Classification of Log Files with Limited Labeled Data

Stefan Hommes, Radu State, Thomas Engel
University of Luxembourg, SnT
4, rue Alphonse Weicker
L-2721 Luxembourg
{stefan.hommes, radu.state, thomas.engel}@uni.lu

ABSTRACT

We address the problem of anomaly detection in log files that consist of a huge number of records. In order to achieve this task, we demonstrate label propagation as a semi-supervised learning technique. The strength of this approach lies in the small amount of labelled data that is needed to label the remaining data. This is an advantage since labelled data needs human expertise which comes at a high cost and becomes infeasible for big datasets. Even though our approach is generally applicable, we focus on the detection of anomalous records in firewall log files. This requires a separation of records into windows which are compared using different distance functions to determine their similarity. Afterwards, we apply label propagation to label a complete dataset in only a limited number of iterations. We demonstrate our approach on a realistic dataset from an ISP.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection (e.g., firewalls)*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Security

Keywords

Label propagation, firewall, log files, Kullback-Leibler divergence

1. INTRODUCTION

Security appliances require a configuration that can adapt to new threats to assure comprehensive protection. One source providing such insights are log files, which are often neglected since their analysis is difficult due to the huge amount of data (e.g. one million records per day). Automated approaches to detect anomalous events with supervised learning techniques require a labelled dataset. Such a fully-labelled training set is not available in most cases due to the high cost associated with the labelling process and privacy issues.

We propose an approach to classify log file records by using label propagation. Latter is a semi-supervised learning technique that requires only a small subset of labelled data which acts as a source that pushs out labels to the unlabelled data. To reduce the amount of required labelled data in large data sets, we split the data in several parts which are labelled in several iterations. After the first part was labelled with the label propagation algorithm, the resulting labelled data can be leveraged again as an input to label the remaining data of the following part. This steps are repeated until the whole dataset is fully labelled.

Even though label propagation is not limited to a specific application scenario, we demonstrate the usage to detect anomalous records in firewall log files. Due to the huge amount of logged connections and the temporal dependencies (e.g. network scan), we split records into windows of a certain size. Based on such windows, we apply a distance function which is required by the label propagation algorithm to represent the degree of similarity between windows. We demonstrate our approach in a realistic dataset from a checkpoint firewall that is used by an local ISP.

The paper is structured as follows: Related work is presented in section 2. We then define a metric to specify the degree of anomalous records in a window-based approach in section 3. The concept and algorithm for iterative label propagation is presented in section 4. The experimental results based on a realistic dataset are presented in section 5. We conclude our work in section 6.

2. RELATED WORK

Firewalls are an important tool in network security management to allow internet-connected devices to communicate with each other in a secure manner. Typical firewalls examine all packets that pass through them according to specific rules set up by the system administrator. These rules might block, allow or log certain connections. Dependent on the configuration of the firewall, all accepted and dropped connections are stored and archived in log files for later inspection. It is common practice that, due to the lack of real-time analysis tools, such log files are only analysed after an incident has occurred.

To find certain connection patterns that could be the result of an attack, the information content of these log files can be used to detect anomalous connection attempts. Since the

IPTComm'13, October 15 - 17 2013, Chicago, IL, USA

size of log files and the amount of events in a system can grow very fast, analysing this data can be very challenging. Furthermore, a certain event by itself may be unimportant but be in fact a precursor of an attack (e.g. denial of service), or an error when occurring in conjunction with a second event. The analysis of firewall configurations was pioneered by the papers [4, 2], where several types of potential conflicts and conflict-resolution algorithms are proposed. While our work does not address the analysis of static firewall rules per se, the detection of firewall misconfiguration could leverage a cross-correlation between the static analysis of firewall rules and observed anomalies and events in the logs. The work described in [1, 3] is more similar to our approach in combining observed traffic with existing and deployed firewall configurations. There are however some major differences. We do not aim to optimize the storage of firewall rules or to reverse-engineer a set of deployed rules, but focus on monitoring the operation of deployed firewall rules in order to detect potential incidents. Our approach, thus, is independent of the set of rules, but assumes that a working and operational configuration is in place. Event management and fault detection has been a major topic in the network management community. Some recent work leveraging association learning is described in [11, 10], where sequences of events are mined in order to learn patterns related to an observed fault. This is complementary to our work, which does not assume a known set of faults, but infers from differences between successive events that some anomaly has occurred. The mining of log data using a rule-based event correlator is addressed in [8, 10]. Even though our approach leverages rules derived from statistical process control, the sequences of triggered alarms could be post-processed using a tool like SEC [9]. Although we have not yet implemented it, such post-processing could also use an unsupervised clustering method similar to the work described in [7]. From an operational perspective, [6] provides a rich and powerful application programming interface for event correlation, but human knowledge and expertise are required to use it.
In the intrusion detection community, [12] describes a method that monitors changes in event intensity and uses probabilistic techniques to evaluate events in network traffic data. In the context of supervised machine learning techniques, the authors of [5] leverage support vector machines to learn the predicting signs for failures. The approach is dependent on a labelled set of data and requires computational resources that are not appropriate for the large-scale online monitoring of a firewall.

3. DISTANCE METRIC

Semi-supervised learning can be used to reduce the cost of the labelling process since only a small set of labelled data is needed. The key to this technique is that unlabelled data and labelled data are used for the training. Thus, the unlabelled data is used to improve learning accuracy, which is very useful since unlabelled data is usually abundant.
If we consider all labelled and unlabelled data points as nodes in a fully connected graph, the labels from each node propagate to neighbouring nodes according to their proximity. The closer the nodes i, j are to each other, the larger the weight which is calculated for a distance. Since a distance needs to be chosen for each data point, the following section defines a distance d that is used as a metric to determine the similarity of log file records.

Table 1: Example of a firewall log file record. Relevant attributes are assigned to distance metrics (J=Jaccard similarity, KL=Kullback-Leibler divergence) for detecting outliers in a series of records.

Attribute	Example	Metric
Number	4237	-
Date	16May2011	-
Time	0:05:11	-
Type	Log	-
Source Port (a_1)	49954	J
Rule (a_2)	298	KL
Current Rule Number	298-Standard	-
Information	-	-
Product	VPN-1 Power/UTM	-
Interface (a_3)	eth-s1/s1p1c3	KL
Origin (a_4)	IP1220-Gare1	KL
Action (a_5)	Drop	KL
Service (a_6)	694	KL
Source IP (a_7)	192.168.8.183	J
Destination IP (a_8)	192.168.8.255	J
Protocol (a_9)	udp	KL
Rule Name	-	-
User	-	-

3.1 Separation of records

In an initial step, we bundle log file records into chunks, or *windows*, for two reasons: firstly, to reduce the amount of data since the number of firewall log file records produced by a company or ISP can be huge due to the large number of network attacks and blocked connections; and secondly, to incorporate the dependencies from temporally related events since a typical attack attempt (e.g. network scan) often results in a series of blocked connections and log file records which all relate to the same attacker. Each window can further be labelled as normal or anomalous and can be built in two different ways. A window w_s contains a fixed number of records, whereas a window w_t contains events from a certain time period. Since the time of occurrence is not considered to be important in our approach, we chose to use the fixed-length window type. This also avoids empty windows that can occur during attack-free intervals, or during periods having no traffic.

3.2 Similarity of attributes

The typical attributes and values from an example firewall record are shown in Table 1. We consider the attributes a_1 to a_9 to be important for anomaly detection. To determine if a window is anomalous or normal, we calculate the distance between two consecutive windows w_i and w_{i+1}. All values from each attribute (e.g. "tcp") in the first window are compared with the corresponding attribute in the second. Since attributes differ in data type and range (e.g. source Port and protocol), we utilize two methods to calculate this distance.

3.2.1 Kullback-Leibler divergence

The Kullback-Leibler divergence can be considered as a metric to estimate the difference between two probabilistic distributions in information theory. In order to calculate this distance for two consecutive windows, we build one histogram per window for each attribute. Each value x for

an attribute a_i can either be a number (e.g. port number) or a string (e.g. interface) and defines its own bin in the histogram that records its frequency of occurrence m. The probability of each value in the histogram can be calculated by

$$p_x = \frac{m_x}{X} \tag{1}$$

where X is the total number of all values per window. We then calculate the Kullback-Leibler distance between $w_i \rightarrow w_{i+1}$ and $w_{i+1} \rightarrow w_i$ for an attribute by

$$\begin{aligned} D(a_{w_i}||a_{w_{i+1}}) &= \sum_{x \in X} p^a_{w_i}(x) \cdot log_2 \frac{p^a_{w_i}(x)}{p^a_{w_{i+1}}(x)} \\ D(a_{w_{i+1}}||a_{w_i}) &= \sum_{x \in X} p^a_{w_{i+1}}(x) \cdot log_2 \frac{p^a_{w_{i+1}}(x)}{p^a_{w_i}(x)} \end{aligned} \tag{2}$$

We calculate the distance for both directions since the Kullback-Leibler divergence is not symmetric. The sum of both distances can then be used to describe the distance $\tilde{D}$ between two windows. If two attributes are exactly the same, we find that $\tilde{D} = 0$.

$$\tilde{D}(a_{w_i}||a_{w_{i+1}}) = D(a_{w_i}||a_{w_{i+1}}) + D(a_{w_{i+1}}||a_{w_i}) \tag{3}$$

3.2.2 Jaccard similarity coefficient

For an attribute that can have a large number of different values (e.g. IP address), the Kullback-Leibler divergence is not suitable due to the large number of different bins in the histogram. For such attributes, we choose the Jaccard similarity to give an estimation of the similarity between two windows. It is defined by the size of the intersection divided by the size of the union for the values from window A and B, so $J = 1$ if all values from both windows are the same.

$$J(a_{w_i}, a_{w_{i+1}}) = \frac{|X^a_{w_i} \cap X^a_{w_{i+1}}|}{|X^a_{w_i} \cup X^a_{w_{i+1}}|} \tag{4}$$

3.3 Distance between windows

To determine a single value that describes the similarity between two windows, we define a distance d in equation 5. It is calculated by summing all Kullback-Leibler distances and Jaccard similarities for all attributes.

$$d = \sum_{i \in I_1}^{n} \tilde{D_{a_i}} + \sum_{i = \in I_2}^{n} J_{a_i} \tag{5}$$

The method used for each attribute is defined in Table 1, where the Kullback-Leibler divergence is calculated for $I_1 = \{a_2, a_3, a_4, a_5, a_6, a_9\}$ and the Jaccard similarity for $I_2 = \{a_1, a_7, a_8\}$.

4. LABEL PROPAGATION

We review the label propagation algorithm as it was described by [13]. The concept of label propagation is shown in Figure 1. The original graph nodes are now replayed by all windows $W = w_1, ..., w_n$.
The labelled data is described as $(w_1, y_1)...(w_l, y_l)$ with $Y_L = \{y_1...y_l\}$ as the class labels. Labelled data item w_i is known to belong to one of two classes C, either normal or anomalous, because we wish to classify malicious windows in the log files. The unlabelled data is described as (w_{l+1}, y_{l+1}) ... (w_{l+u}, y_{l+u}) with $Y_U = \{y_{l+1}...y_{l+u}\}$ as the class labels. Estimating the class labels Y_U from W and Y_L can be described as a transductive learning setting. The main idea behind this mechanism is that unlabelled data items should be labelled with class labels by taking into account similarities to already labelled data items.

The calculation of the weight between two windows i, j is shown in equation 6 and based on the distance d from equation 5. Windows with closer Euclidean distances should have similar class labels and thus a higher weight. The attribute σ is further used to control the weight.

$$\hat{w}_{ij} = \exp\left(-\frac{d^2_{ij}}{\sigma^2}\right) \tag{6}$$

We can calculate a probabilistic transition matrix T based on the distance from all windows compared to each other, which defines the probability of jumping from window j to i. The size of the matrix is defined by $(l+u) \times (l+u)$.

$$T_{ij} = P(j \rightarrow i) = \frac{\hat{w}_{ij}}{\sum_{k=1}^{l+u} \hat{w}_{kj}} \tag{7}$$

In addition, we define a label matrix Y of size $(l+u) \times C$, which contains the probability for each window to belong to class c_1 or c_2.

4.1 Algorithm

The algorithm consists of three steps that are repeated until the label matrix Y converges:

1. Propagate $Y \leftarrow TY$
2. Row-normalize Y
3. Clamp the labelled data

After multiplying the two matrices in step 1, we propagate the probabilities of the labelled data to the unlabelled data. After row normalisation in step 2, we reset the probabilities of the labelled data to the respective value (0 or 1). This is necessary in order to retain persistent elements in the label matrix (clamping). In the next run, the labelled data acts as a source and the unlabelled data is updated.

The steps 1, 2 and 3 in the algorithm are computationally equivalent to a matrix multiplication, which is of $O(n^3)$. In our case the Y matrix has n rows and two columns and thus the computational complexity of steps 1, 2 and 3 is $O(n^2)$. The outer loop of this algorithm (until convergence) is shown to be bounded in [13] and our own practical experience is in line with that result.

4.2 Iterative labelling

Label propagation can reduce the manual effort required since only a small part of the dataset need be labelled by an expert. But since a typical firewall log file consists of a huge

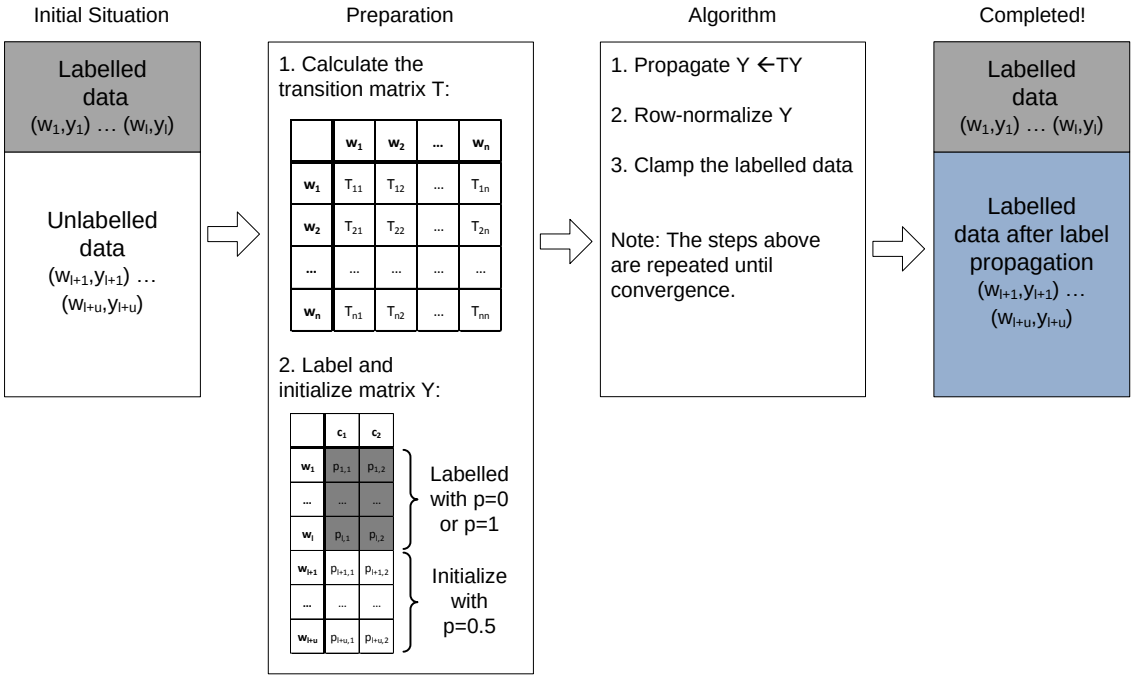

Figure 1: Concept of label propagation; the preparation step requires the calculation of the transition matrix T and labelling some windows by assigning them to the different classes (c_1 or c_2). The algorithm is executed until convergence and assigns the initially unlabelled windows to their respective classes.

number of records, the labelled part can still be uncomfortably large for manual labelling. Our iterative method reduces the amount of manually labelled data needed.
The concept of iterative labelling is shown in Figure 2. Firstly, we separate the log file into a subset and label Y_L with the support of an expert. We then use the label propagation algorithm to label Y_U. We repeat the process, choosing $Y_{L(2)} = Y_L + Y_U$, and use the label propagation algorithm again until the log file is fully labelled.

5. EXPERIMENTAL RESULTS

Our industrial partner, the POST Luxembourg company, provided us with a datasets from a Checkpoint firewall cluster, which is used as an internet firewall for their internet-connected servers. The dataset statistics are presented in Table 2.

The experiments were run on a dual-core 2.8 GHz Intel PC with 4 GB of RAM. The calculation of the transition matrix T is the most time-consuming operation if the window size is small compared to the number of records in the dataset. We chose a fixed window size of $w_s = 100$, since most attacks result in a number of logged entries that are in that range. The dataset was further split into three parts, since the ratio between the amount of labelled and unlabelled data has an impact on the classification. We determined by experiment a ratio of 1 to 5 to be sufficient.

5.1 First iteration

This part consisted of 50 windows in total, where we manually labelled five anomalous and five normal windows. For the 40 unlabelled windows, both classes were initialized with a probability of 0.5. After applying the label propagation algorithm, we detected one window which was labelled with a probability of 0.99 as anomalous and occurred as a block of 116 records:

Number of records	49550	
Time period	1h 39min	
Date	15.05.2011	
Data size	10.4 MB	
	Unique Values	
	total	$w_s = 100$
Source port	21696	70.7
Rule	34	7.1
Interface	26	7
Origin	2	1.9
Action	4	2
Service	3293	19.2
Source	2657	47.8
Destination	449	34.1
Protocol	5	3

Table 2: Dataset statistics: The lower part shows the unique values for each attribute in the the dataset and the average for a window size of $w_s = 100$

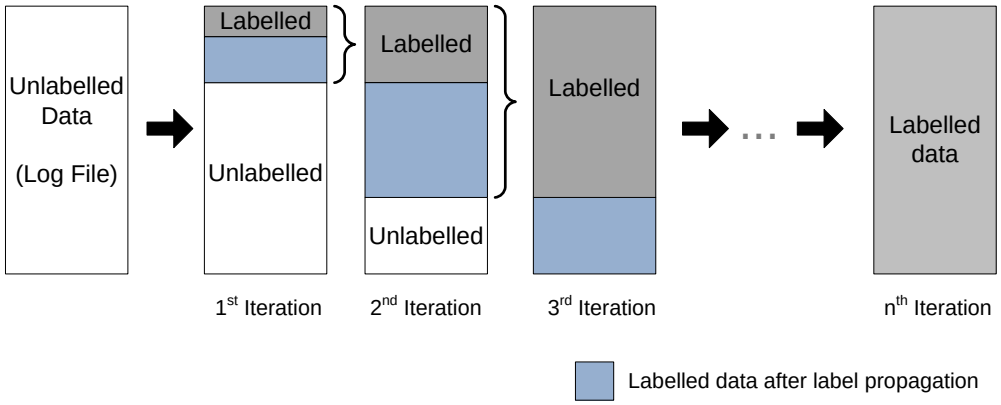

Figure 2: Concept of iterative labelling; only a small amount of labelled data is initially used by the label propagation algorithm, reducing the cost of manual labelling for large datasets.

```
  "2128" "16May2011" "0:02:15" "Log" "60270" "166"
"166-Standard" "service_id:  http" "VPN-1 Power/UTM"
"eth-s2/s2p2c2" "⋆⋆⋆" "Accept" "http"
"⋆⋆⋆.bb.netbynet.ru" "⋆⋆⋆" "tcp" "" ""
```

Investigating this case we looked into the log and correlated the source IP with well-known list of public HTTP proxies. Such proxies are usually used to hide the identity of web users and while this can be legitimate in order to protect privacy in less democratic countries, in our case the incident was more probably a brute-force authentication breaking attack.

5.2 Second iteration

This iteration used the 50 labelled windows from the previous step and added 200 unlabelled windows. The algorithm classified three windows as anomalous: these included two different types of records. The following window was identified with a probability of 0.74 and occurred as a block of 327 records of the following type:

```
  "11141" "16May2011" "0:16:37" "Log" "35961" "166"
"166-Standard" "service_id:  http" "VPN-1 Power/UTM"
"eth-s2/s2p2c2" "⋆⋆⋆" "Accept" "http"
"⋆⋆⋆.bb.netbynet.ru" "⋆⋆⋆" "tcp" "" ""
```

The interpretation of this record is similar to the record from the first iteration. The next window was identified with a probability of 0.73 and occurred as a block of 60 records:

```
  "12182" "16May2011" "0:18:21" "Log" "2364" "298"
"298-Standard" "" "VPN-1 Power/UTM" "eth-s2/s2p2c1"
"⋆⋆⋆" "Drop" "telnet" "⋆⋆⋆" "⋆⋆⋆" "tcp" "" ""
```

Any telnet traffic is highly suspicious and is a clear sign of attack.

5.3 Third iteration

The final iteration took the 250 labelled windows from the previous step, and attempted to label the remaining 246 windows. It classified two further windows as anomalous. The first was identified with a probability of 0.96 and occurred as a block of 73 records:

```
  "27842" "16May2011" "0:48:32" "Log" "5071" "298"
"298-Standard" "" "VPN-1 Power/UTM" "eth-s2/s2p2c1"
"⋆⋆⋆" "Drop" "sip_any" "⋆⋆⋆.calpop.com" "⋆⋆⋆" "udp"
"" ""
```

We looked into the log and the incoming traffic was sourced by a machine located at a popular hosting provider. The many SIP packets were in fact a VoIP scan looking for open SIP proxies.

The second window is also related to SIP but unlike the previous window, the sender is not known:

```
  "27536" "16May2011" "0:47:50" "Log" "sip_any"
"298" "298-Standard" "" "VPN-1 Power/UTM"
"eth-s2/s2p2c1" "⋆⋆⋆" "Drop" "sip_any" "⋆⋆⋆"
"⋆⋆⋆" "udp" "" ""
```

The reason for such connection attempts may either be a badly configured VoIP configuration, or a fraudulent use of SIP. In this specific case, many SIP Register requests occurred without success, and the system identified this anomaly correctly.

5.4 Convergence

We also looked at the convergence properties of the label propagation algorithm. The outer loop of the method is performed until convergence is obtained. We evaluated the

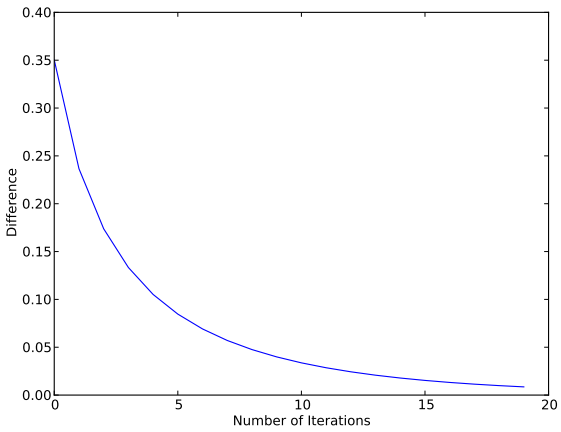

Figure 3: Convergence of the label propagation algorithm (section 4.1) for the second iteration (50 labelled windows, 200 unlabelled windows, $w_s = 100$).

differences between successive matrices using the squared sum of the elements in the difference matrix. The difference between successive iterations is calculated using the following equation:

$$D_{it,it+1} = \sum_{i=1}^{n} |y_{i,1}^{it} - y_{i,1}^{it+1}|^2 \quad (8)$$

The evolution of the label propagation algorithm is displayed in Figure 3 for the dataset during the second iteration (50 labelled, 200 unlabelled). This shows that 10 iterations are sufficient to obtain convergence. This result empirically validates the convergence properties established in [13].

6. CONCLUSION

We have described label propagation as a semi-supervised learning technique to determine anomalous records in firewall log files. These consist of a huge number of entries, making manual inspection a tedious task. Network attacks such as denial-of-service or network scans often result in a block of related records that need to be identified from a log file in order to enhance existing security practices. In the first step, we separate records into windows and calculate the distance between these in order to define a metric for similarity. Label propagation is then used to label all windows in the log file as normal and anomalous with only a limited amount of labelled data. In addition, iterative labelling is an extension that helps to reduce the initial amount of labelled data when the proportion of unlabelled data is too high for a single iteration. We evaluated our approach on a realistic dataset that was retrieved from a checkpoint firewall at a local ISP. The results can support the administrator in modifying existing firewall rules and assessing the threat level of the network.

7. ACKNOWLEDGMENTS

Supported by the Fonds National de la Recherche, Luxembourg (PhD-09-188).

8. REFERENCES

[1] M. Abedin, S. Nessa, L. Khan, E. Al-Shaer, and M. Awad. Analysis of firewall policy rules using traffic mining techniques. *IJIPT*, 5(1/2):3–22, 2010.

[2] E. Al-Shaer. Designing, optimizing, and evaluating network security configuration. In *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, 2008.

[3] E. Al-Shaer, A. El-Atawy, and T. Samak. Automated pseudo-live testing of firewall configuration enforcement. *IEEE Journal on Selected Areas in Communications*, 27(3):302–314, 2009.

[4] E. Al-Shaer, C. R. Kalmanek, and F. Wu. Automated security configuration management. *J. Network Syst. Manage.*, 16(3):231–233, 2008.

[5] E. W. Fulp, G. A. Fink, and J. N. Haack. Predicting computer system failures using support vector machines. In *Proceedings of the First USENIX conference on Analysis of system logs*, WASL'08, pages 5–5, Berkeley, CA, USA, 2008. USENIX Association.

[6] J. Hwang. Splunk, innovation behind. In *Proceedings of the Symposium on Computer Human Interaction for the Management of Information Technology*, CHiMiT '09, New York, NY, USA, 2009. ACM.

[7] A. A. Makanju, A. N. Zincir-Heywood, and E. E. Milios. Clustering event logs using iterative partitioning. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 1255–1264, New York, NY, USA, 2009. ACM.

[8] J. P. Rouillard. Refereed papers: Real-time log file analysis using the simple event correlator (SEC). In *Proceedings of the 18th USENIX conference on System administration*, pages 133–150, Berkeley, CA, USA, 2004. USENIX Association.

[9] R. Vaarandi. Platform independent event correlation tool for network management. In *NOMS*, pages 907–909. IEEE, 2002.

[10] R. Vaarandi. Mining event logs with SLCT and LogHound. In *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pages 1071–1074, 2008.

[11] R. Vaarandi and K. Podins. Network IDS alert classification with frequent itemset mining and data clustering. In *Network and Service Management (CNSM), 2010 International Conference on*, pages 451 –456, oct. 2010.

[12] N. Ye, S. Emran, Q. Chen, and S. Vilbert. Multivariate statistical analysis of audit trails for host-based intrusion detection. *Computers, IEEE Transactions on*, 51(7):810 –820, jul 2002.

[13] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, School of Computer Science - Carnegie Mellon University, 2002.