A Framework for the Application of Association Rule Mining in Large Intrusion Detection Infrastructures

James J. Treinen[1] and Ramakrishna Thurimella[2]

[1] IBM Global Services, Boulder, CO 80301, USA
jamestr@us.ibm.com
[2] University of Denver, Denver, CO 80208, USA
ramki@cs.du.edu

Abstract. The high number of false positive alarms that are generated in large intrusion detection infrastructures makes it difficult for operations staff to separate false alerts from real attacks. One means of reducing this problem is the use of meta alarms, or rules, which identify known attack patterns in alarm streams. The obvious risk with this approach is that the rule base may not be complete with respect to every true attack profile, especially those which are new. Currently, new rules are discovered manually, a process which is both costly and error prone. We present a novel approach using association rule mining to shorten the time that elapses from the appearance of a new attack profile in the data to its definition as a rule in the production monitoring infrastructure.

Keywords: Association Rules, Data Mining, Intrusion Detection, Graph Algorithms.

1 Introduction

Attempts to compromise networked computing resources generally consist of multiple steps. The first of these is the reconnaissance phase, consisting of the identification of target operating systems, port scanning, and vulnerability enumeration. This is followed by the exploitation of the weaknesses discovered during the initial intelligence gathering process. A successful attack often ends with the installation of back door channels so that the attacker can easily gain access to the system in the future [29].

If an intrusion detection infrastructure is in use at the victim network during this process, each action by the attacker has the potential to raise an alarm, alerting the security staff to the presence of malicious activity in the network. Generally speaking, intrusion detection sensors do not have the ability to aggregate the alarms for the discrete activities into an end-to-end attack profile. Given that an alarm is raised for each perceived malicious action, the typical intrusion detection sensor can generate many thousands of alarms per day. Unfortunately, the vast majority of these alarms are false positives [20], and the task of separating the real attacks from false alarms quickly becomes daunting.

D. Zamboni and C. Kruegel (Eds.): RAID 2006, LNCS 4219, pp. 1–18, 2006.

As noted by Lippmann, et al. in [26], the deployment of an inaccurate Intrusion Detection Sensor (IDS) can have undesirable effects in addition to simply missing certain types of attacks. The first of these is the potential to reduce the level of vigilant monitoring by security operations staff, due to the false sense of security provided by the IDS. Secondly, using operations staff to examine all of the alarms produced in a day can make the deployment of a typical IDS system extremely expensive in terms of support and labor costs. These issues are further compounded in large monitoring infrastructures where the number of managed sensors can easily reach into the thousands, generating millions of alerts per day.

The context for our experiments is that of a *large* Managed Security Service Provider (MSSP). Our experiments were conducted on a production data set that was generated by roughly 1000 IDS sensors. The sensor technologies used to generate the data set represented multiple vendors and versions of their software, and were installed across 135 distinct customer networks. The alarm logs generated by the sensors were consolidated at a Security Operations Center (SOC) which used a third party Enterprise Security Manager (ESM) with the ability to monitor the incoming alarm stream and match the alarms against a predefined set of meta rules. It is these meta rules which the operations staff use to detect intrusions across the networks they monitor. Similar to signature based intrusion detection sensors, the ESM uses pattern matching to detect predefined patterns in the incoming alarm streams. If the base alarms arriving at the ESM consolidation point match a predefined attack rule in the monitoring engine, a meta alarm is triggered and displayed on the operations staff's console for inspection.

Because new vulnerabilities are discovered every day, new alarm signatures are continuously installed on the intrusion detection sensors. This highly dynamic environment produces a genuine challenge in terms of keeping the rule base in the ESM current. Our framework provides a means of reducing the amount of labor required to keep the rules current in the ESM, while at the same time significantly reducing the amount of time which elapses from the appearance of a new attack profile in the data to installation of the corresponding rule in the production monitoring environment.

The time from the appearance of new attack profiles to the time when new rules describing them are implemented is *critical*. Any delay in updating the rule base could result in potentially undetected attacks. The amount of manual inspection currently required to discover new rules makes staffing to meet these time demands very expensive. We have found that using our framework to automate this task drastically decreases the amount of manual inspection required. This in turn has the net effect of decreasing the time from discovery to implementation as well as decreasing the over all cost of maintenance.

The concept of association rule mining for intrusion detection was introduced by Lee, et al. in [22], and is extended in [6,24,27]. Their approach is to use the rules returned by the association rule algorithm to prove that causal relationships exist between a user, and the type of entries that are logged in the audit data as a result of their actions on the system. Our research has shown that in the same manner that [22,24] were able to demonstrate the existence of causal relationships between users and the entries logged in system audit data as a result of their actions, it is possible to show causal relationships between an attacker and the combination of alarms which are generated in intrusion detection logs as a result of their behavior in a network. We were then able to use the patterns which were discovered using our data mining technique to configure new rules for the ESM system in a rapid and economical way. As a means of demonstrating this, we include examples of attack activity which answer the following questions:

1. What techniques did the attacker employ?
2. How were these techniques manifested as patterns in the IDS alarm logs?
3. Was our framework able to detect these patterns?
4. How did the discovered patterns result in a new rule in the ESM?

As with all data mining solutions, much up-front work must be done adjusting the parameters for the algorithm so that optimal results are obtained. There is no silver bullet configuration, and it is noted throughout the literature that when using association rule mining, the features which are chosen for examination are critical to the success of the algorithm [24,30].

The remainder of this paper is organized as follows. Related work is discussed in Section 2. Section 3 provides an overview of the experimental environment, a brief description of data mining terminology, and a discussion of representing alarms as directed graphs. Section 4 defines our approach, including a novel alarm filtering technique. Section 5 describes our results, and provides example rules which were generated using our framework. Section 6 presents concluding remarks.

2 Related Work

Many data mining techniques have been applied to intrusion detection. The vast majority of the research has concentrated on mining various types of system audit data, or raw network traffic in order to build more accurate IDS devices [6,13,22,23,24,25,30,33,34,35].

The use of data mining has also been employed to examine alarm logs, specifically using cluster analysis to classify alarms into attack and benign categories [20,24] and to perform root cause analysis regarding the cause of false alarms in [17,18,20,21]. The results obtained using cluster analysis can vary widely depending on which algorithm and distance measure is used. These issues are discussed at length in [10,14,20,22,24,30,33,37].

In order to be truly effective, the use of data mining techniques must be one step in an over all Knowledge Discovery in Databases (KDD) process. This case is made repeatedly in the literature, e.g. [30] who use cluster analysis solely as the initial step in their data exploration. It is reiterated in [17,18,20,21] that although the research tends to focus on the mining algorithm employed, it is only one step in the overall KDD process. They also note that without all of these steps, data mining runs a high risk of finding meaningless or uninteresting patterns. It is for this reason that [37] propose their end-to-end KDD architecture. Julisch outlines the basic KDD steps as follows in [18], as condensed from their original definition in [9] :

1. Understand the application domain
2. Data integration and selection
3. Data mining
4. Pattern evaluation
5. Knowledge presentation

A similar outline is made in [30], who also note that once a group of domain experts is consulted, the entire process should be automated.

3 Preliminaries

3.1 Experimental Environment

Figure 1 describes our data mining architecture. As the alarms arrive at the SOC, they are stored temporarily in a database on the monitoring engine. From this database we extracted the set of all alarms generated in a single day for all networks and loaded them into a data warehouse. It is on this warehouse that we executed the data mining algorithms with the goal of generating new monitoring rules for installation in the ESM.

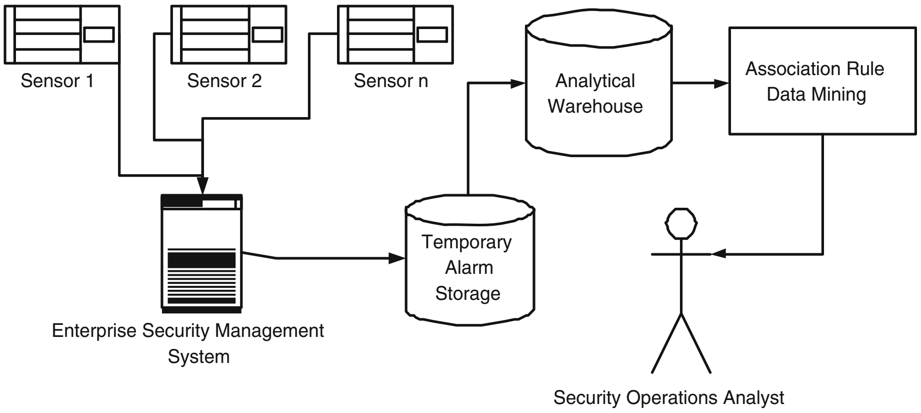
Sensor 1
Sensor 2
Sensor n
Analytical Warehouse
Association Rule Data Mining
Temporary Alarm Storage
Enterprise Security Management System
Security Operations Analyst

Fig. 1. The Association Rules Data Mining Architecture

3.2 Data Mining Terminology

In our analysis, we employ the use of association rule mining [1]. Because the field of data mining is very mature, rather than focusing on improving existing rule algorithms, we make use of the functionality that is available in DB2 Intelligent Miner for Modeling v8.2, which provides a fast algorithm for finding association rules. The main goal of association rule mining is to locate non-obvious interrelationships between members of a large data set [16]. The goal of our analysis is to find associations between the various attack signatures and IP addresses which constitute true attacks on the network, and capture them as rules in the ESM rule engine so that the SOC can easily detect future instances of the attack. The association rules algorithm generates rules in the following form, as well as some statistics which describe their strength and quality.

$$[x][y] \rightarrow [z]$$

$$Support = 50$$

$$Confidence = 80$$

This rule indicates that a relationship exists between the items x, y and z. Specifically, the rule states that whenever x and y were present in a given grouping, known as a transaction, then z was present as well. The Support value states that this specific grouping of three items represents 50 percent of the transactions which were examined. The Confidence value states that 80 percent of the time that the items x and y were found together, the item z was also found [16].

Formally, let $I = \{i_1, i_2, ..., i_n\}$ be a set of items. Given a set of transactions D, where each transaction is defined as a set of items $T \subseteq I$, a transaction T contains X if $X \subseteq T$. An association rule is an implication $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. The association rule $X \Rightarrow Y$ holds in the transaction set D with a Confidence c if c percent of transactions in D which contain X also contain Y. The association rule $X \Rightarrow Y$ has a Support value s in the transaction set D if s percent of the transactions in D contain $X \cup Y$ [1].

In our results, the Support values are typically less than 5 percent. This is due to the fact that thousands of signatures exist in the monitoring infrastructure, and generally the rules which are discovered cover only a small percentage of the total signature set for a given day.

3.3 Modeling Alarms as Directed Graphs

In order to facilitate a novel technique for filtering the number of alarms which must be analyzed during the mining process, we generated a directed graph which modeled the alarms to be examined. Each entry in the data warehouse included both the source IP address and destination IP address for which the alarm was raised. We deduced the direction of each potential attack from this information. We then generated a directed graph $G = (V, E)$ such that each IP address was represented as a vertex in the graph, and each edge was represented

Table 1. Typical Intrusion Detection Alarms

Network ID	Source IP	Destination IP	Signature
Network A	10.0.0.1	10.0.0.4	Signature 1
Network A	10.0.0.2	10.0.0.4	Signature 1
Network A	10.0.0.3	10.0.0.4	Signature 2
Network A	10.0.0.5	10.0.0.7	Signature 2
Network A	10.0.0.6	10.0.0.7	Signature 2
Network A	10.0.0.7	10.0.0.8	Signature 2
Network A	10.0.0.9	10.0.0.13	Signature 3
Network A	10.0.0.10	10.0.0.13	Signature 4
Network A	10.0.0.11	10.0.0.13	Signature 5
Network A	10.0.0.12	10.0.0.13	Signature 6

by a detected alarm. The edge was drawn from the source IP address toward the destination IP address, corresponding to the direction of the alarm.

The results are such that the IDS alarms which are shown in Table 1 are modeled as the directed graph shown in Figure 2.

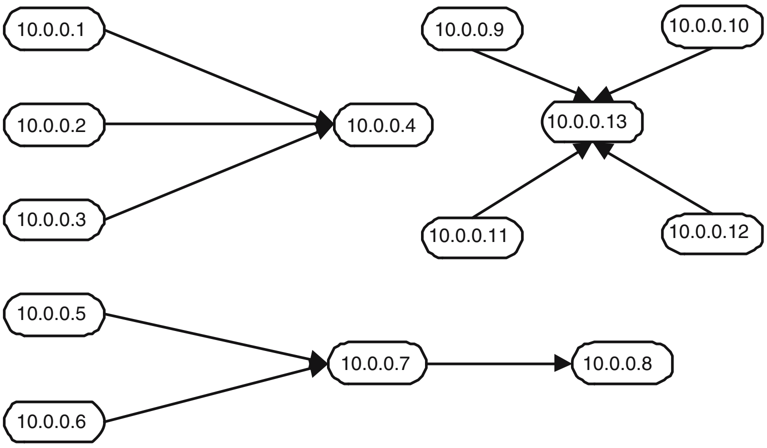

Fig. 2. Intrusion Detection Alarms as a Directed Graph With Three Connected Components

3.4 Data Set Reduction Using the Connected-Component Algorithm

The number of alarms produced in large intrusion detection environments can easily be on the order of millions of rows per day. We have observed raw event counts approaching 10 million events per day. We knew that most of these alarms were false positives, however it was not possible to label precisely which alarms were of genuine concern [17,18,20,21]. Because of the large volumes of data that required analysis, it was beneficial from a performance perspective to trim away any data that we knew to be irrelevant before starting the mining activities. In order to facilitate this, we represented the alarm logs as directed graphs, which allowed us to employ the use of graph algorithms to limit the scope of our inquiry. This process was only possible if we had a priori knowledge of a signature for which we wished to discover new rules.

When considering the problem of finding rules which exist between distinct signature and IP address combinations, it was important to note that there were alarms in the overall data set that could not be related to one another. For example, while examining one set of alarms, if we knew that another set of alarms could not be related to it, we removed the second set from consideration.

Drawing on our earlier discussion of alarm logs as directed graphs, we could translate the set of alarms in Table 1 into the directed graph shown in Figure 2, which displays three easily identified connected components. Limiting our mining activity solely to alarms in the same connected component allowed us to explore only relationships between alarms which could legitimately exist. A complication arose in the case of slave nodes which were controlled by a master who was not represented in the graph. We designated this scenario to be out of scope for our experiments.

When attempting to discover rules for a specific signature, a natural question arises as to why we did not simply limit the alarms to those which were produced by a source IP address that also produced the signature undergoing analysis. Reducing the data set in this manner was possible if we were interested only in the detection of single-source attacks for a specific signature. We would then examine the set of all alarms generated by a source IP address which triggered the signature in question. However, trimming the data in this way would severely limit any further analysis that we wished to perform on the set of alarms. By carrying the other relevant alarms from the connected component, we have access to a greater number of signatures and IP addresses for analysis. We also preserved the ability to perform further analysis by grouping on fields other than the source IP address if we found that a more extensive exploration of the data was warranted.

For example, consider a multi-stage attack consisting of a reconnaissance event which discovered a vulnerability on the target and exploited it in a way that in turn attacked a third system. Table 2 lists alarms which would constitute such a scenario. These alarms are shown graphically in Figure 3. The reconnaissance and subsequent exploit occur between 10.0.0.5 and 10.0.0.7. A successful compromise of 10.0.0.7 by 10.0.0.5 is then used to further attack 10.0.0.8.

If we had specified the reconnaissance signature as the input to the mining process and trimmed away all IP addresses which did not trigger that signature, we would have missed the second half of the attack. As such, limiting the alarms that we examine only to those which occur in the same connected component

Table 2. Intrusion Detection Alarms for a Multi-Stage Attack

Network ID	Source IP	Destination IP	Signature
Network A	10.0.0.5	10.0.0.7	Reconnaissance
Network A	10.0.0.5	10.0.0.7	Exploit 1
Network A	10.0.0.7	10.0.0.8	Exploit 2
Network A	10.0.0.6	10.0.0.7	False Alarm

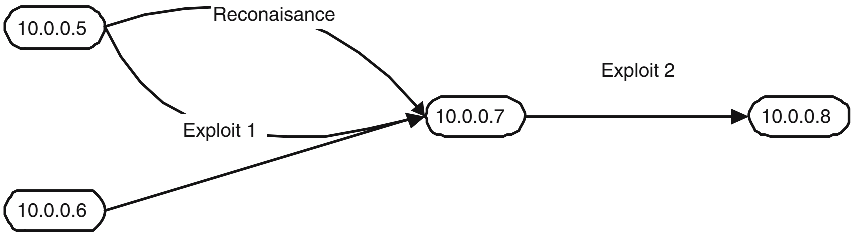

Fig. 3. A Multi-Stage Attack Scenario

provided the appropriate balance of efficiency without interfering with our ability to perform complex analysis of the relevant data. On average, we were able to reduce the amount of data that required analysis by 30 percent. However, our ability to reduce the amount of data we inspected was sometimes diminished in the case of graphs which were nearly fully connected. Because this type of graph produced one large connected component comprised of the majority of the alarms, the amount of data which we were able to trim away prior to executing the association rule algorithm was in some cases reduced to less than 5 percent.

4 The Approach

Our experiments were conducted on the set of alarm logs generated by network-based intrusion detection sensors over a 24-hour period for 135 distinct production networks. The alarms were loaded into a data warehouse specifically engineered to facilitate efficient off-line analysis of intrusion detection alarms using association rule mining techniques. We repeated the experiments on a daily basis for 30 days.

4.1 Generation of Signature Specific Rules

Our first set of experiments were conducted with the goal of discovering new rules for a signature which was thought to be exhibiting suspicious behavior. We accomplished this by first selecting the set of connected components in which the suspected signature was present, and discarding all alarms that were not members of these these connected components. Once we had filtered the data in this way, we then executed the association rule algorithm to see if any rules for this signature were generated. Algorithm 1 describes this technique.

Algorithm 1. Find-Signature-Rules(G,s)

```
Require: $G = (V, E)$, a directed graph of IDS Alarms, $s$ a subject signature
$C$ ←Connected-Components($G$)
for all $C' \in C$ do
    if $s \in C'$ then
        copy all alarms in $C'$ to $T$
    end if
end for
R ← Association-Rules(T)
Return R
```

Of the scenarios that we discuss, signature specific rule generation experienced the lowest occurrence of success. One of the reasons for this was that rather than being identified algorithmically, the signature examined was generally chosen by a human operator who was simply curious as to whether any correlations involving this signature were hidden in the data. The subject signature was most often chosen for analysis based on an abnormally high volume of that signature over a specific time period, or its appearance as a new signature where it had not been previously detected. These scenarios might occur due to the introduction of a previously unforeseen attack scenario into a network, or simply because of software updates on the sensors themselves.

Over the course of our experiments, we were able to successfully generate rules for specific signatures roughly 10 percent of the time. However, given that data mining always requires manual evaluation and exploration of its results, we still believe this to be an effective tool for operations staff to have at their disposal. The skill of the user conducting the analysis had a great impact on the quality of the results, which is consistent with the views expressed in [18,30,37]. We found that as the user's experience with the technique grew, their ability to choose signatures for which rules would be generated grew as well.

Approximately half of the experiments uncovered patterns involving signatures other than those which were the original subject of our exploration. In some cases, the rules algorithm would produce more than 100 rules for a single run. This appeared at first glance to be overwhelming, however, the rules which exhibited very strong Confidence values floated to the top on their own merits, and were easily identifiable.

If we were unable to safely remove significant numbers of rows from consideration by filtering on connected component, the time required for the mining algorithm to generate results grew rapidly. A side effect produced by this complication was the generation of a very large number of rules by the algorithm.

In some cases we observed rule counts as high as 8000 for a single network's data. This number of rules on its own is of limited value, as it does not solve the problem of limiting the amount of data which must be examined manually by operational staff. However, the vast majority of the time, the count of rules for a single network on a single day was below 100. When a spike occurred, we found it to be indicative of significant phenomena in the network being monitored. We discuss these findings in a later section of this paper.

A useful means of tuning the number of rules returned by the association rule algorithm was to adjust the minimum values for the Support and Confidence parameters for the mining algorithm, which had the net effect of limiting the number of rules which were produced. The obvious risk in limiting the rules to those with a very high Support value is that any signature which generated low volumes when compared to the volume of alarms in a single day will simply be lost. It is for this reason that we generally left the Support value at a relatively low setting, while enforcing a constraint of high Confidence values on the result set. By doing this, we were able to limit the results to rules which were found to hold the majority of the time.

4.2 Generation of Single Source Rules

Our framework generated the greatest number of high Confidence rules when we grouped the transactions in the database by source IP address. When using this approach it was not necessary to limit the rows we examined using the connected components algorithm, though it was beneficial from a performance perspective if we knew the signature for which we wished to perform the analysis, and used this information to limit the data set before executing the association rules algorithm. When performing single-source analysis, we also found that setting the minimum values for the Support and Confidence parameters to 0 was useful. Intuitively, providing these low values for the Support and Confidence parameters would produce an overwhelming number of rules. However, over the course of our experiments we found that on average, a single source IP address will trigger less than two signatures in any 24 hour period. Because we were looking for correlations between signatures which were generated by a single source, it was obvious that no rules would be generated for these IP addresses. Because of this, 87 percent of our single-source experiments generated zero rules for a given day's data.

5 Efficacy of the Framework

The Confidence value given for a new rule was critical in determining how effective the rule would be in the production monitoring environment. On average, 66 percent of the rules we produced had a confidence value of 100, and rules with a Confidence value over 80 were produced 86 percent of the time. We found that certain types of attack activity generated very high volumes of rules with a Confidence value of 100 percent. While these rules were not false positives, they skewed the statistics. Disregarding them, the percentage of rules with a Confidence value above 80 percent was 63 and the percentage of rules with Confidence values of 100 was 43.

When applying our technique, we were able to detect attacks that did not trigger meta alarms on the operational console. In one case, we were able to detect an attack on a day where the ESM system received 1,543,997 alarms. The detected attack was comprised of only 6 alarms, and did not result in a meta alarm firing on the operational console. This is of great consequence as this attack would otherwise have been lost in the noise of the 1.5 million other alarms that flowed through the infrastructure that day. It was then possible to code a rule describing this scenario into the ESM system so that future instances would be detected.

5.1 Rule Examples

1. **Web Server Attack**:
   Our first example does not indicate the reconnaissance approach which was used to determine the list of web servers that underwent the detected attack, as no reconnaissance signature was present in the alarm log that generated this rule. It is possible that the technique used did not trigger an alarm, or that the reconnaissance phase of the attack was carried out many days in advance in an attempt to prevent detection. The alarms which were present in the database which generated this rule are indicated in Table 3. The IP addresses have been sanitized to prevent identification of the customer network for which the analysis was performed.

Table 3. IDS Alarms for a Multi-Stage Web Server Attack

Network ID	Source IP	Destination IP	Signature
Network A	24.9.61.170	192.168.2.4	AWStats configdir Command Exec
Network A	24.9.61.170	192.168.2.5	XMLRPC PHP Command Execution
Network B	24.9.61.170	192.168.2.16	AWStats configdir Command Exec
Network B	24.9.61.170	192.168.2.17	XMLRPC PHP Command Execution
...	...	...	...

Rule for Multi-Stage Web Server Attack

[AWStats configdir Command Exec]⇒ [XMLRPC PHP Command Execution]
Confidence = 100
Support = 3.45

This rule involves two signatures generated by an attacker who was attempting to locate a vulnerability to exploit on a web server. The first stage of the attack appeared in the alarm logs as multiple instances of the signature, [AWStats configdir Command Exec], which fired as the attacker attempted to execute an unauthorized command using the *configdir* variable of the *awstats.pl* CGI script. The second phase of the attack appeared in the alarm logs as the signature, [XML RPC PHP command Execution], which was triggered as attempts were made to exploit an XMLRPC vulnerability via SQL injection [7].

Our framework was able to detect this pattern by grouping alarms by the source IP address, and looking for repetitive combinations. When grouped together, these two signatures, when triggered by the same source IP address, are indicative of an attacker who attempted multiple exploits before either compromising the target server, or moving to another victim. Further, because these were the only rules generated for this network on the day in question, we can be almost certain that the activity was legitimate attack activity and not part of an automated vulnerability scan. We observed this same pattern on two distinct monitored networks on the same day, which indicates further that the detected activity was a real attack.

2. **Reconnaissance Attack**:
   This rule was generated using data from a network where an attacker was attempting to locate vulnerable file shares to attack. A pattern was found in the alarm logs for this customer which described a frequently occurring pattern of two TCP-based reconnaissance signatures followed by a LANMan share enumeration, which is a common means of locating vulnerable file shares for future exploitation.

Rule for Reconnaissance Activity
[TCP Port Scan][TCP Probe HTTP]⇒ [LANMan share enum]
Confidence = 66.66
Support = 1.7

3. **Scanning Activity**:
   Rules of this type frequently materialized when a network experienced a series of exploit and probing attempts. This type of brute force attack results in a set of rules where the actual attacks span a wide range of signatures, and are associated with a reconnaissance event in the form of a TCP port scan. The goal of the attacker in these situations was to discover open vulnerabilities on a system to be exploited in future attacks. A special case which had to be considered when searching for these types of attacks was whether or not the scanning activity was legitimate traffic generated as part of a policy verification procedure. This was most commonly caused by the use of an automated scanning appliance under the control of the network security staff as a means of ensuring that the hosts under their control had been updated with the most recent security patches.

Rules for Scanning Activity

[RPC Race Condition Exploitation]⇒ [TCP SYN Port Sweep]
Confidence = 51
Support = 1.8

[SQL Query in HTTP Request]⇒ [TCP SYN Port Sweep]
Confidence = 43
Support = 1.7

[FTP RealPath Buffer Overflow]⇒ [TCP SYN Port Sweep]
Confidence = 100
Support = 0.2

4. **Worm Related Rules**:
   Worms propagate by exploiting vulnerabilities to gain control of a victim server, subsequently scanning the network for other vulnerable machines, as to guarantee rapid and widespread infection before a patch can be implemented. The following example rules define a multi-stage worm attack which took advantage of file sharing vulnerabilities which exist in a widely deployed operating system. The first rule correlates an overflow exploit of an SMB vulnerability, and subsequent access. The existence of the [ICMP L3 Retriever Ping]alert is indicative of Black/Nyxem worm activity.

Rule for Black/Nyxem Worm

[NETBIOS SMB–DS IPC unicode share access][ICMP L3retriever Ping]⇒
[NETBIOS SMB–DS Session Setup And request unicode username overflow attempt]
Confidence = 100
Support = 41

Another example of worm related patterns which we detected describes correlations relevant to the SQL Slammer worm which ravaged the Internet in 2002, and is still frequently detected. This worm exploited a buffer overflow vulnerability to execute malicious code and install itself on the victim machine, after which it scanned for other hosts to which it could propagate. Two mature signatures exist for this worm in our monitoring environment. The first signature describes the initial overflow attempt, followed by a propagation attempt. Our framework was able to determine that a strong correlation exists between these two signatures. Using this information, we can then code a new rule into the ESM which watches for this type of pattern, and raises a meta alarm when it is detected.

While worms such as SQL Slammer are well known, we have shown that our method can consistently detect the patterns which are generated in the alarm stream by their propagation. Based on this, we feel that the techniques presented here can be applied to detect future instances of emerging worm traffic, *independent* of whether the intrusion detection sensors supply

Rule for SQL Slammer Worm

[MS-SQL version overflow attempt]⇒ [MS-SQL Worm Propagation attempt]
Confidence = 100
Support = 35

worm specific signatures, or if the newly emerging worm manifests itself as a combination of existing signatures.

5.2 Identification of High Risk Networks

As mentioned previously, we found that on average, 87 percent of our experiments generated no rules for a given network over a 24-hour period. This translates to the total number of networks for which rules were produced in a single 24-hour period being 17 out of 135. Figure 4 shows a typical count of rules generated per monitored network on a logarithmic scale. In this case, 19 out of the 135 monitored networks produced rules. Of these 19 networks, 12 produced 10 or less rules for that particular day, while one network produced 117 and one produced 2295. Graphing these counts highlights the anomalous networks, which provides a useful tool for operational personnel to see which networks require immediate attention.

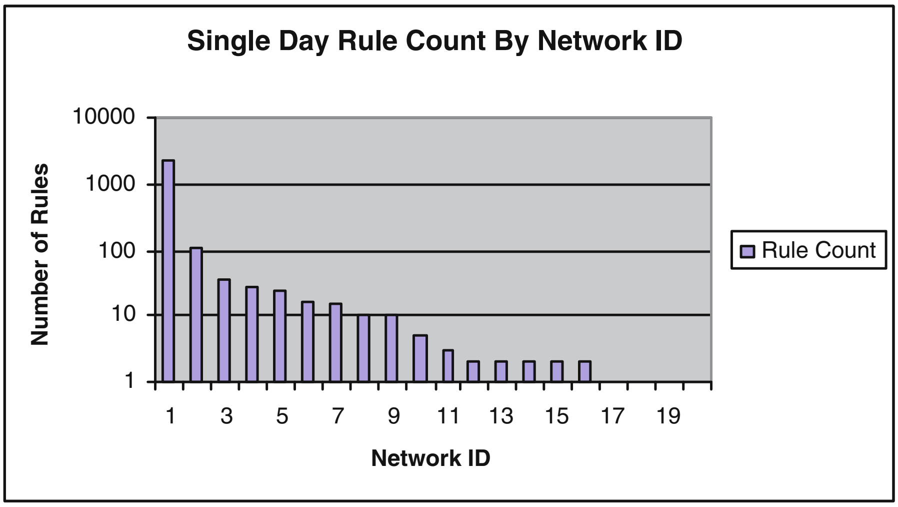

Fig. 4. Anomalous Network Activity as Shown by a Count of Rules Produced Per Network for a Selected Day

5.3 Facilitation of Sensor Tuning and Root Cause Analysis

Much in the same way that Julisch describes the use of cluster analysis for the identification of the root cause of false positive alarms in [18,20,21], we have found that we can facilitate the determination of root causes of certain alarms using our data mining framework.

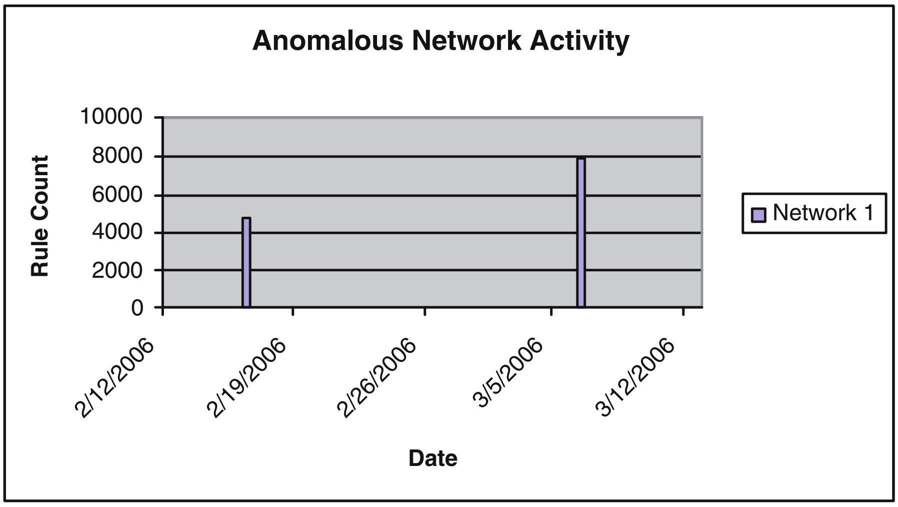

Fig. 5. Spikes Indicating Anomalous Activity For a Single Network

Figure 5 shows a 30-day trend of rule volumes broken out by day for a selected network. The spikes represent the generation of 4854 and 7926 rules on two separate days, respectively. When we inspected these rules, they appeared to describe a denial of service attack on an electronic commerce site. The rules covered 47 percent of the alarms which were generated on the corresponding days, and were comprised of a flood of Half Open SYN signatures, coupled with various other attack signatures. After some investigation, it was discovered that the actual cause of the alarms was a misconfigured IP route between a web application server and an LDAP server. Every time that a user attempted to authenticate to the application, the request was lost due to the IP stack's inability to complete the TCP handshake. The intrusion detection sensors interpreted this as a spoofed source IP address, which resulted in a flood of the corresponding alarms to the security operations center. By fixing this IP routing problem, the corresponding reduction in alarms would provide increased fidelity in the alarm stream for that network as well as increased chances that legitimate attack traffic would not be overlooked.

6 Conclusion

We have outlined a novel framework for the application of association rule mining techniques on the millions of alarms which are received daily at large Managed Security Service Providers. As new attack strategies emerge, our framework is successful at discovering the associated patterns of alarms which occur as a result of the attacker's actions in the victim network. By highlighting these patterns, we reduce the time required for SOC personnel to implement meta rules which ensure the detection of future instances of emerging attacks.

Our framework provides a reliable means of closing the time gap between the appearance of new attack profiles in the alarm logs and the configuration of rules in the ESM. We accomplished this while reducing the human-error factor, as well as the costs associated with manually inspecting large alarm logs.

In addition to the ability to discover new rules for the ESM, we have also shown that our framework can be used to flag suspicious network activity for in-depth analysis by operations staff in an off-line environment. The use of our framework can detect a variety of classes of attacks which may have been lost in the large data volumes due to processing time constraints in the on-line monitoring system.

Acknowledgments

We would like to extend our gratitude to the Security Intelligence and Managed Security Service Delivery teams at IBM for their assistance in carrying out our experiments.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules Between Sets of Items in Large Databases. Proceedings of the ACM SIGMOD Conference on Management of Data (1993) 207-216
2. Ali, K., Manganaris, S., Srikant, R.: Partial Classification Using Association Rules. Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (1997) 115-118
3. Apap, F., Honig, A., Hershkop, S., Eskin, E., Stolfo, S.: Detecting Malicious Software by Monitoring Anomalous Windows Registry Accesses. Proceedings of Recent Advances in Intrusion Detection, 5th International Symposium (2002) 36-53
4. Arcsight Corporation.: Arcsight ESM Product Brief. http://www.arcsight.com/collateral/ArcSight_ESM_brochure.pdf (2005)
5. Arcsight Corporation.: Arcsight Pattern Discovery Product Brief. http://www.arcsight.com/collateral/ArcSight_Pattern_Discovery.pdf (2005)
6. Barbara, D., Couto, J., Jajodia, S., Wu, N.: ADAM: A Testbed for Exploring the Use of Data Mining in Intrusion Detection. SIGMOD Record Volume 30 Number 4 (2001) 15-24
7. Cisco Systems. Network Security Database. http://www.cisco.com/cgi-bin/front.x/csec/idsAllList.pl (2005)
8. Debar, H., Wespi, A.: Aggregation and Correlation of Intrusion-Detection Alerts. Proceedings of Recent Advances in Intrusion Detection, 4th International Symposium (2001) 85-103
9. Fayyad, U. Piatetsky-Shapiro, G. Smyth, P.: The KDD Process for Extracting Useful Knowledge From Volumes of Data . Communications of the ACM (1996) 27-34

10. Guan, Y., Ghorbani, A., Belacel, N.: Y-Means : A Clustering Method for Intrusion Detection. Proceedings of Canadian Conference on Electrical and Computer Engineering (2003)
11. Han, J., Cai, Y., Cercone, N.: Knowledge Discovery in Databases: An Attribute-Oriented Approach. Proceedings of the 18th International Conference on Very Large Data Bases (1992) 547-559
12. Han, J., Cai, Y., Cercone, N.: Data-Driven Discovery of Quantitative Rules in Relational Databases. IEEE Transactions on Knowledge and Data Engineering, Volume 5 (1993) 29-40
13. Honig, A., Howard, A., Eskin, E., Stolfo, S.: Adaptive Model Generation : An Architecture for the Deployment of Data Mining-based Intrusion Detection Systems. Applications of Data Mining in Computer Security, Barbara, D., Sushil, J., eds. Boston : Kluwer Academic Publishers (2002) 153-194
14. Hosel, V., Walcher, S.: Clustering Techniques : A Brief Survey. http://ibb.gsf.de/reports/2001/walcher.ps (2000)
15. IBM Corporation : DB2 Intelligent Miner for Modeling. New York (2005)
16. IBM Corporation : IBM DB2 Intelligent Miner Modeling Administration and Programming Guide v8.2. Second Edition. New York (2004)
17. Julisch, K.: Mining Alarm Clusters to Improve Alarm Handling Efficiency. Proceedings of the 17th Annual Computer Security Applications Conference (2001) 12-21
18. Julisch, K.: Data Mining for Intrusion Detection A Critical Review. Applications of Data Mining in Computer Security, Barbara, D., Sushil, J., eds. Boston : Kluwer Academic Publishers (2002) 33-62
19. Julisch, K., Dacier, M.: Mining Intrusion Detection Alarms for Actionable Knowledge. Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2002) 366-375
20. Julisch, K. Clustering Intrusion Detection Alarms to Support Root Cause Analysis. ACM Transactions on Information and System Security, Volume 6, Number 4 (2003) 443-471
21. Julisch, K. Using Root Cause Analysis to Handle Intrusion Detection Alarms. PhD Thesis. Universität Dortmund (2003)
22. Lee, W., Stolfo, S.: Data Mining Approaches for Intrusion Detection. Proceedings of the 7th USENIX Security Symposium (1998) 79-94
23. Lee, W., Stolfo, W., Mok, K.: Mining Audit Data to Build Intrusion Detection Models. Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (1998) 66-72
24. Lee, W. Stolfo, S. Kui, M.: A Data Mining Framework for Building Intrusion Detection Models. IEEE Symposium on Security and Privacy (1999) 120-132
25. Lee, W., Stolfo, S., Chan, P., Eskin, E., Fan, W., Miller, M., Hershkop, S., Zhang, J.: Real Time Data Mining-based Intrusion Detection. Proceedings of the 2nd DARPA Information Survivability Conference and Exposition (2001)
26. Lippmann, R., Haines, J., Fried, D., Korba, J., Das, K.: The 1999 DARPA Off-Line Intrusion Detection Evaluation. Computer Networks, Volume 34 (2000) 579-595
27. Manganaris, S., Christensen, M., Zerkle, D., Hermiz, K.: A Data Mining Analysis of RTID Alarms. Proceedings of Recent Advances in Intrusion Detection, Second International Workshop (1999)
28. Mchugh, J.: Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. ACM Transactions on Information and System Security, Volume 3, Number 4 (2000) 262-294

29. McLure, S., Scambray, J., Kurtz, G.: Hacking Exposed Fifth Edition: Network Security Secrets & Solutions : McGraw-Hill/Osborne (2005)
30. Nauta, K., Lieble, F.: Offline Network Intrusion Detection: Mining TCPDUMP Data to Identify Suspicious Activity. Proceedings of the AFCEA Federal Database Colloquium (1999)
31. Ning, P., Cui, Y., Reeves, D., Xu, D.: Techniques and Tools for Analyzing Intrusion Alerts. ACM Transaction on Information and System Security. Volume 7, No. 2 (2004) 274-318
32. Noel, S., Wijesekera, D., Youman, C.: Modern Intrusion Detection, Data Mining, and Degrees of Attack Guilt. Applications of Data Mining in Computer Security, Barbara, D., Sushil, J., eds. Boston : Kluwer Academic Publishers (2002) 1-31
33. Portnoy, L., Eskin, E., Stolfo, S.: Intrusion Det
34. Schultz, M., Eskin, E., Zadok, E., Stolfo, S.: Data Mining Methods for Detection of New Malicious Executables. Proceedings of IEEE Symposium on Security and Privacy (2001)
35. Stolfo, S., Lee, W., Chan, P., Fan, W., Eskin, E.: Data Mining-based Intrusion Detectors: An Overview of the Columbia IDS Project. SIGMOD Record, Vol. 30, No. 4 (2001) 5-14
36. Valdes, A., Skinner, K.: Probabilistic Alert Correlation. Proceedings of Recent Advances in Intrusion Detection, Third International Workshop (2001) 54-68
37. Yang, D., Hu, C., Chen, Y.: A Framework of Cooperating Intrusion Detection Based on Clustering Analysis and Expert System. Proceedings of the 3rd international conference on Information Security (2004)