

Mining Intrusion Detection Alarms for Actionable Knowledge

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

Marc Dacier
IBM Research
Zurich Research Laboratory
dac@zurich.ibm.com

ABSTRACT

In response to attacks against enterprise networks, administrators increasingly deploy intrusion detection systems. These systems monitor hosts, networks, and other resources for signs of security violations. The use of intrusion detection has given rise to another difficult problem, namely the handling of a generally large number of alarms. In this paper, we mine historical alarms to learn how future alarms can be handled more efficiently. First, we investigate episode rules with respect to their suitability in this approach. We report the difficulties encountered and the unexpected insights gained. In addition, we introduce a new conceptual clustering technique, and use it in extensive experiments with real-world data to show that intrusion detection alarms can be handled efficiently by using previously mined knowledge.

Keywords

Intrusion detection, alarm investigation, data mining, conceptual clustering, episode rules.

1. INTRODUCTION

Over the past 10 years, the number as well as the severity of network-based computer attacks have significantly increased [1]. As a consequence, classic computer security technologies such as authentication and cryptography have gained in importance. Simultaneously, intrusion detection has emerged as a new and potent approach to protect computer systems [2, 13]. In this approach, so-called *Intrusion Detection Systems (IDSs)* are used to monitor and analyze the events occurring in a computer system. IDSs trigger alarms when they detect signs of security violations. The response to such security incidents is site-dependent, but typically includes law suits, firewall reconfigurations, and the fixing of discovered vulnerabilities.

Practitioners [9, 33] as well as researchers [8, 11, 30] have observed that IDSs can easily trigger thousands of alarms

per day, up to 99% of which are false positives (i.e. alarms that were triggered incorrectly by benign events). This flood of mostly false alarms has made it very difficult to identify the (hidden) true attacks. For example, the manual investigation of alarms has been found to be labor-intensive and error-prone [9, 12, 33]. Tools to automate alarm investigation are being developed [12, 14, 42], but there is currently no silver-bullet solution to this problem.

This paper shows that data mining can be used to support and partially automate the investigation of intrusion detection alarms. Specifically, we mine historical alarms for actionable knowledge that enables us to handle future alarms more efficiently. For example, having discovered patterns of false positives, we can reduce the future alarm load by filtering out these patterns (but see Section 3.1 for the dangers of filtering). As it is unclear which data mining technique is most effective in this learning-based approach, we investigate two different techniques, namely episode rules and a conceptual clustering technique. In doing so, we make the following novel contributions: First, we report the unexpected insights that episode rules have given us into the nature of intrusion detection alarms. Second, we explain why nevertheless, episode rules do not appear to be optimal in the specific environment in which we operate. Third, we explain why and how we adapted a standard conceptual clustering technique to the domain of intrusion detection. The resulting new technique has several desirable properties, which make it of broader interest. Fourth, we run extensive experiments with the newly introduced clustering technique to show that data mining can extract knowledge that enables efficient alarm handling.

The remainder of this paper is organized as follows: Section 2 reviews related work. Section 3 introduces our framework for using data mining to support alarm investigation. Section 4 investigates episode rules in this framework. Section 5 describes a new conceptual clustering technique, and Section 6 uses it to validate our data-mining-based approach to alarm investigation. Section 7 offers our conclusions.

1.1 Alarm Model

IDSs trigger alarms to report presumed security violations. We model alarms as tuples over the Cartesian product $D_{A_1} \times \dots \times D_{A_n}$, where $\{A_1, \dots, A_n\}$ is the set of alarm attributes and D_{A_i} is the *domain* (i.e. the range of possible values) of attribute A_i . The alarm attributes capture intrinsic alarm properties, such as the alarm source, the alarm destination,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGKDD '02 Edmonton, Alberta, Canada

Copyright 2002 ACM 1-58113-567-X/02/0007 ...\$5.00.

the alarm type (which encodes the observed attack), and the time-stamp. We denote the A_i value of an alarm a by $a.A_i$.

2. RELATED WORK

The idea of using data mining to support alarm investigation is not new. Manganaris *et al.* mine association rules over alarm bursts [33]. Subsequently, alarms that are consistent with these association rules are deemed “normal” and are discarded. The risk of losing relevant alarms is not considered in this work, whereas bounding this risk is central to our approach. Clifton and Gengo [11] use episode mining to guide the construction of custom-made filtering rules. Although Section 4 pursues the same idea, it offers new insights into the value of this approach. Also related is our earlier work [30], which introduces the clustering technique of Section 5. Here, we extend this work in three ways: We motivate the design decisions that lead to said clustering technique, we discuss its theoretical properties, and we evaluate it by means of extensive experiments.

Other related work comes from Barbará *et al.*, who use incremental data mining techniques to detect anomalous network traffic patterns in real time [3, 5]. Lee and Stolfo use data mining for feature construction and training of classifiers that detect intrusions [32]. The goal of these research projects is to construct IDSs more systematically and to overcome limitations of existing IDSs. Our work, by contrast, aspires to use existing IDSs more efficiently. There are many other research projects that also applied data mining to intrusion detection. An overview of these projects and a general treatment of data mining in computer security can be found in a recent book edited by Barbará and Jajodia [4]. Data mining for fraud detection is investigated by Fawcett and Provost [15], and by Chan and Stolfo [10].

Alarm correlation systems [12, 14, 40, 42] try to group alarms so that the alarms of the same group pertain to the same phenomenon (e.g. the same attack). In that way, they offer a more condensed view on the security issues raised by an IDS. The work by Dain and Cunningham [12] is noteworthy as it uses data mining techniques to learn correlation rules from hand-labeled training examples. This approach assumes that there is a human expert who (implicitly) knows the correlation rules, so the machine can learn them from him or her. By contrast, we advocate exploratory data mining techniques that assume no prior knowledge on the side of the user. More generally, we employ data mining to understand how to handle alarms more efficiently, be it by means of correlation, filtering, patching of flawed IDS signatures, blocking of attackers at the firewall, or something else.

In the world of telecommunication networks, Klemettinen uses association rules and episode rules to support the development of alarm correlation systems [31]. Hellerstein and Ma pursue the same goal by means of visualization, periodicity analysis, and m-patterns (a variant of association rules requiring mutual implication) [26]. Garofalakis and Rastogi investigate bounded-error lossy compression of network management events [18]. Note that a priori, it is not clear how well these techniques work in intrusion detection. Here, we address this question for episode rules and for a new clustering technique that we derived specifically for intrusion detection.

3. TECHNICAL OVERVIEW

This section introduces our framework for using data mining to support alarm investigation. Moreover, the data used in the experiments is described.

3.1 Framework for Alarm Investigation

Our data mining philosophy can be described as “learning from the past to master the future.” More precisely, we apply data mining techniques to historical intrusion detection alarms to gain new and actionable insights. These insights are subsequently used to reduce the future alarm load that the human analyst has to handle. This can be done in at least two ways: By eliminating alarm root causes so that no more alarms are triggered, or by writing filtering and correlation rules that automate alarm processing. Figure 1 illustrates this process.

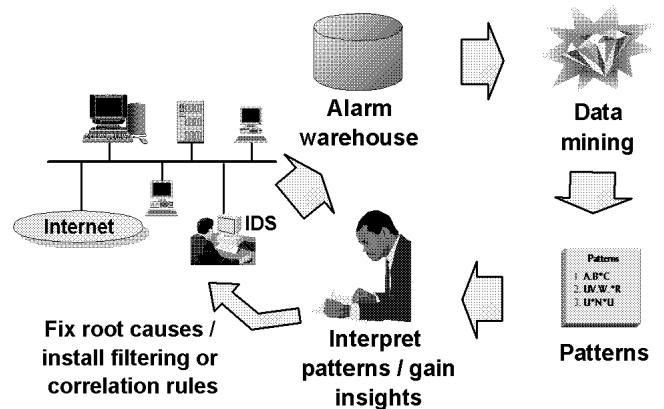


Figure 1: Using data mining for alarm investigation.

Let us first consider how one can eliminate alarm root causes. In one case we observed a misconfigured secondary DNS server that did half-hourly DNS zone transfers from its primary DNS server. The deployed IDS would trigger “DNS Zone Transfer” alarms because DNS zone transfers can be used to “spy out” a target network. By correctly setting up the secondary DNS server, we eliminated the alarm root cause and all its associated alarms. In another case, we fixed a broken TCP/IP stack that generated fragmented traffic and thereby triggered “Fragmented IP” alarms. Similarly, an attacking machine can be sanitized or shunned at the firewall, both of which eliminates the alarm root cause.

On occasion, alarm root causes are not under our control or they are expensive to eliminate. Then, custom-made filtering rules (which automatically discard alarms) or correlation rules (which intelligently group and summarize alarms) can be an alternative. The primary problem with filtering and correlation is that it can destroy valuable information. For example, filtering can discard important alarms and correlation can group alarms in meaningless ways. Our approach addresses this problem as it uses data mining as a supporting technology that enables a human expert to understand *why* alarms are triggered and *how* they should be handled in the future. Based on his or her understanding, the human expert can then devise custom-made filtering and correlation rules that are safe to use.

3.2 Requirements

Note that the framework of Figure 1 does not stipulate any particular data mining technique. However, to be of value, a prospective data mining technique should satisfy the following requirements:

Scalability: IDSs can trigger well over a million alarms per month (cf. the column “Max” of Table 1, which indicates for the year 2001 the maximum number of alarms per month).

Noise tolerance: Intrusion detection alarms can be very noisy [6, 38].

Multiple attribute types: Intrusion detection alarms can contain numerical, categorical, time, and free-text attributes [30]. Ideally, a data mining technique should support and use all of these attribute types.

Ease of use: The people using the data mining techniques are security rather than data mining experts. As a consequence, setting parameters should not require extensive tweaking or an overly strong background in data mining and statistics.

Interpretability & relevance of patterns: The data mining step should only generate highly interpretable and relevant patterns.

To appreciate the importance of the last point (“Interpretability & relevance of patterns”), recall that the results of the data mining step are interpreted and acted upon by a human expert. This raises the need for highly interpretable and relevant patterns as otherwise, the human cost of learning from these patterns would become too high. Moreover, the process of Figure 1 is *iterative* and *site-specific*, which further reinforces the interpretability and relevance requirement. The process is *iterative* as it has to be repeated roughly once a month to keep up with changes in the alarm behavior of IDSs. Such changes occur, for example, as the result of new attacks, updates to the IDS software, or system reconfigurations. Similarly, the process is *site-specific* as each site generally has a very unique alarm mix [36]. As a consequence, each site must be treated individually.

3.3 A Note on the Experiments

A preliminary remark on intrusion detection terminology is in order: IDSs are classified into knowledge-based and behavior-based systems [13]. *Knowledge-based systems* such as STAT [27] use knowledge accumulated about attacks to detect instances of these attacks. *Behavior-based systems* (e.g. IDES [29]) use a reference model of normal behavior and flag deviations from this model as anomalous and potentially intrusive. Another dichotomy splits IDSs according to their audit sources. Specifically, *host-based IDSs* analyze host-bound audit sources such as operating system audit trails, system logs, or application logs, whereas *network-based IDSs* analyze network packets that are captured from a network.

The experiments in this paper use alarms from network- and knowledge-based commercial IDSs that were deployed in operational (i.e. “real-world”) environments. We consider it a

Table 1: Overview of IDSs used in experiments.

IDS	Type	Location	Min	Max	Avg
1	A	Intranet	7643	67593	39396
2	A	Intranet	28585	1946200	270907
3	A	DMZ	11545	777713	310672
4	A	DMZ	21445	1302832	358735
5	A	DMZ	2647	115585	22144
6	A	Extranet	82328	719677	196079
7	A	Internet	4006	43773	20178
8	A	Internet	10762	266845	62289
9	A	Internet	91861	257138	152904
10	B	Intranet	18494	228619	90829
11	B	Intranet	28768	977040	292294
12	B	DMZ	2301	289040	61041
13	B	DMZ	3078	201056	91260
14	B	Internet	14781	1453892	174734
15	B	Internet	248145	1279507	668154
16	B	Internet	7563	634662	168299

strength of our validation that it uses alarms from real-world environments rather than from simulated or laboratory environments, which can have significant limitations [36]. We are not in possession of extensive data collections from host- or behavior-based IDSs and therefore exclude experiments with these IDS types. However, our experiments with the data available for these IDS types gave results comparable to the ones presented in this paper.

Table 1 introduces the sixteen IDSs we use throughout this paper. Our selection criteria was to offer a representative mix of IDSs from different vendors in different operational environments. The sixteen IDSs are deployed at eleven different companies, and no two IDSs are deployed at the same geographic site. The “IDS” column contains a numerical identifier that will be used throughout the paper to reference the IDSs. The “Type” column indicates the IDS type, namely “A” or “B”, both of which are leading commercial IDSs. To avoid unintended commercial implications, we do not reveal the product names or vendors of “A” and “B”. For each IDS, we employ all alarms that were triggered in the year 2001. The minimum, maximum, and average number of alarms per month are listed for each IDS in the “Min”, “Max”, and “Avg” columns, respectively. Finally, the “Location” column indicates where the IDSs are deployed:

Intranet: Denotes an IDS on an internal corporate network without Internet access.

DMZ: Designates an IDS on a perimeter network that is protected by a firewall, but offers services to the Internet.

Extranet: Denotes an IDS on a network that is shared between multiple cooperating companies, but is not accessible from the Internet.

Internet: Indicates an IDS that is deployed before the external firewall on a direct link to the Internet.

Note the generally large difference between the minimum and maximum number of alarms per month. This difference

reflects changes in the alarm mix over time. Analogously, the vastly varying alarm loads *between* IDSs provide intuitive evidence of the uniqueness of each site. This illustrates our earlier point that the alarm investigation process of Figure 1 has to be applied iteratively and site by site. Finally, for confidentiality reasons, we cannot provide more detailed information on the sites where the IDSs are deployed.

4. EPISODE RULES

Episode rules are a data mining technique that was developed to find patterns in event sequences [34, 35]. Intuitively, episode rules are implication rules that predict the occurrence of certain alarms based on the occurrence of other alarms. For example, an episode rule might state that in 50 percent of all cases, an “Authentication Failure” alarm is followed within 30 seconds by a “Guest Login” alarm.

In network management, researchers have successfully used episode rules in a framework like ours [31]. Therefore, episode rules are a natural candidate for the data mining step of Figure 1. In this section, we report our experience with this approach. In addition, we summarize the insights we gained into the nature of intrusion detection alarms.

4.1 Definitions

To formally define episode rules, we need the following terminology [34, 35]: An *alarm predicate* is a boolean expression that tests certain alarm properties such as the alarm type or source. A *serial (parallel) episode* is a sequence (multi-set) $\alpha = \langle P_i \rangle_{1 \leq i \leq n}$ of alarm predicates. Note that the predicates of a serial episode are ordered, whereas they have no order in parallel episodes. Given a parallel episode α and an alarm sequence S , a time interval $[t_s, t_e]$ is an *occurrence* of α if it contains a *distinct* alarm a for each P_i such that $P_i(a)$ holds. For occurrences of serial episodes, the alarm order must additionally match the predicate order (i.e. the alarm a satisfying P_i must occur before the alarm a' satisfying P_{i+1}). The interval $[t_s, t_e]$ is a *minimal occurrence* of α if there is no proper subinterval of $[t_s, t_e]$ that would also be an occurrence of α . Finally, *episode rules* are implication rules of the form

$$\langle P_1, \dots, P_k \rangle \implies \langle P_1, \dots, P_k, \dots, P_n \rangle [s, c, W], \quad (1)$$

where $\langle P_i \rangle_{1 \leq i \leq k}$ is a sub-episode of $\langle P_i \rangle_{1 \leq i \leq n}$, and the two episodes are either both serial or parallel. The parameters s , c , and W are called *support*, *confidence*, and *window width* and their interpretation is the following: Episode $\langle P_i \rangle_{1 \leq i \leq n}$ has s minimal occurrences in sequence S . Moreover, if $t_e - t_s \leq W$ and $[t_s, t_e]$ is a minimal occurrence of episode $\langle P_i \rangle_{1 \leq i \leq k}$, then there is a c percent probability for the super-episode $\langle P_i \rangle_{1 \leq i \leq n}$ to occur in $[t_s, t_s + W]$. Variations of the above definitions are described in [34, 35].

4.2 Experience

We have used episode rules for the data mining step in Figure 1. In our experiments, we have mined the alarms from our experimental IDSs (cf. Table 1) for serial and parallel episode rules. The set of admissible alarm predicates was restricted to predicates of the form $P(a) \equiv (\wedge_i a.A_i = c_i)$, where a is an alarm, A_i are attributes, and c_i are constants. The episodes and episode rules discovered contained many interesting patterns, including the following ones:

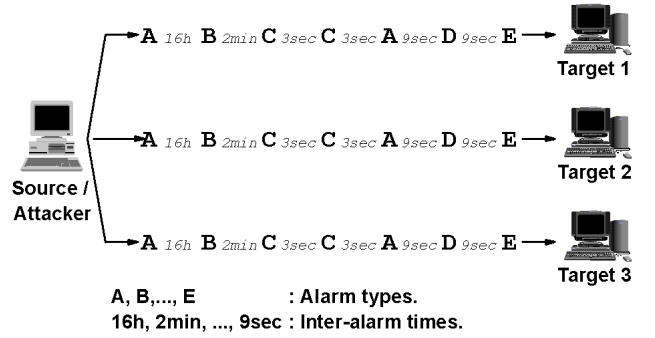


Figure 2: An attack tool being run against three targets.

- We discovered episodes that were characteristic of attack tools. For example, we repeatedly found episodes that resulted from attack scenarios like the one in Figure 2, where a source host triggers the same sequence of alarms against different target hosts. In general, scenarios like this result from an attacker trying out his or her attack tool against different targets.
- We found episode rules where the right-hand side represented a massive attack, while the left-hand side was an early indicator of this attack. Thus, the left-hand side would give early warnings of imminent attacks.
- We discovered alarms that — for some IDS-specific reason — almost always entail other alarms. For example, on IDSs of one type, “TCP FIN Host Sweep” alarms imply “Orphaned FIN Packet” alarms with a confidence of 100% (but not vice versa).
- We discovered episodes that correspond to legitimate system operations such as remote file-system mounts or certain network management tasks.

Clearly, these patterns are of direct relevance to alarm handling. For example, knowing the episodes that correspond to legitimate system operations, it is easy to filter them out in the future. Similarly, if an alarm systematically entails other (redundant) alarms, then one can reduce the overall alarm load by fusing these alarms into a single, semantically richer meta-alarm. Finally, episodes that result from attack tools can be used to detect future attacks in a more reliable and timely manner.

Nevertheless, episode mining has two important drawbacks in the framework of Figure 1. First, the attainable degree of automation is very low. In our experiments, less than one percent of alarms could be handled automatically thanks to previously mined episodes and episode rules. The remaining 99% of alarms still had to be investigated manually. The second drawback is that episode mining tends to produce a large number of irrelevant or redundant patterns [31]. Moreover, many of these patterns were difficult to interpret in terms of real-world phenomena. Thus, given a large number of not always easy to interpret episodes and episode rules, locating the truly interesting ones became a time-consuming activity.

All in all, we felt that the benefit of a one-percent reduction in alarm load did not compensate for the cost of locating the interesting episodes and episode rules. Therefore, we do not use episode rules as a data mining technique in our framework. Despite this negative result, our experiments with episode rules have been important for two reasons: First, we have come to appreciate the difficulty of finding a data mining technique that works well in our framework. As a result, we have tailored a data mining technique to our needs (cf. Section 5). Second, we have gained important new insights into the nature of intrusion detection alarms. These insights are summarized in the following subsection.

4.3 Alarm Characteristics

An important lesson that episode mining has taught us is that intrusion detection alarms are extremely monotonous and repetitive. Specifically, we noticed that almost all high-support episode rules consisted of multiple instances of the same predicate, i.e. $P_i = P_j$ generally held for all i and j in equation (1). The monotony of intrusion detection alarms is illustrated more clearly by the following experiment: Let us randomly choose an IDSs and a source host that has triggered alarms at this IDS. This source host might have triggered many alarms throughout the year 2001, but with a probability of 96% they were all of the same type! Moreover, the probability for all alarms to hit the same target port (target host) is 95% (69%). These probabilities were calculated using the 16 IDSs in Table 1, but we have confirmed them (give or take a few percentage points) using over 90 million alarms from more than 50 different IDSs.

The above observation inspires two ideas for making alarm investigation more efficient. First, source hosts that display diverse behavior, e.g. by triggering alarms of many different types, deserve closer investigation. In fact, hackers generally have little a priori knowledge about their targets and therefore resort to trying different reconnaissance and attack techniques until they are successful or exhausted. In doing so, hackers tend to trigger diverse alarm streams that involve many different alarm types and targets. Given that this kind of diverse behavior is generally rare, it is a rewarding heuristic to investigate it more closely when it occurs. Note, however, that perfectly monotonous behavior (e.g. password guessing) can still constitute an attack. Therefore, zooming in on diverse behavior helps in finding real attacks, but not all attacks are diverse.

The second idea is to automate alarm investigation for the case where a source host keeps triggering alarms of the same type against the same target. Given that this case is so frequent, automating it will vastly relieve the human analyst. Moreover, the dominance of homogeneous and repetitive alarms suggests that intrusion detection alarms have a natural clustering tendency. This leads us to the next section.

5. CONCEPTUAL CLUSTERING

Clustering seeks to group objects into categories (so-called *clusters*) so that members of a category are alike, whereas members of different categories are different [19, 28]. In most clustering techniques, it is an afterthought to represent clusters in an intelligible manner that supports understanding and decision making [19, 28]. *Conceptual clustering*, by contrast, puts cluster representation in the foreground and

searches for clusters that have “good” representations in a given description language [16, 37, 39]. Examples of description languages include variants of predicate logic [7, 37] as well as probabilistic languages that list attribute probabilities [16, 41].

Conceptual clustering has two important advantages in the framework of Figure 1. First, by deriving intelligible descriptions of clusters, it facilitates cluster interpretation. The importance of this point follows from the “Interpretability & relevance of patterns” requirement of Section 3.2. Second, conceptual clustering is particularly good at handling categorical attributes such as IP addresses, port numbers, and alarm types. This strength matters as categorical attributes are known to pose problems for many other clustering techniques [17, 20].

In this paper, we use a variant of the classic *Attribute-Oriented Induction (AOI)* technique [21] as our conceptual clustering tool. AOI was initially introduced as a data summarization technique, but its link to conceptual clustering was subsequently established [23, 25]. Section 5.1 introduces the classic AOI algorithm, and Section 5.2 shows why and how we modified it. The properties of the modified AOI algorithms are discussed in Section 5.3. Our experiments are presented in Section 6.

5.1 Attribute-Oriented Induction

Attribute-oriented induction operates on relational database tables and repeatedly replaces attribute values by more abstract values. The more abstract values are taken from user-defined generalization hierarchies, which, for example, might state that IP addresses can be generalized to networks, timestamps to weekdays, and port numbers to port ranges. Because of generalization, previously distinct alarms become identical and can be merged. In this way, huge relational tables can be condensed into short and highly comprehensible summary tables.

For a more formal treatment, we extend all alarms by a new integer-valued pseudo-attribute, the so-called *count C*. Thus, we henceforth model alarms as tuples over the Cartesian product $D_{A_1} \times \dots \times D_{A_n} \times D_C$. The count attribute is used by the AOI algorithm for book-keeping, only. The alarm attributes A_i are as before. A *generalization hierarchy* is a tree-structured *is-a* hierarchy that shows how concepts are organized into more general concepts. Figure 3.a shows a sample generalization hierarchy for IP addresses. Alarms whose attributes assume non-leaf concepts such as **WWW** or **Net-A** in Figure 3.a are also called *generalized alarms*.

Inputs to the AOI algorithm are a relational table \mathcal{T} over the attributes $\{A_1, \dots, A_n, C\}$, as well as generalization hierarchies \mathcal{H}_i and generalization thresholds d_i for all attributes A_i ($i = 1, \dots, n$). The first step of the AOI algorithm (cf. Figure 4) is to assign the value 1 to the count attribute of each alarm. Subsequently, the main loop (steps 2–8) is iterated: Step 3 selects an attribute A_i and the steps 4 and 5 replace the A_i values of all alarms by their parent values in \mathcal{H}_i . By doing so, previously distinct alarms can become identical. Two alarms a and a' are *identical* if $a.A_i = a'.A_i$ holds for all attributes A_i (but $a.C \neq a'.C$ is possible). Steps 6 and 7 merge identical alarms into a single one whose count value

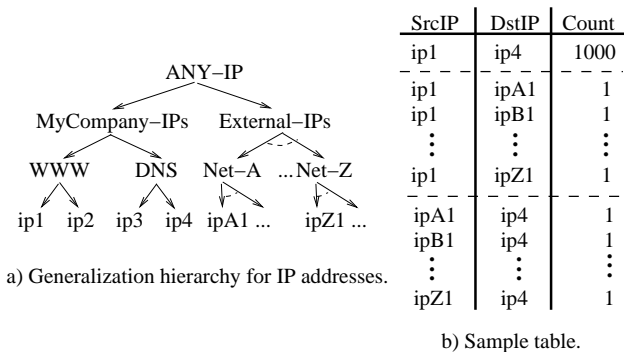


Figure 3: A generalization hierarchy and sample table.

equals the sum of constituent counts. In this way, the count attribute always reflects the number of original alarms that are summarized by a given generalized alarm. Note that each alarm a represents a cluster of size $a.C$. Moreover, the *elements* of a cluster a are the original (ungeneralized) alarms that were merged into a .

One key aspect of the classic AOI algorithm has been left open, namely, how the attributes A_i are selected in step 3. The selection criterion is that any attribute A_i that assumes more than d_i distinct values in table \mathcal{T} can be selected. (Recall that d_i , the generalization threshold, is an input parameter to the algorithm.) The main loop terminates in step 2 if no such attribute exists.

5.2 Modifications

To summarize, an attribute A_i is generalized until it assumes at most d_i distinct values ($i = 1, \dots, n$). This strategy guarantees that the final generalized table contains at most $\prod_{i=1, \dots, n} d_i$ generalized alarms. This strategy of bounding the number of distinct attribute values can lead to excessive generalization, in which too much detail is lost (so-called *over-generalization*). This section illustrates the problem of over-generalization and shows how we modified the classic AOI algorithm to mitigate it.

Figure 3.b shows a sample table having the alarm attributes “SrcIP” (the source-IP) and “DstIP” (the destination-IP). Note that the first tuple in the table represents 1000 occurrences of the alarm (ip1, ip4). We use the generalization hierarchy of Figure 3.a for both alarm attributes, and we assume that both generalization thresholds have been set to 10 (i.e. $d_1 = d_2 = 10$). Given that both alarm attributes assume 27 distinct values, they are both generalized once. This yields a new table whose alarm attributes still have 27 distinct values. Therefore, both attributes are generalized again. The resulting table, which contains the generalized alarms (MyCompany-IPs, MyCompany-IPs, 1000), (MyCompany-IPs, External-IPs, 26), and (External-IPs, MyCompany-IPs, 26), is the final result of classic AOI. Note that this result is (over-)generalized to a point where important details have been lost. In fact, instead of the above result we had rather obtained the alarms (ip1, ip4, 1000), (ip1, External-IPs, 26), and (External-IPs, ip4, 26), which are more specific and informative.

```

1: for all alarms  $a$  in  $\mathcal{T}$  do  $a.C := 1$ ; // Init counts
2: while table  $\mathcal{T}$  is not abstract enough do {
3:   Select an alarm attribute  $A_i$ ;
4:   for all alarms  $a$  in  $\mathcal{T}$  do // Generalize  $A_i$ 
5:      $a.A_i :=$  father of  $a.A_i$  in  $\mathcal{H}_i$ ;
6:   while identical alarms  $a, a'$  exist do // Merge
7:     Set  $a.C := a.C + a'.C$  and delete  $a'$  from  $\mathcal{T}$ ;
8: }
  
```

Figure 4: The classic AOI algorithm.

A major source of over-generalization is “noise”. Indeed, noise forces up the number of distinct attribute values and thereby controls the generalization process. In the above example, there was one main signal (the tuple (ip1, ip4, 1000) of Table 3.b) and five percent of “noise” (the remaining 52 tuples). However, the noise dominated the generalization process and caused the alarm (ip1, ip4, 1000) to be generalized four times, so it became (MyCompany-IPs, MyCompany-IPs, 1000). Noise-induced over-generalization is a serious problem in intrusion detection (cf. Section 3.2), and it motivates our first modification of the classic AOI algorithm.

MODIFICATION 1. We abandon the generalization thresholds d_i as well as the associated strategy of bounding the number of distinct attribute values. Our new strategy tries to find generalized alarms that cover “many” of the original (ungeneralized) alarms. Formally, we search alarms $a \in \mathcal{T}$ that have a count bigger than min_size (i.e. $a.C > \text{min_size}$), where $\text{min_size} \in \mathbb{N}$ is a user-defined constant. Whenever such an alarm is found, it is removed from table \mathcal{T} and reported to the user. Processing continues with the table $\mathcal{T}' := \mathcal{T} \setminus a$. \square

Recall that each alarm a represents a cluster of size $a.C$. The above modification has two effects: First, by imposing a minimum cluster size of min_size , it forces the AOI algorithm to find “large” clusters. Second, by preventing further generalization of an alarm a that satisfies $a.C > \text{min_size}$, it tries to avoid over-generalization. The combined effect is to bias the algorithm towards large clusters that nonetheless have specific representations in the form of generalized alarms. Finally, Modification 1 also raises the need for a new attribute selection criteria for step 3 of Figure 4. To counteract over-generalization, we use the following heuristic criteria, which tries to minimize the total number of attribute generalizations:

MODIFICATION 2. For each alarm attribute A_i , let $F_i := \max\{f_i(v) \mid v \in D_{A_i}\}$ be the maximum of the function

$$f_i(v) := \text{SELECT sum}(C) \text{ FROM } \mathcal{T} \text{ WHERE } A_i = v,$$

which sums the counts C of all alarms $a \in \mathcal{T}$ with $a.A_i = v$. Step 3 of Figure 4 selects an attribute A_i whose F_i value is minimal, i.e. $F_i \leq F_j$ must hold for all j . \square

The motivation for this heuristic is that an alarm a with $a.C > \text{min_size}$ cannot exist unless $F_i > \text{min_size}$ holds for all attributes A_i . Therefore, we use it as a heuristic to increase the smallest F_i value by generalizing its corresponding attribute A_i . Other heuristics are conceivable, but the one given here works well in practice (cf. Section 6).

To see what we have achieved so far, let us reconsider the example of Figure 3 and let min_size equal 20. The alarm (`ip1,ip4,1000`) has a count larger than 20, and is immediately removed from the table and presented to the user. All remaining alarms have counts of one, so that generalization starts. Because of $F_1 = F_2 = 26$, either of the two attributes “SrcIP” or “DstIP” can be selected for generalization. Without loss of generality, we assume that the attribute “SrcIP” is chosen and generalized. The resulting alarms still have counts of one, which is why a second generalization step is initiated. Again, we assume that the “SrcIP” is selected and generalized. The resulting table contains the alarm (`External-IPs,ip4,26`), which is removed and reported to the user. Finally, the “DstIP” attribute is generalized twice, the alarm (`MyCompany-IPs,External-IPs,26`) is reported, and processing ends.

A problem becomes apparent. Although generalizing the attribute “SrcIP” has allowed us to find the alarm (`External-IPs,ip4,26`), it has irrevocably over-generalized the “SrcIP” of the 26 alarms that remain after (`External-IPs,ip4,26`) was removed. As a consequence, the last alarm reported is (`MyCompany-IPs,External-IPs,26`) instead of the more specific (`ip1,External-IPs,26`). This problem, that a generalization step can be opportune in the short run while having undesirable late effects, motivates the Modification 3.

MODIFICATION 3. *After reporting an alarm $a \in \mathcal{T}$ with $a.C > min_size$, we used to continue processing with table $T' := T \setminus a$. Henceforth, we first undo all generalization steps in T' . This involves replacing all generalized alarms by their constituent ungeneralized alarms. Then, processing resumes with the resulting table, in which all counts equal one.* \square

Now, let us reconsider the above example: Processing is unchanged up to the point where the alarm (`External-IPs,ip4,26`) is removed from the table. Then, Modification 3 kicks in and resets the “SrcIP” attribute of the remaining 26 alarms to its original value, namely `ip1`. Finally, the “DstIP” attribute is generalized twice, the alarm (`ip1,External-IPs,26`) is reported, and processing ends.

Our modified version of AOI also supports dynamic generalization hierarchies [22]. *Dynamic generalization hierarchies* are constructed at run-time to fit the data distribution. For example, instead of a static generalization hierarchy that imposes concepts such as “morning”, “evening”, “night”, etc., we dynamically construct generalization hierarchies for the time attribute. Frequently, this yields much better results as the temporal characteristics of alarm patterns are not well-described by static concepts. Similarly, some IDSs extend their alarms by a free-text attribute that stores the supposedly intrusive event. To tap the semantic information of free-text attributes, we dynamically build generalization hierarchies that reflect the *is-substring-of* relationship between frequent substrings. Space restrictions preclude a more detailed discussion of these aspects.

5.3 Discussion

In Section 3.2, we have listed five requirements that a data mining technique should satisfy to be suitable for our framework. Next, we examine how well the modified AOI algorithm meets these requirements:

Scalability: The excellent scalability of classic AOI [23] is mostly preserved by our modifications. For instance, our implementation of the modified algorithm runs on a 700 MHz Pentium III CPU with 1 GB RAM, and processes two million alarms in less than four minutes.

Noise tolerance: By design, the modified AOI algorithm tolerates the noise typically found in intrusion detection alarms (see the discussion before Modification 1).

Multiple attribute types: Using static and dynamic generalization hierarchies, AOI can take advantage of a wide variety of attribute types, including numerical, categorical, time, and free-text attributes.

Ease of use: The min_size parameter has a very intuitive interpretation and is the only one to be set. Some expertise in the application domain is needed to define meaningful generalization hierarchies. However, once defined, these hierarchies are generally static.

Interpretability & relevance of patterns: As Section 6 will point out, we found the results of the modified AOI algorithm to be relevant and easy to interpret.

It is a well-known problem that clustering techniques can “impose” clustering structures that are not warranted by the underlying data [19]. *Cluster validation* addresses this problem by quantitatively assessing the merits of a derived clustering structure [19]. Here, we focus on the properties of the modified AOI algorithm that might prevent or favor the generation of invalid clusters.

To begin with, the modified AOI algorithm has a favorable control flow. In fact, beginning with the most apparent cluster, it repeatedly finds clusters and removes them from the data set. Up to min_size alarms that did not fit well into any cluster might be left over at the end. This, it has been argued [24], is more natural than assigning all alarms (noise and outliers included) to clusters. Another important point is that a high min_size threshold can compromise cluster validity by forcing the algorithm to merge unrelated alarms. However, the merging of unrelated alarms tends to produce over-generalized results, which helps to identify the problem. Then, the algorithm can be re-run with a smaller min_size value. Moreover, “drilling down” into a cluster (i.e. reversely traversing the merge steps that have produced it) is a practical way to identify potentially invalid clusters. Finally, the attribute selection heuristic of Modification 2 can influence the validity of resulting clusters. We plan to investigate this influence in our future work.

6. EXPERIMENTAL RESULTS

This section assesses the modified AOI algorithm with respect to its ability to extract actionable knowledge from intrusion detection alarms. Section 6.1 describes our experiments and the results obtained. Section 6.2 provides evidence that one can expect similar results for different data sets.

6.1 Practical Experience

This section evaluates the modified AOI algorithm in the framework of Figure 1. To this end, we iterate the following experiment with varying parameters: First, we choose

a month m in the year 2001 and an IDS I from Table 1. Then, we use the modified AOI algorithm to analyze all alarms that were triggered by IDS I in month m . Based on the results obtained, we *manually* derive and implement filtering and correlation rules. We apply these rules to the alarms triggered by IDS I in month $m + 1$, and we calculate the percentage of alarms that are handled automatically. This percentage is called the *alarm load reduction* in month $m + 1$. Clearly, a high alarm load reduction is desirable as it means that the human analyst has to handle fewer alarms.

Throughout our experiments, we found the results of the modified AOI algorithm to be intuitive and interpretable. The algorithm returned between eight and thirty-two alarm clusters per IDS and month. Interpreting these clusters, and deriving appropriate filtering and correlation rules took approximately two hours per IDS-month, but this time decreased as we gained more experience. Moreover, when keeping the IDS I fixed, we found a non-negligible overlap between the alarm clusters of successive months. That further reduced the cost of the overall process.

To assess the effectiveness of derived filtering and correlation rules, we ran two sets of experiments. In a first set, we fixed the parameter m to be November 2001, and let the parameter I run over all IDSs in Table 1 (i.e. $I = 1, 2, \dots, 16$). Figure 5 shows the alarm load reductions attained in month $m + 1$ (i.e. in December 2001). As is apparent from the figure, approximately 75% of the December alarms were handled automatically by the rules derived from the November alarms. In a second set of experiments, we fixed $I := 8$ and let m vary. Figure 6 shows for each month the alarm load reduction achieved by the rules from the previous month.

Note that the curve in Figure 6 has a sharp drop in October. Based on our investigation, there was a temporary networking problem in October. The IDS confused this problem with an attack and triggered countless alarms. Clearly, the rules derived from the September alarms could not anticipate this networking problem. As a consequence, they were ineffective and the alarm load reduction dropped sharply. Note, however, that the alarm load reduction across IDSs (cf. Figures 5) and over time (cf. Figure 6) is generally good.

To summarize, we found it intuitive and straightforward to interpret the output of the modified AOI algorithm. Moreover, we achieved an average alarm load reduction of 75%, in our experiments. Based on these results, we postulate that our approach to alarm handling is practical and effective. The next section presents experimental evidence that we would have reached the same conclusion if we had chosen different data sets for the experiments.

6.2 Stability of Results

Note that in the last section, we have strictly adhered to the process of Figure 1. In particular, we have *manually* derived filtering and correlation rules from the data mining results. These experiments were important, because they assessed precisely the approach that is advocated in this paper.

Here, we present experiments, where we *automatically* derive the filtering rules. Everything else is unchanged. To derive the filtering rules, we use a program that proceeds in three steps. First, it reads the output of the modified

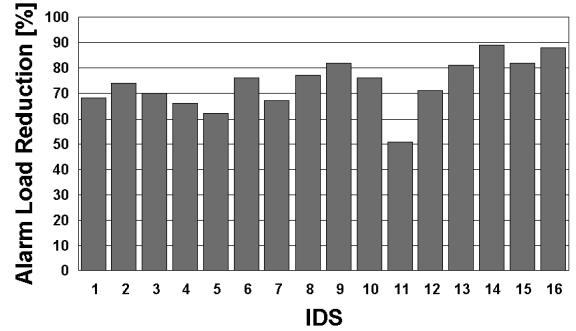


Figure 5: Alarm load reduction in Dec 2001.

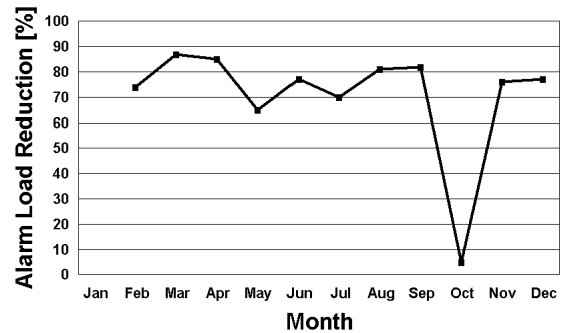


Figure 6: Alarm load reduction for IDS 8.

AOI algorithm. Second, it discards all generalized alarms that — according to user-defined generalization thresholds — are too general. In that way, generalized alarms that a human expert might consider incomprehensible and useless are excluded from further processing. In the third step, the remaining generalized alarms are translated one-to-one into filtering rules. For example, the alarm (ip1,ip4,1000) of Figure 3.b would be translated into the filtering rule “if SrcIP=ip1 and DstIP=ip4 then discard alarm;”.

Our new experimental setup is as follows: For each tuple (I, m) consisting of an IDS I ($I = 1, \dots, 16$) and a month m ($m = \text{Jan}, \dots, \text{Nov}$), we used said program to derive filtering rules from the output of the modified AOI algorithm. Then, we applied these filtering rules to the alarms that IDS I triggered in month $m + 1$. We recorded the resulting alarm load reductions and calculated for each IDS the average alarm load reduction over all eleven months. Figure 7 presents the resulting statistics.

Note that the average alarm load reduction in Figure 7 is around 66%, which is less than the 75% we reported in the previous section. This reflects the fact that the automatically derived filtering rules tend to be more restrictive than the manually derived ones. The important point, however, is that using the alarms from $16 \times 11 = 176$ IDS-months in a different experimental setup, we could confirm the overall effectiveness of our approach to alarm investigation.

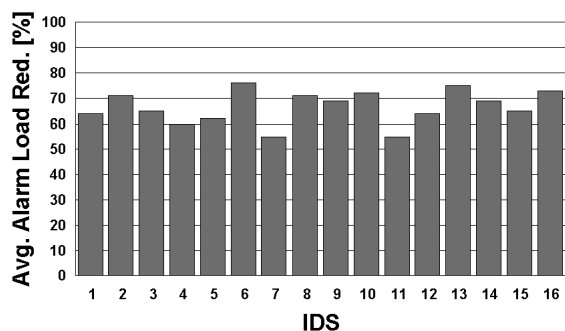


Figure 7: Average alarm load reduction over the year 2001.

7. CONCLUSION

The use of intrusion detection has created the problem to investigate a generally large number of alarms. This paper has shown that data mining can be used to support and partially automate this investigation process. The approach pursued is to mine historical alarms for new and actionable insights. These insights are subsequently used to automate alarm processing, or to pin down and fix alarm root causes. The viability of this approach is demonstrated by extensive experiments with real-world data.

We have seen that not all data mining techniques are suitable for the specific environment in which we operate. Specifically, we found that the cost of finding relevant episode rules outweighs the benefit derived from them. Nonetheless, there is value in using episode rules as evidenced by the interesting alarm patterns discovered. Moreover, the results of episode mining have shown that intrusion detection alarms are very homogeneous and repetitive. This observation facilitates attack detection. Specifically, we have explained that a source host triggering a heterogeneous stream of alarms is likely to be an attacker.

We have also investigated the suitability of attribute-oriented induction (a conceptual clustering technique) for our purposes. As the classic technique tends to produce too general results, we have modified it in three ways: First, we have biased the technique to find *large* clusters. Second, we have introduced a new heuristic way of selecting attributes for generalization. Third, we occasionally undo all generalization steps to prevent them from accumulating to the point where over-generalization occurs. The resulting technique was shown to work well in our framework. In addition, it has several desirable properties, which make it of broader interest. Our future work will concentrate on objectively assessing the validity of the clusters derived by the modified attribute-oriented induction technique.

Acknowledgments

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

8. REFERENCES

- [1] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Stoner. State of the Practice of Intrusion Detection Technologies. Technical report, Carnegie Mellon University, January 2000. <http://www.cert.org/archive/pdf/99tr028.pdf>.
- [2] R. Bace. *Intrusion Detection*. Macmillan Technical Publishing, 2000.
- [3] D. Barbará, J. Couto, S. Jajodia, L. Popyack, and N. Wu. ADAM: Detecting Intrusions by Data Mining. In *IEEE Workshop on Information Assurance and Security*, 2001.
- [4] D. Barbará and S. Jajodia, editors. *Applications of Data Mining in Computer Security*. Kluwer Academic Publisher, Boston, 2002.
- [5] D. Barbará, N. Wu, and S. Jajodia. Detecting Novel Network Intrusions Using Bayes Estimators. In *First SIAM Int'l Conf. on Data Mining (SDM'01)*, 2001.
- [6] S. M. Bellovin. Packets Found on an Internet. *Computer Communications Review*, 23(3):26–31, 1993.
- [7] G. Bisson. Conceptual Clustering in a First Order Logic Representation. In *10th European Conf. on Artificial Intelligence*, pages 458–462, 1992.
- [8] E. Bloedorn, B. Hill, A. Christiansen, C. Skorupka, L. Talboot, and J. Tivel. Data Mining for Improving Intrusion Detection, 2000. http://www.mitre.org/support/papers/tech_papers99_00/.
- [9] J. Broderick – Editor. IBM Outsourced Solution, 1998. <http://www.infoworld.com/cgi-bin/displayTC.pl?/980504sb3-ibm.htm>.
- [10] P. Chan and S. Stolfo. Toward Scalable Learning with Non-Uniform Class and Cost Distributions: A Case Study in Credit Card Fraud Detection. In *4th Int'l Conf. on Knowledge Discovery and Data Mining*, pages 164–168, 1998.
- [11] C. Clifton and G. Gengo. Developing Custom Intrusion Detection Filters Using Data Mining. In *Military Communications Int'l Symposium (MILCOM2000)*, October 2000.
- [12] O. Dain and R. K. Cunningham. Fusing Heterogeneous Alert Streams into Scenarios. In Barbará and Jajodia [4].
- [13] H. Debar, M. Dacier, and A. Wespi. A Revised Taxonomy for Intrusion Detection Systems. *Annales des Télécommunications*, 55(7–8):361–378, 2000.
- [14] H. Debar and A. Wespi. Aggregation and Correlation of Intrusion-Detection Alerts. In *4th Workshop on Recent Advances in Intrusion Detection (RAID)*, LNCS, pages 85–103. Springer Verlag, 2001.
- [15] T. Fawcett and F. Provost. Adaptive Fraud Detection. *Data Mining and Knowledge Discovery*, 1:291–316, 1997.

- [16] D. H. Fisher. Knowledge Acquisition Via Incremental Conceptual Clustering. *Machine Learning*, 2:139–172, 1987.
- [17] V. Ganti, J. Gehrke, and R. Ramakrishnan. CACTUS – Clustering Categorical Data Using Summaries. In *5th ACM SIGKDD Int'l Conf. on Knowledge Discovery in Databases (SIGKDD)*, pages 73–83, 1999.
- [18] M. Garofalakis and R. Rastogi. Data Mining Meets Network Management: The Nemesis Project. In *ACM SIGMOD Int'l Workshop on Research Issues in Data Mining and Knowledge Discovery*, May 2001.
- [19] A. Gordon. *Classification*. Chapman and Hall, 1999.
- [20] S. Guha, R. Rastogi, and K. Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes. *Information Systems*, 25(5):345–366, 2000.
- [21] 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.
- [22] J. Han and Y. Fu. Dynamic Generation and Refinement of Concept Hierarchies for Knowledge Discovery in Databases. In *Workshop on Knowledge Discovery in Databases*, pages 157–168, 1994.
- [23] J. Han and Y. Fu. Exploration of the Power of Attribute-Oriented Induction in Data Mining. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. AAAI Press/MIT Press, 1996.
- [24] P. Hansen, B. Jaumard, and N. Mladenovic. How to Choose K Entries Among N . *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 19:105–116, 1995.
- [25] O. Heinonen and H. Mannila. Attribute-Oriented Induction and Conceptual Clustering. Technical Report Report C-1996-2, University of Helsinki, 1996.
- [26] J. L. Hellerstein and S. Ma. Mining Event Data for Actionable Patterns. In *The Computer Measurement Group*, 2000.
- [27] K. Ilgun, R. A. Kemmerer, and P. A. Porras. State Transition Analysis: A Rule-Based Intrusion Detection System. *IEEE Transactions on Software Engineering*, 21(3):181–199, 1995.
- [28] A. Jain, M. Murty, and P. Flynn. Data Clustering: A Review. *ACM Computing Surveys*, 31(3):264–323, 1999.
- [29] H. S. Javitz and A. Valdes. The SRI IDES Statistical Anomaly Detector. In *IEEE Symposium on Security and Privacy, Oakland, CA*. SRI International, May 1991.
- [30] K. Julisch. Mining Alarm Clusters to Improve Alarm Handling Efficiency. In *17th Annual Computer Security Applications Conference (ACSAC)*, pages 12–21, December 2001.
- [31] M. Klemettinen. *A Knowledge Discovery Methodology for Telecommunication Network Alarm Data*. PhD thesis, University of Helsinki (Finland), 1999.
- [32] W. Lee and S. J. Stolfo. A Framework for Constructing Features and Models for Intrusion Detection Systems. *ACM Transactions on Information and System Security*, 3(4):227–261, 2000.
- [33] S. Manganaris, M. Christensen, D. Zerkle, and K. Hermiz. A Data Mining Analysis of RTID Alarms. *Computer Networks*, 34(4), October 2000.
- [34] H. Mannila and H. Toivonen. Discovering Generalized Episodes Using Minimal Occurrences. In *2nd Int'l Conf. on Knowledge Discovery and Data Mining*, pages 146–151, 1996.
- [35] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, 1:259–289, 1997.
- [36] J. McHugh. The 1998 Lincoln Laboratory IDS Evaluation – A Critique. In *3th Workshop on Recent Advances in Intrusion Detection (RAID)*, LNCS, pages 145–161. Springer Verlag, 2000.
- [37] R. S. Michalski and R. E. Stepp. Automated Construction of Classifications: Conceptual Clustering Versus Numerical Taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(4):396–410, 1983.
- [38] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks*, 31(23–24):2435–2463, 1999.
- [39] L. Pitt and R. E. Reinke. Criteria for Polynomial Time (Conceptual) Clustering. *Machine Learning*, 2(4):371–396, 1987.
- [40] 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.
- [41] L. Talavera and J. Béjar. Generality-Based Conceptual Clustering with Probabilistic Concepts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):196–206, 2001.
- [42] A. Valdes and K. Skinner. Probabilistic Alert Correlation. In *4th Workshop on Recent Advances in Intrusion Detection (RAID)*, LNCS, pages 54–68. Springer Verlag, 2001.