Intrusion Detection in Enterprise Systems by Combining and Clustering Diverse Monitor Data

Atul Bohara[*]
Department of Computer Science
University of Illinois at Urbana-Champaign
abohara2@illinois.edu

Uttam Thakore[*]
Department of Computer Science
University of Illinois at Urbana-Champaign
thakore1@illinois.edu

William H. Sanders
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
whs@illinois.edu

ABSTRACT

Intrusion detection using multiple security devices has received much attention recently. The large volume of information generated by these tools, however, increases the burden on both computing resources and security administrators. Moreover, attack detection does not improve as expected if these tools work without any coordination.

In this work, we propose a simple method to join information generated by security monitors with diverse data formats. We present a novel intrusion detection technique that uses unsupervised clustering algorithms to identify malicious behavior within large volumes of diverse security monitor data. First, we extract a set of features from network-level and host-level security logs that aid in detecting malicious host behavior and flooding-based network attacks in an enterprise network system. We then apply clustering algorithms to the separate and joined logs and use statistical tools to identify anomalous usage behaviors captured by the logs. We evaluate our approach on an enterprise network data set, which contains network and host activity logs. Our approach correctly identifies and prioritizes anomalous behaviors in the logs by their likelihood of maliciousness. By combining network and host logs, we are able to detect malicious behavior that cannot be detected by either log alone.

CCS Concepts

•Security and privacy → Intrusion detection systems; •Theory of computation → *Unsupervised learning and clustering;*

Keywords

Security, Monitoring, Intrusion Detection, Anomaly Detection, Machine Learning, Clustering

[*]These authors contributed equal work to the paper and are listed in alphabetical order.

HotSoS '16, April 19 - 21, 2016, Pittsburgh, PA, USA

© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4277-3/16/04...$15.00

DOI: http://dx.doi.org/10.1145/2898375.2898400

1. INTRODUCTION

Securing large computer networks and distributed systems is becoming increasingly challenging because of the growing scale and complexity of these systems. Security breaches continue to occur, despite the use of several layers of protection. Such compromises result in unauthorized access to sensitive information and misuse of computing resources, leading to significant losses to organizations, both financially and socially [6, 2, 3]. Statistics published in Symantec's 2015 Internet Security Threat Report indicate a significant increase in both vulnerabilities and attack attempts over the last year [4]. There is ongoing demand for research on better intrusion detection and analysis methods.

Manifold problems make the task of securing a large networked system very difficult. New attack techniques are invented very frequently. Most of the sophisticated attack techniques try to resemble normal behavior very closely, and thus are very difficult to detect. The large scale and high complexity of large enterprise systems make intrusion detection even more challenging. Furthermore, monitoring system-wide activities for the purpose of intrusion detection results in volumes of diverse monitor data that easily overwhelm security experts and online intrusion detection systems [1]. As a result, in many cases, intrusions are detected long after significant losses have already been incurred.

In this work, we try to address some of the aforementioned problems and take a significant step towards improving intrusion detection. We present an approach to detect intrusions by identifying patterns of anomalous usage of an enterprise system based on the data captured in diverse monitors deployed in the system. We do not rely on any labeling of monitoring data, and instead use completely unsupervised machine learning techniques.

Recent research has shown that host-based and network-based monitor data can be used together to provide additional context for improved intrusion detection [8, 14, 11, 9]. Many modern attacks use techniques that seem normal to individual monitors. One of the effective ways to detect these attacks is to look at the combined pattern of events across different monitors.

In our approach, we combine the host-level context, which is captured by monitors such as system logs that are deployed on individual hosts, with the network-level context, which is captured by network-level monitors such as firewall and network intrusion detection systems, and we use the aggregated profile in detecting anomalous behavior.

First, we introduce the VAST 2011 Mini Challenge 2 dataset, which contains network-level and host-level logs and a set of injected attacks. We present a threat model that describes a set of attack types that can be detected using anomaly detection algorithms and encompasses many of the attacks in our dataset. We then extract meaningful features from the log data, and use the features to combine the network-level and host-level data in a way that facilitates anomaly detection over the joined data. Subsequently, we perform clustering on the log data to identify usage profiles, which we classify as normal or anomalous based on statistical tests. We devise a method to order the anomalous clusters in terms of their likely maliciousness, which can aid a security administrator in prioritizing which clusters to investigate manually. Finally, we show the efficacy of our approach by evaluating it on the VAST dataset.

We make the following contributions:

1. **Identify important features:** We reason about and identify a good set of features to detect flooding-based network attacks and suspicious host behavior in our dataset. We aggregate and combine the data from network-level and host-level monitors based on the extracted features.

2. **Detect anomalies:** We apply the k-means and DBSCAN clustering algorithms to detect different behavioral patterns in system logs and firewall logs. We use statistical techniques to choose parameters for the clustering algorithms to minimize the amount of tuning required by a domain expert. We present a metric to analyze the output of the clustering algorithms to identify anomalous clusters, and demonstrate its efficacy on our dataset.

3. **Support intrusion detection:** We propose a technique to prioritize the anomalous clusters in order of likely maliciousness based on analysis of the distributions of the features in each cluster, and show that the intrusions present in the dataset are captured by the top few anomalous clusters.

2. DATASET DETAILS AND THREAT MODEL

In this section, we introduce the dataset that we examine and describe the types of attacks that we want to detect using our unsupervised anomaly detection approach.

2.1 Dataset

In this paper, we work with the dataset published by the Visual Analytics Community for Mini Challenge 2 of the VAST 2011 Challenge [5]. The dataset contains over 1.5 GB of enterprise network and host logs for network operations performed on the workstations and servers of a fictional freight shipping company over three days. We analyzed the logs from the firewall (Cisco Adaptive Security Appliance 5510), intrusion detection system (Snort), and Windows operating system security event logs (Windows 2008 Server).

The firewall logs in the dataset contain attributes about communications between internal and external hosts, such as source and destination IP address, communication protocol, and destination service. The Windows server security logs, which we refer to as *system logs*, record events such as valid and invalid logon attempts, authentication events, and subject and target user IDs. In the dataset's scenario, the Snort IDS has been configured to generate alerts based on the default Snort rule set.

The dataset is accompanied by a document that describes the attacks that were injected into the data and the monitors in which evidence for the attacks appears. The dataset is injected with the following types of attacks: denial of service from external hosts, a worm installed on some internal hosts, port scans by internal hosts, a socially engineered e-mail sent by an outside network address, a remote desktop connection violating company policy from outside network, and the appearance of a suspicious host on the enterprise network. We utilize the information in the dataset documentation to evaluate our intrusion detection approach, as explained in Section 6.

2.2 Threat Model

Our threat model consists of two categories of security incidents in an enterprise network environment. The first category of attacks includes network scan attacks and flooding-based network attacks. Examples of such attacks include port scan attacks, in which a malicious subject scans a critical server in the network to gather information about the services running on different ports; port sweep attacks, in which a malicious subject scans multiple hosts for a single type of service; and network-layer Denial-of-Service (DoS) or Distributed DoS attacks, in which a malicious subject attempts to disrupt service availability by overwhelming the system. The goal of these attacks is to disrupt a network-based service, with a range of possible motivations, such as revenge or financial gain.

The second category of attacks that we aim to detect is the presence of malware on a host in the network, specifically viruses and worms. These types of malware can change the behavior of an infected host by running suspicious processes, scanning the network, or attempting to authenticate to other hosts in the network. Such malware can cause serious damage to the enterprise by stealing information, disrupting service, or using the machines for illicit purposes.

The goal of our work is to identify hosts that exhibit behavior that deviates significantly from normal usage patterns in order to detect the aforementioned types of attacks. We do not wish, however, to explicitly profile normal usage; rather, we want to learn the profiles in an unsupervised manner from the data. In later sections, we present our approach to detecting anomalies without explicit profiling.

3. OVERVIEW OF APPROACH

Since we know that there are different types of attacks within the data, but do not have labels for the individual log entries, we cannot use supervised algorithms to classify the data. Furthermore, there is no guarantee that the types of attacks taking place in the system will remain static. We must therefore use a learning algorithm that can discover host behavior patterns in the data in an unsupervised way.

Clustering is an unsupervised machine learning technique especially suitable for identifying structure in unlabeled data. It partitions a given set of objects into subsets of similar objects and separates dissimilar objects into different clusters. We use clustering algorithms in our approach, as they can identify behavior patterns in the data by using self-similarity and can distinguish behavior patterns that could signify anomalous behavior.

First, we identify the features that are most useful for detecting the types of attacks described in our threat model. Next, we apply dimensionality reduction and clustering on the features we extract from the dataset to discover clusters in the data, and then analyze the feature distributions of the clusters to identify potential anomalous behavior. Finally, we tag the anomalous clusters with the possible types of attacks they represent, which can then be analyzed manually by a security administrator to decide the best course of action to take. We compare the results of our approach with the ground truth provided with the dataset and evaluate the efficacy of our approach.

4. FEATURE EXTRACTION

4.1 Feature Extraction

We perform feature extraction and clustering on only the firewall and system logs in the dataset. Since Snort requires a rule set to generate meaningful alerts, it relies on a system administrator to select an adequate rule set, which presents the possibility of missing alerts. Snort also aggregates network traffic before generating some alerts, so it provides results comparable to those of our own analysis. Therefore, we use Snort only to validate the results of our approach.

We choose features based on both the attacks discussed in our threat model and domain knowledge of the enterprise network on which the dataset is based. Domain knowledge includes the types and locations of critical servers and acceptable resource usage policies. We identify the following four categories of features, which we use to capture usage behavior and join the data across different monitors.

4.1.1 Identification features

We use the IPv4 address as the unique identifier of each host in the network. For both the firewall logs and system logs, for each one-minute time window, we aggregate the raw log entries for a given IP address into a single feature vector. Thus, every data point can be uniquely identified with the IP address and the timestamp. All of the features described subsequently are extracted and aggregated for a distinct pair of an IP address and a timestamp. As we show in the next section, these identification features are very useful in correlating the results of cluster analysis with intrusions.

One important point to note is that in doing the time-based aggregation of features, we assume that the time is synchronized across all hosts in the system and across all monitors. We account for the time difference in the firewall logs and system logs in our dataset manually, but in other systems, it could be accomplished using a centralized network time server.

4.1.2 Network traffic-based features

We extract the following features related to network communication from the firewall data: the number of unique source and destination ports used, the number of TCP connections built and torn down, and the number of distinct administrative services used (e.g., Telnet, finger, FTP). We use the source IP address to identify the feature vectors during cluster analysis.

4.1.3 Service-based features

To incorporate the diversity of services accessed and the types and locations of machines contacted by a host in our analysis, we extract the following features from the firewall data for each source IP address: the number of accesses to workstations; the number of accesses to the domain controller servers, which also provide the DNS service; the number of accesses to database servers; and the number of accesses to any other type of server.

4.1.4 Authentication-based features

Finally, since the system log data comes from the domain controller server, which services authentication requests for all workstations and servers in the network, we extract features from the system log data by counting the following properties of system log events: failed logon attempts, logon attempts using explicit credentials, special privilege assignments to new logons, computer account changes, generation of Kerberos authentication tickets, NTLM authentication attempts, administrative logons, anonymous user logons, anonymous target user names, local interactive logons, remote desktop logons, requests for session keys, port numbers that are either zero or blank, and logons by distinct process IDs. We aggregate the log entries by the source IP address of the authentication request for all events for which the source IP is given in the log.

4.2 Analysis of Features

After extracting the features, we analyze the distribution of values each feature takes in the dataset to understand the relative importance of the features and their possible effect on the clustering algorithms. Specifically, we look at the shape, variance, and modality of the distributions.

First, we examine the network traffic and service-based features extracted from the firewall data. Figure 1 shows the empirical cumulative distribution functions (CDFs) for a subset of the features for the firewall data aggregated by source IP address. The features take many different distributions, but our first observation is that many of them are not Gaussian and are right-tailed. It is also important to note that a significant proportion of the probability mass is on very small values for most of the features. For example, for the first feature, which represents the number of unique destination IP addresses visited per minute by a source, almost 75% of the data points have a value of 1. As a result, we expect that the tails of the distributions—that is, the data points that take high values—will be of most interest in the clustering. We also expect that the clustering approaches will place the vast majority of points into a few dense clusters.

Next, we examine the authentication-based features extracted from the system log data. Figure 2 shows the empirical CDFs for a subset of the features for the system log data aggregated by source IP address. As with the firewall data, the features take many different distributions, but many are not Gaussian and right-tailed. The vast majority of the probability mass for each feature (over 95%) is on the values 0 or 1. For the system log data, we therefore expect even more densely packed clusters than for firewall data.

Through examination of our data, we observe that dimensionality reduction by Principal Component Analysis (PCA) prior to clustering does not provide useful results. Briefly, PCA projects data in high-dimensional space onto axes called *principal components*, each of which point in the

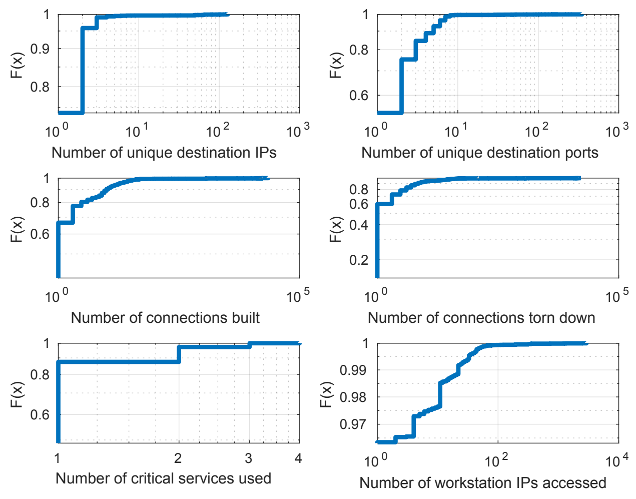

Figure 1: Empirical cumulative distribution functions (CDFs) for selected features in the firewall data aggregated by source IP address, shown with log scales for both the X and Y axes.

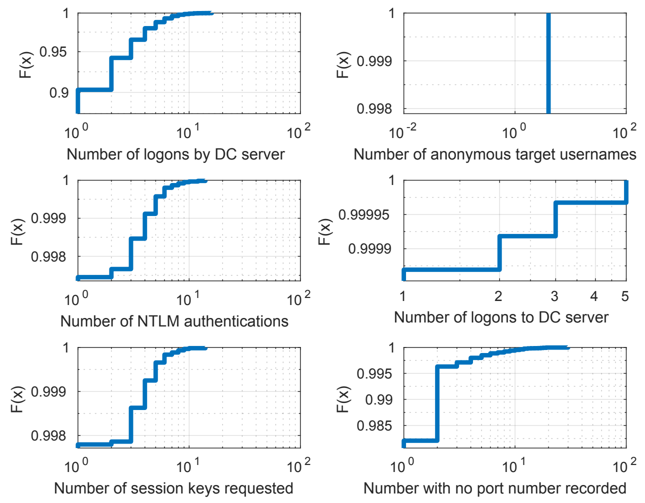

Figure 2: Empirical cumulative distribution functions (CDFs) for selected features in the system log data aggregated by source IP address, shown with log scales for both the X and Y axes.

direction of maximum variance given the variance already captured by preceding components. By projecting the data onto the first few principal components, it is possible to reduce the dimensionality of the data while retaining the bulk of the variance in the data.

In our data set, however, variance does not describe the expressivity of features well. First, many of the features in our data have constricted ranges, so variance can be artificially limited by the range of the feature values. As an example, the number of possible source ports in TCP is 2^{16}, so the features corresponding to the number of distinct source ports used by a host will be limited to the range $[0, 2^{16}]$. Additionally, the number of failed logon events, which is a strong indication of malicious behavior, takes a small range of values and therefore has low variance compared to the number of successful logons, which has a much higher variance but is relatively unuseful in detecting attacks.

Furthermore, we wish to detect anomalous behavior, which can manifest as a small number of data points with outlying values. However, the principal components determined by PCA that represent most of the variance in the data may not capture such behavior. For example, the first three principal components for the firewall data, which together capture over 99% of the variance in the data, represent only the number of TCP connections built, the number of TCP connections torn down, and the number of unique source ports used. However, in our analysis of the clustering results, we find that the number of destination IP addresses contacted is the primary distinguishing feature for one of the attacks in the dataset, and clustering on the basis of the data projected onto the first few principal components would miss this attack altogether.

As a result, we perform clustering on the features without performing dimensionality reduction first. We do, however, refine the feature set before doing the clustering by looking at the correlation between different features and their relative importance for clustering, as we discuss in Section 4.3.

Also, to ensure that the clustering algorithms are not influenced by the difference in ranges in the different features, we normalize the data prior to clustering such that all features have a range of $[0, 1]$.

4.3 Feature Selection

We extract all the features described above and then use two different techniques to refine the feature set to use for clustering. First, if the features are strongly correlated to each other, they will contain redundant information and increase the complexity of the clustering algorithms. To remove this redundancy, we keep only one feature from each set of strongly correlated features.

We use the Pearson correlation coefficient to measure the linear dependence between each pair of feature vectors. We consider a pair as strongly correlated if the correlation coefficient is greater than 0.99. We believe that the Pearson correlation coefficient will provide fairly accurate estimation for correlation between features because the ranges of all the features are bounded and the size of our dataset is moderately high.

Second, if the features do not contribute towards clustering, then we discard them. For example, the number of logon events in each time interval is an important feature to characterize each host in the network, but in this dataset, it does not help distinguish between clusters. We prune such features before performing our clustering experiments.

To determine which features to prune, we run a clustering algorithm and analyze the average normalized feature scores in each cluster, which we define later in Section 5.3. We identify the features having almost the same average score in each cluster, and discard them for the final clustering experiments.

5. INTRUSION DETECTION USING CLUSTER ANALYSIS

In this section, we describe our intrusion detection approach in detail. First, we apply clustering algorithms to the data to discover clusters of similar data points, and then analyze the joint distributions of features in each cluster to identify the types of behavior associated with each cluster. We hypothesize that different types of attacks will manifest as different distributions of the features in the data, and data points corresponding to different attack types will cluster together because of their similarity with each other or their dissimilarity with other clusters. Based on the feature distribution analysis of the clusters, we then identify anomalous clusters and sort them by their dissimilarity with clusters that represent normal behavior. Finally, we manually analyze the anomalous clusters and identify the types of attacks they represent.

5.1 Overview of Algorithms Used

We use the k-means and DBSCAN clustering algorithms to classify different host behavior profiles.

k-means is a centroid-based clustering algorithm that partitions a given dataset into k clusters, such that each observation belongs to the cluster with the closest mean. For our analysis, we apply Lloyd's basic k-means algorithm [18] and use Euclidean distance as the distance metric for clustering.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [12] creates clusters on the basis of the density of clusters formed. The DBSCAN algorithm starts with a random point and adds it to a cluster if a certain minimum number of points lie within a certain radius of the point. The algorithm then iteratively grows the cluster by similarly considering all points in the radius and repeating until no more points are added. After the assignment of points to one cluster completes, a new arbitrary point is selected, and the process is repeated. This algorithm also marks outlier points that lie alone in low-density regions. For our analysis, as with k-means, we use the Euclidean distance as the distance metric for the DBSCAN algorithm.

We chose these two algorithms because they are simple and widely used, and we hypothesize that given the right set of features and analysis techniques, they can yield good results. Since the algorithms use different approaches to cluster points, we expect them to work well in different types of situations. k-means is relatively simpler and computationally faster than DBSCAN, but it is sensitive to noise and works well only when the clusters are spherical (i.e., have equal variance in all dimensions) and have a roughly equal number of data points. DBSCAN, on the other hand, can generate arbitrarily shaped clusters by identifying connected regions of high density, and is designed to ignore noisy points. We compare the results of each algorithm on our dataset and make decisions on which to use based on the goodness of the clusters they generate.

In addition to running the two clustering algorithms on the firewall data and system log data separately, we join the two data sources by performing an inner join for values that share source IP address and timestamp values, and run clustering on the joined data as well. We hypothesize that clustering on joined data will reveal attacks that would not be evident from either data set independently.

5.2 Parameter Selection for Clustering Algorithms

5.2.1 Parameter selection for k-means

The k-means algorithm requires a single parameter: the number of clusters assumed to be in the data, k. To determine the number of clusters to use for k-means, we examine the cluster tightness and the separation between clusters.

To choose the number of clusters that best represents the data, we start by choosing a range of values for k that would likely best fit the data. For each candidate value of k, we run the k-means algorithm for a fixed number of iterations and take the cluster means that minimize the *within-cluster sum of distances (WCSD)*, which is the average sum of Euclidean distances from each point to the center of the cluster to which it is classified. We use this procedure to account for the initialization bias encountered with k-means, which could otherwise lead the k-means result to reside in a local optimum.

Then, to test the appropriateness of the value of k, we look at the silhouette values [22] of all points in the data set. The silhouette value of a point quantifies how well it belongs to the cluster to which it is assigned as compared to all other clusters. For a point $\mathbf{x}$, let $a(\mathbf{x})$ represent the average distance from $\mathbf{x}$ to all other points in its cluster, and $b(\mathbf{x})$ represent the average distance from $\mathbf{x}$ to all other points in the cluster for which such distance is minimized (the next closest cluster). Then, the silhouette value of $\mathbf{x}$, $s(\mathbf{x})$, is given by

$$s(\mathbf{x}) = \frac{b(\mathbf{x}) - a(\mathbf{x})}{\max\left(a(\mathbf{x}), b(\mathbf{x})\right)} \tag{1}$$

The values of $s(\mathbf{x})$ can range from -1 to 1, and values closest to 1 indicate clusters that are tight and well-separated. Therefore, we choose the value of k for which the run with the smallest WCSD has the maximum average silhouette value for all points.

5.2.2 Parameter selection for DBSCAN

The DBSCAN algorithm requires two input parameters: ϵ, the neighborhood radius for each point, and $minPts$, the minimum number of points required to form a cluster. We use the guidelines presented in the original DBSCAN paper [12] along with evaluation of the average silhouette values to compute suitable values for the parameters.

According to [12], the value of $minPts$ should be at least $D + 1$, where D is the number of dimensions of the dataset used for clustering. The authors recommend choosing a larger value of $minPts$ as the size of the dataset or the amount of noise in the dataset increases. After deciding on a reasonable estimate for the value of $minPts$, we examine the *sorted k-distance graph* for our dataset to determine the value of ϵ. The k-distance graph depicts the set of distances from each data point to its k^{th}-nearest neighbor, sorted in decreasing order of magnitude, where $k = minPts$. A good value of ϵ is one for which the k-distance graph shows a strong bend or *knee point*.

Using the approach given above, we estimate a small range of values for each of the parameters. For each pair of $(\epsilon, minPts)$, we then run the DBSCAN algorithm on the dataset and compute the average silhouette score as explained above for k-means clustering, ignoring the outlier points. We choose the pair of parameter values corresponding to the best average silhouette score. In this manner, we select the parameter values by considering the properties of our dataset, the properties of the DBSCAN algorithm, and the quality of clusters generated.

5.3 Cluster Analysis and Classification

After running k-means and DBSCAN with the parameter values selected by the methods described above, we analyze the clusters obtained to identify those that correspond to anomalous behavior. For each clustering algorithm and log type, we examine the cluster sizes and distribution of hosts within the clusters to distinguish between "normal" clusters and "anomalous" clusters.

For each cluster c in the data set, let $\mathcal{S}_c$ represent the set of points in cluster c, and let $\mathcal{H}_c$ represent the set of unique hosts, identified by source IP address, in the cluster. We consider a cluster c to represent normal behavior in the system if $|\mathcal{S}_c| \geq s$, where s is a minimum threshold for the number of points, and $|\mathcal{H}_c| \geq h$, where h is a minimum threshold for the number of unique hosts. That decision was motivated by our observation that the distributions of feature values for both the firewall and system log data have very high probability mass at low values, so most of the normal data points will cluster into a few densely packed clusters containing most of the hosts. Anomalous behavior represents either 1) a large number of attacking hosts acting maliciously over a short period of time, as in the case of a denial-of-service (DoS) attack, which would result in a small value of $|\mathcal{S}_c|$, or 2) a small number of attacking hosts acting maliciously over an arbitrary period of time, as in the case of a port scan, which would result in a small value of $|\mathcal{H}_c|$.

Indeed, we found that for all data types in our dataset, over 80% of the data points always fell into the first two or three clusters, and that these clusters individually contained well over half of the hosts. Based on those observations, we chose the values of $s = S/k$ and $h = H/k$, where k is the number of clusters, $S = \sum_{c=1}^{k} |\mathcal{S}_c|$ is the total number of data points, and $H = \left|\cup_{c=1}^{k} \mathcal{H}_c\right|$ is the total number of hosts in the system. Intuitively, a cluster will be considered normal if more than a proportional fraction of the data points and hosts fall within the cluster. In our actual data, the normal clusters contained much more than the threshold value of data points and almost always contained all of the hosts.

We represent the set of normal clusters with

$$\mathcal{N} = \{c : |\mathcal{S}_c| \geq s \wedge |\mathcal{H}_c| \geq h\} \tag{2}$$

and the set of abnormal clusters with

$$\mathcal{A} = \{c : |\mathcal{S}_c| < s \vee |\mathcal{H}_c| < h\}. \tag{3}$$

Next, we compute the *normalized average feature value vector*, $\hat{\mathbf{f}}^c$, for each cluster c as follows:

$$\mathbf{f}^c = \sum_{\mathbf{x} \in \mathcal{S}_c} \frac{\mathbf{x}}{|\mathcal{S}_c|} \tag{4}$$

$$\hat{\mathbf{f}}^c = \frac{\mathbf{f}^c}{\max_i \mathbf{f}_i^c} \tag{5}$$

where i is the feature index, S_c is the set of points in cluster c, and F is the number of features. For each cluster, $\mathbf{f}^c$ is the centroid of cluster c, and $\hat{\mathbf{f}}^c$ is the centroid normalized by the supremum norm of the centroid. $\hat{\mathbf{f}}^c$ describes the features that are active in a given cluster and makes it easy to compare their relative strengths, as the maximum value in $\hat{\mathbf{f}}^c = 1$. We use $\hat{\mathbf{f}}^c$ to understand the feature distributions for each of the clusters and to determine which clusters are more likely to be malicious, as we describe in the next section.

5.4 Intrusion Detection

The last step in our approach is to determine which of the clusters representing anomalous behavior actually describe *malicious* behavior.

Using the $\hat{\mathbf{f}}^c$ vectors, we compare the joint distributions of features for each anomalous cluster with those for normal clusters. Specifically, we compute the difference between the feature distributions of two clusters c and c' by computing the L_1 distance between the $\hat{\mathbf{f}}$ values for each cluster. We define the *cluster difference*, $D_{c,c'}$ as follows:

$$D_{c,c'} = \left\| \hat{\mathbf{f}}^c - \hat{\mathbf{f}}^{c'} \right\|_1 \tag{6}$$

In our analysis of the dataset, we observed that the feature vectors were sparse; that is, only a few features took nonzero values in each cluster, and of those, only a few had high values. Furthermore, since we removed highly correlated features before clustering, the cluster difference for different clusters was not small unless the clusters themselves were geometrically close to each other. Thus, we posit that anomalous clusters with the greatest difference from normal clusters in terms of normalized feature distributions are more likely to be malicious.

As a result, we define the *cluster normalcy*, Δ_c, for an anomalous cluster c as the cluster difference between the anomalous cluster and its closest normal cluster. In other words,

$$\Delta_c = \min_{c' \in \mathcal{N}} D_{c,c'}. \tag{7}$$

We ultimately provide a prioritized list of the anomalous clusters, ordered by decreasing value of Δ_c, to the security administrator to manually evaluate. By separating the clusters by behavior, reducing the clusters to those that are anomalous, and prioritizing the clusters by likely maliciousness, we significantly simplify the work that must be performed by the administrator.

Given the administrator's input, we envision the possibility of training a classifier that could automatically determine whether a new data point is likely to be malicious. We discuss the idea briefly as part of future work in Section 8.

6. EXPERIMENTAL EVALUATION

To experimentally evaluate our approach, we implemented our feature extraction code in Python and ran all of the machine learning and cluster analysis on the data in MATLAB. We used MATLAB's stats toolbox implementation of k-means and our own implementation of DBSCAN.

As mentioned in Section 2.1, a ground truth document was provided with the dataset that contains the attacks that were injected into the data and the monitors in which evidence for the attacks appears. Analysis of this document and the challenge description showed that because of the network configuration of the dataset's system, the firewall was not able to observe all of the port scanning/sweeping attacks that took place in the dataset. As a result, we evaluated our approach only with those attacks detectable using only the firewall logs and system logs. This yielded the following list of attacks: denial of service (DoS) from hosts 10.200.150.201 and 10.200.150.206–209, worm likely installed on hosts 192.168.2.171–175, port scans by hosts 192.168.2.174–175, a socially engineered e-mail sent by host 10.200.150.6, a remote desktop connection violating company policy from host 10.200.150.201, and the appearance of a suspicious host with IP address 192.169.2.151.

Based on our threat model, we further narrowed the set

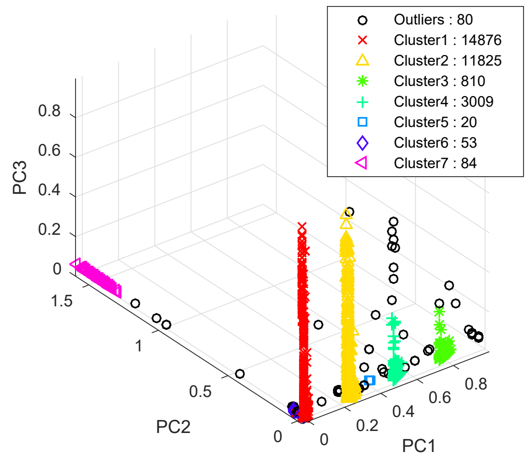

Figure 3: DBSCAN clusters for the firewall data for parameter values $\epsilon = 0.15$ and $minPts = 21$, projected to the first three Principal Components (PC) for visualization purposes. Clustering was performed in the original 10-dimensional space.

of attacks we aim to detect to those representing network flooding attacks and suspicious behavior by hosts. Any attack that was a direct violation of policy or required investigation of raw packet data (such as the socially engineered e-mail) would not be detectable under our threat model, or indeed with just the firewall and system log data and no other information. Thus, the set of attacks we expected to be able to detect decreased to DoS from hosts 10.200.150.201 and 10.200.150.206–209, worm likely installed on hosts 192.168.2.171–175, and port scans by hosts 192.168.2.174–175.

To evaluate the ability of our approach to identify the attacks actually present in the dataset, for each anomalous cluster, we examine the IP addresses of the hosts in the cluster, the timestamps of the entries, and the distribution of feature values, and compare these to the ground truth document. If the hosts and timestamps in the cluster correspond to a known attack in the ground truth document, we consider the cluster to detect the attack.

From the firewall data, as described in Section 4, we extracted a total of 12 features. Two of the features—IP address and timestamp—were identification features, but the remaining 10 features were of use for clustering: unique destination IPs, unique source ports, unique destination ports, connections built, connections torn down, critical services used, workstations accessed, DNS server accesses, database server accesses, and other critical server accesses. For each feature, we aggregated the value of the feature per minute for each source IP.

From the system log data, we extracted a total of 36 features, of which the same two as for the firewall were identification features. After we ran our feature selection approach from Section 4.2, the number of uncorrelated features was reduced to 20, and all of them were used for clustering.

6.1 Detecting Flooding-Based Network Attacks

Between the two types of monitors we used in our analysis, flooding-based network attacks manifest most directly in firewall data. Because external hosts cannot communicate

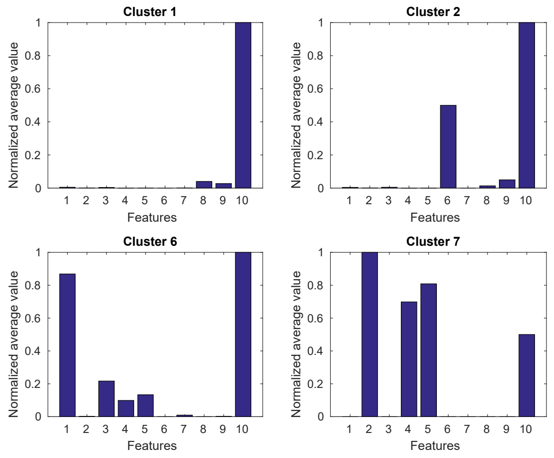

Figure 4: Normalized average feature values for four DBSCAN clusters on the firewall data.

with the domain controller according to the dataset scenario, hosts in the 10.200.0.0/24 subnet do not appear in the system log data at all, so the only way for us to detect attacks from outside hosts with our approach is by analyzing the firewall logs.

Extracting per-minute data points from the firewall logs yielded over 150,000 data points, which exceeded the computational and memory capabilities of our machines when running DBSCAN. To accommodate, we uniformly randomly downsampled the data to 20% of its original size before running both clustering algorithms. Since we downsampled uniformly, the overall distribution of points did not change. Furthermore, the types of attacks we aim to detect generate multiple data points in the data set, so there was no threat that important clusters would disappear from the dataset. We also adjusted the parameters of DBSCAN to account for the decreased density of the dataset.

We ran the parameter value selection algorithms described in Section 5.2 on the data set to obtain the optimal parameters for the clustering algorithms. The parameter values we obtained were $k = 8$, $\epsilon = 0.15$, and $minPts = 21$. The results of running DBSCAN on the firewall logs with said parameters are shown in Figure 3. With the parameters specified, DBSCAN identified 6 clusters and a small set of outlier points. While it is not evident from the graph, the clusters are actually quite distant from each other in the higher-dimensional space in which clustering was performed.

We found that k-means performed poorly on the firewall data, clustering together data points that did not have similar feature distributions and absorbing smaller clusters into their larger, neighboring clusters. That was likely due to the non-Gaussian distribution of the data points and the large variance in the number of points in each cluster. DBSCAN, on the other hand, performed exceptionally well. For the remainder of this analysis, we use the DBSCAN results.

Next, we analyzed the size of each cluster and the number of unique hosts in each. Applying Equations 2 and 3, we found that clusters 1 and 2, which together represented over 86% of the data points, were classified as normal, and all others were classified as anomalous. We then performed the cluster difference analysis and ordered the clusters based on their Δ_c values, as described in Section 5.4. For the firewall data, excluding the outliers, cluster 6 was considered most

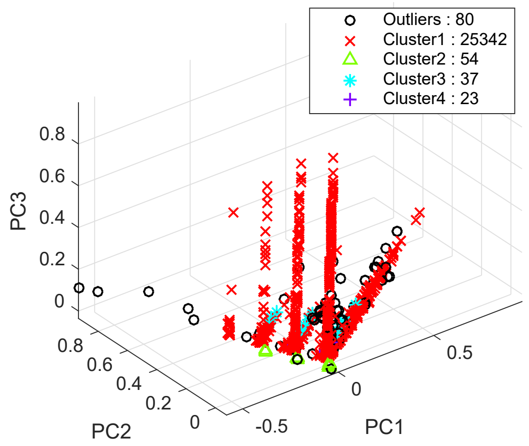

Figure 5: DBSCAN clusters for the combined firewall and system log data for parameter values $\epsilon = 0.25$ and $minPts = 20$, projected to the first three Principal Components (PC) for visualization purposes. Clustering was performed in the original, 28-dimensional space.

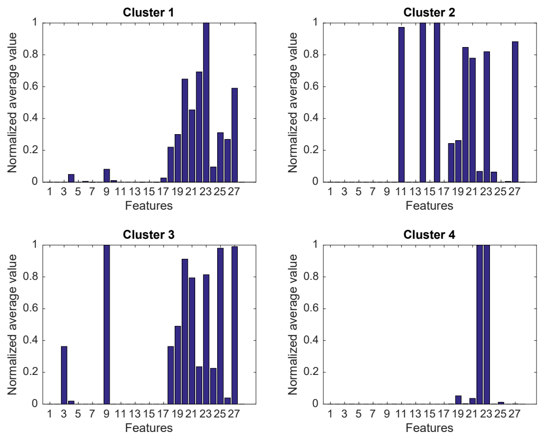

Figure 6: Normalized average feature values for four DBSCAN clusters on combined firewall and system log data.

likely to be malicious, followed by clusters 5, 3, and 4.

When we analyzed the hosts and timestamps for each of the anomalous clusters, we found that the order given by the cluster normalcy-based prioritization exactly matched the ground truth, which showed that clusters 6 and 5 corresponded to attacks and clusters 3 and 4 were not malicious.

Cluster 6 contained only the IP addresses 10.200.150.201 and 10.200.150.206–209 and had very high feature values for the number of unique source ports, number of connections built, and number of connections torn down. The cluster corresponds to outside attackers' performing a DoS attempt on one of the servers inside the network. The timestamp values for the points in cluster 6 correspond to some of the times that the hosts were performing the DoS attack.

Cluster 5 contained only the IP addresses 192.168.2.174–175 and 192.168.1.6, and had very high feature values for the number of destination IPs accessed. Examining the timestamp values for the points showed that while 192.168.1.6 appeared only at three very different times in the cluster, the other two hosts appeared over 20 times within the same 4-hour period. The cluster corresponds to two hosts inside the network that were performing port scans of other machines in the network in the time window in which the two hosts appeared in the cluster. It is likely that 192.168.1.6, which is a mail server, was classified into this cluster because it sent mails to many hosts at the three times it appeared in the cluster, and thus had a feature distribution similar to that of the port scanners.

For the firewall data, our approach performed incredibly well, separating out the two attacks—DoS from the external network and port scan from the internal network—and providing those two clusters as those most likely to be malicious. As a sanity check, when we compare the performance of our completely unsupervised approach to that of Snort, we find that Snort detected the same two attacks in the dataset, but required an administrator to sift through many lines of logs.

6.2 Detecting Suspicious Host Behavior

To detect suspicious behavior of internal hosts and to find the attacks we mention above, we experimented with both system log data alone and system log data combined with firewall data. As was the case with the firewall data alone, and for many of the same reasons, k-means did not perform well in cleanly separating clusters by host behavior. Therefore, here, too, we focus on the clustering results we obtained from DBSCAN.

First, we performed DBSCAN clustering on the system log data alone. Because of the size of the data, we uniformly randomly downsampled the data to 50% of its original size. Running the parameter selection algorithm yielded $\epsilon = 0.15$ and $minPts = 30$, and running DBSCAN on the system log data with those parameters yielded five clusters and a set of outliers. Performing the analysis described in Sections 5.3 and 5.4, we found that cluster 1 was classified as normal, and all others as anomalous. Of those, only cluster 3, which had the highest Δ_c value, represented a malicious host.

Next, we performed DBSCAN clustering on the combined firewall and system logs, which were joined by the approach explained in Section 5.1. The optimal parameters for the joined data were $\epsilon = 0.25$ and $minPts = 20$, for which the results of DBSCAN are shown in Figure 5. There are four clusters and one group of outlier points. While the clusters do not seem separable in the graph, they are quite distant in the higher-dimensional space in which clustering was performed.

Performing the analysis described in Sections 5.3 and 5.4, we found that cluster 1 represented normal behavior, and clusters 2, 3, and 4 were classified as anomalous. In order to analyze the anomalous clusters for attacks, we computed the normalized average feature vector for each cluster, as shown in Figure 6. Features 1 – 18 correspond to the features extracted from system log data, and features 19 – 28 correspond to those extracted from the firewall log.

When we ordered the anomalous clusters by their cluster normalcy values, we found that cluster 2 was most likely to be malicious, followed by clusters 4 and 3. Based on the unique IP addresses in cluster 3, we concluded that it contained only benign hosts. However, in line with the prioritization of the clusters, clusters 2 and 4 corresponded to the attacks present in the dataset.

Cluster 4 contained only the IP addresses 192.168.2.174–175 and had very high feature values for the number of connections built and torn down. The cluster corresponds to the two hosts' performing port scans on the internal network that were detected by the firewall. Note, however, that this cluster did not erroneously contain the mail server.

Cluster 2 was the only cluster that had relatively higher values for both system log and firewall log features. The cluster had high average values for the number of anonymous target usernames used, the number of NTLM authentications, and the number of session keys requested. This cluster contained only the IP address 192.168.2.172. From the features for which it had high values, it can be inferred that the host was trying to make connections and log on to other hosts on the internal network. While the dataset did not explicitly mark this host as being infected, this situation indicates a very high probability that the host had been infected by a worm.

Finally, as a sanity check, we compared clusters 2 and 4 against the attacks detected by the Snort IDS. While the Snort logs captured the port scan attack, they did not capture the malicious behavior of host 192.168.2.172. That behavior was detectable only through clustering on the combined data, which demonstrates the value of our approach of clustering on joined data instead of looking at each type of data source separately.

6.3 Discussion

As evidenced by the results in Sections 6.1 and 6.2, our approach performed well on the dataset for the types of attacks we describe in our threat model. Without labeling or preprocessing the data by hand, we were able to extract generic, time-aware features, use clustering to identify anomalous behavior, and prioritize the anomalous behaviors in order of decreasing likelihood of maliciousness. For each of the log types in the dataset, our approach correctly identified and prioritized the anomalous clusters corresponding to attacks.

Furthermore, our approach provides the system administrator with additional visibility into the behavior of hosts in the system. By looking at the average feature distributions for each cluster in the dataset, the administrator can determine how different sets of hosts behave in the system and can easily drill down into the anomalous hosts to determine which ones are behaving maliciously.

Since the features we extract describe generic network-based and authentication-based attributes, we believe that our approach can generalize well to any attack that dramatically changes the behavior of a host. For example, we believe that with the same set of features, it would be possible to detect brute-force attempts or data exfiltration attacks.

However, our approach cannot detect all types of intrusions with the same level of accuracy. Intrusions that occur silently, such as zero-day attacks, and those that do not disturb the outward behavior of the host significantly, such as privilege escalation attacks, would not cause the host to appear anomalous.

As detailed in Section 6, we base the evaluation of our intrusion detection technique on the ground truth document describing all the injected attacks in the dataset. We do not claim that there is no possibility of the presence of attacks other than the ones that are given in that document.

With additional features and more diverse sources of information, it might be possible to expand the types of attacks supported by our approach. We leave such exploration to future work.

7. RELATED WORK

Unsupervised anomaly detection in enterprise network systems is not itself a novel idea. Significant research has been done in the domain of network traffic anomaly detection using techniques including PCA-based anomaly detection [16] and entropy and clustering-based anomaly detection [17]. Anomaly detection based on cluster analysis is also used in [26] and [21] to find intrusions in enterprise network environments.

We analyze a diverse set of monitoring data from different security monitors to detect anomalies and tag them as possible intrusions. We also provide a formal approach for selecting parameters for the different algorithms we use. Furthermore, we directly work with raw logs captured from different security monitors and do not rely on any commercial SIEM system. These contributions differentiate our work from previous related work [26, 21].

Another distinguishing feature of our work is that we do not need a model or normal profile of the system. Some of the prior work [13, 15, 25] that relies on modeling of normal system behavior to detect anomalies can fail to model a good reference profile in a malicious environment like the one represented by our dataset. In contrast, we adopt a completely unsupervised technique for detecting anomalies. We also reason about the relative importance of various features in detecting certain types of anomalies, very similar to the work done in [19], but we use an unlabeled dataset for experimentation.

Our work is also related to information fusion techniques for security [8, 7]. However, most of those prior solutions either are too complicated to be used in practice, or need a significant amount of model training or parameter setting.

Finally, our work is also related to the topic of alert correlation, which aims to convert security information into knowledge that is more intuitive and understandable to human administrators. Alert correlation techniques find nontrivial relationships between alerts from multiple sources and group them together to satisfy a variety of goals, such as summarizing IDS logs into higher-level incidents or improving the detection of collaborative attacks [10, 24, 23, 20]. These techniques operate on IDS data, which are higher-level than the data we use, and perform correlation using heuristics or predefined rule sets. In contrast, we categorize data points into normal or anomalous clusters based solely on the similarity of individual data points to each other and the feature distributions of the clusters formed.

8. CONCLUSION

In this paper, we present an approach to identifying anomalous behavior in unlabeled system log and network log data using unsupervised machine learning. Our approach and analysis are based on the monitors available in the VAST 2011 Mini Challenge 2 data set. We first establish a threat model and extract meaningful features from the log data that aid in attack detection. We then describe a method to combine the data using features we have defined that allows us to perform anomaly detection over the joined network and host log data. Next, we use the k-means and DBSCAN clustering algorithms to categorize the data into different usage profiles, and we compare the feature distributions of different clusters to identify anomalous behavior. We then propose a cluster difference metric that we use to prioritize the anomalous clusters based on their likely maliciousness. Finally, we manually analyze the clusters to correlate them with known attacks and evaluate our approach. We find that our approach performs very well for the data set, detecting all attacks present except those that require additional information to detect. Using the joined network-level and host-level data, we are also able to detect attacks that are not detectable with either data source alone.

In future work, we plan to examine the usefulness of other features and to expand the types of logs that we can combine to perform coordinated anomaly detection. We also plan to investigate the relative efficacy of other clustering algorithms that make different assumptions when clustering, such as distribution-based or hierarchical clustering, and to look into ways of classifying new log entries given the profiles constructed by clustering and manual cluster analysis.

Acknowledgments

We would like to thank Ahmed Fawaz, Arjun Athreya, and Gabriel Weaver for their helpful comments, and Jenny Applequist for her editorial revisions to the paper.

This material is based in part on research sponsored by the Air Force Research Laboratory and the Air Force Office of Scientific Research, under agreement number FA8750-11-2-0084.

9. REFERENCES

[1] Cloud Security Alliance. Big Data Analytics for Security Intelligence, 2013. Technical Report. https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Big_Data_Analytics_for_Security_Intelligence.pdf.

[2] Imperva. Man in the Cloud (MITC) Attacks, 2015. Technical Report. https://www.imperva.com/docs/HII_Man_In_The_Cloud_Attacks.pdf.

[3] Sony Pictures Entertainment. System Disruption Notice Letter, 2014. http://oag.ca.gov/system/files/12%2008%2014%20letter_0.pdf.

[4] Symantec. Internet Security Threat Report, Volume 20, 2015. http://www.symantec.com/security_response/publications/threatreport.jsp.

[5] Visual Analytics Community. VAST Challenge, 2011. http://vacommunity.org/VAST+Challenge.

[6] Wikipedia. JPMorgan Chase Data Breach, 2014. https://en.wikipedia.org/wiki/2014_JPMorgan_Chase_data_breach.

[7] M. Almgren, U. Lindqvist, and E. Jonsson. A Multi-Sensor Model to Improve Automated Attack Detection. In *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection*, pages 291–310. Springer-Verlag, 2008.

[8] T. Bass. Intrusion Detection Systems and Multisensor Data Fusion. *Commun. ACM*, 43(4):99–105, 2000.

[9] P. Cao, E. Badger, Z. Kalbarczyk, R. Iyer, and A. Slagell. Preemptive Intrusion Detection: Theoretical Framework and Real-world Measurements. In *Proceedings of the 2015 Symposium and Bootcamp on the Science of Security*, pages 5:1–5:12. ACM, 2015.

[10] H. Debar and A. Wespi. Aggregation and Correlation of Intrusion-Detection Alerts. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pages 85–103. Springer-Verlag, 2001.

[11] H. Dreger, C. Kreibich, V. Paxson, and R. Sommer. Enhancing the Accuracy of Network-based Intrusion Detection with Host-based Context. In *Proceedings of the Second International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 206–221. Springer-Verlag, 2005.

[12] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, volume 96, pages 226–231, 1996.

[13] C. Hood and C. Ji. Proactive Network-Fault Detection [Telecommunications]. *IEEE Transactions on Reliability*, 46(3):333–341, 1997.

[14] S. Katti, B. Krishnamurthy, and D. Katabi. Collaborating Against Common Enemies. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, pages 365–378. USENIX Association, 2005.

[15] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based Change Detection: Methods, Evaluation, and Applications. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, pages 234–247. ACM, 2003.

[16] A. Lakhina, M. Crovella, and C. Diot. Diagnosing Network-wide Traffic Anomalies. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 219–230. ACM, 2004.

[17] A. Lakhina, M. Crovella, and C. Diot. Mining Anomalies Using Traffic Feature Distributions. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 217–228. ACM, 2005.

[18] S. Lloyd. Least Squares Quantization in PCM. *IEEE Trans. Inf. Theor.*, 28(2):129–137, 2006.

[19] W. Ma, D. Tran, and D. Sharma. A Study on the Feature Selection of Network Traffic for Intrusion Detection Purpose. In *IEEE International Conference on Intelligence and Security Informatics, 2008. ISI 2008*, pages 245–247, 2008.

[20] P. Ning, Y. Cui, and D. S. Reeves. Constructing Attack Scenarios Through Correlation of Intrusion Alerts. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 245–254. ACM, 2002.

[21] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion Detection with Unlabeled Data using Clustering. In *Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*, pages 5–8, 2001.

[22] P. Rousseeuw. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *J. Comput. Appl. Math.*, 20(1):53–65, 1987.

[23] E. Totel, B. Vivinis, and L. Mé. A Language Driven Intrusion Detection System for Event and Alert Correlation. In *Proceedings at the 19th IFIP International Information Security Conference*, pages 209–224. Springer, 2004.

[24] A. Valdes and K. Skinner. Probabilistic Alert Correlation. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pages 54–68. Springer-Verlag, 2001.

[25] N. Ye. A Markov Chain Model of Temporal Behavior for Anomaly Detection. In *Proceedings of the 2000 IEEE Workshop on Information Assurance and Security*, pages 171–174. United States Military Academy, West Point, NY, 2000.

[26] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda. Beehive: Large-scale Log Analysis for Detecting Suspicious Activity in Enterprise Networks. In *Proceedings of the 29th Annual Computer Security Applications Conference*, pages 199–208. ACM, 2013.