J Comput Virol (2010) 6:43–55
DOI 10.1007/s11416-008-0104-2

ORIGINAL PAPER

New data mining technique to enhance IDS alarms quality

Safaa O. Al-Mamory · Hongli Zhang

Received: 6 April 2008 / Revised: 16 August 2008 / Accepted: 20 August 2008 / Published online: 10 September 2008

Abstract The intrusion detection systems (IDSs) generate large number of alarms most of which 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 abstracted as 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 new data mining technique was verified with many datasets, and the averaged reduction ratio was about 82% 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].

Clustering is used by many researchers to mitigate the flood of alarms especially when alarm processing deals with huge number of alarms [3–5]. Julisch [2,6] propounded an interesting method using data mining. They introduced alarm clustering as a method to support root cause discovery. The root cause of an alarm was defined as the "reason for which it occurs." He argued that root causes are primarily responsible for the large number of redundant alarms and most of these root causes are generated because of configuration problems and thus can be fixed by manual interception. His work outlines semi-automatic approach for reducing false positives in alarms by identifying the root causes automatically and then writing rules to filter them. Such rules can drastically reduce future alarms load. However, this method overlooked the following shortcomings: First, a weak measure was used to compute distances between the alarms. Second, to solve an overgeneralization problem, Julisch's algorithm passes over a table of alarms many times [6]. Finally, the stop condition is complex which depends on an input threshold.

The primary focus of this paper is on addressing these shortcomings. We developed the new approximation algorithm to cluster the alarms; these clusters are abstracted as 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 every cluster. By representing each cluster by a cluster's 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 every cluster's alarms. The contribution of this paper is developing new

S. O. Al-Mamory (✉) · H. Zhang
School of Computer Science and Technology,
Harbin Institute of Technology, 150001 Harbin, China
e-mail: safaa_vb@yahoo.com

H. Zhang
e-mail: zhl@pact518.hit.edu.cn

data mining method which has new features generalization technique to avoid overgeneralization, and has good distance measure between features values. Application of this method to alarms log greatly helps the analyst in identifying the root causes; and then reduces the alarm load in the future.

This paper is organized as follows. Section 2 reviews related works. Section 3 shows the proposed system framework and the requirements. Section 4 presents the proposed alarms clustering algorithm. Section 5 presents the empirical results. The discussion is presented in Sects. 6 and 7 concludes this paper and suggests future work.

2 Related works

In the last decade, researchers have designed many systems that deal 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. [5] proposed a clustering technique to reconstruct attack scenarii. Probabilistic alert correlation finds similarity between alerts that match closely, if not exactly [8]. According to Valdes et al., probabilistic alert correlation correlates attacks over time, over multiple attempts, and from multiple sensors. Alert correlation tasks consist of [8]: 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. [9] 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. [4] used clustering to introduce a concise view about attacks and to reduce the volume of alarms. Julisch [2,6] 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 alerts 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 is the work of Ning et al. [10] and the like. This method generates correlation graphs depending on prerequisites/consequences knowledge of individual alerts. They proposed a correlation model based on the inherent observation that most intrusions consist of many stages, with the early stages preparing for the later ones. The correlation model is built upon two aspects of intrusions that are, prerequisites/consequences knowledge. With knowledge of prerequisites/consequences, their model can correlate related alerts by finding causal relationships between them. They used hyper alert correlation graphs to represent the alerts, where the nodes represent hyper alerts and the edges represent *prepare for* relation.

The Morin et al. model (M2D2) [11] has provided concepts and relations relevant to the information system security. It relies on a formal description of sensor capabilities in terms of scope and positioning, to determine if an alert is false positive. This model can be used to facilitate event aggregation and correlation.

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 states some basic concepts that is used in the system. Section 3.1 presents some data mining principals and the requirements which should be satisfied by the proposed method. Section 3.2 describes the structure of our system.

3.1 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 [13].

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 [13]. 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 [6], which should be fulfilled by the proposed technique:

- *Scalability* Scalability of pattern and rule discovery algorithms is obviously an important issue [13]. Because of the fact that more than a million alarms are triggered by IDSs per month [6], 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 [13].
- *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 [14]. 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 [6]. 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 [13].

3.2 Framework overview

As defined by Julisch [6], a root cause (e.g., a worm) affects one or more components (e.g., the hosts in a subnet), which in turn causes these components to trigger alarms (when the worm spreads by attacking other machines). Similarly, a non-standard protocol can affect a server and cause it to trigger "Suspicious protocol" alarms. Root cause analysis is concerned with identifying the type and locations of root causes

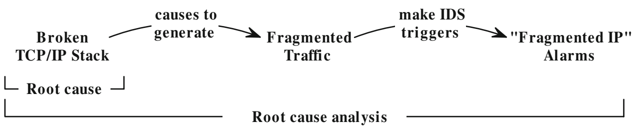

Fig. 1 An example states what are the root cause and root cause analysis

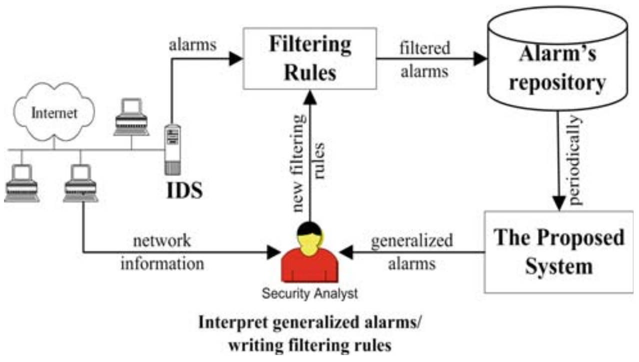

Fig. 2 The framework of the proposed system

[6]. Figure 1 states these two concepts with an example taken from [6].

The suggested data mining technique do extract alarm patterns. Our method makes use of the 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 can extract a pattern from each cluster. The extracted patterns help the security analyst in specifying the root causes behind these false alarms and writing accurate filtering rules.

The framework of the proposed system can be seen in Fig. 2. 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. It 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.

4 The proposed algorithm

This section describes the proposed algorithm in details. Section 4.1 shows the generalization hierarchies as a background knowledge. Section 4.2 proposes the distance measure which has been suggested to compute the distance between alarms. Section 4.3 presents our algorithm and Sect. 4.4 focuses on finding a heuristic to setting the control parameter.

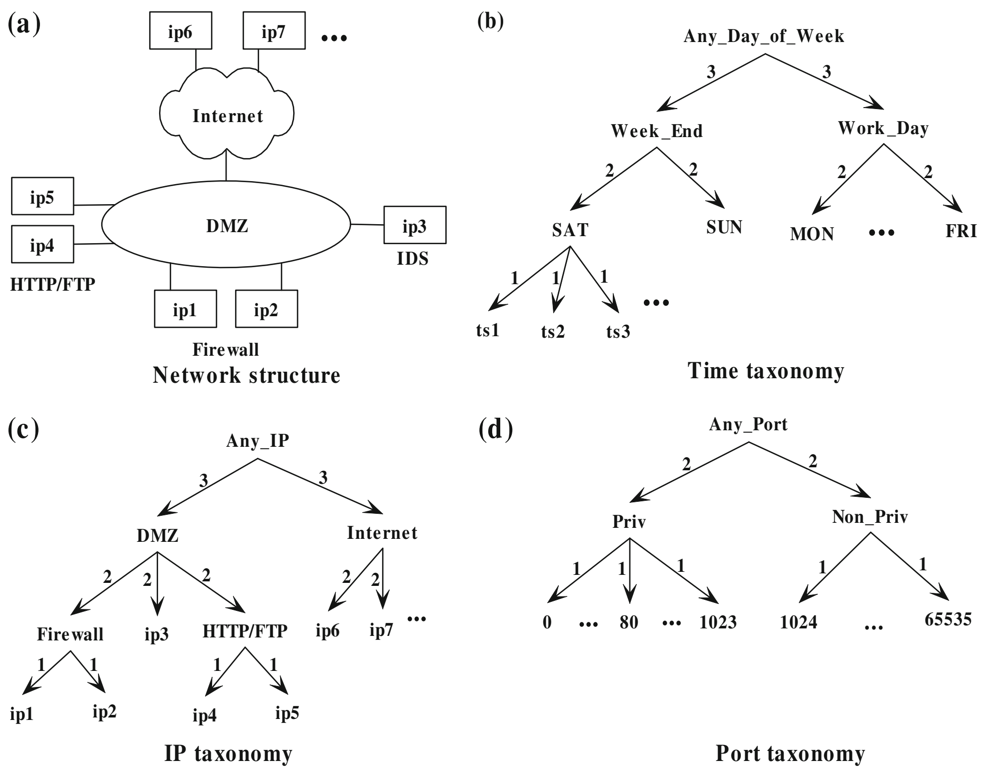

Fig. 3 Network structure and sample generalization hierarchies for IP address, port, and time features

4.1 Generalization hierarchies

The availability of certain background knowledge, such as conceptual hierarchies (hierarchies for short), can 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 [15]. A hierarchy defines a sequence of mappings from a set of concepts to their higher-level correspondences [16].

Hierarchies represent necessary background knowledge which directs the generalization process [15]. Knowledge about hierarchies can be directly provided by domain experts. Hierarchy information may also be implicitly stored in the database. Alternatively, a hierarchy can be constructed automatically based on clustering and database statistics [16]. Our algorithm assumes that meaningful hierarchies have been defined for all alarm's features (features for short). Some hierarchies can be seen in Fig. 3 [2].

We have adopted edge numbering (as can be seen in Fig. 3b–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. 3 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 *i*'th 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. 3c. Same thing is assumed with port feature, where values 20, 21, 80 are generalized to *PRIV* as can be seen in Fig. 3d. General(f_i) is the set of generalized values of f_i. As a result, a BigDom(f_i) is Dom(f_i) ∪ General(f_i) where Dom(f_i) ∩ General(f_i) = ∅. 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. 3b–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 g and T be the table of alarms.

Definition 1 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 as $\alpha \rightarrow \beta$. 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 1, as shown in Fig. 3c, 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. 3 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(α, β), 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. 3. As a result, the distance is computed using Eq. (1). Here, these two measures will be used in order to compare Julisch's measure, M_A, with our measure, M_B. The using of measure M_B is the first difference with Julisch's work.

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

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 $L(NCA(.,.))$ is, the lesser the distance. This will make the generalization to be more accurate. For demonstration, see Fig. 3c, ip_5 is nearer to ip_1 than *Internet*. In other words, $\mathrm{dist}(ip_5, ip_1) <$ dist (ip_5,*Internet*). The measure M_B distinguishes between $\mathrm{dist}(ip_5, ip_1) = 6$ and $\mathrm{dist}(ip_5, Internet) = 9$, while M_A measure does not distinguish this, e.g., $\mathrm{dist}(ip_5, ip_1) = \mathrm{dist}(ip_5, Internet) = 4$. 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 $\sum \mathrm{dist}(x[f_i], y[f_i])$ $\forall\, i$: $1 \leq i \leq n$.

4.3 Generalized alarms generation

The alarm clustering problem is NP-Complete as proved by Julisch [17]. The first work that used root causes to filter IDS alarms is Julisch's work, so we have compared our algorithm with his algorithm. In addition, we believe that there is a room for enhancement in his work. 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 Sect. 4.2. Second, it uses a different stop condition. Third, any generalization does not happen unless there is need. And finally, it 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 (i.e., the stop condition, this is the second difference with Julisch's work). Hereafter, the identical generalized alarms would be merged. More formally, our system works in the following steps:

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

For more detail, Fig. 4 shows the *Generate_Generalized_Alarms* procedure. Clusters centers are stored in G, line 1 sets this tables to empty; moreover, they are considered as the generalized alarms. An important feature of this procedure is that it passes over T for once which makes it very fast as shown in line 2. Line 3 selects the nearest cluster's center (G_k) to the new alarm A_i. If the distance, between A_i and G_k, is less than or equal to *NOD* (line 4), then A_i would be inserted in this cluster (line 5) and the $NCA(A_i, G_k)$ will be computed between the identical features (line 6). Figure 5 shows how the generalization process occurs. Otherwise, the new alarm is considered as a new cluster's center and is appended to G (line 8). The identical generalized alarms are merged, in line 9 to 10, into a single generalized alarm whose count value is equal to sum of individual counts. Line 11 returns all

```
Procedure Generate_Generalized_Alarms
Input: An alarm clustering problem ($T$, $NOD$, $\tau_1,\ldots,\tau_n$)
Output: Set of generalized alarms
Method
 1: Set $G$ to Empty ;
 2: for each alarm $A_i \in T$ do {
 3:   Select the nearest cluster $G_k$ to $A_i$ from $G$;
 4:   if $Dist(A_i, G_k) \le NOD$ then {
 5:     Set $A_i$ belong to $G_k$ ;
 6:     Keep $NCA(A_i[f_j], G_k[f_j])$ $\forall f_j$ as new features values for $G_k$;
 7:   }else
 8:     Consider $A_i$ as a new cluster's center and save it in $G$;}
 9: for each identical alarms g, $\acute{g} \in G$ do
10:   g[count] := g[count]+ $\acute{g}$[count] and delete $\acute{g}$ from $G$ ;
11:Return $\forall$g $\in G$;
```

Fig. 4 Pseudo code of the proposed algorithm

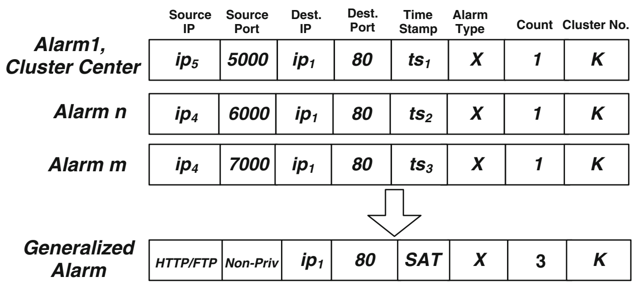

Fig. 5 An example of the generalization process

generalized alarms in G, where the count refers to the size of the cluster.

Preposition 1 *The time complexity of the proposed algorithm is $O(n{*}k)$, where n is the number of the alarms and k is the number of clusters in these alarms.*

Proof 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 $\le NOD$ value. As shown in Fig. 4, 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 3. The other steps of the algorithm, line 4 to line 8, take a constant time. Steps 9 and 10 merge duplicated clusters taking a quadratic time with respect to k; this time can be ignored if 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 *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 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 *NCA* as shown in Fig. 5.

Our algorithm is a variation of attribute-oriented induction (AOI) algorithm. AOI algorithm is a set-oriented database mining method which generalizes the task-relevant subset of data attribute-by-attribute, compresses it into a generalized relation, and extracts from it the general features of data [15]. Our algorithm makes use of the generalization concept of AOI algorithm and the neighboring concept.

The generalization used by our method is different from the one used in AOI algorithm [15]. AOI algorithm generalizes one concept to its parent in the hierarchy whereas our generalization is done by finding the NCA 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 new generalization technique is the third difference with Julisch's work.

In this paper, the generalization is performed by finding *NCA*(.,.) 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 $\le NOD$ holds, then *NCA*(.,.) between identical features of the new alarm and the generalized alarm of the matching cluster will be computed. Put 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 *NCA*(.,.) 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 NOD setting

Most of the clustering algorithms have control parameters [18]. 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. The using of *NOD* parameter is the fourth difference with Julisch's work. In this section, we will present a heuristic to find the best value for the *NOD* parameter.

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 [6]. For any dataset, the best partitioning will maximize (or minimize) the validity index. More formally, for a given dataset, we:

- Perform n clustering runs $\mathcal{R}_{c,i}$, where i ∈ [1, n]. Each run is with different NOD_i value.
- Compute $IDX_{\mathcal{R}c,i}$ validity index value for each $\mathcal{R}_{c,i}$.
- Choose the NOD_i value that is associated with the maximum (or minimum) $IDX_{\mathcal{R}c,i}$ 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 to evaluate the results of a clustering algorithm. Three approaches to investigate cluster validity are exist [19]: 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 [20]. 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 [21]: 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 [22–24,19,25]. There are many validity indices for the relative criteria, among them are: the Silhouette [26], the Dunn [27], the Davies-Bouldin [28], and the SD [20] indices. We have selected SD index because its time complexity is $O(n)$ [22]. In the sequel of this section, 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 [22] is defined by Eq. (2):

$$\mathrm{Scat}(c) = \frac{1}{c}\sum_{i=1}^{c}\frac{\|\sigma(v_i)\|}{\|\sigma(X)\|} \tag{2}$$

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 scattering (separation) between clusters is given by Eq. (3) [22].

$$\mathrm{Dis}(c) = \frac{D_{\max}}{D_{\min}}\sum_{k=1}^{c}\left(\sum_{z=1}^{c}\|v_k - v_z\|\right)^{-1} \tag{3}$$

where $D_{\max} = \max(\|v_i - v_j\|)$ ∀ i, j ∈ {1, 2, 3,…, c} is the maximum distance between cluster centers. The $D_{\min} = \min(\|v_i - v_j\|)$ ∀ i, j ∈{1, 2,…,c} is the minimum distance between cluster centers. Now, a validity index SD can be computed using Eq. (4) [22].

$$SD(c) = \alpha\ \mathrm{Scat}(c) + \mathrm{Dis}(c) \tag{4}$$

where α is a weighting factor equal to $\mathrm{Dis}(c_{\max})$ where $c_{\max}$ is the maximum number of clusters. The first term (Scat(c) that is defined by Eq. (2)) 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 [20]. The *NOD* parameter value that minimizes the SD index can be considered as the best value.

5 Experiments

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. Two different datasets were used in our experiments: a real dataset and DARPA 1999 dataset [29]. For both of these datasets we ran Snort [30], an open-source signature-based IDS. The resulting alarms have been labeled manually by us and using attack truth tables, for real dataset and DARPA 1999 dataset, respectively.

5.1 Results on a live network

This section presents experiments which we have performed by clustering a log of historical alarms. One of the objectives from this experiment is to see the interpretability of resulted clusters. 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 subsequent filtering stage ($i + 1$). The alarms log was taken from Snort IDS [30], during a period of two months containing 302,473 alarms; 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. 3a. The dataset used in this experiment was collected during the daily operation of an educational network. 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. 3.

The resulting generalized alarms (from the first month) of the six largest alarm clusters are shown in Table 1; 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 the machines as in Fig. 3a.

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 1 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

Table 1 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	25,511
http_inspect: BARE BYTE UNICODE ENCODING	Inside	Non-priv	ip_2	80	Any	9,021
ICMP redirect host	ip_1	Undefined	*DMZ*	Undefined	Work_day	7,291
Telnet Access	Internet	Non-priv	ip_{16}	23	Work_day	2,867

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 1. 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.

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 *NOD* parameter. The dataset used in this set of experiments is DARPA 1999 dataset. We used this dataset to compare our algorithm with others.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.

Figure 6 shows the performance of our system for all clustering and filtering runs for DARPA 1999 dataset. It should be noted that 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 week are converted to rules in order to filter out the false alarms in the subsequent week. This can be noted from the third and fourth columns in Fig. 6.

The reduction ratio with this experiment was about 70% as can be seen in Fig. 7 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.

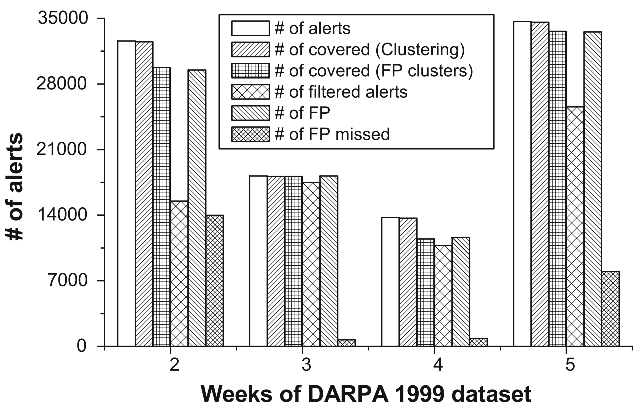

Fig. 6 Clustering and filtering runs for the DARPA 1999 dataset

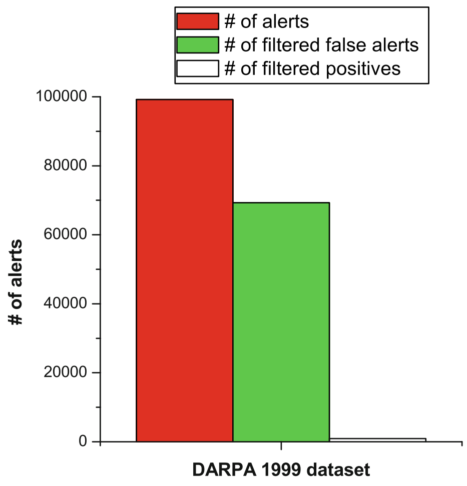

Fig. 7 Total alarms reduction and the missed true positives for the DARPA 1999 dataset

Those true alarms belong to the following attacks: training attacks #20 (ip sweep, week2, 761 alarms), test attacks #41 (port sweep, week4, 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, 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 [31]. However, intrusion detection is a multistage 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 [32].

We will try to analysis the clustering results by using the measures suggested by Pietraszek [32]. These measures are as follows: First, average fraction of an incident covered by clusters. The average incident coverage shows whether and how completely incidents are covered by clusters. Ideally, we want this value to be either close to 0 or close to 1. In the first case, the incidents are not covered at all (means that only false alarms are clustered). In the second case, the incident is covered completely (allowing for the derivation of rules for true alarms). Second, total fraction of all incidents covered. Finally, the average number of clusters covering an incident, the average number of clusters per incident shows how specific the clusters are. This value should be close to 1 as this

Table 2 The application of three measures on DARPA 1999 dataset with different *NOD* parameter values (Clustering stage)

NOD value	# of resulted clusters	Average incident coverage	Fraction of all incident events covered	Average # of clusters covering an incident
4	4,278	0.69	0.96	2.21
6	3,352	0.7	0.97	2.13
8	2,539	0.7	0.97	2.11
10	1,807	0.71	0.99	2.11
12	1,663	0.71	0.99	2.1
14	1,383	0.71	0.99	2.08
16	1,027	0.73	0.99	2.06
18	**878**	**0.73**	**0.99**	**2.06**
20	701	0.75	0.99	2.05
22	448	0.81	0.99	2.09

is the case, in which clusters accurately model the intrusions [32]. The values of these three measures are presented in Table 2 with different *NOD* values.

Based on the clustering results appeared in Table 2, we do note (from the third column) that most of the incident events are covered in the clustering stage. In addition, the value of the fraction of all incident events covered measure (i.e., the fourth column) is higher than the average incident coverage measure, which means that the small incidents is not covered. When we labelled the alarms manually, we noted that sometimes Snort had triggered more than one alarm for the same incident. This was because all Snort rules were used in the alarms generation. Therefore, the value of the average number of clusters covering an incident measure (the last column) did not approach to one.

The influence of the *NOD* parameter can be seen in Table 2. The small value for it causes the spread of the root causes over the clusters; in addition, the number of clusters would be increased. The *NOD* parameter value must be adjusted carefully. To do that, we have used the SD validity index which appeared in Eq. (4), the results can be shown in Fig. 8. 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. This value can also be seen in Table 2 as the best (or near the best) value. We conclude from this figure that the SD index is capable of locating the best value for *NOD* parameter.

We have seen that the complexity of our algorithm is $O(n*k)$. The processing time per alarm (averaged per 5,000 alarms) can be seen in Fig. 9. In addition, the impact of *NOD* parameter on the execution time can be seen in Fig. 10. In this figure, we do note that when the value of *NOD* parameter approaches to zero, the execution time will increase and also the number of resulted clusters will increase.

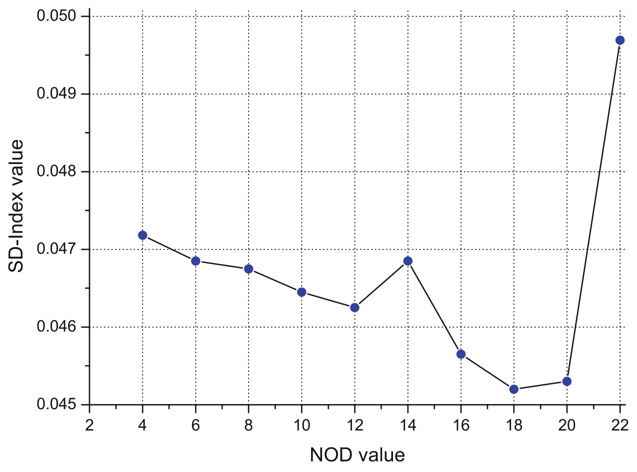

Fig. 8 The application of SD validity index (Eq. 4) on the DARPA 1999 dataset with different *NOD* values suggests that the best value for the *NOD* parameter is 18

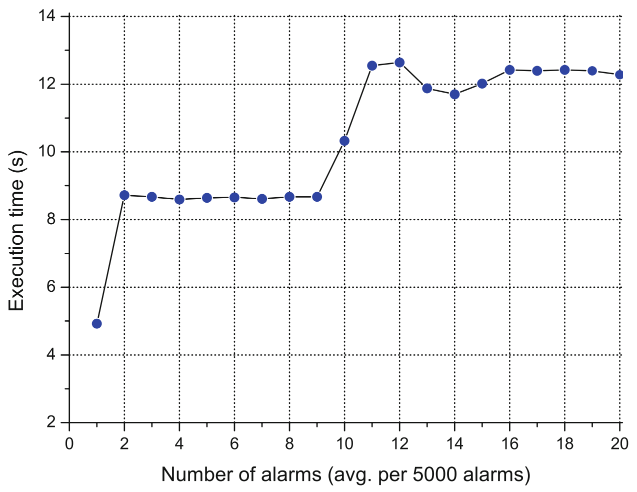

Fig. 9 The performance of the proposed algorithm

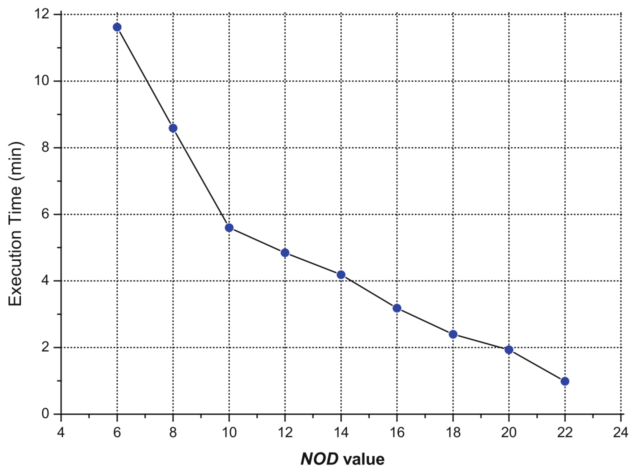

Fig. 10 The impact of the *NOD* parameter on the proposed algorithm performance

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. From the literature, Julisch [2] outlined 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.

Next, we examine how well the proposed algorithm meets the requirements listed in Sect. 3.2:

- *Scalability* The performance of our algorithm depends on the number of received alarms and the number of clusters in them as we have seen in *preposition 1*. 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 *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 [6]. 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. A heuristic has been presented in Sect. 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 *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 a 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 more accurate the measure. As have shown in Sect. 4.2, the used measure minimized the distances in such case while the semantics of the distances are retained. The more accurate the measure is, the lesser the generalization steps. Minimizing the generalization steps means that the results capture more detailed information about the alarms; which facilitate the interpretation of the results.

One interesting thing of the proposed algorithm is its stop condition. It passes over the table of alarms for once and tries to find the nearest cluster for any alarm. If an alarm is far from all clusters centers then a new cluster would be created, this process continues until all alarms are considered.

We have compared our method with five different methods having different techniques which are filtering [32,33], classification [32], aggregation [4], and prioritization [31]. The comparison measure is the Reduction Ratio (RR) which is defined by Valeur [34] as in Eq. (5). Some of the previous methods had used samples from DARPA 1999 dataset; however, we have used all the five weeks in DARPA 1999 dataset for comparison. Table 3 presents the comparison results. We can see that our method has the best reduction ratio. The nearest reduction ratio to ours is the work of Wang [33]; however, the size of training and test data are definitely too small to generate meaningful results. Our algorithm reduced the alarms load by about 70% making the security analyst focuses on the remaining 30% alarms.

$$\text{Reduction Ratio} = \frac{\#\text{input alarms} - \#\text{output alarms}}{\#\text{input alarms}} \tag{5}$$

Table 3 Comparison between our system and the other systems

	Julisch [32]	Perdisci [4]	Pietraszek [32]	Valeur [31]	Wang [33]	Our algorithm
# alarms	59,812	52,540	59,812	7,985	13	233,615
Duration	5 weeks	3 days	5 weeks	2 weeks	sample	5 weeks
Technique	Filtering	Aggregation	Classification	Prioritization	Filtering	Filtering
RR (%)	53	58.9	60	67.8	69.1	69.9

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.

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 [6]. 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 [6]. 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 different hierarchies [35]. Consequently, the usefulness of resulting clusters would depend on the quality and possibly granularity of the hierarchies.

7 Conclusions and future work

This paper has addressed the problem of IDSs overburdening the security analyst by generating thousand of alarms per day. A new data mining technique has been proposed to cluster alarms and to extract filters from these clusters. The extracted filters is related to problems in the network (may be configuration problems) and can be used to filter out many false positives. A new measure has been used which depends on background knowledge of the monitored network. The proposed technique is designed to pass over the table of alarms for once. The averaged reduction ratio which was obtained from different datasets was about 82% making the security analyst focuses on the remaining alarms.

Our future work has three main directions. First, applying of this method across different networks to average the results is worthy. Second, depending on vulnerabilities information of the protected network, the process of the extracting the good filters can be automated. Finally, we want to use the proposed technique to solve problems in another research field.

Acknowledgments The authors thank the anonymous reviewers for their valuable comments and suggestions in improving this paper. This research was supported by Project 973 (2007CB311101).

References

1. Manganaris, S., Christensen, M., Zerkle, D., Hermiz, K.: A data mining analysis of RTID alarms. J. Comput. Netw. **34**, 571–577 (2000)
2. Julisch, K.: Clustering intrusion detection alarms to support root cause analysis. ACM Trans. Inf. Syst. Secur. **6**, 443–471 (2003)
3. Yu, J., Reddy, Y.V.R., Selliah, S., Reddy, S., Bharadwaj, V., Kankanahalli, S.: TRINETR: an architecture for collaborative intrusion detection and knowledge-based alert evaluation. J. Adv. Eng. Inf. **19**, 93–101 (2005)
4. Perdisci, R., Giacinto, G., Roli, F.: Alarm clustering for intrusion detection systems in computer networks. J. Eng. Appl. Artif. Intell. **19**, 429–438 (2006)
5. Siraj, A., Vaughn, R.: 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)
6. Julisch, K.: Using root cause analysis to handle intrusion detection alarms, PhD dissertation, University of Dortmund (2003)
7. Al-Mamory, S.O., Zhang, H.: A survey on IDS alerts processing techniques. In: Proceeding of the 6th WSEAS International Conference on Information Security and Privacy (ISP '07), Spain, pp. 69–78 (2007)
8. Valdes, A., Skinner, K.: Probabilistic alert correlation. In: Proceeding of the Recent Advances in Intrusion Detection. LNCS, vol. 2212, pp. 54–68 (2001)
9. Dain, O.M., Cunningham, R.K.: Fusing a heterogeneous alert stream into scenarios. In: Proceeding of the 2001 ACM Workshop on Data Mining for Security Applications, pp. 231–235 (2001)
10. Ning, P., Cui, Y., Reeves, D.S., Xu, D.: Techniques and tools for analyzing intrusion alerts. ACM Trans. Inf. Syst. Secur. **7**, 274–318 (2004)
11. Morin, B., Me, L., Debar, H., Ducasse, M.: M2D2: A formal data model for IDS alert correlation. In: Proceeding of the International Symposium on Recent Advances in Intrusion Detection, pp. 115–137 (2002)
12. Pietraszek, T.: Using adaptive alert classification to reduce false positives in intrusion detection. In: Proceeding of the Recent advances in intrusion detection, France, pp. 102–124 (2004)
13. Hand, D., Mannila, H., Smyth, P.: Principles of Data Mining. The MIT Press, Cambridge (2001)
14. Bellovin, S.M.: Packets found on an Internet. J. Comput. Commun. Rev. **23**, 26–31 (1993)
15. Han, J., Fu, Y.: Exploration of the power of attribute-oriented induction in data mining. In: Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.) Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press, Cambridge, pp. 399–421 (1996)
16. Han, J., Cai, Y., Cercone, N.: Data-driven discovery of quantitative rules in relational databases. IEEE Trans. Knowl. Data Eng. **5**, 29–40 (1993)
17. Julisch, K.: Mining alarm clusters to improve alarm handling efficiency. In: Proceeding of the 17th Annual Computer Security Applications Conference, New Orleans, pp. 12–21 (2001)
18. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**, 264–323 (1999)
19. Theodoridis, S., Koutroubas, K.: Pattern Recognition. Academic Press, New York (1999)

20. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. J. Intell. Inf. Syst. **17**, 107–145 (2001)
21. Berry, M.J.A., Linoff, G.: Data Mining Techniques for Marketing, Sales and Customer Support. Wiley, New York (1996)
22. Halkidi, M., Vazirgiannis, M., Batistakis, I.: Quality scheme assessment in the clustering process. In: Proceeding of the 4th European Conference on Principles of Data Mining and Knowledge Discovery, pp. 265–276 (2000)
23. Rezaee, M.R., Lelieveldt, B.B.F., Reiber, J.H.C.: A new cluster validity index for the fuzzy c-mean. Pattern Recognit. Lett. **19**, 237–246 (1998)
24. Sharma, S.C.: Applied Multivariate Techniques. Wiley, New York (1996)
25. Xie, X.L., Beni, G.: A validity measure for fuzzy clustering. IEEE Trans. Pattern Anal. Mach. Intell. **13**(8), 841–847 (1991)
26. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**(1), 53–65 (1987)
27. Bezdek, J.C., Pal, N.R.: Some new indexes of cluster validity. IEEE Trans. Syst. Man Cybern. Part B **28**(3), 301–315 (1998)
28. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. **1**(2), 224–227 (1979)
29. MIT Lincoln Laboratory: 1999 DARPA intrusion detection evaluation data set (1999). http://www.ll.mit.edu/IST/ideval/data/1999/1999dataindex.html
30. Roesch, M.: Snort-lightweight intrusion detection for networks. In: Proceeding of the 1999 USENIX LISA Conference, pp. 229–238 (1999)
31. Valeur, F., Vigna, G., Kruegel, C., Kemmerer, R.A.: A comprehensive approach to intrusion detection alert correlation. In: IEEE Transactions on Dependable and Secure Computing **1**(3) (2004)
32. Pietraszek, T.: Alert classification to reduce false positives in intrusion detection. PhD dissertation, Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Germany, July 2006
33. Wang, J., Lee, I.: Measuring false-positive by automated real-time correlated hacking behavior analysis. In: Proceedings of the 4th International Conference on Information Security. LNCS, vol. 2200, pp. 512–535 (2001)
34. Valeur, F.: Real-time intrusion detection alert correlation. PhD dissertation, University of California, Santa Barbara, June 2006
35. Han, J., Cai, Y., Cercone, N.: Knowledge discovery in databases: an attribute-oriented approach. In: Proceeding of the 18th International Conference on Very Large Databases, pp. 547–559 (1992)