GENERATING SECURITY EVENT CORRELATION RULES THROUGH K-MEANS CLUSTERING

Nelson Uto, Helen Teixeira, Andre Blazko, Marcos Ferreira de Paula, Renata Cicilini Teixeira
CPqD Telecom & IT Solutions
Rod. Campinas-Mogi Mirim (SP-340) km 118.5 CEP 13086-902 Campinas-SP Brazil

Mamede Lima Marques
Universidade de Brasilia
Campus Universitario Darcy Ribeiro, CP 4422, CEP 70919-970 Brasilia-DF Brazil

Keywords: Network security, security event correlation, semi-automatic rule generation, data mining.

Abstract: Current implementations of intrusion detection systems (IDSs) have two drawbacks: 1) they normally generate far too many false positives, overloading human operators to such an extent that they can not respond effectively to the real alerts; 2) depending on the proportion of genuine attacks within the total network traffic, an IDS may never be effective. One approach to overcoming these obstacles is to correlate information from a wide variety of networks sensors, not just IDSs, in order to obtain a more complete picture on which to base decisions as to whether alerted events represent malicious activity or not. The challenge in such an analysis is the generation of the correlation rules that are to be used. At present, creating these rules is a time consuming manual task that requires expert knowledge. This work describes how data mining, specifically the k-means clustering technique, can be employed to assist in the semi-automatic generation of such correlation rules.

1 INTRODUCTION

Intrusion detection systems (IDSs) aim to detect malicious or suspicious activity against a system. Current IDS implementations suffer from the fact that they trigger many false positives (false alarms), inundating their human operators with so much information that they are not able to effectively respond to the real threats. Another problem with IDSs is that real attacks can often go unnoticed (false negatives), especially in the case of new types of attacks for which detection signatures have not yet been created. In fact, when discussing the drawbacks of IDSs, it is imperative to take into consideration that the process of detecting intrusions might be harder than previously thought (Axelsson, 1999). Based on a probabilistic model, Axelsson showed that the factor limiting the effectiveness of an IDS (i.e. the capability of detecting real attacks and raising few false alarms) is the false positive rate rather than the detection rate. The worrying issue of Axelsson's analysis was that if attacks are rare compared to valid behaviour, then one must work with an unrealistically low false positive rate in order to have an effective IDS.

One approach to improve the effectiveness of an IDS is to work on a higher abstraction level. The idea is to correlate alerts collected by several sensors (e.g. firewalls, IDSs, routers, etc.) placed in a network, in order to have more evidence on which to base a decision as to whether a detected activity should be labeled suspicious or not. The drawback of this technique is that it is not easy to manually write the correlation rules that are to be used to perform this task. Furthermore, expert knowledge is required for this.

Up until now, there have been few studies on mechanisms for automatically generating correlation rules. Examples include (Burns et al., 2000) and (Kreibich and Crowcroft, 2004) whose techniques and results are discussed in Section 3.2. In this paper, we present preliminary results from using k-means clustering for the semi-automatic generation of security event correlation rules. Our initial work has applied this technique only to alerts from IDSs to find correlation patterns. Our next step will be to apply k-means clustering to alerts generated by heterogenous sensors.

The remainder of this paper is organized as follows: Section 2 introduces concepts, data sources and some techniques used for security event correlation. Section 3 introduces the k-means data mining technique and discusses related work that has been performed to date in this area. Section 4 proposes how k-means clustering may be exploited for the semi-automatic generation of correlation rules and finally Section 5 presents the next steps that are to be performed in this analysis.

Uto N., Teixeira H., Blazko A., Ferreira de Paula M., Cicilini Teixeira R. and Lima Marques M. (2005).
GENERATING SECURITY EVENT CORRELATION RULES THROUGH K-MEANS CLUSTERING.
In *Proceedings of the Second International Conference on e-Business and Telecommunication Networks*, pages 376-381
DOI: 10.5220/0001417903760381
Copyright © SciTePress

2 SECURITY EVENT CORRELATION (SEC)

Security event correlation, also called security alert correlation, consists of relating alerts collected by several sensors in order to improve the reasoning used to decide whether an attack is taking place or not. Since each new sensor can potentially monitor additional information about the behaviour of a system, then by increasing the number of different sensors used to generate alerts, one can obtain a more complete picture of network activity on which to base an analysis. Such an analysis can assist in rejecting false positives generated by IDSs due to a lack of corroborating alerts.

In this section we introduce the nomenclature commonly employed to describe security events, the main data sources used by correlation algorithms and some examples of how correlation can be employed to improve the detection process.

2.1 Definitions

The following terms are commonly employed to describe network security correlation systems and will be used throughout the remainder of this paper:

- **Event** – a change in the state of a system.
- **Alert** – a message triggered by an entity (hardware or software) in response to the occurence of an event.
- **Attack** – a deliberate attempt to compromise the security of a system, normally violating the adopted security policy. It is worthwhile mentioning that an attack may be comprised of a single or multiple steps.
- **Data source** – a source of information that can be used by the correlation process.
- **Sensor** – a component responsible for monitoring system elements and raising alerts when preconfigured conditions are met.
- **Correlation rule** – a set of logical expressions that describe the relationship between alerts and the actions to be taken when the rule is satisfied. A rule should be specific enough as to allow the proper detection of the fingerprint of an attack whilst simultaneously flexible enough to recognize variations.

2.2 Data Sources

Sensors can monitor several components (data sources), be they software or hardware, and trigger alerts when certain conditions occur. As present, there exists no standardized format for generating alerts. Each sensor implementation specifies its own format, thus requiring an ad-hoc treatment for every sensor. In order to simplify the manipulation of event data, normalization can be performed prior to executing the correlation algorithm. One initiative in this direction is that of the IETF's Intrusion Detection Message Exchange Format (IDMEF) which defines a normalization schema for information generated by IDSs (Debar et al., 2005).

The following list of data sources (sensors) for network security event correlation is based upon (Yin et al., 2003) and it is not intended to be exhaustive:

- **Firewalls** – can be used to monitor the types of services requested and used, common external IP addresses accessing internal services, port scans and outbound connections.
- **Intrusion detection systems** – aim to detect malicious or suspicious activity against a system.
- **Operating systems** – can be configured to log login attempts, resource access, program usage and system rebooting.
- **Routers** – can log system errors, network and interface state changes and traffic violating access control lists.
- **SMTP** – can record the address of the sender and receiver, the date and time of the transmission and the message size.
- **SNMP** – SNMP agents can notify events that occur in monitored elements according to their respective management information base (MIB).
- **Web servers** – can log diagnostic information, processing errors, successful requests, and the input and output of server scripts.

2.3 Some Examples

The following examples aim to capture the essence of security event correlation, showing how it can be employed to lower the false positive rate whilst improving the detection rate.

Our first example is a simple one. Suppose that an intrusion detection system detects an IP packet whose payload can exploit a vulnerability in an IIS Web server and triggers an alert for the target (destination IP). If the target is actually running the Apache Web server, rather than the Microsoft one, then such an alert is clearly a false positive and should not be sent to the system administrator. This example illustrates how the use of more than one data source, in this case IDS data with an inventory of applications running on each destination IP, can contribute to formulating a better conclusion.

Our second example is related to the notification of new vulnerabilities. One of the functionalities to be included in the Open-Source Vulnerability Database

(OSVDB) project is the automation of vulnerability queries satisfying predetermined conditions. After being notified, the system administrator can look for the vulnerable machines and apply the proper patches or workarounds. However, often there might not be machines in the system affected by the vulnerability just found. A better approach would consist of letting the correlation module run a cross-check of the inventory information in order to trigger patch alerts only for susceptible machines.

Our third example is taken from (Jiang and Cybenko, 2004). Worms, such as *CodeRed* and *Nimda*, randomly scan IP addresses searching for potential targets. Since many of the IP addresses tested are not assigned to a network, the scanning process can generate a large volume of ICMP *host unreachable* messages. The worm propagation may also affect the network latency and the Border Gateway Protocol. By correlating information and releasing an alert signalling the occurrence of a worm only when all three events occur simultaneously, the possibility of emitting a false positive is greatly reduced. Another correlation approach could be to consider the volume of ICMP messages generated as a function of time and trigger an alert only if this measure follows a temporal pattern related to the stages of the worm lifecycle.

For a more complete example, we refer the reader to (Drew, 2003) who presents a more complex scenario where correlation can be used to detect attacks exploring vulnerabilities in CGI scripts.

2.4 Some Correlation Techniques

Several correlation techniques have been proposed in the literature so far. In this section we discuss three of them in order to illustrate how they work and to show that, essentially, they all rely on the same basic information, though expressed in different formats. We do not consider heuristic algorithms because they can not take advantage of the correlation rule generation mechanism we propose in this paper.

The approach presented in (Yemini et al., 1996) is based on coding techniques and explores the fact that, in general, a particular problem can be identified by a set of symptons. Thus, each problem (in our case, each attack) is associated to a **code** representing the induced symptons. The correlation process is divided into two phases: in the **selection phase**, one chooses a subset of the possible events to be used in the detection process. This subset, called the **codebook**, must be optimal in the sense that it is sufficient to distinguish between different types of attacks. In the **decoding phase**, the events in the codebook are monitored and analyzed in real-time to determine when known attacks are occuring. Put another way, the correlation can be seen as the analysis of causal relationships between events represented as codes.

(Ning et al., 2002) propose the use of prerequisites of intrusions as a means to correlate security alerts. A **prerequisite of intrusion** is the set of necessary conditions for an attack to be successful. For example, an attack against a vulnerability in the IIS Web server requires that the target machine runs the affected version of that software. In turn, the system state to be reached as a result of the attack is called the **(possible) consequence of the attack**. As we have already mentioned, an attack might be composed of several steps. In such cases, the consequence of any step (obviously excluding the last one) must be the prerequisite of the subsequent one. In this way, the triple (attributes, prerequisite, consequence) defines a **hyperalert type** which models the atomic actions an attacker must execute to complete the attack. Correlation then consists of finding instances of hyperalert types connected by consequence/prerequisite.

A third correlation technique is the application of chronicles to event correlation, as introduced by (Morin and Debar, 2003). **Chronicles** are temporal patterns that represent a possible evolution of the system being observed. To quote the authors: "chronicles provide a framework for modeling dynamic systems. They include an evolution monitoring mechanism to track changes in the modeled system. Recognition of chronicles is based on a formalism in which time is fundamental". A chronicle is composed of a set of predicates and temporal constraints and can be used to model attack scenarios. As events are being detected, the recognition process filters those chronicles whose predicates are violated. When a chronicle has all its assertions satisfied an alert is triggered.

Comparing the three approaches described above, we can see that a correlation algorithm is comprised of a knowledge base (used to describe attack scenarios) and a detection process. As such, one can say that all correlation techniques are essentially the same; their only difference is the format in which the correlation knowledge is represented. This is an important point as it implies that a mechanism for semi-automatically generating correlation rules may be independent of the correlation technique to be used. One just needs a way of mapping the derived rules to the specific knowledge representation employed by a particular algorithm. However, it must also be highlighted that certain temporal constraints cannot be represented by every algorithm. An example could be a constraint that requires a certain time interval (window) between two events.

3 DATA MINING

Over the last decade, data mining has emerged as a promising analytical technique to detect hidden

Table 1: Principal data mining techniques

Technique	Description
Classification	Prediction of the category to which a particular record belongs
Association rules	Determines the correlation and causality between sets of objects
Clustering	Separation of data into subsets that share common properties
Statistical analysis	Determines the likelihood of characteristics and associations in the selected data
Visualization	Presents a graphical summary of the data

knowledge (i.e. relationships and behaviours that can be described by rules, regularities, patterns and constraints) from within large volumes of data. It has been used in many fields and for various goals, such as market analysis, risk analysis and detecting unusual patterns that are often indicative of fraud or unusual behaviour (Han and Kamber, 2000).

Table 1 lists the principal data mining techniques. Classification is an example of *supervised* data mining, whose goal is to use available data to build a model that can be used to describe (classify) the rest of the available data. The remaining techniques in Table 1 are examples of *unsupervised* data mining. In these cases, no variable is singled out as the target; instead, the goal is to establish relationships amongst the variables.

In the information security area, various data mining techniques have been applied to the study of alerts from IDSs. However, despite the large gamma of experiments that have been performed (Brugger, 2004), little work has been directed at using this technique to generate event correlation rules. It is this area that will be explored in our work.

In this section, we present a brief overview of the k-means clustering algorithm and discuss relevant data mining experiments that have been performed in this area.

3.1 K-means technique

In the case of unsupervised learning, clustering techniques are the most appropriate as, a priori, they do not require any initial knowledge. In our work, we have concentrated on using the k-means partitioning algorithm to perform the clustering, whose steps are listed below:

1. Choose a value for k, the total number of clusters.
2. Randomly choose k points as cluster centers.
3. Assign the remaining instances to their closest cluster center.
4. Calculate a new cluster center for each cluster.
5. Repeat steps 3-5 until the cluster centers do not change.

Every point (or element) is a data object that is composed of various attributes. A similarity/dissimilarity metric is used to perform Steps 3 and 4 and is calculated as a function of all such attributes.

In general, the k-means technique requires real-valued data and one must preselect the number of clusters present in the data. Whilst the technique lacks explanation capabilities and the ability to attribute significance to the observed clusters, we argue that it can be employed as a useful tool to separate non-frequent alerts from frequent alerts, enabling an expert to concentrate his analysis on the former when generating SEC rules. As was illustrated by (Burns et al., 2000) and (Manganaris et al., 2000), it is the non-frequent events that are of more interest for detecting malicious activity.

3.2 Previous data mining results

At present, we are only aware of two other experiments in which data mining techniques have been employed for semi-automatic rule generation. (Burns et al., 2000) have used frequency analysis and visualization techniques to assist an expert in recognizing interesting patterns of activity that can subsequently be incorporated into correlation rules for event management. Whilst this technique has been demonstrated to work, we argue that it relies heavily upon manual input from an expert user to visualize the non-frequent behaviour. In our approach, k-means clustering is a more efficient, non-interactive, method for separating frequent from non-frequent behaviour.

In the second experiment, (Kreibich and Crowcroft, 2004) have employed honeypots to collect malicious activity and have used pattern detection algorithms, on the network traffic within the honeypot, to automatically derive new intrusion detection signatures for unknown attacks. Whilst the goal of this work was not to generate correlation rules per se, the newly detected signatures can subsequently be incorporated into correlation rules. In future, this work could be used to complement our approach. However, it should be noted that our objective is slightly different; i.e. using data mining techniques to analyze the alerts emitted by different types of network sensors, rather than analyzing honeypot traffic.

Another relevant application of data mining techniques, though not with the aim of generating correlation rules, is that of (Manganaris et al., 2000). This work performed association analysis on real-time IDS data collected at a Network Operations Center (NOC). Two weeks of historical alerts were used to model the association rules. Subsequently, the rules were applied to the alerts collected at the NOC during a separate week. The results were promising with the authors detecting all of the incidents that had been recorded manually by the NOC operators and, furthermore, detecting three additional incidents that had gone unnoticed at the NOC. This is practical proof of the suitability of data mining to the incident detection process.

As far as clustering technique is concerned, its use to aid network intrusion detection is not new (Brugger, 2004). However, the majority of this work has been performed on common network traffic, in the format described by the "libcap" libraries, and the goal of these experiments has been to detect anomalous behaviour. In contrast, our work focuses on clustering the alerts produced by IDSs and other network sensors and our aim is to generate SEC correlation rules. We consider our approach to be more realistic in that, for today's enterprise networks, it is not feasible to collect traffic from a whole network for training or analysis purposes.

4 GENERATING SEC RULES

4.1 Clustering experiment

In our initial experiment, a database of security events was generated by collecting data for a period of six weeks from a network-based Snort sensor installed on a corporate DMZ. The IDS was placed on the same network segment as the corporate Web, FTP and DNS servers. It was configured with the most recent signature rules and, to perform the most general test possible, all classes of signatures were enabled in the rules engine.

In order to be able to investigate the temporal dependence of the alerts, we preprocessed the alert data to form "connections". Taking $ts(A_i)$ and $T(A_i)$ to be, respectively, the timestamp and triple $(SourceIP(A_i),\ DestIP(A_i),\ DestPort(A_i))$, associated to the i-th alert A_i, we defined a "connection" as a sequence $A_1, A_2, ..., A_n$ of alerts, that satisfy the following requirements:

- $T(A_1) = T(A_2) = ... = T(A_n)$
- $\forall A_i, 1 \leq i \leq n-1 : ts(A_{i+1}) - ts(A_i) \leq 10 \text{ minutes}$
- $\neg\exists A_0 \mid ts(A_1) - ts(A_0) \leq 10 \text{ minutes} \wedge T(A_0) = T(A_1)$
- $\neg\exists A_{n+1} \mid ts(A_{n+1}) - ts(A_n) \leq 10 \text{ minutes} \wedge T(A_{n+1}) = T(A_n)$

By applying the above definition our initial data set of 170K alerts was reduced to 26K "connections" to be used as input for the clustering analysis.

As pointed out in Section 3.1, the k-means clustering technique requires a similarity/dissimilarity function, which is based upon the attributes of the elements being clustered. In our experiment, the following "connection" attributes were considered: the source IP, destination IP, and destination port of the packet that triggered the alert; the connection's duration, the total number of alerts, the total number of distinct signatures and signature classes, the total number of SYN, RST, FIN and SYNFIN flags, the total number of alerts of each priority category and the total number of alerts of each distinct signature class and signature. All of the attributes were assigned the same weight for the calculation of the similarity metric.

The k-means clustering was applied to the connection data set with the number of clusters arbitrarily set to 3, 5, 8 and 10. In all four cases, over 99% of "connections" were grouped into one single cluster. As for the remaining clusters, each contained "connections" for which certain attributes had a signficantly different proportion than those observed in the other clusters. We consider that the optimal number of clusters in our experiment was 5. For $k > 5$ there were very few "connections" in the additional clusters and the connection attributes were very similar to those of another cluster.

4.2 Mapping a cluster to a SEC rule

When clustering is employed to discover behaviour that is very rare in the overall data set, it is a common occurrence for nearly all of the data to be placed into one cluster (assumed to be normal behaviour) and for the remaining clusters to contain the rare (assumed abnormal) behaviour. Applying this reasoning, our initial assumption is that the biggest cluster is indicative of normal behaviour (i.e. false-positives from the IDS), and that the remaining clusters contain "connections" with a greater proportion of real-positives, i.e. non-frequent, malicious behaviour.

Each cluster is discerned from the others by a set of attributes presenting uncommon values when compared to the majority of the elements of the analysed data set. In our case, the infrequent attribute values correspond to unusual occurences of some types of alerts within the same "connection". Thus, a correlation rule might be written for this *attack fingerprint*, by specifying the detection condition as the occurence of the peculiar set of alerts, within a certain time window.

4.3 Discussion and future work

We consider clustering to be an appropriate technique for our purposes as it does not require the use of a labeled data set for training. As such, it is more appropriate for the analysis of IDS alerts (or those produced in abundance from the logs of network devices) where the outcome of the suspicious activity cannot be classified as malicious (real-positive) or non-malicious (false-positive) a priori. A known limitation of the k-means technique is that it is sensitive to extreme values or *outliers* (Han and Kamber, 2000). In our case, we consider this fact to be beneficial and our aim is to explore this property to cluster anomalous behaviour that is infrequent within the network traffic.

Our next steps are to perform further experiments to confirm whether the assumption presented in Section 4.2 is valid and under what conditions. To this end, we propose to use results from Honeypot experiments to classify combinations of alerts known to be associated with malicious behaviour and compare these to the results obtained from clustering. Furthermore, future work will incorporate alerts from a wide-variety of network devices (firewalls, routers, application servers, etc.) and we expect that such a richer information base will yield more accurate correlation rules (as discussed in Section 2.3).

The time parameter used for the “connection” definition, i.e. the allowed time interval between two subsequent alerts (Section 4.1), also needs to be explored. Values other than the 10 minutes employed in the first experiment need to be tested to determine how this influences the clusters obtained. It is logical that the smaller this value is, the more difficult it will be to catch attacks perpetrated at a slower rate. On the other hand, if this paramenter is too large, it may induce the creation of “connections” of unrelated alerts that happen to fall within this window.

Finally, we plan to test whether repeatedly applying the clustering algorithm to the smaller clusters is beneficial to providing a more detailed determination of the inter-relationships between the alerts. The alternative approach would be to choose a bigger value for the parameter k (the number of clusters).

5 CONCLUSION

We have discussed the limitations of IDSs and how the correlation of events from various network elements can be used to improve the intrusion detection rate and reduce the number of false-positives. As the generation of correlation rules is a time-consuming task that requires expert knowledge, semi-automatic techniques that can assist this process are clearly desirable. To this end, we propose the use of data mining to separate frequent (false-positives) from non-frequent behaviour. In succession, logical expressions for correlation rules can be written focusing solely on the latter. Our initial results suggest that this technique is promising, though more tests are needed to formulate a precise conclusion.

REFERENCES

Axelsson, S. (1999). The base-rate fallacy and its implications for the difficulty of intrusion detection. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 1–7.

Brugger, S. T. (2004). Data mining methods for network intrusion detection. `http://www.bruggerink.com/~zow/papers/brugger_dmnid_survey.pdf`.

Burns, L., Hellerstein, J. L., Ma, S., Peng, C. S., Rabenhorst, D. A., and Taylor, D. (2000). A systematic approach to discovering correlation rules for event management. IBM Research Report RC 21847, IBM.

Debar, H., Curry, D., and Feinstein, B. (2005). The intrusion detection message exchange format. http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-14.txt.

Drew, S. (2003). Intrusion detection faq – what is the role of security event correlation in intrusion detection? http://www.sans.org/resources/idfaq/role.php.

Han, J. and Kamber, M. (2000). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

Jiang, G. and Cybenko, G. (2004). Temporal and spatial distributed event correlation for network security. In *Proc. of American Control Conf.*, pages 996–1001.

Kreibich, C. and Crowcroft, J. (2004). Honeycomb - creating intrusion detection signatures using honeypots. *SIGCOMM Comput. Commun. Rev.*, 34(1):51–56.

Manganaris, S., Christensen, M., Zerkle, D., and Hermiz, K. (2000). A data mining analysis of rtid alarms. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 34(4):571–577.

Morin, B. and Debar, H. (2003). Correlation of intrusion symptoms: an application of chronicles. In *RAID 2003*, volume 2820 of *LNCS*, pages 94–112. Springer.

Ning, P., Cui, Y., and Reeves, D. S. (2002). Analyzing intensive intrusion alerts via correlation. In *RAID 2002*, volume 2516 of *LNCS*, pages 74–94. Springer.

Yemini, S. A., Kliger, S., Mozes, E., Yemini, Y., and Ohsie, D. (1996). High speed and robust event correlation. *IEEE Communications Magazine*, 34(5):82–90.

Yin, X., Lakkaraju, K., Li, Y., and Yurcik, W. (2003). Selecting log data sources to correlate attack traces for computer network security: Preliminary results. In *Proc. of the 11th Intl. Conference on Telecommunication Systems, Modeling and Analysis (ICTSM11)*.