Mining Alarm Clusters to Improve Alarm Handling Efficiency

Klaus Julisch
IBM Research
Zurich Research Laboratory
kju@zurich.ibm.com

Abstract

It is a well-known problem that intrusion detection systems overload their human operators by triggering thousands of alarms per day. As a matter of fact, we have been asked by one of our service divisions to help them deal with this problem. This paper presents the results of our research, validated thanks to a large set of operational data. We show that alarms should be managed by identifying and resolving their root causes. Alarm clustering is introduced as a method that supports the discovery of root causes. The general alarm clustering problem is proved to be NP-complete, an approximation algorithm is proposed, and experiments are presented.

1. Introduction

In response to attacks against enterprise networks, administrators are increasingly deploying intrusion detection systems (IDSs). These systems monitor hosts, networks, or other resources using a variety of techniques [25, Chapter 2]. Unfortunately, the use of intrusion detection has given rise to another difficult problem, namely the handling of a generally large number of alarm messages (alarms, for short). In fact, it is not uncommon for an IDS to trigger thousands of alarms per day, up to 99% of which are false positives [17, 24]. Investigating alarms manually is not only error-prone but also a waste of human time and energy [3, 24]. This paper presents a novel semi-automatic approach for handling intrusion detection alarms efficiently.

The *root cause* of an alarm is the reason for which it occurs. We have made the key observation that a small number of root causes generally accounts for over 90% of all alarms. Moreover, these root causes are mostly configuration problems and do not disappear unless someone fixes them. Clearly, these 90% of alarms are highly redundant as their root causes are stable; even worse, these alarms distract the ID analyst from finding the real attacks.

Motivated by the above observations, we have investigated techniques to efficiently handle *large* groups of *redundant* alarms. The outcome of this research was a semi-automatic process that consists of two steps: Step one identifies root causes that account for *large* numbers of alarms. Step two removes these root causes and thereby dramatically reduces the future alarm load. Indeed, the experiment of Section 4 shows how the one-time effort of identifying and removing root causes pays off by reducing the future alarm load by 82%. This is significant as it enables the ID analyst to henceforth concentrate on the remaining 18% of alarms. As a general rule, root cause identification and removal should be done roughly once a month in order to keep up with changes in the computing environment.

This paper focuses on the first of the above two steps, namely the identification of root causes. To support the identification of root causes, we have developed a novel technique, called *alarm clustering*. The motivation for alarm clustering stems from the observation that the alarms of a given root cause are generally "similar" (Section 2.3 will formally define our notion of similarity). Alarm clustering reverses this implication and groups similar alarms together, assuming that these alarms also have the same root cause. For each alarm cluster, a single so-called *generalized alarm* is derived. Intuitively, a generalized alarm is a pattern that an alarm must match in order to belong to the respective cluster. We show that knowledge of generalized alarms vastly simplifies the identification of root causes.

By way of illustration, here is a short list of root causes that we have observed in our work:[1]

- A broken TCP/IP stack that generates fragmented traffic and thereby triggers "Fragmented IP" alarms.
- A firewall that has Network Address Translation (NAT) enabled funnels the traffic of many users and thereby occasionally seems to perform host scans. Indeed, a NAT-enabled firewall acts as proxy for many users. When these users *simultaneously* request external services, then the firewall will proxy these requests and the resulting SYN packets resemble SYN host sweeps.

[1] Related examples by different authors can be found in [15] and [27].

– A misconfigured secondary DNS server which does half-hourly DNS zone transfers from its primary DNS server. The resulting "DNS Zone Transfer" alarms are no surprise.

It is instructive to examine the first example of the broken TCP/IP stack more closely. Obviously, all the resulting "Fragmented IP" alarms have the same source address. Now, let us suppose that the broken TCP/IP stack belongs to a Web server. Then, the source ports are identical, too (namely 80). Furthermore, assuming that the Web server is used primarily on workdays, we will observe that the bulk of alarms occurs on workdays. In summary, this example shows that all alarms of a given root cause are "similar". Our alarm clustering algorithm groups these alarms together and summarizes them by a single generalized alarm such as "Your Web server triggers many 'Fragmented IP' alarms on workdays!".

Clearly, a generalized alarm like the above facilitates the identification of root causes, but human expertise is still needed. Therefore, alarm clustering only supports the discovery of root causes, but does not completely automate it. However, as we gain experience with alarm clustering, we observe that certain types of generalized alarms always occur for the same root causes. Therefore, one can build a knowledge-based system that automatically maps generalized alarms to their most probable root causes. That way, root cause analysis can become fully automatic, but the details of such a knowledge-based system are beyond the scope of this paper.

The novel contribution of this paper is fourfold: First, root cause identification and removal is introduced as a safe way to manage intrusion detection alarms efficiently. Second, we present and validate a novel clustering algorithm that groups similar alarms together and summarizes them by a single generalized alarm. Third, we show that it is usually quite straightforward to derive the root cause of a given generalized alarm. Finally, we show that removing root causes can dramatically reduce the future alarm load, thus enabling a much more thorough analysis of the remaining alarms.

1.1. Related Work

So far, the problem of alarm handling has received little attention in the intrusion detection community. In [24], S. Manganaris *et al.* mine association rules between adjacent alarms. These association rules are subsequently used to implement anomaly detection on the IDS alarm stream. Also related is the work by A. Valdes and K. Skinner [31]. Their approach uses similarity-based alarm correlation, both to suppress false positives and to improve sensitivity to true positives. However, their overall framework, formalism, and notion of similarity are very different from ours. Finally, our earlier work [17] uses alarm patterns to filter out false positives, but proposes neither a systematic approach nor tool support to discover the alarm patterns.

In the Spice/Spade project [29], S. Staniford *et al.* have designed a port-scan correlator that groups/correlates probes that belong to the same stealthy port-scans, but the design has yet to be validated. Moreover, the port-scan correlator has been optimized for detecting *stealthy scans*. By contrast, our focus is to find *predominant alarm patterns* so that we can identify and remove their root causes. That way, we decrease the future alarm load and enable a more thorough analysis of the remaining alarms.

Other related work comes from W. Lee *et al.*, who used data mining for feature selection and training of classifiers that detect intrusions in audit data [21]. The goal of that work was to construct intrusion detection systems more systematically. By contrast, we aspire to use existing IDSs more efficiently. In addition, very different data mining algorithms are employed in our work.

In the world of telecommunication networks, M. Klemettinen has used association rules and frequent episodes to discover alarm patterns, which were subsequently used in the development of alarm correlation systems [19]. Similar work in the field of network management has been done by J.L. Hellerstein *et al.* [14]. However, the methods used in this thrust of research do not capture the notion of alarm similarity. Furthermore, association rules and frequent episodes have the drawback of generating many uninteresting and redundant alarm patterns [19]. Our alarm clustering algorithm does not have these drawbacks.

Finally, it is worth noting that the idea of root cause identification has been pioneered by the network management community [18, 23, 33]. However, to the author's knowledge, the current paper is the first to apply root cause identification to intrusion detection.

1.2. Paper Overview

The remainder of this paper is organized as follows: Section 2 introduces our notion of similarity, defines alarm clusters, and proves the general alarm clustering problem to be NP-complete. Section 3 presents a novel algorithm that offers an approximation solution to the alarm clustering problem. This makes alarm clustering a practical technique. Section 4 presents our experience with alarm clustering and Section 5 offers conclusions and areas of future work.

2. A Framework for Similarity and Clustering

This section presents a framework for similarity and clustering. We call it a framework because our concepts are very flexible, site-independent, and IDS-independent.

2.1. Introduction and Background

This section introduces clustering in general and discusses the problems that are intrinsic to alarm clustering in particular. In general, clustering seeks to group objects so that the objects within a given group/cluster are similar, whereas the objects of different groups/clusters are dissimilar [13, 16]. Obviously, the notion of similarity is key to this definition. In practice, similarity is easy to define for numerical attributes, but categorical, time, or string attributes pose serious problems [7, 8, 10, 13]. Unfortunately, alarm messages can contain all of these attribute types:

Categorical Attributes: The domain of categorical attributes is discrete and unordered [13]. Examples of categorical attributes include IP addresses, port numbers, and header flags.

Numerical Attributes: Examples of numerical attributes include counters (e.g. for the number of SYN packets in a SYN flooding alarm), and size fields (e.g. for the packet size in "Large ICMP Traffic" alarms [4]).

Time Attributes: All alarms must be time-stamped. Please note that time should not be regarded as a numerical attribute. Indeed, doing so would mean to ignore its unique semantics (including the notions of periodicity, workdays versus weekends, etc.).

String Attributes: String attributes assume arbitrary and unforeseeable text values. To begin with a negative example, attributes such as "Attack Name" or "Recommended Action" are *not* string attributes because their domains are small and fixed. However, NetRanger's context field [5] is a typical string attribute. The context field is optional, but, when present, it stores the supposedly malicious network packet. For example, the context field frequently contains supposedly malicious URLs or command sequences that seem to constitute an FTP or Telnet exploit. Evidently, the difficulty with string attributes is to extract the information that they carry.

In fact, the mere presence of categorical attributes makes alarm clustering a difficult problem [2, 8, 9, 10]. The presence of time and string attributes adds further complexity. This section introduces a similarity concept that embraces all of the above attribute types. Furthermore, the clustering problem is restated in the light of this similarity concept.

In a nutshell, to define the similarity between alarms, we use taxonomies on the alarm attributes. Examples of alarm attributes include the source and destination IP address. A taxonomy is a generalization hierarchy on these attributes. For example, a taxonomy might state that IP address $10.3.3.3$ is a Web server, is a machine in the DMZ, is a .... Intuitively, two alarms are all the more similar the closer their attributes are related by way of these taxonomies. We will formalize this notion and define alarm clusters as alarm sets that maximize intra-cluster alarm similarity, while having a user-defined minimum size. In the output, we require that each alarm cluster be summarized by a single "generalized" alarm. We consider this an important feature of our approach because it is not practical to communicate alarm clusters by means of their constituent alarms.

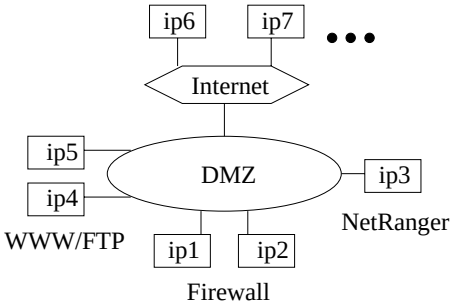
2.2. Notation

At present, different IDSs report their alarms in different formats. In anticipation of a unifying standard [6], a similar, yet simplified alarm format will be used. Specifically, alarms are modeled as tuples over a multi-dimensional space. The dimensions are called *alarm attributes* or *attributes* for short. Examples of alarm attributes include the source and destination IP address, the source and destination port, the alarm type (which encodes the observed attack), and the timestamp (which also includes the date).

Formally, alarms are defined as tuples over the Cartesian product $\mathsf{X}_{1\leq i\leq n}\text{dom}(A_i)$, where $\{A_1, \ldots, A_n\}$ is the set of attributes and $\text{dom}(A_i)$ is the *domain* (i.e. the range of possible values) of attribute A_i. Furthermore, for an alarm a and an attribute A_i, the projection $a[A_i]$ is defined in the usual way, namely as the A_i value of alarm a. Next, an *alarm log* is modeled as a set of alarms. This model is correct if the alarms of alarm logs are pairwise distinct – an assumption that we make to keep our notation simple. (Note that unique alarm-IDs can be used to make all alarms pairwise distinct.)

Let A_i be an alarm attribute. A tree $\mathcal{T}_i$ on the elements of $\text{dom}(A_i)$ is called a *taxonomy* (or a *generalization hierarchy*). For two elements $x, \hat{x} \in \text{dom}(A_i)$, we call $\hat{x}$ a *parent* of x, and x a *child* of $\hat{x}$ if there is an edge $\hat{x} \longrightarrow x$ in $\mathcal{T}_i$. Furthermore, we call $\hat{x}$ a *generalization* of x if the taxonomy $\mathcal{T}_i$ contains a path from $\hat{x}$ to x (in symbols: $x \trianglelefteq \hat{x}$). The length of this path is called the *distance* $\delta(x, \hat{x})$ between x and $\hat{x}$. Please note that $\delta(x, \hat{x})$ is undefined if $x \trianglelefteq \hat{x}$ is not satisfied. Finally, $x \trianglelefteq \hat{x}$ is trivially satisfied for $x = \hat{x}$, and $\delta(x, \hat{x})$ equals 0 in this case.

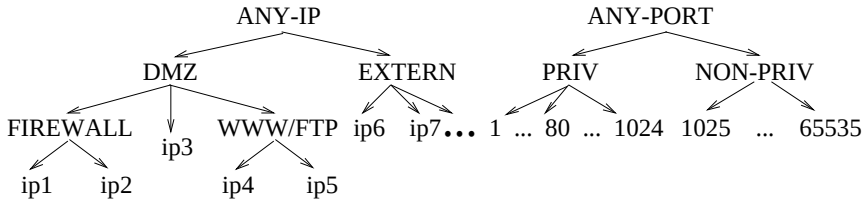
By way of illustration, Figure 1 shows a network topology and the taxonomies one might want to use for IP addresses and port numbers in this environment. Please note that the domain of IP addresses is the union of "elementary" IP addresses (i.e. the set $\{p.q.r.s \mid p, q, r, s \in \{0, \ldots, 255\}\}$) and "generalized" IP addresses (i.e. the set $\{$*FIREWALL*, *WWW/FTP*, *DMZ*, *EXTERN*, *ANY-IP*$\}$). Analogously, the domain of port numbers is $\{1, \ldots, 65535, \textit{PRIV}, \textit{NON-PRIV}, \textit{ANY-PORT}\}$. Next, according to Figure 1c, the IP address *ip1* is a firewall, is a DMZ machine, is any IP address. More succinctly,



a) Network

Target-IP	Target-Port
ip1	80
ip2	1234
ip2	23
ip4	8080
ip4	80
ip4	21
ip5	80

b) Sample alarm log



c) Taxonomies for IP addresses and port numbers

Figure 1. Network, alarm log, and taxonomies of our running example.

this relationship can be expressed as $ip1 \trianglelefteq$ *FIREWALL* $\trianglelefteq$ *DMZ* $\trianglelefteq$ *ANY-IP*. Furthermore, we have $\delta(ip1, \text{ANY-IP}) = 1 + \delta(\text{FIREWALL}, \text{ANY-IP}) = 1 + 1 + \delta(\text{DMZ}, \text{ANY-IP}) = 1+1+1+\delta(\text{ANY-IP}, \text{ANY-IP}) = 1+1+1+0 = 3$. Finally, $\delta(ip1, ip2)$ is not defined because $ip1 \trianglelefteq ip2$ is false.

Next, we extend our notation from attributes to alarms. To this end, let $a, \hat{a} \in \mathsf{X}_{1 \le i \le n} \text{dom}(A_i)$ denote two alarms. The alarm $\hat{a}$ is called a *generalization* of alarm a iff $a[A_i] \trianglelefteq \hat{a}[A_i]$ holds for all attributes A_i. In this case, we write $a \trianglelefteq \hat{a}$. Furthermore, if $a \trianglelefteq \hat{a}$ holds, then the *distance* $\delta(a, \hat{a})$ between the alarms a and $\hat{a}$ is defined as $\delta(a, \hat{a}) := \sum_{i=1}^{n} \delta(a[A_i], \hat{a}[A_i])$. If $a \trianglelefteq \hat{a}$ is not satisfied, then $\delta(a, \hat{a})$ is undefined. Finally, in the case of $a \trianglelefteq \hat{a}$, we say that a is more *specific* than $\hat{a}$, and $\hat{a}$ is more *abstract* than a.

As a convention in this paper, we use the symbols $A_1, \ldots, A_n$ to stand for alarm attributes. Furthermore, the symbols $\mathcal{T}_1, \ldots, \mathcal{T}_n$ are reserved for taxonomies on the respective attributes. Finally, the symbol $\mathcal{L}$ will be used to denote an alarm log.

2.3. Similarity and Alarm Clusters

We begin by defining similarity. To this end, let $\mathcal{S} \subseteq \mathcal{L}$ denote a set of alarms. The *cover* of $\mathcal{S}$ is the most specific alarm $c \in \mathsf{X}_{1 \le i \le n} \text{dom}(A_i)$ to which all alarms a in $\mathcal{S}$ can be generalized. Thus, the cover c satisfies $\forall a \in \mathcal{S} : a \trianglelefteq c$, and there is no more specific alarm c' ($c' \trianglelefteq c$) that would also have this property. We denote the cover of $\mathcal{S}$ by $\text{cover}(\mathcal{S})$. For example, in Figure 1, we have $\text{cover}(\{(ip1, 80), (ip4, 21)\}) = (\text{DMZ}, \text{PRIV})$. Finally, the *dissipation* of $\mathcal{S}$ is defined as

$$\Delta(\mathcal{S}) := {}^{1}/_{|\mathcal{S}|} \sum_{a \in \mathcal{S}} \delta(a, \text{cover}(\mathcal{S})). \quad (1)$$

Let us verify that $\Delta(\{(ip1, 80), (ip4, 21)\}) = {}^{1}/_{2} \times (3 + 3) = 3$. Intuitively, the dissipation measures the average distance between the alarms of $\mathcal{S}$ and their cover. The important thing to note here is that the alarms in $\mathcal{S}$ are all the more similar, the smaller the value of $\Delta(\mathcal{S})$. Therefore, we will strive to *minimize* dissipation in order to *maximize* intra-cluster alarm similarity.

Next, we define the alarm clustering problem. To this end, let $\mathcal{L}$ be an alarm log, $min\_size \in \mathbb{N}$ an integer, and $\mathcal{T}_i$, $i = 1, \ldots, n$, a taxonomy for each attribute A_i. We define:

Definition 1 (Alarm Clustering Problem) *Let* $(\mathcal{L}, min\_size, \mathcal{T}_1, \ldots, \mathcal{T}_n)$ *be an* $(n+2)$*-tuple with symbols as defined above. The* alarm clustering problem *is to find a set* $\mathcal{C} \subseteq \mathcal{L}$ *that* minimizes *the dissipation* $\Delta(\mathcal{C})$*, subject to the constraint that* $|\mathcal{C}| \ge min\_size$ *holds. We call* $\mathcal{C}$ *an* alarm cluster *or* cluster *for short.*

In other words, among all sets $\mathcal{C} \subseteq \mathcal{L}$ that satisfy $|\mathcal{C}| \ge min\_size$, a set with minimum dissipation has to be found. If there are multiple such sets, then any one of them can be picked. Note that once the cluster $\mathcal{C}$ has been found, the remaining alarms in $\mathcal{L} \setminus \mathcal{C}$ can be mined for additional

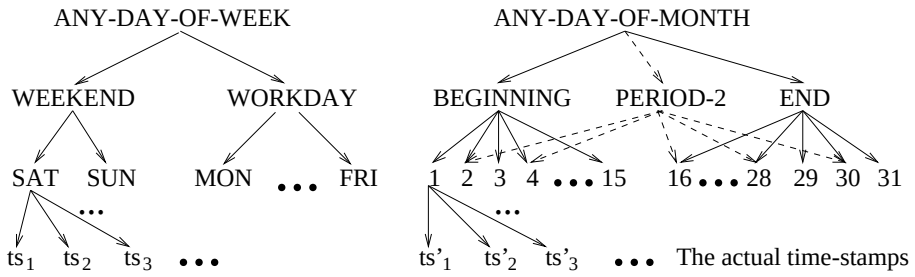


Figure 2. Sample taxonomies for time attributes.

clusters. One might consider to use a different min_size value for $\mathcal{L} \setminus \mathcal{C}$, an option that is very useful in practice.

Imposing a minimum size on alarm clusters has two advantages. First, it decreases the risk of clustering small sets of unrelated but coincidentally similar alarms. Second, large clusters are of particular interest because identifying and resolving their root causes has a high payoff. Finally, the decision to maximize similarity as soon as the minimum size has been exceeded minimizes the risk of including unrelated alarms in a cluster.

Clearly, stealthy attacks that trigger fewer than min_size alarms do not yield any clusters, but that is not the point. Indeed, let us recall the practical goal that we are pursuing, namely to identify predominant root causes that account for *large* amounts of alarms. By removing these root causes, we can significantly reduce the number of newly generated alarms. This reduction is exactly our goal as screening the reduced alarm stream for attacks is much more efficient. Note that even though important, this screening step is beyond the scope of this paper.

Given the need for a practical alarm clustering algorithm, the following result is relevant:

Theorem 1 *The alarm clustering problem* $(\mathcal{L}, min\_size, \mathcal{T}_1, \ldots, \mathcal{T}_n)$ *is NP-complete.*

Proof : The proof is obtained by reducing the CLIQUE problem [26] to the alarm clustering problem. For details, please refer to Appendix A. □

In Section 3, we will describe an approximation algorithm for the alarm clustering problem. For now, let us assume that we are able to discover alarm clusters. Then, the question arises how alarm clusters are best presented to the user. As will be shown in Section 4.2, alarm clusters can easily comprise thousands of alarms. Therefore, it is not viable to represent clusters by means of their constituent alarms. Indeed, doing so would mean to overwhelm the user with a vast amount of information that is hard to make sense of. To solve this problem, we represent clusters by their covers. Note that covers correspond to what we have informally called "generalized alarms".

Representing clusters by their covers works well as long as the covers are reasonably specific. Indeed, abstract covers such as $(DMZ, ANY\text{-}PORT)$ (cf. Figure 1) reveal very little about the alarms that the cover was derived from. Specific covers such as $(FIREWALL, 80)$ are preferable (even if they don't explicitly reveal the root cause either). Fortunately, by clustering alarms of maximum similarity, we favor cluster that have specific covers. This relationship, which is established by equation (1), is another argument for clustering alarms of maximum similarity (and minimum dissipation).

In order to obtain generalized alarms that are meaningful and indicative of their root causes, we clearly need to consider more attribute types than just the ones in the above examples. In particular, string and time attributes can contain valuable information, and the following subsection shows how to include these attribute types in our framework.

2.4. Taxonomies for Time and String Attributes

The time and string attributes of ID alarms can be tied into the above notion of similarity. For brevity, this section will rely on examples, but the generalizations are obvious.

Let us consider time attributes first. Typically, one wishes to capture temporal information such as the distinction between weekends and workdays, between business hours and off hours, or between the beginning of the month and the end of the month. To make the clustering algorithm aware of concepts like these, one can use taxonomies such as the ones in Figure 2 (please ignore the dashed arrows for the moment). For example, the left-hand taxonomy shows that the time-stamp ts_1 can be generalized to the concepts *SAT*, *WEEKEND*, and ultimately, *ANY-DAY-OF-WEEK*.

A limitation of the current framework is that taxonomies have to be trees, whereas directed acyclic graphs (DAGs) are more flexible. Indeed, it might be desirable to use the two taxonomies in Figure 2 simultaneously. However, combining the taxonomies into a single one is not possible because each time-stamp would have to have two parents – one that corresponds to its day of the week and one that

Input: An alarm clustering problem $(\mathcal{L}, min\_size, \mathcal{T}_1, \ldots, \mathcal{T}_n)$
Output: An approximation solution for $(\mathcal{L}, min\_size, \mathcal{T}_1, \ldots, \mathcal{T}_n)$
Algorithm:

```
 1: G := L;                          // Make a copy of L.
 2: loop forever {
 3:     for each alarm a ∈ G do {
 4:         c := number of alarms x ∈ L with x ⊴ a;
 5:         if c ≥ min_size then terminate and return alarm a;
 6:     }
 7:     use heuristics to select an attribute A_i, i ∈ {1, ..., n};
 8:     for each alarm a ∈ G do     // Generalize a[A_i]
 9:         a[A_i] := parent(a[A_i], T_i);
10: }
```

Figure 3. Alarm clustering algorithm.

corresponds to its day of the month. This violates the tree property of taxonomies.

To reach a solution, we replicate the time attribute and assign each taxonomy tree to a separate replica. That way, one replica plays the role "Day of the Week", whereas the other plays the role "Day of the Month". Furthermore, each replica is generalized according to its own taxonomy. Similarly, suppose that we want to include the concept of periodicity into the right-hand taxonomy of Figure 2. Again, the most obvious approach (indicated by dashed arrows) is not possible as it destroys the tree property of the taxonomy. Fortunately, the replication trick can be applied here, too.

Next, we consider string attributes. String attributes can assume text values with completely unforeseeable contents. Therefore, the challenge lies in tapping the semantic information of the strings. We solve this problem by means of a feature extraction step that precedes the actual alarm clustering. *Features* are crisp bits of semantic information that, once extracted, replace the original strings. Thus, each string is replaced by the set of its features. Note that subset-inclusion defines a natural taxonomy on feature sets. For example, the feature set $\{f_1, f_2, f_3\}$ can be generalized to the sets $\{f_1, f_2\}$, $\{f_1, f_3\}$, or $\{f_2, f_3\}$, which in turn can be generalized to $\{f_1\}$, $\{f_2\}$, or $\{f_3\}$. The next level is the empty set, which corresponds to "ANY-FEATURE". Evidently, the resulting taxonomy is a DAG, not a tree. To transform this DAG into a tree, we use the above replication trick combined with the pruning of edges.

We consider it the IDS analyst's responsibility to select features that capture as much semantic information as possible. Fortunately, there are well-established techniques to support feature selection [20, 22, 32]. As will be shown in Section 4.1, our current implementation uses frequent substrings as features.

3. Algorithm for Alarm Clustering

Given the NP completeness of alarm clustering, we have developed an approximation algorithm. An *approximation algorithm* for the problem $(\mathcal{L}, min\_size, \mathcal{T}_1, \ldots, \mathcal{T}_n)$ finds a cluster $\mathcal{C} \subseteq \mathcal{L}$ that satisfies $|\mathcal{C}| \geq min\_size$, but does not necessarily minimize $\Delta(\mathcal{C})$. Obviously, the further an approximation algorithm pushes $\Delta(\mathcal{C})$ to its minimum, the better it is.

The approximation algorithm we have developed is a variant of attribute-oriented induction (AOI) [11, 12], a well-established technique in the field of data mining. Our modification over the classical AOI is twofold: First, we generalize attributes more conservatively than classical AOI. Second, we use a different termination criterion, which is reminiscent of density-based clustering [1, 13].

To begin with, our approximation algorithm *directly* constructs the (single) generalized alarm that constitutes the algorithm's output. In other words, the algorithm does *not* make the detour over first finding an alarm cluster and then deriving its cover. The algorithm starts with the alarm log $\mathcal{L}$ and repeatedly generalizes the alarms in $\mathcal{L}$. Generalizing the alarms in $\mathcal{L}$ is done by choosing an attribute A_i and replacing the A_i values of all alarms by their parents in $\mathcal{T}_i$. This process continues until an alarm has been found to which at least min_size of the original alarms can be generalized. This alarm constitutes the output of the algorithm. Figure 3 shows the resulting algorithm.

In more detail, line 1 makes a copy of the initial alarm log $\mathcal{L}$. This is necessary as the initial alarm log is needed in line 4. In line 5, the algorithm terminates when a generalized alarm a has been found to which at least min_size alarms $x \in \mathcal{L}$ can be generalized. If the algorithm did not terminate, then the generalization step (lines 8 and 9) is executed. Here, selecting an attribute A_i is guided by the following heuristic:

For each attribute A_i, let $f_i \in \mathbb{N}$ be maximum with the property that there is an alarm $a^* \in \mathcal{G}$ such that $x[A_i] \trianglelefteq a^*[A_i]$ holds for f_i of the original alarms $x \in \mathcal{L}$. If f_i is smaller than min_size, then we know that we will not find a solution without generalizing A_i and, therefore, select A_i for generalization. Note that this will not eliminate the optimal solution from the search space. If, on the other hand, $f_i \geq min\_size$ holds for all attributes, then we select the attribute A_i with the smallest f_i value. Clearly, this choice might not be optimal!

We make no formal claim about the above heuristic except that it works well in practice (cf. Section 4). Evidently, the heuristic does influence the resulting clusters and we plan to investigate this influence. Also, one could conceive a completely different approximation algorithm, for example one that is based on partitioning or hierarchical clustering [13, 16]. Our choice for the above algorithm is based mainly on its simplicity, scalability, and particularly good noise tolerance. However, we acknowledge that there is no such thing as a single best clustering algorithm [7, 10, 13].

4. Experience with Alarm Clustering

This section summarizes our experience with alarm clustering. To this end, we provide a detailed description of how we applied alarm clustering to solve a real-world alarm handling problem that we were facing.

4.1. Experiment Setup

This section presents a clustering experiment that we have conducted on a log of historical alarms. The alarm log is taken from a commercial IDS, spans a time period of one month, and contains 156380 alarm messages. The IDS sensor was deployed in a network that is isomorphic to the one in Figure 1a. Please note that this experiment involves *real* alarm data that was collected during the day-to-day operation of a commercially used network!

Also, we want to stress that we have applied the same algorithm to many more data sets of various environments. The results obtained with these different data sets are consistent with the results presented here. However, this section focuses on a single data set in order to have a detailed discussion of the results.

For the purpose of our experiment, we model alarms as 7-tuples. In detail, the individual alarm attributes are the source and destination IP address, the source and destination port, the alarm type, the timestamp, and the context field. Please recall that the context field is optional, but when present, contains the suspicious network packet.

For IP addresses and port numbers, we use the taxonomies in Figure 1c. For timestamps, we use the taxonomies in Figure 2, but without the dashed part. No taxonomy is defined for the alarm types. Finally, for the context field (a string attribute) we use frequent substrings as features. More precisely, let $\mathcal{V} := <a[Context] \mid a \in \mathcal{L}>$ denote the multi-set (or bag) of values that the context field assumes in the alarm log $\mathcal{L}$. Then, the Teiresias algorithm [28] is run on $\mathcal{V}$ in order to find all substrings that have a user-defined minimum length and minimum frequency. These substrings are the features and each original string s is replaced by the (single) most frequent feature that is also a substring of s. Thus, all feature sets have size one. Finally, each feature set can only be generalized to the "ANY-FEATURE" level. An important strength of this feature extraction algorithm is that the resulting features are very understandable and interpretable, thus increasing the overall understandability of alarm clusters!

4.2. Results

This section presents the thirteen largest alarm clusters that we have found (cf. Table 1). Each line of the table describes one cluster and the "Size" column indicates the cluster's size. The AT column shows the Alarm Types, for which we have provided mnemonic names below the table. Within the table, "any" is generically written for attributes that have been generalized to the root of their taxonomies. It is worth noting that only alarm types 1 and 2 have context attributes. Therefore, the context attribute is undefined for all the other alarm types. Also, the port attributes are occasionally undefined. For example, the ICMP protocol has no notion of ports [30]. As a consequence, the port attributes of alarm type 5 are undefined. Finally, recall that the names *ip1*, *ip2*, ... refer to the machines in Figure 1a.

It is worth noting that the clusters in Table 1 cover 95% of all alarms. Thus, we have found a very crisp summary of almost the entire alarm log. Moreover, using this summary for root cause discovery is a huge simplification over using the original alarm log (more on this in the following subsection). To estimate the payoff of resolving the root causes, we have written filters that remove all alarms that match one of the clusters. We then applied these filters to the alarms of the following month. The result was that 82 percent of all alarm messages were automatically discarded by the filter! Thus, if the root causes had been resolved, then 82 percent less work would have been the payoff in the subsequent month.

4.3. Discussion

This section shows how one can derive hypothetical root causes from the generalized alarms in Table 1. Of course, the hypothetical root causes have to be validated, but in our practical work, this validation has almost always confirmed

Table 1. The thirteen largest alarm clusters.

AT	Src-Port	Src-IP	Dst-Port	Dst-IP	Time	Context	Size
1	*NON-PRIV*	*EXTERN*	80	*ip4*	any	see text	54310
1	*NON-PRIV*	*EXTERN*	80	*ip5*	any	see text	54013
1	*NON-PRIV*	*FIREWALL*	80	*EXTERN*	any	any	17830
2	*NON-PRIV*	*FIREWALL*	21	*EXTERN*	any	any	6439
2	*NON-PRIV*	*EXTERN*	21	*WWW/FTP*	any	any	4181
3	undefined	*ip6*	undefined	*ip1*	*WORKDAY*	undefined	4581
3	undefined	*ip6*	undefined	*ip2*	*WORKDAY*	undefined	3708
4	*NON-PRIV*	*ip1*	80	any	any	undefined	761
4	*NON-PRIV*	*ip2*	80	any	any	undefined	663
4	*NON-PRIV*	*FIREWALL*	25	any	any	undefined	253
5	undefined	*EXTERN*	undefined	*ip4*	any	undefined	823
5	undefined	*EXTERN*	undefined	*ip5*	any	undefined	711
6	undefined	*ip7*	undefined	*FIREWALL*	*END-OF-MONTH*, *TUESDAY*	undefined	861

Alarm Types (AT): 1 ≡ "WWW IIS View Source Attack"; 2 ≡ "FTP SYST Command Attempt"; 3 ≡ "IP Fragment Attack"; 4 ≡ "TCP SYN Host Sweep"; 5 ≡ "Fragmented ICMP Traffic"; 6 ≡ "Unknown Protocol Field in IP Packet";

our first hypothesis. This shows the value of generalized alarms in root cause analysis. The following discussion proceeds by alarm type (cf. Table 1):

Alarm Type 1: The first two alarm clusters in Table 1 contain the following (sanitized) substring in their context fields:

GET /search_cgi/cgi?action=View&VdkVgwKey=
http%3A%2F%2Fwww**%2E**xxx**%2E**gov

This request is completely legal and, based on Table 1, it has been issued more than 100000 times. Thus, the most likely root cause is an under-specified signature. Indeed, a detailed analysis has shown that alarms of type 1 occur whenever a URL contains "%2E", as is the case for the above URL. Finally, the third type-1 alarm turned out to be the dual problem: Internal clients requesting external Web pages, the URL of which contains "%2E".

Alarm Type 2: These alarms simply highlight the fact that many FTP clients issue the SYST command – a legal command that returns information about the FTP server. The root cause seems to be a signature that is triggered whenever the keyword "SYST" is sent to the FTP port. This root cause has been validated subsequently.

Alarm Type 3: Either *ip6* is maliciously sending fragmented packets to the firewalls or there is a router that fragments the packets between *ip6* and the firewalls. Our investigation has shown that the second hypothesis is correct.

Alarm Type 4: Here, the IDS thinks that the firewalls are running host sweeps. In reality, however, the firewalls proxy the HTTP (port 80) and SMTP (port 25) requests of their clients. While exercising this function, the firewalls occasionally contact many external machines at virtually the same time. The resulting traffic resembles host sweeps.

Alarm Type 5: After investigating the source IPs, we realized that they all belong to various Internet Service Providers. Therefore, we conjectured that there is some link between fragmented ICMP traffic and modem access to the Internet.

Alarm Type 6: At the end of the month, a machine on the Internet starts using an unknown transport layer protocol to communicate with the firewall. As many security tools ignore protocols that they do not understand, attackers occasionally use unknown protocols to establish covert channels. A closer investigation of this generalized alarm seems to indicate that, indeed, *ip7* is trying to set up a covert channel.

In summary, knowledge of generalized alarms has made it rather straightforward to derive the above root causes. Moreover, by deriving only six different root causes (one per alarm type), we have managed to understand 95% of the

156380 alarms! Section 4.2 has shown that the future alarm load will decrease by 82% if these root causes are removed. That, in turn, enables a much more thorough analysis of the remaining alarms! Note that it is beyond the scope of this paper to discuss this analysis of the remaining alarms.

5. Conclusion & Future Work

This paper has considered the problem of intrusion detection systems overloading their human operators by triggering thousands of alarms per day. Our solution to this problem exploits the observation that 90% of these alarms can be attributed to one out of a small number of root causes. These 90% of alarms are highly redundant and distract the ID analyst from more stealthy activities. Therefore, we argue that – roughly once a month – one should take the time to identify and remove the most predominant root causes. To make this idea practical, we introduce alarm clustering as a method that supports the discovery of root causes. Furthermore, we show that the future alarm load decreases dramatically if the discovered root causes are removed. Thanks to this reduction in alarm load, analyzing intrusion detection alarms becomes much more cost-effective and much less error-prone.

Our future work has two main axes:

- We will investigate extensions of our framework. In particular, support for DAG-structured taxonomies is an interesting research direction.
- We will work on a knowledge-based tool that automatically derives the most likely root cause from a given generalized alarm.

Acknowledgments

The author thanks Marc Dacier for his valuable comments on earlier versions of this article.

This research was supported by the European IST Project MAFTIA (IST-1999-11583), which is partially funded by the European Commission and the Swiss Department for Education and Science. The views herein are those of the author and do not necessarily reflect the views of the supporting agencies.

References

[1] R. Agrawal *et al.* Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In *ACM SIGMOD Int'l Conf. on Management of Data*, pages 94–105, 1998.

[2] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering Gene Expression Patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

[3] J. Broderick – Editor. IBM outsourced solution, 1998. `http://www.infoworld.com/cgi-bin/displayTC.pl?/980504sb3-ibm.htm`.

[4] CERT Advisory CA-1996-26: Denial-of-Service Attack via ping. `http://www.cert.org`.

[5] Cisco Systems, Inc. NetRanger Documentation. `http://www.cisco.com/`.

[6] M. Erlinger and S. Staniford-Chen. Intrusion Detection Exchange Format (idwg). `http://www.ietf.org/html.charters/idwg-charter.html`.

[7] D. Fasulo. An Analysis of Recent Work on Clustering Algorithms, 1999. `http://www.cs.washington.edu/homes/dfasulo/`.

[8] V. Ganti, J. Gehrke, and R. Ramakrishnan. CACTUS – Clustering Categorical Data Using Summaries. In *5th ACM SIGKDD Int'l Conf. on Knowldege Discovery in Databases (SIGKDD)*, pages 73–83, 1999.

[9] D. Gibson, J. M. Kleinberg, and P. Raghavan. Clustering Categorical Data: An Approach Based on Dynamical Systems. *VLDB Journal*, 8(3):222–236, 2000.

[10] S. Guha, R. Rastogi, and K. Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes. *Information Systems*, 25(5):345–366, 2000.

[11] J. Han, Y. Cai, and N. Cercone. Knowledge Discovery in Databases: An Attribute-Oriented Approach. In *18th VLDB Conference*, pages 547–559, 1992.

[12] J. Han, Y. Cai, and N. Cercone. Data-Driven Discovery of Quantitative Rules in Relational Databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(1):29–40, 1993.

[13] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publisher, 2000.

[14] J. L. Hellerstein and S. Ma. Mining Event Data for Actionable Patterns. In *The Computer Measurement Group*, 2000.

[15] Internet Security Systems, Inc. RealSecure Signatures Reference Guide, 2001. `http://documents.iss.net/literature/RealSecure/RS_Signatures_6.0.pdf`.

[16] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[17] K. Julisch. Dealing with False Positives in Intrusion Detection. In *3nd Workshop on Recent Advances in Intrusion Detection*, 2000. `http://www.raid-symposium.org/raid2000/program.html`.

[18] I. Katzela and M. Schwartz. Schemes for fault identification in communication networks. *IEEE/ACM Transactions on Networking*, 3(6):753–764, 1995.

[19] M. Klemettinen. *A Knowledge Discovery Methodology for Telecommunication Network Alarm Data*. PhD thesis, University of Helsinky (Finland), 1999.

[20] D. Koller and M. Sahami. Toward Optimal Feature Selection. In *13th Int'l Conf. on Machine Learning (ICML-96)*, pages 284–292, 1996.

[21] W. Lee. *A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems*. PhD thesis, Computer Science Department, Columbia University, NY, 1999.

[22] H. Liu and R. Setiono. A Probabilistic Approach to Feature Selection – a Filter Solution. In *13th Int'l Conf. on Machine Learning*, pages 319–327. Morgan Kaufmann, 1996.

[23] C.-C. Lo and S.-H. Chen. Robust Event Correlation Scheme for Fault Identification in Communication Network. *IEEE Global Telecommunications Conference (GLOBECOM)*, 6:3745–3750, 1998.

[24] S. Manganaris *et al.* A Data Mining Analysis of RTID Alarms. In *2nd Workshop on Recent Advances in Intrusion Detection*, 1999. `http://www.raid-symposium.org/raid99/index.html`.

[25] A. Mounji. *Languages and Tools for Rule-Based Distributed Intrusion Detection*. PhD thesis, Facultes Universitaires Notre-Dame de la Paix Namur (Belgium), 1997.

[26] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

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

[28] I. Rigoutsos and A. Floratos. Combinatorial pattern discovery in biological sequences: The TEIRESIAS Algorithm. *Bioinformatics*, 14(1):55–67, 1998.

[29] S. Staniford, J. A. Hoagland, and J. M. McAlerney. Practical Automated Detection of Stealthy Portscans. In *ACM Computer and Communications Security IDS Workshop*, pages 1–7, 2000.

[30] A. S. Tanenbaum. *Computer Networks*. Prentice-Hall, 1996.

[31] A. Valdes. Probabilistic Alert Correlation. In *4th Workshop on Recent Advances in Intrusion Detection*, 2001.

[32] Y. Yang and J. O. Pedersen. A Comparative Study on Feature Selection in Text Categorization. In *14th Int'l Conf. on Machine Learning*, pages 412–420, 1997.

[33] S. Yemini *et al.* High speed and robust event correlation. *IEEE Communications Magazine*, 34(5):82–90, 1996.

A. NP Completeness of Alarm Clustering

The alarm clustering problem is NP-complete. More precisely, let $\mathcal{L}$ be an alarm log, min_size a minimum cluster size, and $\mathcal{T}_i$ a taxonomy for each attribute A_i, $i \in \{1, \ldots, n\}$, in $\mathcal{L}$. Then, the problem of finding a cluster $\mathcal{C} \subseteq \mathcal{L}$ that minimizes $\Delta(\mathcal{C})$ while satisfying $|\mathcal{C}| \geq min\_size$ is NP-complete.

Proof: We shall reduce the CLIQUE problem [26] to the alarm clustering problem. In the CLIQUE problem, which is known to be NP-complete [26], we are given a graph G and an integer k. The goal is to decide whether G contains a k-*clique*, i.e. a fully connected subgraph of size k. We assume that G contains at least $\binom{k}{2}$ edges because otherwise, there is trivially no k-clique in G. Now, the following steps will reduce the CLIQUE problem on the alarm clustering problem (Figure 4):

(1) We assign a separate attribute A_i to each node in G. Let $\{A_1, \ldots, A_n\}$ be the resulting set of attributes.

(2) We define $\text{dom}(A_i) := \{0, 1\}$ and $\mathcal{T}_i := 1 \longrightarrow 0$ for $i = 1, \ldots, n$.

(3) For each edge e in G, we add one alarm a to $\mathcal{L}$. Let A_i and A_j ($A_i \neq A_j$) denote the attributes that correspond to the endpoints of e. Then, the attributes A_i and A_j of alarm a are set to 1, whereas all the other attributes are set to 0.

(4) We set $min\_size := \binom{k}{2}$, which corresponds to the number of edges in a k-clique.

(5) Finally, we solve the alarm clustering problem that has just been defined. Let $\mathcal{C} \subseteq \mathcal{L}$ be the optimal alarm cluster, i.e. the alarm cluster that minimizes $\Delta(\mathcal{C})$ while satisfying $|\mathcal{C}| \geq min\_size$. Furthermore, let $c = (c_1, \ldots, c_n)$ be the cover of $\mathcal{C}$. Then, G contains a k-clique if and only if $\sum_{i=1}^{n} c_i = k$.

Two facts are key to understanding step 5. First, the cover $c = (c_1, \ldots, c_n)$ satisfies $c_i = \vee_{a \in \mathcal{C}} a[A_i]$ for all i and second, $\Delta(\mathcal{C}) = l - 2$, where $l = \sum_{i=1}^{n} c_i$ is the number of ones in cover c. The equation $\Delta(\mathcal{C}) = l - 2$ follows from the definition of $\Delta(\mathcal{C})$ (see equation (1)) and the fact that $\delta(a, c) = l - 2$, which holds for all alarms $a \in \mathcal{C}$.

Finally, going back to step 5, suppose that $\sum_{i=1}^{n} c_i = k$ holds. Then, $\{A_i | c_i = 1\}$ defines a k-clique (and the alarms in $\mathcal{C}$ correspond to its edges). Conversely, suppose $\sum_{i=1}^{n} c_i > k$ ($\sum_{i=1}^{n} c_i < k$ is impossible). In this case, there exists no k-clique in G. Indeed, let us assume that G did contain a k-clique. Then, let $\mathcal{C}'$ denote the cluster of alarms that correspond to the edges in the hypothetical k-clique. Furthermore, let $c' = (c'_1, \ldots, c'_n)$ be the cover of $\mathcal{C}'$. Now, the equations $\sum_{i=1}^{n} c'_i = k$ and ultimately $\Delta(\mathcal{C}') = k - 2$ follow. This contradicts the assumption that $\Delta(\mathcal{C}) = \sum_{i=1}^{n} c_i - 2$ is minimal (indeed $\Delta(\mathcal{C}') < \Delta(\mathcal{C})$).

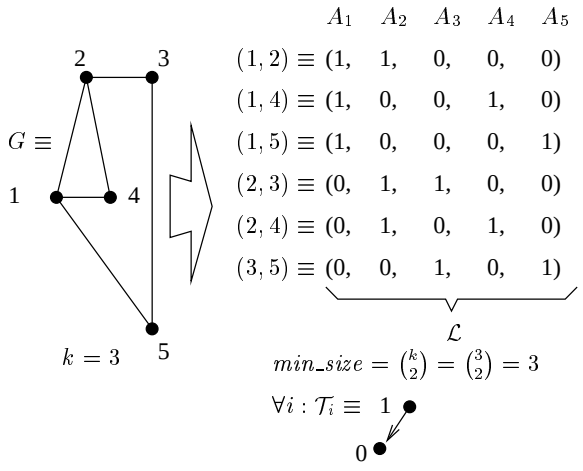


Figure 4. Example of reducing CLIQUE to alarm clustering.