [31]

$$Dis(c) = \frac{D_{max}}{D_{min}} \sum_{k=1}^{c} \left(\sum_{z=1}^{c} \|v_k - v_z\| \right)^{-1}, \tag{5}$$

where $D_{max} = \max(\|v_i - v_j\|)\forall i,\ j \in \{1, 2, \ldots, c\}$ is the maximum distance between cluster centers. The $D_{min} = \min(\|v_i - v_j\|)\forall i, j \in \{1, 2, \ldots, c\}$ is the minimum distance between cluster centers. Now, a validity index SD can be computed using Eq. (6) [31]

$$SD(c) = \alpha Scat(c) + Dis(c), \tag{6}$$

where α is a weighting factor equal to $Dis(c_{max})$ where c_{max} is the maximum number of clusters. The first term ($Scat(c)$ that is defined by Eq. (4)) indicates the average compactness of clusters. A small value for this term indicates compact clusters and as the scattering within clusters increases (i.e., they become less compact) the value of $Scat(c)$ also increases. The second term $Dis(c)$ indicates the total separation between the c clusters. Contrary to the first term the second one, $Dis(c)$, is influenced by the geometry of the clusters centers and increase with the number of clusters [29]. The NOD parameter value that minimizes the SD index can be considered as the best value.

5. Experiments and evaluation

In this section, we describe the experiments conducted to evaluate our system. The proposed system was tested on an AMD Athelon processor 2.01 GHz with 512 RAM running Windows XP. Three different datasets were used in our experiments: a real dataset, DARPA 1998 dataset [38], and DARPA 1999 dataset [39]. The real dataset was collected during the daily operation of an educational network. For all of these datasets we ran Snort [15], an open-source signature-based IDS, to generate alarms. Table 2 presents some details about these datasets. In the following three subsections, we will see the results with these datasets.

Different methods have been used to label the resulting alarms. For the real dataset, we were depending on our experience to label the alarms. We used an automatic approach to label DARPA 1998 datasets based on attack truth table. Three criteria were considered to label any alarm as attack; these criteria are: (i) matching source IP address, (ii) matching target IP address, and (iii) alarm time stamp in the time window in which the attack has occurred. For the DARPA 1999 dataset, we manually labeled IDS alarms using attack truth tables. One of the reasons behind the using of DARPA datasets is to let other researchers reproduce our results (or compare their results with ours).

The work with our algorithm composed of two stages; clustering and filtering stages. The resulting clusters from clustering stage (i) are subsequently used as rules in the following filtering stage ($i+1$). Each dataset has been divided into periods so as to filter the alarms of the current period using the filters resulted from preceding period. The period in the real dataset is selected to be one month while it is selected to be one week for the DARPA datasets. For DARPA datasets, we did not include the first week because we can not apply filters on it; but it can be used as a classifier for the second week. The false alarms clusters of any period are converted to rules in order to filter out the false alarms in the subsequent period.

5.1. Results on a live network

This section presents experiments which we have performed by clustering a log of historical alarms. The objectives from this experiment are to see the interpretability of resulted clusters, and to show the influence of the distance measure on reduction ratio. The alarms log was taken from Snort IDS [15], during a period of two months; the first month was used for generating filtering rules which have been used to classify the second month alarms. The Snort IDS sensor has been deployed in a network which is similar to the one in Fig. 2a. For this dataset, the IP hierarchy was derived from the background knowledge of our network which composed of Inside, Internet, Router, DMZ, etc., as concepts. We used the same time and port hierarchies in Fig. 2.

The resulting generalized alarms (from the first month) of the six largest alarm clusters are shown in Table 3; they summarize about 96% of all alarms. Each line of the table represents one alarm cluster; the count column indicates cluster size. The *undefined* value in the port columns indicates that the IDS did not generate any value for the feature such as the ICMP protocol which has no ports notion. For privacy purposes, the IP addresses of our network are sanitized as ip_1, ip_2, etc. referring to machines as in Fig. 2a.

It should be noted that the generalized alarms can perfectly suggest root causes. Furthermore, these root causes need to be validated thus requiring experience in network security and environment information. We have analyzed the generalized alarms presented in Table 3 as follows.

FULL XMAS scan. It is possible for Snort IDS to generate a false alarm that looks like this alarm. Here, the IDS thought that the router was running a scan. After investigating the router, we discovered that it was faulty which caused various unnatural packets to be sent. The problem involved bits from the port specification being copied into flags field, causing bogus flags combinations to be set and this was the root cause.

(ftp_telnet) FTP traffic encrypted. Many encrypted FTP traffic alarms were triggered, which was strange because we were not using any FTP clients at all. Using the *eMule* program, as our analysis showed, was the root cause. Because all ports are freely configurable in *eMule*, some users were setting the ports of *eMule* identical to FTP ports. In other words, *eMule*'s connection on FTP port causes reports of FTP attacks.

DDOS Stacheldraht agent->handler skillz. The Snort IDS showed very large amount of Stacheldraht alarms. It could tell that Stacheldraht is a distributed denial-of-service tool (DDoS). Further investigations showed us that Stacheldraht is the DDoS tool that attacks systems with floods of ICMP traffic. Since we were quite certain that our network was not suffering that many DDoS attacks, we further analyzed the traffic. This showed that a system in the network of the internet service provider was misconfigured, sending packets that contained the text-string: "This is not an attack".

http_inspect: BARE BYTE UNICODE ENCODING. This alarm simply highlights the fact that many clients issue this alarm. The root cause, after investigation, was that this alarm came up all the time when the *McAfee* virus scan enterprise clients contacted *McAfee* server over port 80.

ICMP Redirect Host. There was large number of ICMP Redirect Host alarms that came from a DMZ host directed to other DMZ hosts. A closer investigation of this generalized alarm indicated that a host was sending messages to other DMZ hosts about the existence of the firewall; therefore, these alarms can be omitted.

Telnet Access. This alarm was generated after each successful telnet connection. This event occurs when a remote user from outside the internal network successfully connects to a telnet server. Because authorized users are allowed, in our policy, to connect remotely using telnet; this alarm was considered as false positive.

Note that for the third alarm type, we found that it is difficult to specify the actual root cause; it is still unclear to us what made the IDS decides that there was a DDoS attack. This, however, is no limitation of our alarm clustering method. In fact, even when we looked at raw intrusion detection alarms, we could not ascertain the root causes. Too much information was missing because the IDS provides too little information about the alarms. Hence, with or without alarm clustering, there are cases where we do not have enough information to identify root causes with certainty. However, in most cases, the obtained generalized alarms was easy to interpret.

We used the historical alarms to generate the generalized alarms which are presented in Table 3. Instead of removing these root causes, the discovered root causes were converted to rules. These rules were then used to filter the alarms of the second month, reducing the alarms load by about 93%. Therefore, resolving of root causes reduces the work of the subsequent month.

The distance measure has a big influence on the results as can be seen in Fig. 4. The results for the two measures, M_A and M_B, were approximately similar but when using M_B the results were more

Table 2
Properties of the datasets used in this paper.

Datasets	Public	Experiments duration	# of alerts
Real dataset	No	2 months	302,473
DARPA 1998	Yes	5 weeks	295,484
DARPA 1999	Yes	5 weeks	233,615

Table 3
The main generalized alarms produced by our algorithm with real dataset.

Alarm type	Src-IP	Src-Port	Dst-IP	Dst-Port	Time	Count
FULL XMAS scan	Router	Any	Inside	Any	Any	51,935
(ftp_telnet) FTP traffic encrypted	Internet	Non-priv	Inside	21	Any	35,780
DDOS Stacheldraht agent → handler skillz	Internet	Undefined	Inside	Undefined	Any	25511
http_inspect: BARE BYTE UNICODE ENCODING	Inside	Non-priv	ip_2	80	Any	9021
ICMP Redirect Host	ip_1	Undefined	*DMZ*	Undefined	Work_day	7291
Telnet Access	Internet	Non-priv	ip_{16}	23	Work_day	2867

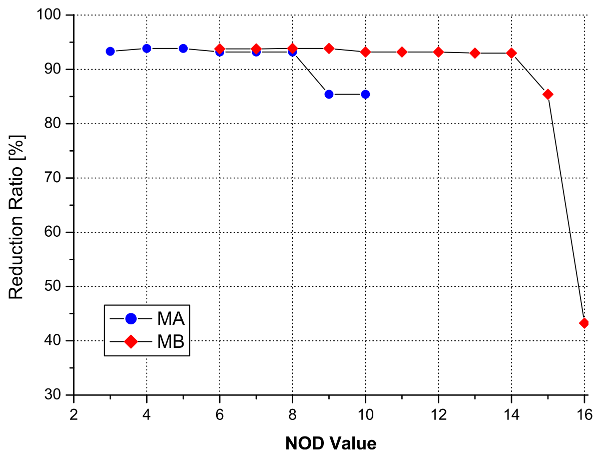


Fig. 4. The influence of distance measures, M_A and M_B, on the reduction ratio.

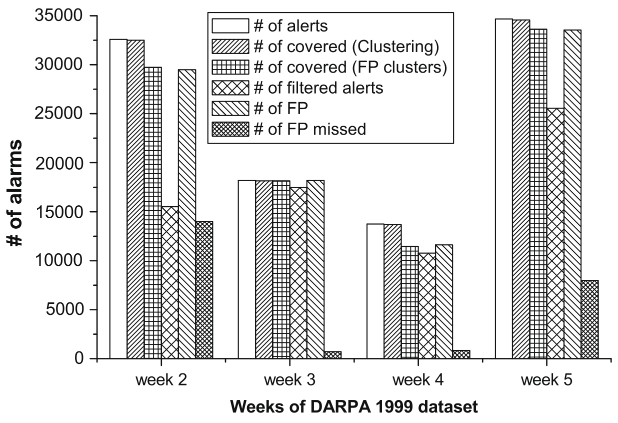


Fig. 5. Clustering and filtering runs for the DARPA 1999 dataset.

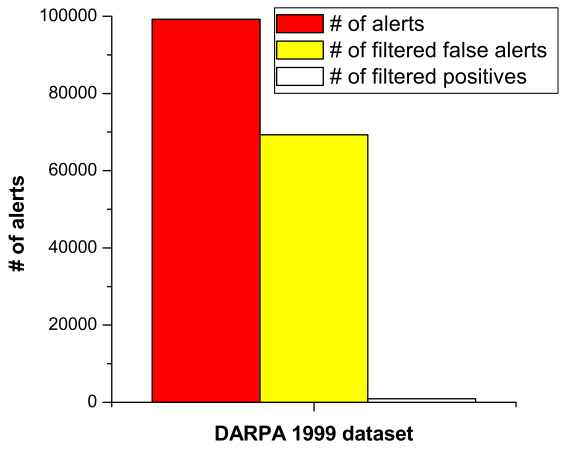


Fig. 6. Total alarms reduction and the missed true positives for the DARPA 1999 dataset.

stable due to less sensing to changes of the NOD value. This figure also shows the impact of NOD parameter on the results. It can be noted from the mentioned figure that whenever the NOD value increases, then the reduction ratio decreases. This is because some of root causes were lost due to overgeneralization. In addition, the reduction ratio was decreased when the NOD value was too small. We will see more results on the impact of the NOD parameter in the next section.

5.2. Results on DARPA 1999 dataset

The objective of this set of experiments is to analyze the clustering results, to show that the filtering with our method is safe, to state how the results affected by the NOD parameter, and to find the best value for the NOD parameter. The dataset used in this set of experiments is DARPA 1999 dataset.

The alarms with this dataset are divided into five weekly logs. We performed five clustering runs and four filtering runs on this dataset; each of the runs producing up to several hundred clusters. Fig. 5 shows the performance of our system for all clustering and filtering runs for DARPA 1999 dataset.

The reduction ratio with this experiment was about 70% as can be seen in Fig. 6 while the number of missed true alarms was 955 alarms. When we investigated the filtered true alarms, we found that all of them were scan alarms. Those true alarms belong to the following attacks: training attack #20 (ip sweep, week 2, 761 alarms), test attacks #41 (port sweep, week 4, 170 alarms), #54 (probe, week 5, 16 alarms), #55 (port sweep, week 5, 8 alarms).

When we investigated the filtering rule that filters out attack #20, we found that there was the same attack in week one but labelled as false alarms because week one is free of attacks. This leaves us with the remaining three incidents, namely testing attacks #41, #54, #55 for which our algorithm removed true alarms. As we have note, all those incidents are scanning incidents (i.e. port sweep and probe). Depending on site policies, port scans have become so common on the Internet that most administrators consider them as mere nuisances and do not spend any time or resources in tracking down their sources [40]. However, intrusion detection is a multi-stage process, in which after an incident has been detected, all incidents it comprises are found using link-mining (analyzing related alarms). This means that even if those alarms were removed, the incidents they belong to would not have been missed, and they are likely to be rediscovered in the forensic stage [23].

The NOD parameter value must be adjusted carefully. To do that, we have suggested two methods to setting its value (i.e. method A and method B). Using method A, we did note that $10 \leqslant \text{NOD} \leqslant 21$ holds. When we used method B, different SD validity indices were obtained from several clustering executions (using Eq. (6)); the results can be shown in Fig. 7. In this figure, we do note that the best value for the NOD parameter is associated with the minimum SD index value; which is 18. We conclude from this figure that the SD index is capable of locating the best value for NOD parameter. Moreover, the best value obtained from method B belongs to the range of values which have been obtained from method A.

To show more statistics on the DARPA 1999 datasets results, we have classified the resulted clusters to three classes which are false-alarms clusters (FA-only clusters, which contain only false alarms), true-alarms clusters (TA-only clusters, which contain only true alarms), and mixed clusters (which contain false and true alarms). Table 4 shows the three classes of clusters, in the clustering stage, with different NOD values. For example, DARPA 1999 dataset generated 959 clusters for the entire experiment (with NOD value equal to 18), 878 of which (covering 96% of clustered alerts) were FA-only clusters, 43 (covering 2% of alarms) were TA-only clusters and the remaining 38 clusters (covering 2% of clustered alarms) were mixed clusters. This means that only 878 clusters would be used for the filtering stage. In addition, we can clearly note, from Table 4, the influence of the NOD parameter on the results. The small value for it causes the spread of the root causes over the clusters; moreover, the number of clusters would be increased.

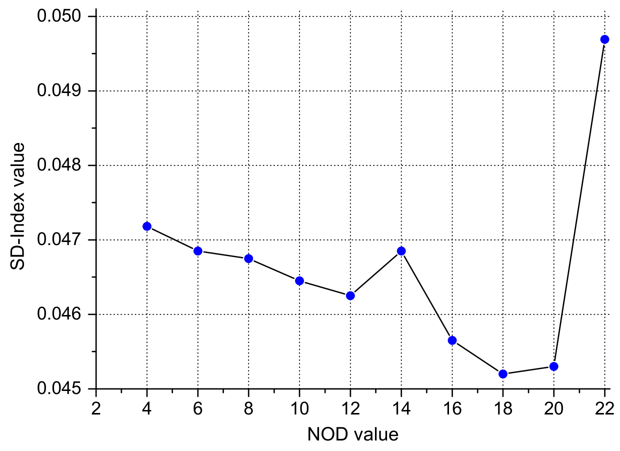


Fig. 7. The application of SD validity index (Eq. (6)) on the DARPA 1999 dataset with different NOD values suggests that the best value for the NOD parameter is 18.

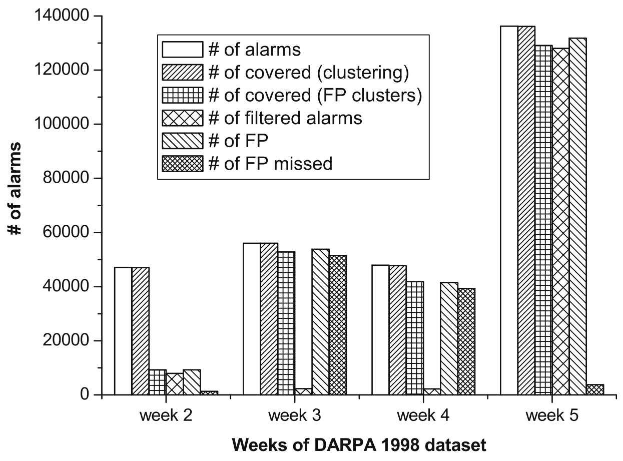


Fig. 8. Clustering and filtering runs for the DARPA 1998 dataset.

5.3. Results on DARPA 1998 dataset

The DARPA 1998 dataset is used in this experiment. One of the goals from using this dataset was to average the reduction ratio of our algorithm. The reduction ratio with this experiment was about 59.41%. We think that there are errors in the labeling process because we used automatic labeling, as mentioned in the beginning of Section 5. Unlike the labeling of DARPA 1999 dataset which was labeled manually making the labeling process more accurate. These errors have led to miss several filters, which cause this low reduction ratio.

The performance of the system far all clustering and filtering runs can be seen in Fig. 8. For each week in this figure, the second column shows the total number of alarms covered in the clustering stage, and the third column shows only those covered alarms which are FA-only clusters. Those FA-only clusters (the third column) in any week were used to filter the alarms (the fourth column) in the subsequent week. The fifth column represents the false alarms existing in every week. We do note that the second week contains few false alarms because this week contains several attacks; week three contains several false alarms because it is attack free. The last column shows the false alarms which have evaded filtering.

6. Discussion

Data mining is an important field for intrusion detection, where the alarm filters can be used to remove the most prevalent and uninteresting false positives. We have proposed an effective approach to reduce false positives in sensor alarms reports by clustering them with abstraction and then using the clusters to discover and understand the root causes of alarms. We have discussed the effectiveness of our approach in Section 6.1. Section 6.2 presents the assumptions and limitations of our system.

Table 4
Clustering statistics for the clustering stage with different NOD values.

NOD value	# FA-only clusters	Fraction alarms	# TA-only clusters	Fraction alarms	# Mixed clusters	Fraction alarms
4	4278	0.966	50	0.02	34	0.015
6	3352	0.964	45	0.02	38	0.016
8	2539	0.962	43	0.02	37	0.018
10	1807	0.962	43	0.02	39	0.018
12	1663	0.962	43	0.02	39	0.018
14	1383	0.960	43	0.02	38	0.020
16	1027	0.960	43	0.02	38	0.020
18	878	0.960	43	0.02	38	0.020
20	701	0.949	40	0.02	38	0.031
22	448	0.933	36	0.014	36	0.053

6.1. Effectiveness of the proposed system

Now, we examine how well the proposed algorithm meets the requirements listed in Section 3.2:

- *Scalability*: The performance of our algorithm depends on the number of received alarms and the number of clusters in them. The proposed system was tested on an AMD Athelon processor 2.01 GHz with 512 RAM running Windows XP, and processed about 150,000 alarms in less than three minutes.
- *Noise tolerance*: The noise, typically found in intrusion detection alarms, leads to overgeneralization problem. The proposed algorithm uses the conditional generalization to avoid this problem. The conditional generalization depends on carefully tuning the NOD parameter. The good value for the NOD parameter will isolate the noise out of the main clusters because the noise has different features values with them.
- *Multiple feature types*: The AOI has the capability of dealing with a wide variety of attributes types, such as numerical attributes, categorical attributes, time attributes, etc. [4]. The proposed algorithm makes use of the generalization concept of AOI and the neighboring concept with resemblance to AOI. As a consequence, it can deal with a variety of attribute types. we have seen how the proposed technique processed different features types of alarms like IP addresses, time, count, and port features.
- *Ease of use*: The two main inputs to the algorithm are the hierarchies and the NOD parameter. The defining of hierarchies requires some expertise in the application domain. The hierarchies are defined for once, so they are static. Two methods have been presented in Section 4.4 to set the NOD parameter.
- *Interpretability*: By avoiding overgeneralization, we were got precise generalized alarms which facilitate the interpretation process.

To avoid overgeneralization, we compute the distance between the new alarm and all clusters' centers, then the generalization occurs if the distance value is below or equal to NOD value; this checking is done before generalization. Furthermore, the generalization does not occur unless there is a guarantee that all the features of an alarm and the compared cluster's center are similar.

The suggested measure, M_B, has a great influence on the results. Given two alarms a_1 and a_2, which belong to the same generalized alarm; the smaller the $Dist(a_1, a_2)$ is, the measure is more accurate. As have shown in Section 4.2, the suggested measure has minimized the distances in such case while the semantics of the distances have been retained. The more accurate the measure is, the generalization steps are lesser. Minimizing the generalization steps means that the results capture more detailed information about the alarms; which facilitate the interpretation of the results.

The reduction ratio greatly depends on the position of the IDS and individual network (i.e. number of false alarms). For example, the IDS located in the intranet triggers huge number of false alarms due to network management systems; hence the reduction ratio would be high. On the other hand, most of alarms triggered by the IDS, which monitors the internet, are real attacks. The real attacks are fleeting and then have small influence on the reduction ratio. As a result, it does not seem to expect this method to perform equally on other networks.

To find the class of a given alarm, it should be checked against the existing clusters, then assign it to the nearest cluster if the distance between that alarm and the nearest cluster's center $\leqslant$NOD value holds. As shown in Algorithm 1, the time for processing each new alarm with the proposed algorithm is linear in k which is the number of detected clusters; this appears in line 5. The other steps of the algorithm, lines 6–14, take a constant time. Steps 16–18 merge duplicated clusters taking a quadratic time with respect to k; this time can be ignored if it is compared to the number of alarms, n, because $k \ll n$ holds. Hence, the performance of the proposed algorithm depends on the number of received alarms and the number of clusters in them. In other words, the time complexity of the proposed algorithm is $O(n * k)$ which appears to be attractive. The processing time per alarm (averaged per 5000 alarms) can be seen in Fig. 9.

The NOD parameter has a big influence on the time complexity of the algorithm. On one hand, the too big NOD value enforces the algorithm to produce few clusters, which accelerates the execution time. On the another hand, the too small value for NOD parameter causes the algorithm to produce many clusters, which slows down the running time. In other words, the zero value for NOD parameter makes the time complexity of the algorithm to be $O(n^2)$ because every alarm will be an individual cluster, but this is not an interesting case.

The application of our algorithm to DARPA 1998 dataset, DARPA 1999 dataset, and real dataset confirming that on average up to 74% of false alarms can be discarded, which reduced the work of the security analyst by about three-quarters. The total alarm reduction for all datasets can be shown in Fig. 10. We have noted that the reduction ratio of real dataset is larger than DARPA datasets. The reason for this is that the DARPA datasets include higher percentage of attacks than the real dataset.

6.2. Assumptions and limitations

In this paper, we have made several assumptions. These assumptions should be had in mind when implementing our system. First, we assumed that our system focuses on discovering the root causes which manifest themselves in large groups of alarms; most of these root causes related to problems in the computing infrastructure. The proposed system does not look for small, stealthy attacks in the alarm logs, but aims to reduce the false alarms to make it easier to identify real attacks in the subsequent analysis. It is not responsible for detecting the bad designed rules of the IDS. Furthermore, it is not looking for the false alarms which are related to bad design of the cluster.

Second, in the filtering stage, when the new alarm is covered by more than one filter (i.e. the multiple membership problem) we took the first matching filter. However, the nearest matching filter idea can be adopted as future work.

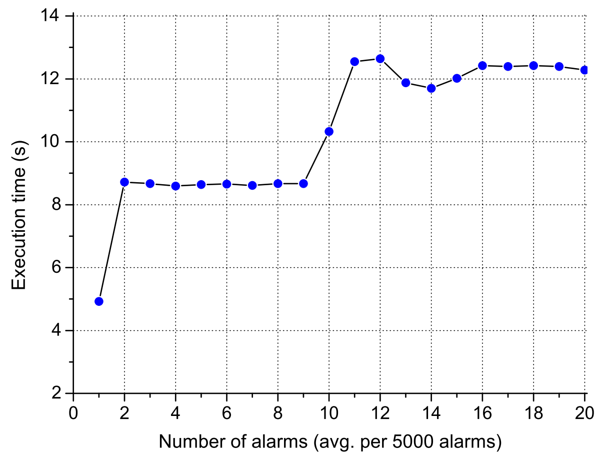


Fig. 9. The performance of the proposed algorithm.

Third, in spit of the fact that there are several methods to build the generalization hierarchies automatically, we assume that there are well-defined generalization hierarchies for alarms' attributes. In addition, we assume that all those hierarchies are represented by trees, while they can be represented by lattice, directed acyclic graph, or tables.

It should be noted that this type of work has some limitations. First, the filtering would be unsafe in case of misclassifying true positives as false positives. The writing of filtering rules requires care and experience; otherwise, the risk of discarding true positives can be high. The filtering in this work does not increase undetected attacks or miss true positives. As suggested by Julisch, if the security analyst is skilled, the environment does not change in ways that the security analyst could not anticipate, and the attacker does not know the filtering rules, then the filtering is safe [4]. These factors, however, is no limitation of our method. Nevertheless, write specific rules, keep rules secret, remove outdated rules, and filter when not vulnerable are guidelines recommended by Julisch for safe filtering [4].

Second, different hierarchies can be constructed on the same feature based on different viewpoints or preferences. Moreover, different rules can be extracted from the same set of data using dif-

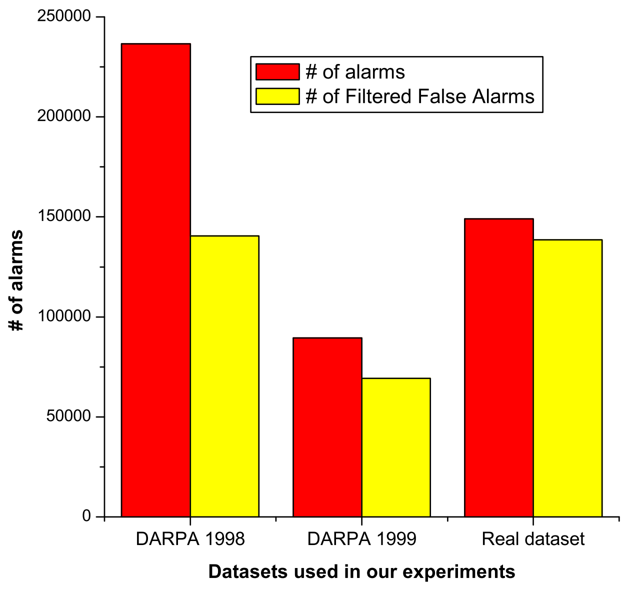


Fig. 10. Total alarm reduction for DARPA 1998, DARPA 1999, and real datasets.

ferent hierarchies [41]. Consequently, the usefulness of resulting clusters would depend on the quality and possibly granularity of the hierarchies.

7. Conclusions

Most of the generated alarms from IDS are false positives. Fortunately, there are reasons for triggering alarms where most of these reasons are not attacks. In this paper, a new data mining technique has been developed to group alarms and to produce clusters. Hereafter, each cluster is modeled by a generalized alarm. The generalized alarms related to root causes are converted to filters to reduce future alarms load. The proposed algorithm makes use of nearest neighboring and generalization concepts to cluster alarms. As a clustering algorithm, the proposed algorithm uses a new measure to compute distances between alarms' features values. This measure depends on background knowledge of the monitored network, making it robust and meaningful. The averaged reduction ratio of the new data mining technique was about 74% of the total alarms.

Acknowledgements

The authors thank the anonymous reviewers for their valuable comments and suggestions in improving this paper.

References

[1] S. Manganaris, M. Christensen, D. Zerkle, K. Hermiz, A data mining analysis of RTID alarms, Journal of Computer Network 34 (2000) 571–577.
[2] K. Julisch, Clustering intrusion detection alarms to support root cause analysis, ACM Transaction on Information and System Security 6 (2003) 443–471.
[3] R. Perdisci, G. Giacinto, F. Roli, Alarm clustering for intrusion detection systems in computer networks, Journal of Engineering Application of Artificial Intelligence 19 (2006) 429–438.
[4] K. Julisch, Using root cause analysis to handle intrusion detection alarms, Ph.D. dissertation, University of Dortmund, 2003.
[5] O.M. Dain, R.K Cunningham, Fusing a heterogeneous alert stream into scenarios, in: Proceeding of the 2001 ACM Workshop on Data Mining for Security Applications, 2001, pp. 231–235.
[6] P. Ning, Y. Cui, D.S. Reeves, D. Xu, Techniques and tools for analyzing intrusion alerts, ACM Transaction on Information and System Security 7 (2004) 274–318.
[7] S.O. Al-Mamory, H. Zhang, A survey on IDS alerts processing techniques, in: Proceeding of the 6th WSEAS International Conference on Information Security and Privacy (ISP'07), Spain, 2007, pp. 69–78.
[8] A. Siraj, R. Vaughn, Multi-level alert clustering for intrusion detection sensor data, in: Proceeding of North American Fuzzy Information Processing Society International Conference on Soft Computing for Real World Applications, Michigan, 2005.
[9] A. Valdes, K. Skinner, Probabilistic alert correlation, in: Proceeding of the Recent Advances in Intrusion Detection, LNCS 2212, 2001, pp. 54–68.
[10] B. Zhu, A.A. Ghorbani, Alert correlation for extracting attack strategies, International Journal of Network Security 3 (3) (2006) 244–258.
[11] R. Smith, N. Japkowicz, M. Dondo, P. Mason, Using unsupervised learning for network alert correlation, LNCS Advances in Artificial Intelligence (2008) 308–319.
[12] T. Pietraszek, Using adaptive alert classification to reduce false positives in intrusion detection, in: Proceeding of the Recent Advances in Intrusion Detection, France, 2004, pp. 102–124.
[13] A.D. Livingston, G. Jackson, K. Priestley, Root Cause Analysis: Literature Review, HSE Books, 2001. pp. 1–62.
[14] M. Paradies, D. Busch, Root cause analysis at Savannah river plant, in: Proceeding of the IEEE Conference on Human Factors and Power Plants, 1988, pp. 479–483.
[15] M. Roesch, Snort-lightweight intrusion detection for networks, in: Proceeding of the 1999 USENIX LISA Conference, 1999, pp. 229–238.
[16] D. Hand, H. Mannila, P. Smyth, Principles of Data Mining, The MIT Press, 2001.
[17] S.M. Bellovin, Packets found on an internet, Journal of Computer Communication Review 23 (1993) 26–31.
[18] J. Han, Y. Fu, Exploration of the power of attribute-oriented induction in data mining, in: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.), Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, 1996, pp. 399–421.
[19] J. Han, Y. Cai, N. Cercone, Data-driven discovery of quantitative rules in relational databases, IEEE Transaction on Knowledge and Data Engineering 5 (1993) 29–40.
[20] J. Dougherty, R. Kohavi, M. Sahami, Supervised and unsupervised discretization of continuous features, in: Proceedings of the 12th International Conference on Machine Learning, 1995, pp. 194–202.
[21] J. Han, Y. Fu, Dynamic generation and refinement of concept hierarchies for knowledge discovery in databases, in: Proceedings of the AAAI Workshop on Knowledge Discovery in Databases, 1994, pp. 157–168.
[22] Y. Lu, Concept Hierarchy in Data Mining: Specification, Generation, and Implementation, Master's Thesis, Simon Fraser University, Canada, 1997.
[23] T. Pietraszek, Alert classification to reduce false positives in intrusion detection, Ph.D. dissertation, Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Germany, July 2006.
[24] J. Han, M. Kamber, Data Mining: Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems), Morgan Kaufmann, 2000.
[25] O. Heinonen H. Mannila, Attribute-oriented induction and conceptual clustering, Technical Report, University of Helsinki, Department of Computer Science, 1996.
[26] K. Julisch, Mining alarm clusters to improve alarm handling efficiency, in: Proceeding of the 17th Annual Computer Security Applications Conference, New Orleans, 2001, pp. 12–21.
[27] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Computing Surveys 31 (1999) 264–323.
[28] S. Theodoridis, K. Koutroubas, Pattern Recognition, Academic Press, 1999.
[29] M. Halkidi, Y. Batistakis, M. Vazirgiannis, On clustering validation techniques, Journal of Intelligent Information Systems 17 (2001) 107–145.
[30] M.J.A. Berry, G. Linoff, Data Mining Techniques for Marketing, Sales and Customer Support, John Wiley & Sons, Inc., 1996.
[31] M. Halkidi, M. Vazirgiannis, I. Batistakis, Quality scheme assessment in the clustering process, in: Proceeding of the 4th European Conference on Principles of Data Mining and Knowledge Discovery, 2000, pp. 265–276.
[32] M.R. Rezaee, B.B.F. Lelieveldt, J.H.C. Reiber, A new cluster validity index for the fuzzy c-mean, Pattern Recognition Letters 19 (1998) 237–246.
[33] S.C. Sharma, Applied Multivariate Techniques, John Wiley & Sons, 1996.
[34] X.L. Xie, G. Beni, A validity measure for fuzzy clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (8) (1991) 841–847.
[35] P.J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics 20 (1) (1987) 53–65.
[36] J.C. Bezdek, N.R. Pal, Some new indexes of cluster validity, IEEE Transactions on Systems, Man, and Cybernetics, Part B 28 (3) (1998) 301–315.
[37] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Transactions on Pattern Analysis and Machine Intelligence 1 (2) (1979) 224–227.
[38] J. Mchugh, Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory, ACM Transactions on Information and System Security 3 (4) (2000) 262–294.
[39] MIT Lincoln Laboratory, 1999 DARPA intrusion detection evaluation data set, 1999. Web page at <http://www.ll.mit.edu/IST/ideval/data/1999/1999dataindex.html>.
[40] F. Valeur, G. Vigna, C. Kruegel, R.A. Kemmerer, A comprehensive approach to intrusion detection alert correlation, IEEE Transactions on Dependable and Secure Computing 1 (3) (2004).
[41] J. Han, Y. Cai, N. Cercone, Knowledge discovery in databases: an attribute-oriented approach, in: Proceeding of the 18th International Conference on Very Large Databases, 1992, pp. 547–559.