# Cyber Security Network Anomaly Detection and Visualization

*Major Qualifying Project*

Advisors:

PROFESSORS LANE HARRISON, RANDY PAFFENROTH

Written By:

HERIC FLORES-HUERTA
JACOB LINK
CASSIDY LITCH

A Major Qualifying Project
WORCESTER POLYTECHNIC INSTITUTE

Submitted to the Faculty of the Worcester Polytechnic Institute in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science.

OCTOBER 25, 2016 - APRIL 27, 2017

Approved By: Professor Lane Harrison & Professor Randy Paffenroth

# Abstract

In this Major Qualifying Project, we present a novel anomaly detection system for computer networks and a visualization system to help users explore network captures. The detection algorithm uses Robust Principal Component Analysis to produce a lower dimensional subspace of the original data for which a sparse matrix of outliers occurs. This low dimensional data subspace is determined by a novel contribution of this MQP to the standard practice of robust principal component analysis revolving around the relaxation parameter $\lambda$. These outliers from the low dimensional subspace can be thought of as features. Based on these features, firewall logs are classified as anomalous through a Random Forest classification trained on one type of anomaly. One of our key contributions is understanding the relationship between $\lambda$ and the underlying random forest detection algorithm. This detection system produces anomaly predictions which are visualized in this MQP. The visualization system has been designed to help cyber security analysts sort anomalies by attribute and view them in the context of normal network activity. The system consists of an overview of firewall logs, a detail view of each log, and a feature view where an analyst can see which features of the firewall log were implicated in the anomaly detection algorithm. The ability of each view to influence its sub-view allows users to sort through the traffic based on firewall attribute and refine their search down to individual anomalies.

# Table of Contents

# LIST OF FIGURES

# Chapter 1

# Introduction

## 1.1 Introduction

In recent years computer networks have become more prevalent throughout society. As the uses for computer networks have expanded, the networks themselves have become integral in the functions of everyday life. As such an important part of society, these computer networks must be both stable and reliable, and maintaining these networks is a quickly evolving field of study. This project focuses on detecting anomalies in network traffic as a way to assist with the process of network protection.

As an initial measure, we must define what exactly a cyber security network anomaly is. A cyber security network anomaly in real-world data must be defined in the context of computer network analysis as well as mathematics. In terms of computer networks, numerous types of anomalies may occur including:

- extreme anomalies (such as a distributed denial of service attack (DDoS))
- moderate anomalies (such as port scans)
- subtle anomalies (such as a buffer overflow attack)

Therefore, anomalies can be defined as cases that differ in some way from the normal operations of a network. In particular, one method for detecting a specific type of anomaly is understanding the causes of the anomaly. Once the causes of the anomaly are understood, a list of rules can be developed to determine if a new case follows the same format as a previous anomaly. Therefore, each new case that follows the same guidelines as a previous anomaly would be labeled anomalous. For this method of anomaly detection, knowing the type of anomalies that can occur in the data is important.

Creating lists of rules for each type of anomaly is not feasible for all data, especially when a new type of anomaly is introduced into the data. Without having a previous rule set to detect an anomaly, a new type of anomaly would not be detected. For this report, we will focus on understanding normal cases of the computer network, and we will define anomalies as cases that are differ from normal.

In order to determine normal cases for computer networks, we used low-dimensional representations of data measured from computer networks to define anomalies [2]. For real world data, such as firewall logs from a cyber security network, a low dimensional subspace of the data can often be found such that the low-dimensional subspace maintains the information of the normal space. A classic approach to this type of anomaly detection is using Principal Component Analysis (PCA) to reduce the data to a low-dimensional subspace and then detect data which does not lie on the low-dimensional subspace to be anomalous [2].

An important piece of information to note, is that the selection of features that create the original data set can greatly affect the results. For this report, we preprocessed the network data and performed feature engineering to generate adequate features for our firewall logs. We will demonstrate that the selection of these features can produce a low-dimensional subspace with good detection accuracy for anomalies.

Traditionally, computing a low-dimensional subspace that approximates a set of data is done by PCA [3]. Given a collection of points, PCA computes a linear projection of the points to a low-dimensional subspace that minimizes the Frobenius norm of the difference between the original points and the projected points [2, 3].

However, PCA is sensitive to all forms of outliers. This leads the distance between the subspace and the outliers to be small, and challenging for the anomaly detection problem. Therefore, this report will focus on robust Principal Component Analysis (RPCA). In [4], RPCA expands upon PCA to incorporate a unique parameter which can reduce the sensitivity to outliers. In [4], the focus is on the problem of recovery. The recovery problem focuses on recovering the true low-dimensional subspace which spans the given data. For this problem, the optimal value for the unique parameter is proven. For our problem of determining a low-dimensional subspace which detects anomalies the best, the optimal parameter is not known. Therefore, one focus of this report is on the determination of the optimal parameter in RPCA.

As previously mentioned, one method for detecting anomalies is to predict all data which does not lie on a low-dimensional data subspace to be anomalous. In some cases, not all of these outliers are actually anomalous. Therefore, detection/classification algorithms can be implemented to determine which of these outliers are anomalous. This report will also focus on improving the basic detection algorithm utilized in [2].

Once anomalies are detected, we create a web-based visualization system to assist users with exploring the data set and investigating the detected anomalies. Our system allows users to drill down to see specific details about the anomalies, and provides users with context between the anomaly and regular network traffic. Although anomaly detection and visualization of anomalies have both been studied previously, this report uniquely combines the two topics. This report is a novel product of the intersection of these two fields of anomaly detection and visualization within mathematics and computer science.

The highlights of our results include:

1. Feature Engineering with Network Knowledge to develop a set of features describing measurements from the network.

2. Development of an empirical parameter for Robust Principal Component Analysis (RPCA) that performs substantially better than the standard parameter provided by theory when applied to the anomaly detection problem.

3. Development of a new decision rule for detecting anomalies. The decision rule produces more accurate predictions of anomalies within the computer network.

4. Development of a novel visualization system for representing a network and the anomalies within the network. This visualization allows users to see and interact with the results of the anomaly detection algorithm to better understand significant attacks on a computer network.

In this report we will discuss the data set we used for this project as well as background information on related works in anomaly detection and visualization. We will also describe both the anomaly detection and visualization systems, the processes that we used to develop them, and how these systems, working in concert, can serve as a tool to assist analysts in the field of cyber security.

# Chapter 2

# Data Set

## 2.1 VAST 2011 Data Set

The VAST (Visual Analytics in Science and Technology) Challenge[1] provides a competition in visual analytics to advance the field. For this project, we have utilized the data from the VAST 2011 Mini Challenge 2 (MC2) data set. This challenge focused on a fictitious company: All Freight Corporation. The creators of the challenge provide a visualization for the company's enterprise as a whole as well as a solution document detailing the attacks that took place on the network. VAST 2011 MC2 provides firewall logs, packet capture (PCAP) files, and Snort Intrusion Detection System (IDS) logs for three days during which anomalies occurred. For this project, we used the firewall logs from this data set.

The Network diagram, shown in Figure 2.1, is provided by the issuers of the VAST 2011 MC2 data set. We used this figure to engineer our set of features from the original firewall log files.

The firewall logs that we analyzed from the VAST 2011 data set contained a three day span of regular network traffic as well as multiple instances of cyber security attacks carried out on the network. Originally we planned to use the PCAP data files from the VAST 2011 MC2, but we found that the attacks we were interested in were not extant in the PCAP data. This was because the attacks all involved packets from the Internet going to the company's External Web Server. The packet capture files only captured data that passed through the company's Cisco Switch. Because the attacks only went from the Internet to the External Web Server, they never passed through the switch, only the company's firewall. Therefore firewall logs were chosen for the anomaly detection algorithms and visualizations presented in this report.

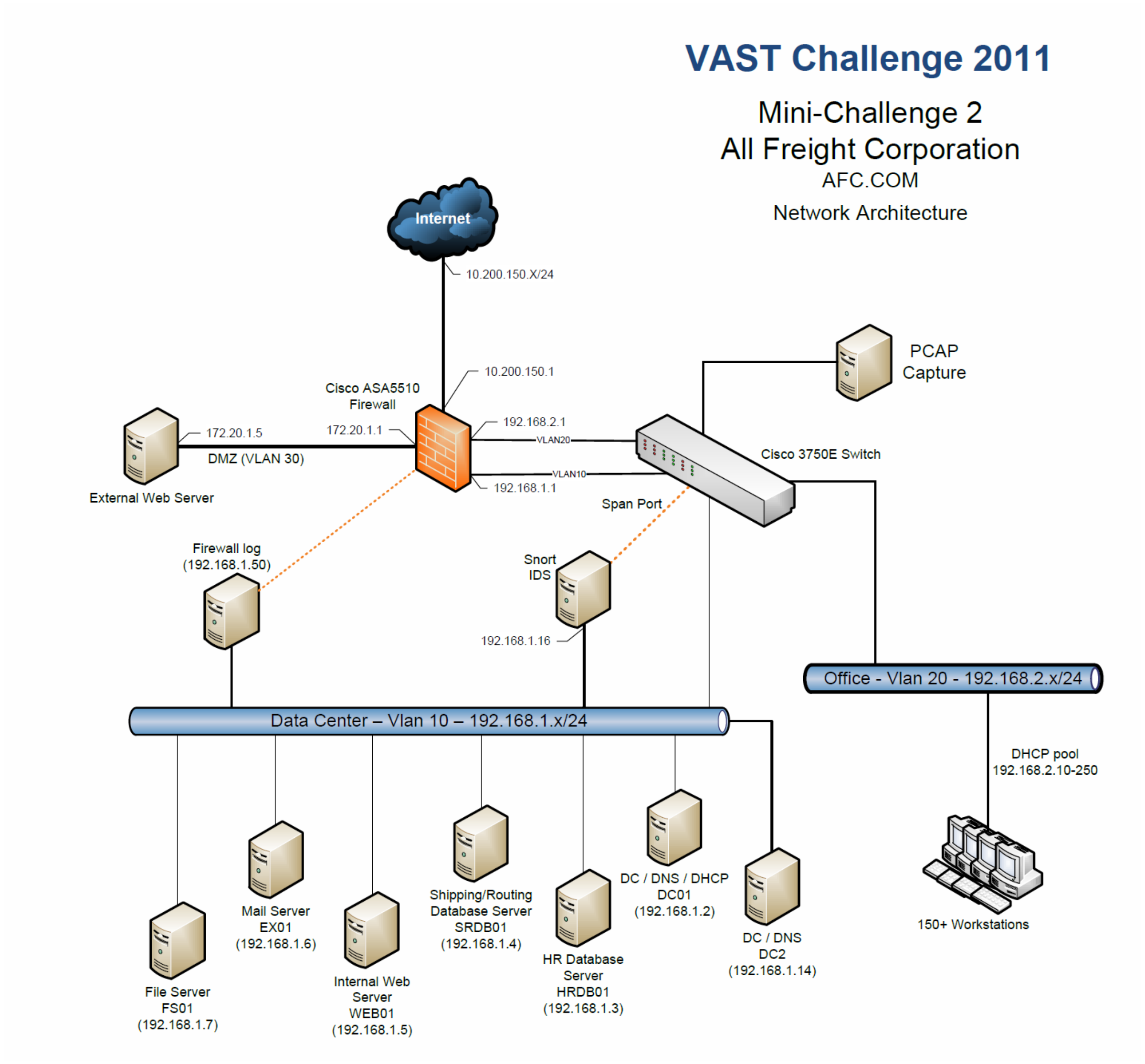

VAST Challenge 2011
Mini-Challenge 2
All Freight Corporation
AFC.COM
Network Architecture
Internet
10.200.150.X/24
10.200.150.1
Cisco ASA5510 Firewall
172.20.1.5
172.20.1.1
DMZ (VLAN 30)
External Web Server
192.168.2.1
VLAN20
VLAN10
192.168.1.1
PCAP Capture
Cisco 3750E Switch
Span Port
Firewall log (192.168.1.50)
Snort IDS
192.168.1.16
Office - Vlan 20 - 192.168.2.x/24
Data Center – Vlan 10 – 192.168.1.x/24
DHCP pool 192.168.2.10-250
150+ Workstations
File Server FS01 (192.168.1.7)
Mail Server EX01 (192.168.1.6)
Internal Web Server WEB01 (192.168.1.5)
Shipping/Routing Database Server SRDB01 (192.168.1.4)
HR Database Server HRDB01 (192.168.1.3)
DC / DNS / DHCP DC01 (192.168.1.2)
DC / DNS DC2 (192.168.1.14)

**Figure 2.1:** *Network Diagram - VAST 2011 Mini Challenge 2 [1]*

## 2.2 Ground Truth

The VAST 2011 MC2 data set was obtained from the Visual Analytics Benchmark Repository provided by the University of Maryland. The mini-challenge comes with a task description, a network diagram, and MC2 core data document. These documents describe the challenge and go into detail on the setup of the company and its network. The actual dataset consists of PCAP files, firewall logs, Snort IDS logs, and Nessus security scans for all three days, separated by day. It also contains an aggregated file of syslogs for all the hosts on the network. This mini-challenge comes complete with an answer guide which details each attack, gives context to the reader on where the attacks can be found, and provides examples of lines in the firewall log where the attacks appear. We refer to attack information provided by this document as ground truth information. All of the attacks that we focused on in this project were manifested in the firewall log, as explained in section 2.1.

The answer guide document for this data set allowed us to see which attacks on the network we would be able to pursue with our anomaly detection methods. We used this information to choose three attacks, which we will discuss in depth in Section 2.4. The knowledge of exact firewall logs that contained attacks was utilized in testing the accuracy of the detection algorithm.

## 2.3 Data Set Error

During the course of mapping attacks to the firewall logs they corresponded to, we realized that there was a disconnect between the description of the Remote Desktop Protocol policy violation and its manifestation in the firewall logs. The RDP attack was described as taking place only on day 2 of the data set, yet this attack spanned back one day prior to when it was described in the answer guide. Since the total size of all the firewall logs was so large, we avoided running queries on the entire length to prevent long wait times. Instead, we opted to map activity day by day. Due to discrepancies, such as this one in the Remote Desktop Protocol policy violation, we had to go back and manually ensure we did not miss any attack information outside of the days given by the answer key.

## 2.4 Significant Attacks

These firewall logs, provided by the VAST 2011 Challenge, contained connection status information about machines whose connections passed through the company firewall. They captured information on several attacks embedded within regular network traffic. These attacks were characterized by a violation of either company policies or United States law (Computer Fraud and Abuse Act). We focused on three distinct types of attacks, even though the data set included others. We chose to avoid attacks that involved social engineering, because the network data representations of these attacks would not be meaningfully different from normal traffic. The attacks that we chose to focus on were:

1. Denial of Service (DoS)
2. Unauthorized Remote Desktop Protocol (RDP)
3. Undocumented Computer added to the network

The Denial of Service attack occurred on day one of the firewall captures. This attack consisted of a sustained flood of network traffic originating from five separate Internet based IP addresses. This flood of network traffic amounted to 9,994,443 firewall logs in a one hour time span ( 166,574 connections per minute). This amount of network traffic was enough to bring down a single web server, assuming each log corresponded to a connection request. Connections from each IP were overlayed, although mostly separate such that each IP sustained their own flood of connections separately from the time of the others. The attack was likely orchestrated to slow down or impede use of the company's network. Because of the nature of the attack (a massive amount of packets/firewall logs), the attack was easily identified in the Network Overview component of our visualization, which displays traffic density over time.

The RDP attack consisted of 8 firewall logs, spanning two separate days, where a company machine was remotely connected to from an Internet IP address. This is an action that was prohibited according to company policy. This attack was much harder to spot from the simple overview traffic visualization, so we had to begin engineering features that could help us to identify this attack and others like it. The attack became apparent in the visualization when sorting traffic by firewall log attributes.

Much like the RDP attack, the Undocumented Computer Added to the network event consisted of very few firewall logs. There were three firewall logs spanning two minutes that captured the addition of this undocumented computer. These logs also did not differ as significantly from normal network traffic as the logs did in the previous attacks, so this attack proved the most difficult to detect.

To begin running anomaly detection and analysis on these attacks, we had to build a rich set of features to represent each firewall log.

# CHAPTER 3
# FEATURE ENGINEERING

## 3.1 Data Setup

Our data source consisted of the daily firewall log CSV files from the VAST2011 MC2, described in Section 2.1. Each log in the firewall files consisted of one row with 15 columns, or features. These rows were iteratively parsed and processed using csvparser in Python. The task of processing each log consisted of extracting a feature set in a format that would be useful to the anomaly detection algorithm. Originally, our feature set consisted of an enumeration of the fields from the .csv files and was tested by running Python's matplotlib Principal Component Analysis (PCA) functions on the output feature set.

In this report, we implemented a linear dimension reductions algorithm that was a modified version of Robust Principal Component Analysis. Since the algorithm implemented was linear, the format of our features needed to be one-hot encoded [5]. For this method of one-hot encoding, a feature whose response that was not binary was converted into several features with binary responses. For instance, a feature such as IP or port numbers, could have values that were numerically similar, (eg. port 20 and 22), yet meaningfully different (FTP (File Transfer Protocol) data transfer vs SSH (Secure Shell) logins). Therefore, changing the feature of port number to several features with binary responses permitted the linear dimension reduction algorithm to view these features as different, even though they were numerically similar.

Because of this, we transitioned towards a more uniform feature set. Each field had only a value of 0 or 1, and the features were things such as "Source IP: Workstation," where the value would be 1 if the firewall log had a source IP within the range of workstation IP's specified by the VAST 2011 MC2 network map. In this way, we created a feature set that had general coverage of the network and encompassed source and destination IP's, source and destination ports, and protocols. However, this left us with only around 40 features, which was not enough to run the algorithms for anomaly detection.

## 3.2 Insufficient Features Example

For the anomaly detection algorithm to accurately predict attacks, the firewall logs being predicted on needed to be distinguishable. Firewall logs that were known to be attacks needed to vary from the firewall logs that were known to not be attacks. The set of features was instrumental in ensuring this. If there were too few features describing the firewall logs, then some attack logs would have been identical to non-attack logs. This was the case with our set of 40 features, described in the previous section.

5,000 lines from the firewall logs were chosen during the first day of VAST 2011 data where there were no attacks. Of these 5,000 logs, all duplicate logs were eliminated. This left 33 unique logs in accordance to the features chosen. The anomaly detection algorithm, explained in section 4.3, used these logs as training data.

Additionally, 5,000 firewall logs were chosen from the second day, which included the RDP anomaly. Of these 5,000 firewall logs, one log was an anomaly. This set of logs also had duplicate logs removed such that only unique firewall logs remained. Of the 5,000 logs, there were 27 unique logs based off the features chosen. Each unique log was marked with the count of how many times the log occurred in the original slice of data. The remote desktop protocol anomaly log was represented as the 11th unique log out of 27.

The count of the 11th unique log was 2,409. This indicated that the attack log was identical to 2,408 other logs, which were not attacks, which we knew from the ground truth data. Therefore the anomaly detection algorithm had no method to detect between these identical logs because the features were not adequate to distinguish the anomaly from regular traffic.

Additionally, the attack log from the RDP attack was identical to logs within the original slice of data from day 1. The data from day 1 was what the algorithm was trained on and included no attacks. This implied that the attack log would never be marked as anomalous since the algorithm was trained on data that contained an identical log, and was told the data contained no attacks. Therefore the features were again not sufficient to detect the Remote Desktop Protocol anomaly.

From this information, it was imperative to improve the feature set to make sure no attack logs were identical to any non-attack logs.

## 3.3 Creation of Sufficient Features

In order to create a richer feature set we decided to make features programmatically, rather than extracting feature's solely based on their presence as a firewall log column. For every firewall attribute we extracted each unique value the column could take on and turned those values into another output feature. For example, priority can take on the following values: Info, Warning, Emerg (meaning emergency). These values were turned into the output features: isInfo, isWarning, isEmerg. Since we lacked administrative knowledge of the company network, we decided that using a script to automate this process on the entire firewall logs would be an appropriate substitute to manually covering all internal machines IPs, destination service ports, and Cisco message codes. This gave us full coverage of every IP range, destination services, and Cisco message codes used in the three day timespan. We used knowledge of destination service ports to structure additional features based on destination ports. This meant covering port ranges such as 0 - 1024 with additional higher ports for services such as RDP (port 3389) while avoiding a complete enumeration of 65535 port numbers or 65535 additional output features. This became an iterative processes as we went back and forth deciding which additional features were necessary to uniquely identify each known anomaly. The process eventually yielded a final set of 555 unique features, which were enough to run the algorithms necessary to train on the data and detect anomalies. The size of our eventual feature set meant that enough information was captured to make anomalous firewall logs distinct from regular firewall logs.

## 3.4 Slices of Firewall Logs

Once the set of features was created, we split up the firewall logs into small slices in order to run anomaly detection on each type of attack. From the three days of firewall logs, we analyzed five slices of 5,000 entries. These 5,000 entries contained information from the 555 features explained above. We needed a slice that covered each attack that we wanted to analyze, as well as two slices that captured only regular network traffic. The regular traffic slices were used in the training of the anomaly detection algorithm (explained in section 4.3).

### 3.4.1 Slice 1

The first slice of firewall logs was from day 1. This chunk of data included no known attacks and is used for training the anomaly detection algorithm on clean data.

### 3.4.2 Slice 2 (DoS)

This second slice of firewall logs was also from day 1. These logs included the Denial of Service (DoS) attack within them. We started this slice less than a second before the attack began. Additionally, because of the nature of DoS attack, for these 5,000 logs, 4,963 of the firewall logs were part of the DoS attacks. This chunk of data was used in the detection algorithm and is referred to as the DoS attack slice.

### 3.4.3 Slice 3 (RDP)

This third slice of firewall logs was from day 2 during the remote desktop protocol (RDP) anomaly. This included the two firewall logs which contained this anomaly. The rest of the slice was very similar to the "clean" data from slices 1 and 4. This slice of data was used in the detection algorithm and is referred to as the RDP attack.

### 3.4.4 Slice 4

This fourth slice was another slice of data, like the first slice, which contained no known anomalies. This data was pulled from day 3 and was also used to train the anomaly detection algorithm, along with slice 1.

### 3.4.5 Slice 5 (UC)

The final slice of data analyzed is a section of firewall logs containing the Undocumented Computer Added to the network on day 3. This section of data was determined from the solution set from the VAST challenge. Due to a typo in the solution document for the VAST 2011 MC2, the original section of logs that were indicated to contain the anomaly did not. After finding the actual time that the attack occurred in the firewall logs, we chose a slice of 5,000 firewall logs which contained the 4 logs related to the undocumented computer being added to the network.

## 3.5 Removing Duplicate Firewall logs

Once the five slices of firewall logs, each containing 5,000 logs with 555 features represented, were created they were modified to optimize the speed of the Robust Principal Component Analysis. In each of the slices containing 5,000 firewall logs, there were repeat logs. A repeat log is a log which is identical to a previous log based on the features chosen. These duplicate logs do not affect the dimension reduction algorithm and therefore are unnecessary to include until testing the accuracy of the predictions. With this understanding, new slices were created which contained only the unique firewall logs and kept track of the locations of the attacks.

A python program was written to read in the .csv file of original slices provided from the feature engineering. This .csv file is a 5,000 by 555 matrix where the rows are the firewall logs and the columns are the features. Each duplicate firewall log, or row, was removed. Each unique log was marked with the count of how many times the log occurred in the original slice and the new locations of known attacks were saved. This left the the slices in order with 1,212, 21, 106, 519, and 1,309 unique firewall logs. The 2,963 attack firewall logs in the DoS attack slice were all duplicates of 3 unique firewall logs. Otherwise, the two attack firewall logs in the RDP attack were unique as well as the four attack firewall logs in the undocumented computer added to the network attack.

# CHAPTER 4

# ANOMALY DETECTION METHODS

Once the slices of firewall logs were selected and duplicates were removed, the next step was anomaly detection. In this section, background information on dimension reduction algorithms, as well as the methods used in [2] for anomaly detection, will be presented.

## 4.1 PCA

Principal Component Analysis (PCA) can be used for several different analyses of data. PCA can be used to extract the most important information from a data set, compress the data while keeping the most important information, and simplify the data [6]. This report will focus on the dimension reduction, or compression methods, of PCA.

For this report, Y $\in \mathbb{R}^{m \times n}$ where each column vector, $[y_0,...,y_n]$, represents a firewall log from the VAST 2011 data set. Each column vector $y_i \in \mathbb{R}^m$ contains the m features of the firewall log determined by the feature engineering of the data set described in section 3.3.

Given a matrix Y $\in \mathbb{R}^{m \times n}$, Y can be factored into the form:

$$Y = U\Sigma V^T \tag{4.1}$$

where U $\in \mathbb{R}^{m \times m}$ is unitary, $\Sigma \in \mathbb{R}^{m \times n}$ is a diagonal matrix whose entries ($\sigma$) are the singular values of Y, and $V^T \in \mathbb{R}^{n \times n}$ is a unitary matrix whose columns are eigenvectors of Y [7]. Singular values are similar to eigenvalues but are not always equivalent to. Singular values and eigenvalues are only equivalent when the original matrix is a square, real, symmetric matrix with nonnegative eigenvalues (positive-definite matrix). A low dimensional projection of Y can be computed with singular values.

**Definition 4.1.1.** A low dimensional projection of Y, is a matrix $\hat{Y}$ such that $\hat{Y} \in \mathbb{R}^{k \times n}$ where k $\in \mathbb{R}$ and k < m.

Therefore dimensional reduction of Y produces a new matrix $\hat{Y}$ such that there are fewer rows, or features. This matrix is able to keep the important information from the original matrix. This matrix dimension reduction uses the equation:

$$\hat{Y} = \hat{\Sigma} V^T \tag{4.2}$$

[3] to rewrite Y as $\hat{Y}$. $\hat{\Sigma}$ is computed by setting the m-k smallest singular values ($\sigma_i$) to 0 (only the k largest singular values remain) and $V^T$ is the same as in equation (4.1). $\hat{\Sigma} \in \mathbb{R}^{m \times n}$, $V^T \in \mathbb{R}^{n \times n}$, and when multiplied $\hat{Y} \in \mathbb{R}^{m \times n}$. Due to the structure of $\hat{\Sigma}$ having the last m-k rows all be 0, the last m-k rows of $\hat{Y}$ are also all 0. Therefore these rows can be removed, and $\hat{Y} \in \mathbb{R}^{k \times n}$. This new $\hat{Y}$ is the low dimensional projection of Y.

Since equation (4.1) states that $U, \Sigma$, and $V^T$ can be computed for any matrix, any matrix can use the method in equation (4.2) for dimension reduction. For real data in m-dimensional space, each data point often falls within a lower dimensional subspace of the original data set. If this occurs, like in the figure 4.1, then there will be several singular values close to or equal to 0. A singular value close to 0 indicates that a given feature can be predicted by a linear combination of other features. These singular values can be turned to 0 and replaced in $\hat{\Sigma}$ using equation (4.2). This equation reduces the dimension of the data set without loosing information since the features eliminated can be predicted from remaining features.

These singular values close to zero occur in real data because real data is never truly random. A truly random matrix with entries between 0 and 1 will have data evenly dispersed between dimensions and therefore the singular values will not be near 0. This implies that randomly dispersed data can not be dimensionally reduced as seen in figure 4.2.

Thankfully, real data can often be dimensionally reduced [2]. When singular values are 0, this reduction is trivial, but deciding what $\sigma_i$ is small enough to remove depends on the circumstances of the problem. The main dilemma is determining how much data should be preserved. The number (k) of singular values kept are determined by the percent of information that must be retained. For instance if 90% of the data was to be preserved, summing the values, from largest to smallest, until k singular values whose sum was 90% of the total sum of the singular values. Then the algorithm would remove all of the (m-k) singular values that were not included in the top 90%. This is shown in the equation below where the original matrix had dimension m×n.

$$\sum_{i=1}^{k} \sigma_i = (.9) \sum_{i=1}^{min(n,m)} \sigma_i \tag{4.3}$$

Once these m-k $\sigma$'s were determined, they would be changed to 0 to create the $\hat{\Sigma}$. The corresponding rows and columns of zero's on the diagonal of matrix $\hat{\Sigma}$ would be 0 in $\hat{Y}$ when equation (4.2) is applied. This new $\hat{Y}$ preserves 90% of the variance of Y while being lower dimensional.

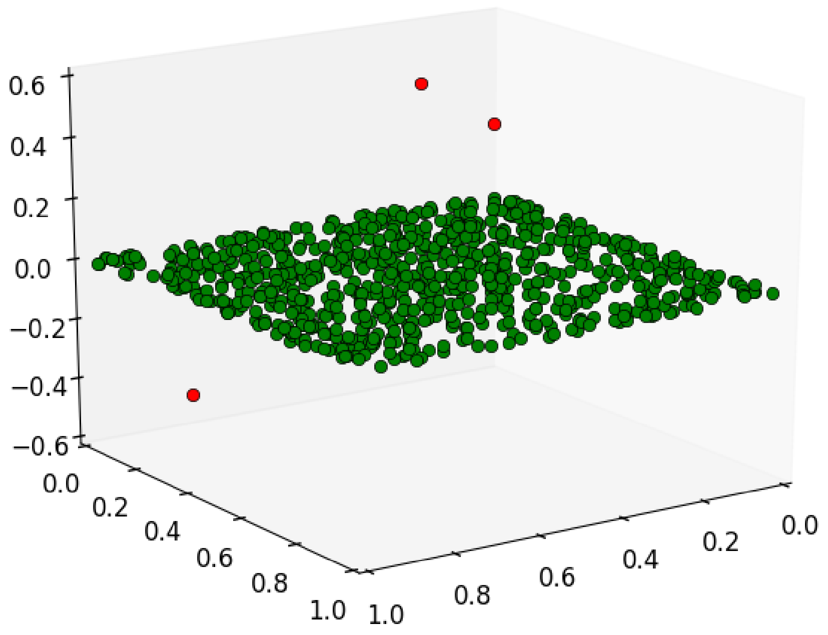


**Figure 4.1:** *In this figure the principal of PCA dimension reduction is depicted. The green dots are data points from a clean data set (no attacks). These points, although given in 3-dimensional space, actually fit almost perfectly into the 2-dimension subspace. Therefore, using PCA, the data can be compressed onto that plane with low error between the projected points and the original points. Once the low dimensional data is produced, anomalous logs can be determined if they do not lay on the green plane. The red points, which do not lay on the plane with the green dots, are marked as anomalous. Note: this image is a cartoon to depict PCA dimension reduction, the true data lies on much higher dimensional space*

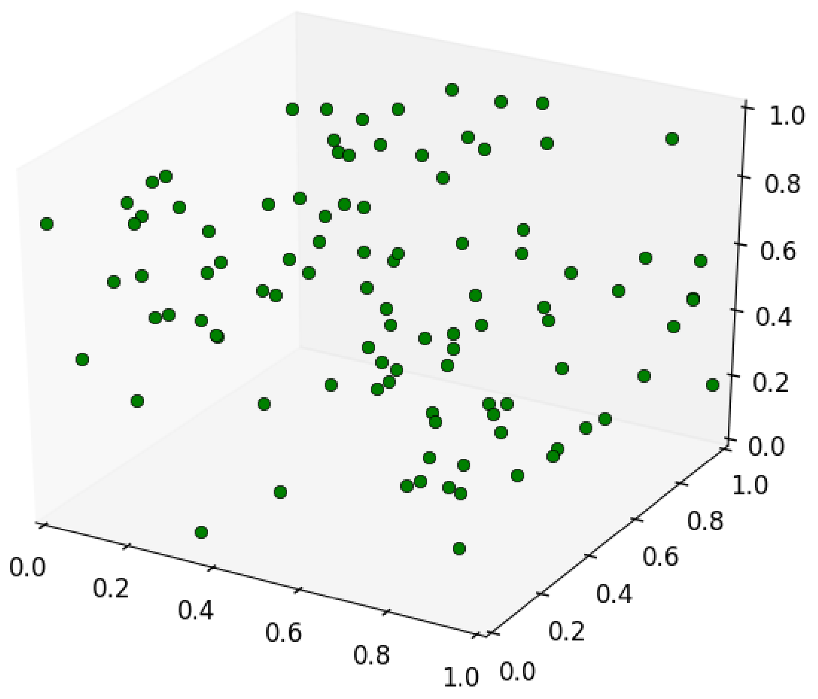


**Figure 4.2:** *In this figure random 3-dimensional data is depicted to represent that truly random data cannot be nicely fit into a lower dimension.*

### 4.1.1 Alternative to SVD PCA

Although SVD is the more traditional derivation of PCA, another approach will become useful for this report. Eckart and Young in their 1936 report [3] proved that the optimization problem in equation (4.4) is identical to the SVD method of PCA in equation (4.2). This optimization problem was the original method for PCA and involves a minimization problem in place of the more well known SVD. The same matrix Y $\in \mathbb{R}^{m \times n}$ is assumed. The columns of Y represent the firewall logs from the VAST 2011 data set and the rows are features for each firewall log. As with SVD, the goal of the minimization problem is to determine a k-dimensional subspace of Y such that the important information persists. To obtain the desired k-dimensional subspace, the minimization equation below can be utilized.

$$\min_{L} \|L - Y\|_F^2 \quad (4.4)$$

$$\text{subject to } \rho(L) < k$$

where L $\in \mathbb{R}^{m \times n}$, $\|L - Y\|_F^2$ is the Frobenius Norm of L-Y (sum of the squares of entries of L-Y),

$\rho$(L) is the rank of L (number of nonzero singular values of the matrix L) [2**?** ]. The methodology is to minimize the error between the original points and the projected points to create the most accurate data in a low dimensional subspace. This method provides a more flexible framework for improving the algorithm to create Robust Principal Component Analysis, the dimension reduction algorithm used in this report.

## 4.2 RPCA

Robust Principle Component Analysis (RPCA), is another form of dimension reduction of data. While PCA is sensitive to outliers, RPCA can detect a more accurate representation of a low dimensional space [2]. Figure 4.3 depicts a situation where PCA would not be able to handle the data as well as RPCA. Note, this sensitivity to outliers in PCA can not easily be fixed by choosing another value for k. In PCA, every singular value ($\sigma_i$) would be transformed by a single outlier. For example a singular value in nominal data that would be close to 0 could change to arbitrarily large if even a single outlier was added to the data set. Therefore RPCA is necessary to handle this situation.

The problem of RPCA is to determine a low rank matrix L and sparse matrix S whose only entries are the anomalies from Y. This problem must be solved without knowing k (the dimension of L) or the location of the anomalous entries in S. In [4], an optimization problem was found to solve RPCA. This optimization problem is:

$$\min_{L,S} \rho(L) + \lambda \|S\|_0 \tag{4.5}$$

$$\text{subject to } |Y - (L+S)| = 0$$

In this equation, L is the low rank matrix, $\rho$(L) is the rank of L, $\lambda$ is the coupling constant (trade off between L and S), and $\|S\|_0$ is the number of nonzero entries in S (ie the number of anomalies detected [4]. Unfortunately this equation, although would result in the correct discovery of L and S without knowing k or the locations of the anomalous entries in S, has no closed form solution. Additionally a brute force approach would lead to an exponentially hard problem to solve. In the 2011 paper from Candes, Ma, and Wright [4], a few requirements were found, for the optimization problem in equation (4.5), such that the discovery of L and S was guaranteed. Other works such as [8–12] provide details on the guarantees for recovering L and S matrices. By including a few additional requirements for the equation (4.5), the optimization can be solved. The requirements are:

1. $\rho$(L) is not too large

2. S is reasonably sparse

3. require the columns of L to be incoherently far from the Standard basis

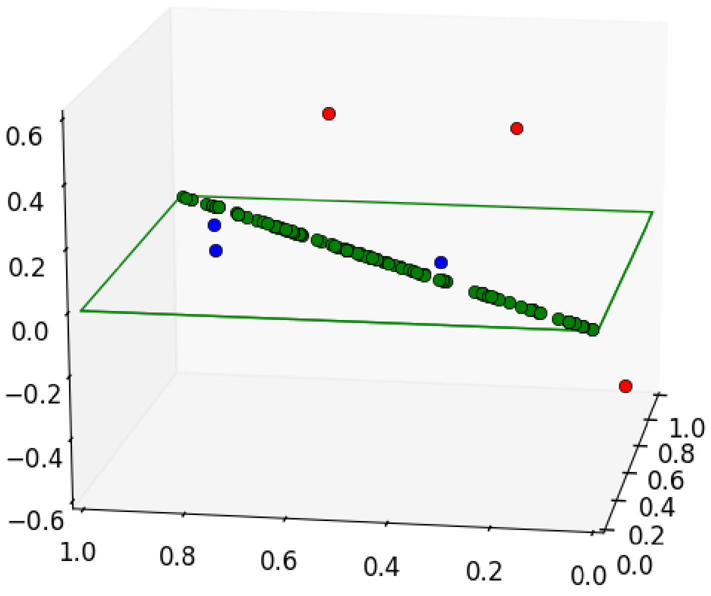


**Figure 4.3:** *This figure represents a possible set of data in 3-dimensional space. The training data includes green and blue points which are all contained on the green plane. Therefore PCA would reduce the 3-dimensional space to the 2-dimensional subspace of the green plane. In contrast RPCA is able to recognize that the majority of the data is actually on a 1-dimensional line with very few outliers. Therefore RPCA is able to reduce the dimensions even more than PCA would be able to recognize. Both algorithms would mark the red points as anomalies, but RPCA is able to determine the blue points are also slightly anomalous even though they are not specifically attacks. Note: this image is a cartoon to depict PCA dimension reduction, the true data lies on much higher dimensional space*

4. require non-zero entries of S to be distributed evenly

These requirements create a solvable minimization problem, although these requirements may not be possible to fulfill in real data.

With these requirements met, a solvable RPCA equation can be created [4].

$$\min_{L,S} \|L\|_* + \lambda\|S\|_1 \tag{4.6}$$

$$\text{subject to } |Y - (L+S)| \leq \epsilon$$

This convex relaxation can recover L and S. In this equation, $\|L\|_*$ is the nuclear norm of L (sum of the singular values of L), $\|S\|_1$ is the sum of the entries of S, $\lambda$ is the coupling constant, and $\epsilon$ is the set of point-wise error constraints to improve the noise from real world data. In Theorem 1.1 in [4], the theoretical best $\lambda$ was shown to be $\frac{1}{\sqrt{max(m,n)}}$. Similar results were also found in [12]. This $\lambda$ perfectly recovers the low-rank and the sparse components of a matrix, but recovery is not the problem this report is addressing. Indeed, in the context of anomaly detection the true low-rank and sparse matrices are not the most important piece of information, but rather the correct predictions of anomalies is important. Therefore the theoretical $\lambda$ may not be the best $\lambda$ for the problem of anomaly detection. The selection of a new $\lambda$ is one of the main focuses of this report.

## 4.3 Previous Work

The work of this report is a continuation off of the discoveries made in [2]. [2] improved the RPCA optimization through careful selection of $\lambda$. The coupling constant ($\lambda$) determines how many outliers, or anomalies, are put into the S matrix. If $\lambda$ is large, then no anomalies are placed in the S matrix and the matrix is empty. This is equivalent to PCA. Otherwise, if $\lambda$ is sufficiently small then any slight anomaly will be placed in S. Therefore the paper [2], showed that choosing a $\lambda$ trained on the data was able to have a more accurate prediction rate on the testing data than the theoretical $\lambda$ given by [4] for the recovery problem. The process for implementing the work performed in [2] is described in the algorithm below.

1. Select $Y_0$, an initial slice of data without an attacks present. The slice will include firewall logs as columns and features of the firewall logs as the rows.

2. Compute a $L_0$ and $S_0$ using equation (4.6) and several possible $\lambda$'s. (Note: $S_0$ will not be empty if $\lambda$ is not sufficiently large since RPCA will predict anomalous data in $S_0$, not just attack data)

3. Select $Y_a$, another slice of data which contains attacks. The slice will have the same number of features as $Y_0$ and contain attack firewall logs as well as not attack firewall logs.

4. Project $Y_a$ onto the subspace spanned by each $L_0$ (one $L_0$ for each $\lambda$ tested) to compute $L_a$.

5. Determine the sparse anomalous matrix for $Y_a$ by setting $S_a = Y_a - L_a$

6. Columns, or firewall logs, in $S_a$ are flagged as anomalous based on a detection algorithm. The detection algorithm in [2] originally flagged anomalies if the largest entry in the column was greater than a threshold $\alpha$. The best accuracy at predicting anomalies of the $S_a$'s was the chosen $\lambda$. This detection process will be explained more in depth in section 7, Detection Algorithms.

This report will use the dimension reduction of RPCA with a trained $\lambda$ to produce a projection of the original data onto a lower dimensional subspace. This projection will minimize the error between the original points and the projected points to have the most accurate, low dimensional data. This lower dimensional data can then be used for a classification problem to determine if a test firewall log is anomalous or not.

# Chapter 5

# Anomaly Detection Algorithm Setup

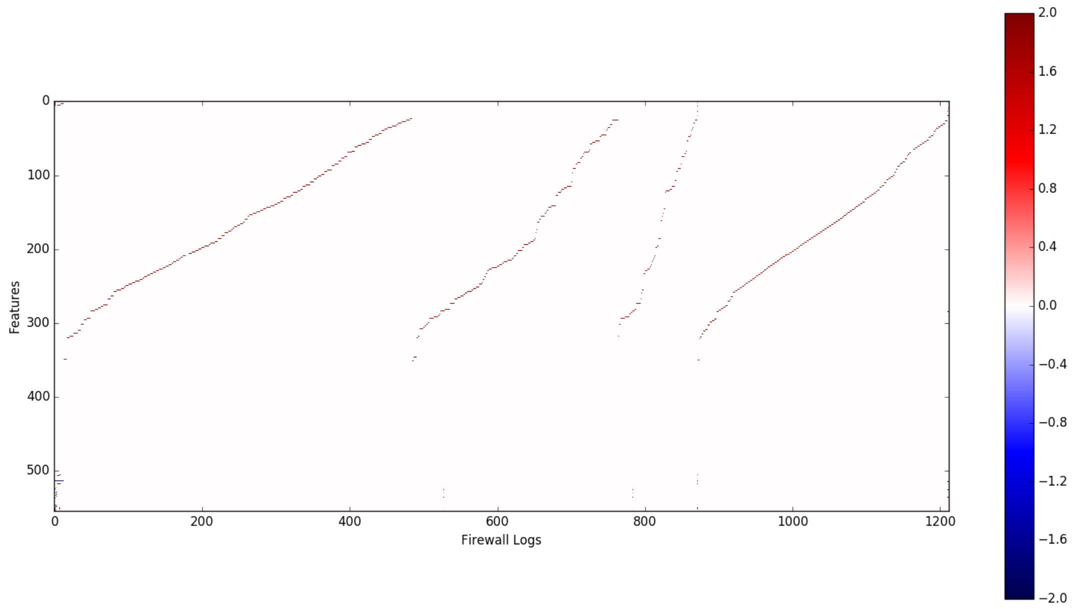
In order to use the modified RPCA algorithm to create a classification problem to detect anomalies, the data must be set up appropriately. First, data without any anomalies must be used to train the modified RPCA algorithm. This is the creation of $Y_0$ from the first step in the algorithm defined in section 4.3. This data is taken from the first day of the VAST 2011 data set during a time period where no attacks occurred and combined with data from the third day of the data set during a time when no attacks occurred. From the VAST 2011 data set, 5,000 firewall logs from each section of data were converted into a matrix. This matrix, known as $Y_0$, has firewall logs as columns and features of the firewall logs as rows. Duplicate firewall log data was then removed and counted using the process defined in section 3.5. After duplicate firewall logs were removed from the $Y_0$ matrix, the matrix contained 555 features and 1,368 firewall logs.

## 5.1 Normalization

Once $Y_0$ was created, the data needed to be normalized. Normalization of data makes variability in the data more significant. For this normalization, we used the z-score normalization presented in [13]:

$$\frac{x_i - \mu}{\sigma} \tag{5.1}$$

$x_i$ is the row of the $Y_0$ matrix (i.e. a feature), $\mu$ is the mean of the elements in $x_i$, and $\sigma$ is the standard deviation of the elements in $x_i$. Since our features are one-hot encoded, all elements in the $Y_0$ matrix are either 0 or 1 as explained in section 3.1. A possible feature could be not true (0) for almost all firewall logs except for a couple firewall logs in $Y_0$. Normalizing this feature would lead to the common 0 entries staying close to 0, but the few 1 entries would be larger. The other extreme is a feature that is not true (0) for a few firewall logs and true (1) for most firewall logs in $Y_0$. Normalizing this feature would lead to the previous 0 entries to become farther from 0 and the previous 1 entires to become close to 0. This normalization makes the infrequent entry of a feature more prominent to the RPCA algorithm. This normalization allows the RPCA algorithm to more easily identify variations and anomalies in the data. The normalized v.s. not normalized graphs of $S_0$ can be found in figure 5.1.



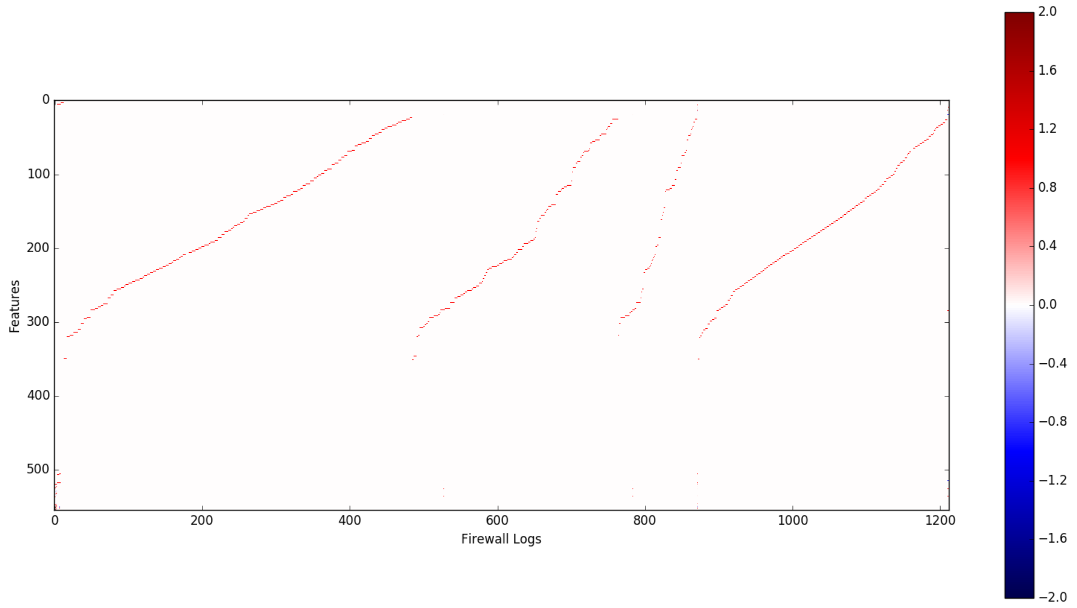


**Figure 5.1:** *In this figure the normalized $S_0$ matrix is on top where every row (feature) is normalized using the z-score normalization. The values of the data are slightly higher in the normalized data matrix. In the bottom heat map the $S_0$ matrix is not normalized. The values are slightly lower (less saturated red). The shape of the data in the matrices does not change, just the values.*

## 5.2 Apply RPCA

Once $Y_0$ is determine and normalized, the modified RPCA algorithm was used to decompose the matrix $Y_0$ into two matrices, $S_0$ and $L_0$. This decomposition, computed for a variety of different values for $\lambda$, was determined by equation (4.6). The decomposition for $Y_0$ is:

$$Y_0 = L_0 + \lambda S_0 \tag{5.2}$$

$L_0$ will be a low rank matrix and $S_0$ is the sparse matrix containing the anomalies of $Y_0$ further explained in section 4.2 [2]. Although $Y_0$ contains no attacks, not all of the logs are the same. There are rare events that occur in $Y_0$ that are not attacks, but are picked up in $S_0$. Therefore even though there are no attacks in $S_0$, the matrix is not empty. Additionally $\lambda$, defined in section 4.2, determines how expensive it is to put anomalies into the $S_0$ matrix. As the value of $\lambda$ increases, the number of elements placed in $S_0$ decreases. Therefore, the variety of possible $\lambda$ values determine the sensitivity of the algorithm.

For this report, $L_0$ and $S_0$ were determined using 20 different $\lambda$'s between .025 and 1. This step is equivalent to the second step in the algorithm defined in section 4.3. This range of possible $\lambda$'s were determined since the minimum $\lambda$ was small and close to the theoretical $\lambda$ discovered in [4]. The given $Y_0$ matrix contained 1,368 unique firewall logs. Therefore the theoretical best $\lambda$ for the recovery problem was $\frac{1}{\sqrt{1368}} \approx 0.027$ which was rounded down to .025, the minimum $\lambda$. Additionally, the maximum $\lambda$ of 1 was determined by the closes $\lambda$ to when $S_0$ was empty. This large $\lambda$ is equivalent to PCA. Figure 5.2 is a graphic depicting possible $S_0$ matrices with a variety of possible $\lambda$ values. For this graph, the there are 30 firewall logs (columns) and about 50 features (rows). This small portion of data was used to visualize possible $S_0$'s as $\lambda$ varied.

Once $L_0$ and $S_0$ matrices were computed for each of the twenty $\lambda$'s, step 3 in the algorithm in section 4.3 was completed. A new piece of the VAST 2011 Challenge Data containing 5,000 firewall logs and one type of attack is put in the same form as $Y_0$. This new matrix ($Y_1$) has features as rows and firewall logs as columns, is normalized as described in section 5.1, and duplicate firewall logs are removed using the method in 3.5. This $Y_1$ matrix is equivalent to the $Y_a$ matrix described in the third step of the algorithm in section 4.3.

Next, the new slice ($Y_1$) is projected into the same low dimensional subspace spanned by each $L_0$ to return a new matrix $L_1$ for each $\lambda$. This is equivalent to extracting the nominal part of $Y_1$. Then to compute $S_1$, the equation below can be used [2].

$$S_1 = Y_1 - L_1 \tag{5.3}$$

This computation of $S_1$ is the fifth step in the algorithm defined in section 4.3. For this report, the detection algorithms are trained on $Y_1$ and then tested on new Y matrices. These new Y matrices contain the other two types of anomalies and are set up in the same way as $Y_0$ and $Y_1$. The detection algorithms are trained on $Y_1$ to determine which $\lambda$ produces the most accurate predictions on $Y_1$. Once the algorithm is trained and the best $\lambda$ is selected, the best $\lambda$ is then

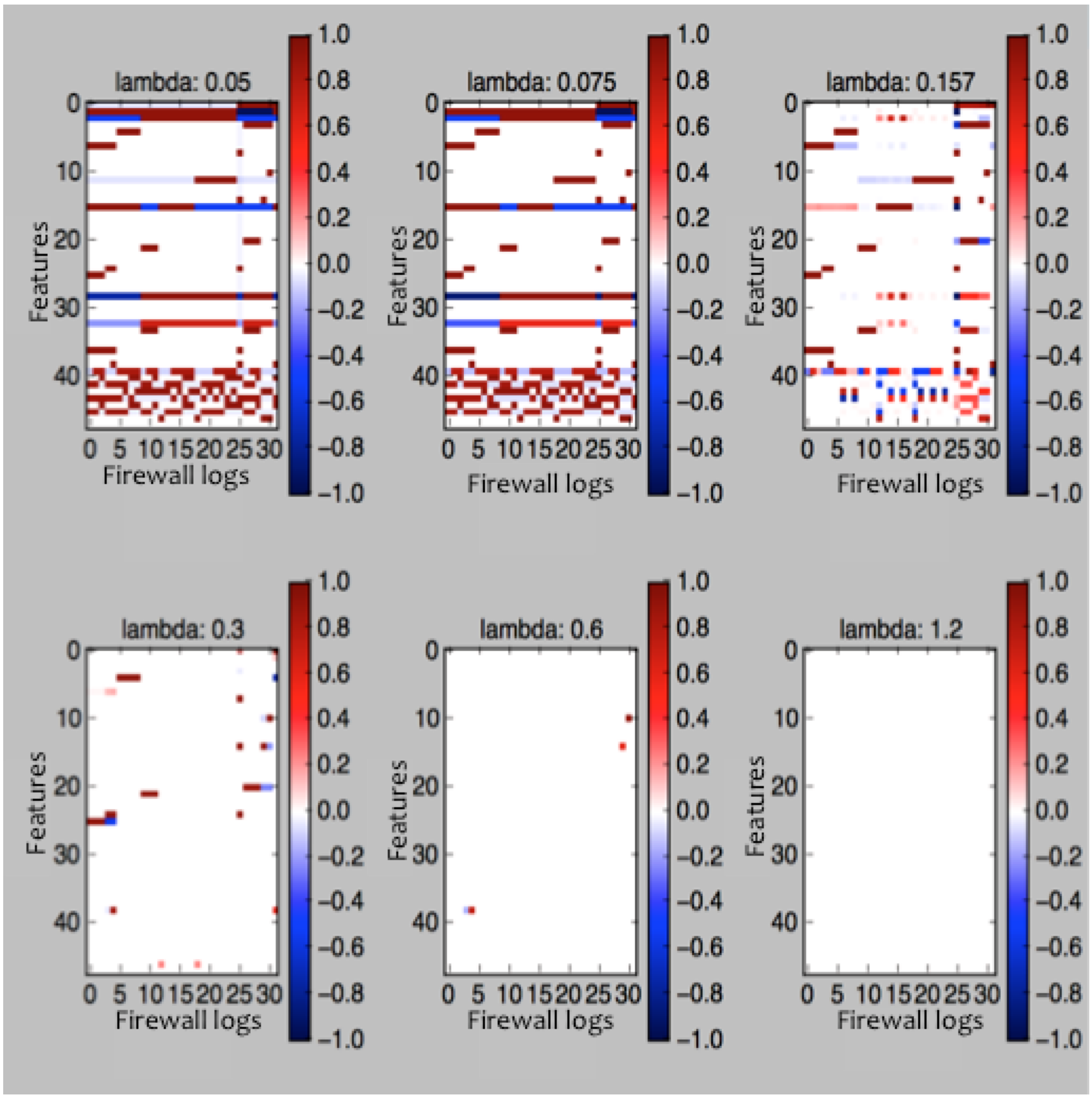


**Figure 5.2:** *This figure represents a possible $S_0$ matrix and its variations as $\lambda$ increases. The top left heat map is a small $\lambda$ which represents the theoretical $\lambda$ for the recovery problem. With a small $\lambda$, $S_0$ is more dense as more outliers are placed into the $S_0$ matrix. In the bottom right corner heat map has a large $\lambda$ which represents PCA. With a large $\lambda$, the $L_0$ matrix will contain all of the elements and therefore no outliers will be placed in the $S_0$ matrix.*

applied to create a new $Y_2$ and $Y_3$ for which $S_2$ and $S_3$ are calculated, as explained earlier, for the detection algorithm to test the accuracy of anomaly predictions on. The accuracy on the new Y matrices is the true accuracy of the prediction model. This process is explained in further detail in chapter 7.

# Chapter 6

# Possible Detection Algorithms

Once the matrices for each slice of firewall logs were projected into a low dimensional subspace and the S matrices were defined, a detection algorithm was needed to identify the anomalies. In the previous paper by Kathleen Kay, Randy Paffenroth, and L Servi [2], a Threshold Detection algorithm was used as the classification algorithm applied to the S matrices. Threshold Detection will be further explained in chapter 7. This classification algorithm is simple, and a novel aspect of this report is implementing a more complex classification algorithm on the S matrices. In this section several classification algorithms will be defined for which the final classification algorithm of random forests was implemented and tested in this report.

## 6.1 Classification Trees

The first type of classification algorithm presented in this report is classification trees. Further information of classification trees can be found in [7, 14]. The output of classification trees are qualitative values, for our purposes the responses are anomalous and not anomalous. These trees could be used to split up the final S matrix into categories (branches) based on the features to determine if a specific column (firewall log) is anomalous or not. A firewall log would be placed into one branch of the tree based on its subset of features which matched the subset of features for the branch of the tree. Then the firewall log would be marked as anomalous if the majority of training points with the same subset of features were also anomalous. This would lead to a classification error rate which is the percent of training firewall logs in the branch which differed from the response of the branch. The equation for the classification error rate is below [7]:

$$E = 1 - \max_k \hat{p}_{mk} \tag{6.1}$$

where $\hat{p}_{mk}$ is the proportion of firewall logs in the $m^{th}$ region (branch) that are from the $k^{th}$ class. The $k^{th}$ class is either anomalous or not anomalous and is determined by which response is more prominent in the current branch. Therefore if 40 firewall logs were in one branch of the classification tree based on their features, and 35 of the firewall logs were anomalous, that branch would be anomalous and the classification error rate would be $\frac{5}{40}$=.125. However, classification error rate is not sensitive for tree growing and the tree would be unbalanced. Therefore the Gini index, or cross-entropy measures, are preferred. The Gini index is below: [7]:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{6.2}$$

The Gini index ensures a balanced tree where $\hat{p}_{mk}$, m, and k are the same as the classification error rate. If the $m^{th}$ branch is pure (all firewall logs are of the same classification) then the Gini index will be the lowest. Therefore minimizing the Gini index ensures that the tree will be balanced and accurate. The Gini index ensures that a significant majority of firewall logs in a single branch of the tree will be of a single classification. The other measure which is commonly used to ensure a balanced tree is cross-entropy. The equation for cross-entropy is below: [7]:

$$D = \sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk} \tag{6.3}$$

The variables for the cross-entropy measure are the same as Gini index and also ensure a balanced tree. When the $m^{th}$ branch is pure the cross-entropy measure is also the lowest. Therefore the Gini index and cross-entropy measures are numerically similar when the branches are pure. Any of the three measures: classification error, Gini index, or cross-entropy can be used to prune a tree.

Pruning a tree is when the entire tree is created and then the tree is cut so only the top portion of the tree remains. The final size of the pruned tree can be obtained by cross-validation (testing on different subsections of the training data to determine which final size of the tree is most accurate in predictions). A full tree is prone to overfitting to the training set, but pruning a tree reduces variance. Variance is the amount by which a statistical learning model changes for different training data [7].

A high variance causes small changes in the training set which results in large changes in the statistical learning model. This can effect the efficiency of the model on the testing data. A low variance can increase the accuracy of a model. Although, there is a trade-off as the variance of a model decreases. As variance decreases, bias of the model increases.

Bias is the error associated with using a model that is too simple for the complexity of a data set [7]. For example, using a linear regression model on complicated problem would probably result in bias, since it is unlikely a real-world problem would have a strictly linear relationship. Therefore, by assuming the relationship is linear, bias is introduced into the problem. This may also occur in more complicated problems.

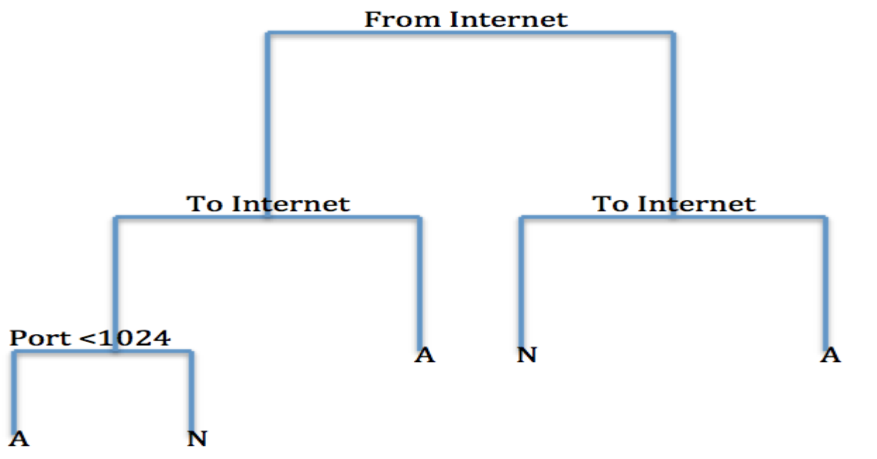


**Figure 6.1:** *This figure is an example of a possible pruned classification tree for the VAST 2011 data set. This tree has the first split on whether or not the firewall log is from the Internet. If the log is from the Internet then it follows to the left, if it is not from the Internet then it follows to the right. The next subsection is whether or not the firewall log went to the Internet. Again, left means the log did go to the Internet and right means the log did not. Finally, if the firewall log is goes to and from the Internet, then the port number is checked. The letters at the bottom indicate whether or not the majority of the training firewall logs in each branch were anomalous or not. Where A stands for anomalous and N stands for not anomalous. This tree is pruned since, in full tree, all 555 features would be utilized as splits in the tree, yet this tree only looks at the top 3 splits.*

A statistical model must balance the trade-off of variance and bias. Classification trees can be improved to reduce the variance without increasing the bias of the model significantly. One method to reduce variance is through bagging.

## 6.2 Bagging

Bagging can be applied to improve statistical learning methods by decreasing variance. Classification trees have a high variance since results vary greatly based on the training data given. For instance if the data for the tree was split into two, and a unique tree was fit to both data sets, the results for each tree could be very different. If numerous trees were fit to unique data sets and the results were averaged, the variance would be lower. Bagging is the processes of fitting several models and taking the average of their results in order to reduce variance. More information on bagging can be found in [7, 15, 16].

Assume there is a set of observations, in our case this would be a training set of firewall logs. This set has a variance of $\sigma^2$. This variance can be reduced by averaging the several (n) sets of data to get a variance of $\frac{\sigma^2}{n}$. This new variance is smaller than the original variance of $\sigma^2$. Therefore averaging the training sets of data decreases the variance of the data. A simple example of bagging applied to the VAST 2011 data can be found in figure 6.2.

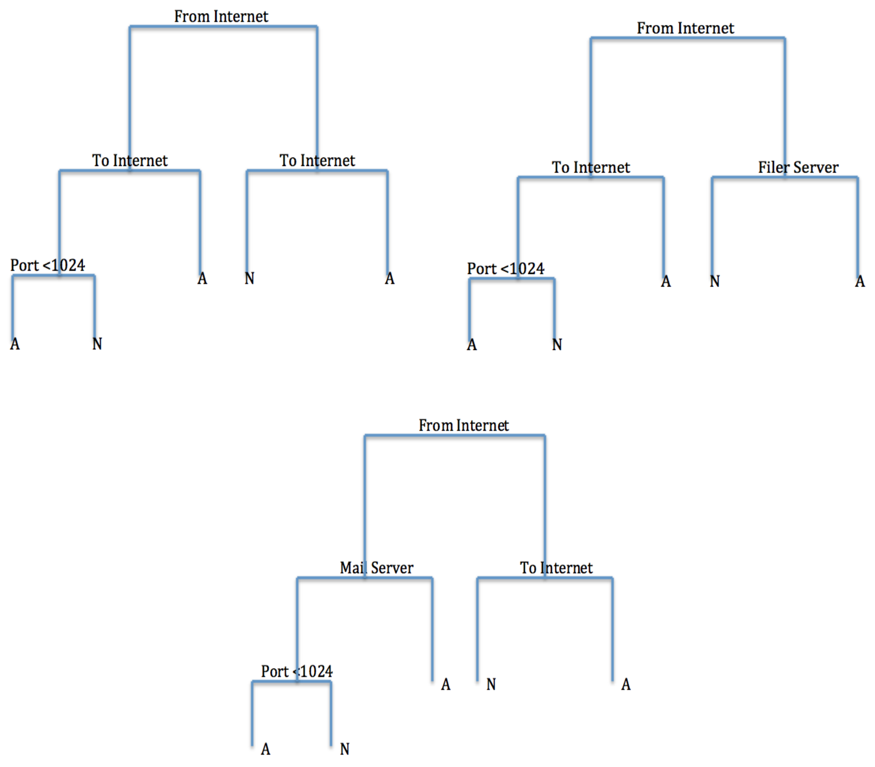


**Figure 6.2:** *In this figure is an example of bagging for the VAST 2011 data set. Each of the above trees would have been fit to a new set of data. It is important to note that a true bagging example would not include pruned trees such as this example, but the pruned trees are necessary to reasonably visually represent the features. Then to determine if a new firewall log was an anomaly the average response from the trees would be taken. Therefore if a new log was from the Internet, not to Internet, to a mail server, and had a port number larger than 1024, the first and second trees would predict the new log was an anomaly while the third tree would predict that the the log is not an anomaly. Therefore the overall prediction of bagging would be $\frac{2}{3}$ change the log was anomalous based on the trees.*

Unfortunately, acquiring n additional training sets of data to perform this average on is not a viable solution. Therefore bootstrapping can be used to create additional data sets needed for bagging to decrease the variance of a model.

## 6.3 Bootstrapping

Bootstrapping is the process of taking repeated random samples from one data set to get many smaller sets. This can be accomplished by random sampling with replacement. This is depicted in the figure 6.3. This method of bootstrapping data can be performed numerous times to get new data. From these newly created data sets, bagging can be applied to the several sets created from the original data. Bootstrapping eliminates limitations of bagging by creating new

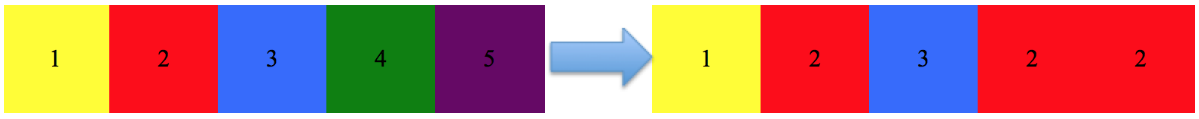


**Figure 6.3:** *In this figure an example of bootstrapping, sampling with replacement, is shown. The left rectangle is a set of 5 data points to be bootstrapped. The right rectangle is a possible bootstrapped data set. The new data set is the same length as the original, where each new point is a random selection from the entire original set. Therefore there can be repeats in the new bootstrapped sample.*

data sets from the original instead of having to obtain more original data. This method could be applied to the VAST 2011 data set by taking a bunch of random samples with replacement from the training set and using them as unique training sets to average and therefore reduce variance.

From B different bootstrapped sets, B classification trees can be calculated and averaging the results will improve the variance of the data. For the best results, these trees are not pruned and are deep to increase the variance and decrease the bias. The equation for this single, low-variance statistical learning model is below, where $\hat{f}^{*b}(x)$ is each classification tree [7].

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x) \tag{6.4}$$

## 6.4 Random Forest Classifier

Random Forest Classifier builds upon the strategy used in bagging and is the final method chosen for the classification algorithm in this report. Random Forest Classification was proposed by Leo Breiman in 2001 [17]. The Random Forest Classifier takes the average of several classification trees, like bagging, but focuses more on how the classification trees are built from the bootstrapped data. In bagging, the trees are built using all predictors and the best predictor is always chosen for each split in the tree. If there is one strong predictor, then most trees will use this predictor first and the trees will not vary too much. Random Forest Classifier ensures the trees vary from each other so that taking the average of the trees reduces the variance of the model. Ensuring the variance of the trees is accomplished by only using a random sample of m predictors for each tree. m < p (total number of predictors) and for random forests m is usually $\sqrt{p}$ [7, 17]. This allows for trees to use different first predictors and therefore allows the trees to vary more. This variance in the trees reduces the overall variance of the statistical learning model created using equation (6.4).

The random forest trees in figure 6.4 are different from the bagging trees in figure 6.1 because each tree only contains a subset of the features. In the bagging example, the from Internet feature is much stronger than any other feature and is continuously the first feature of the tree. This provides less variability in the trees, but the random forest doesn't always have the from Internet feature first and therefore the trees are much more variable. This variability is the

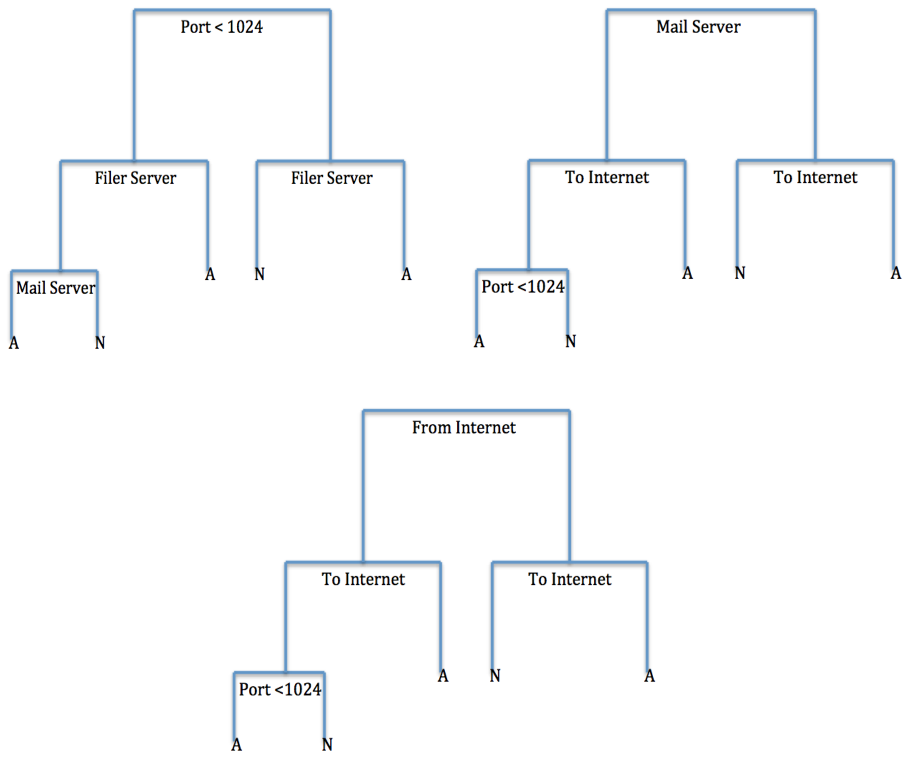


**Figure 6.4:** *In this figure is a small example of a random forest on the VAST 2011 data set. Each of the above trees would have been fit to a new set of data. Each of these trees only had 3 of the 5 total features used throughout the trees. For instance, the first tree only has port < 1024, Filer Server, and Mail Server. Then the second tree has a different subset of features including Mail Server, To Internet, and Port < 1024.*

unique addition given by random forests. Once the forest is created, the process to determine if a new log is anomalous or not is the same as for Bagging.

# Chapter 7

# Implemented Detection Algorithms

Once the $S_1$ values were determined for each possible $\lambda$ (as explain in section 5.2), a detection algorithm had to be trained on $S_1$ to determine which $\lambda$ to use to calculate $S_2$ and $S_3$. The detection algorithm was then utilized to determine the accuracy of predictions on $S_2$ and $S_3$. The detection algorithm determined which columns (firewall logs) of $S_2$ and $S_3$ were anomalies. Three separate anomaly detection algorithms were tested in this report.

## 7.1 Threshold Detection

The most basic detection algorithm utilized in this report is threshold detection. This detection algorithm is the same as the detection algorithm used in [2]. The main principal of this detection algorithm determines a column of S to be anomalous if the maximum element in the column, or infinity norm, is above some threshold ($\alpha$). The best $\lambda$ and $\alpha$ are both unknown and therefore need to be determined from data.

To simplify the number of $\alpha$'s tested, a small amount of data snooping was performed to determine the best $\alpha$. For each firewall log in $S_1$, the max elements was determined. The known attack columns were noted and the max elements of the known attack columns were recorded. From there the smallest of these elements was chosen as the $\alpha$ for the set. With this $\alpha$, the true positive rate (TPR) and false positive rates (FPR), were recorded. This process was completed for each $\lambda$ and $S_1$ pair and the $\lambda$ which produced the highest TPR and lowest FPR was selected as the $\lambda$ for the algorithm.

In our results for threshold detection, we trained the detection algorithm on the Remote Desktop Protocol attack. For this attack, $\lambda$ = .175 and $\alpha$ = 2.284. $\alpha$ was tested between 2.23 and 2.28 and an $\alpha$ = 2.26 produced a TPR = 1 and FPR = 0. A small section of this $S_1$ matrix can be found in figure 7.1. The dark red areas represent max elements over the threshold of $\alpha$, therefore

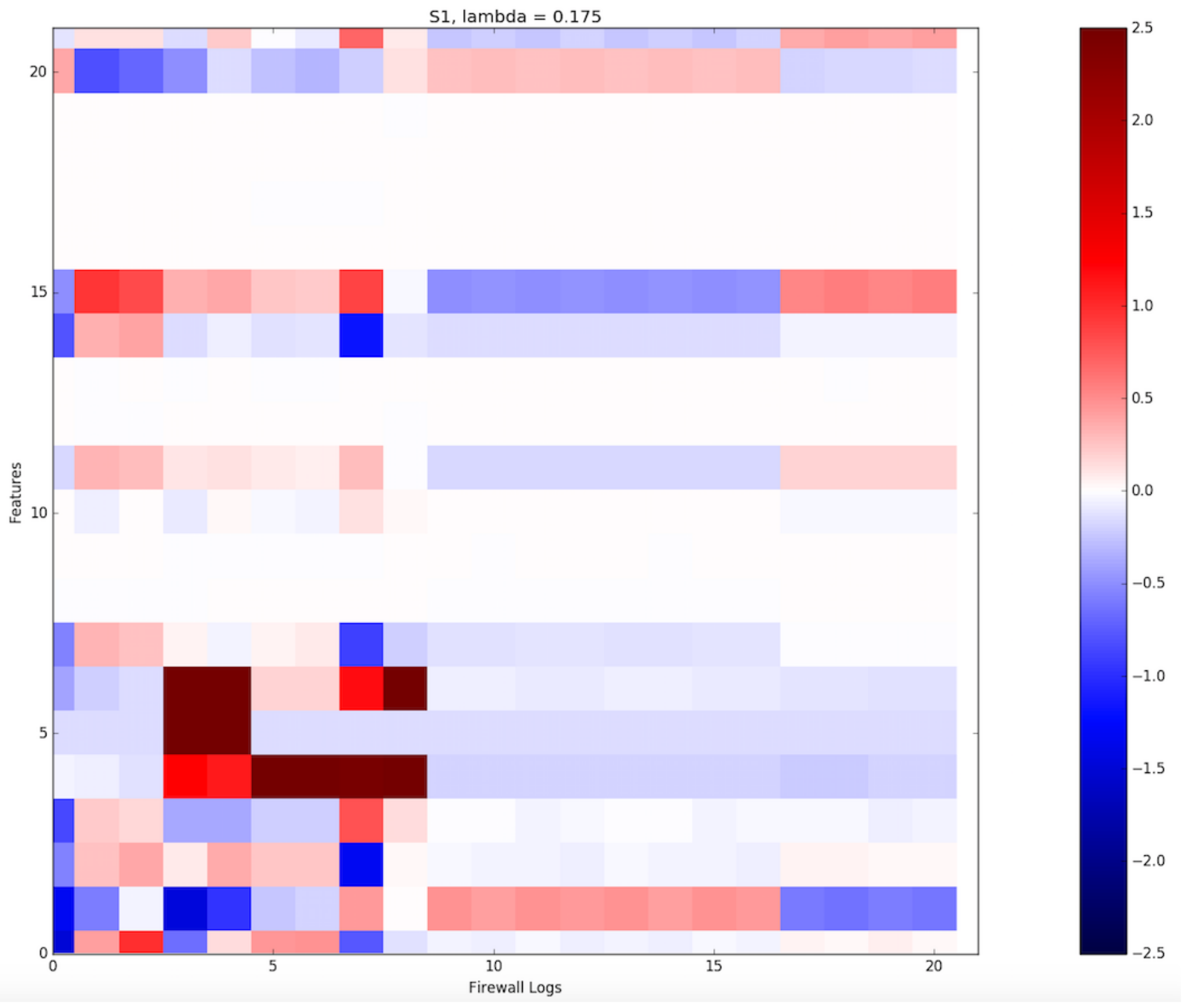


**Figure 7.1:** *In this figure is a small section of the $S_1$ matrix for the RDP attack where only the first 21 firewall logs and 21 features are depicted. Firewall logs 3-8 contain features with which have values greater than 2.26. Therefore these firewall logs would be marked as anomalous.*

the firewall logs (columns) which contain dark colors would be marked as anomalous. These firewall logs were all correctly marked as anomalous to produce the TPR of 1 and FPR of 0 which led to the use of $\lambda$ = .175 and $\alpha$ = 2.26 as the variables for the threshold detection algorithm

This $\lambda$ and $\alpha$ were then used to determine anomalies for two other sets of data in the VAST 2011 data set. A $Y_2$ and $Y_3$ were created each containing a new, unique type of attack from the VAST 2011 data. $Y_2$ contained the DoS attack and $Y_3$ contained the undocumented computer added to the network attack. This data was fit into a matrix in the same way as $Y_0$ and $Y_1$ and normalized as explained in section 5. Then an $S_2$ and $S_3$ were calculated using the $\lambda$ determined by the detection algorithm as explained in section 5.2. This detection algorithm is applied to $S_2$ and $S_3$ using the threshold $\alpha$. Therefore attacks can be predicted on an entirely new set of data without knowing where the real attacks are. The results from this threshold detection are depicted in figure 7.2 where training on the RDP attack correctly predicted the DoS attack but not the undocumented computer attack.

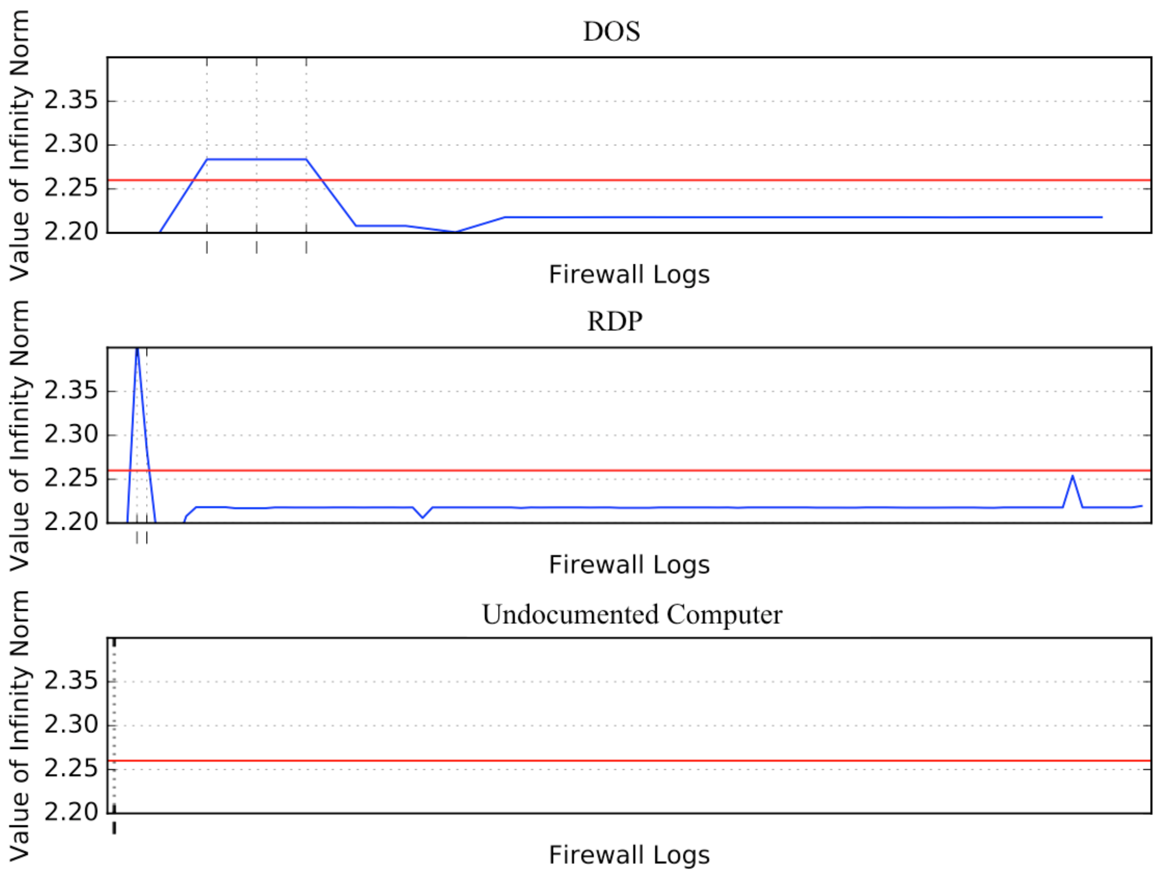


**Figure 7.2:** *In this figure the results of Threshold Detection trained on the RDP attack with $\lambda$ = .175 and $\alpha$ = 2.26 are presented. The blue line represent the max element (infinity norm) in each column(firewall log). The red line is the threshold ($\alpha$) where any firewall logs with infinity norm greater than the red line are marked as anomalous. The dotted black lines are the true attack firewall logs which were unknown to the detection algorithm for the DoS and undocumented computer attacks. The detection algorithm correctly marked all of the true attacks as anomalies in the DoS and RDP attacks. Since the detection algorithm is trained on RDP, this result was previously known that the TPR =1 and FPR = 0. The interesting result is that the DoS attack, which is entirely new data with a different type of attack, is also predicted correctly from the detection algorithm. The DoS attack has three true attacks which are the three firewall logs with infinity norms above $\alpha$. Therefore DoS attack also has a TPR of 1 and a FPR of 0. Unfortunately this accurate predicting does not hold true for the undocumented computer attack which had a TPR = 0 and a FPR = 0 since no firewall logs were marked as anomalous.*

To analyze the result from the undocumented computer attack, the infinity norms were analyzed. While the minimum infinity norm for any column in the DoS or RDP attack slices was above 2, the minimum infinity norm for undocumented computer attack was 0.924. Additionally the maximum infinity norm for the undocumented computer attack was 1.01, which is not close to the set threshold $\alpha$. Therefore no columns of the undocumented computer attack are close to the threshold. This result will be analyzed more in section 9.1.

## 7.2 Top Two Elements

Another detection algorithm tested was similar to the threshold detection algorithm, and also determined which columns (firewall logs) of $S_1$ were anomalous by which values were above a threshold ($\beta$). The threshold $\beta$ is larger than the previous threshold $\alpha$. A firewall log was anomalous if the sum of the two largest elements in a column (firewall log) of an S matrix were above the threshold $\beta$. This detection algorithm is trained on $S_1$ with several $\lambda$'s to determine the "best" combination of $\lambda$ and $\beta$. This combination produces the highest TPR and lowest FPR. This means that the most real attacks are marked as anomalies and the least non-attacks are marked as anomalies. The more important feature is making sure the detection algorithm marks all real attacks as anomalous.

This detection algorithm was tested using the same method as the threshold detection algorithm. Top two element detection was also trained on the RDP attack. Again forty different $\lambda$'s were tested between .075 and .275. Of these $\lambda$'s, 3 produced a TPR of 1 and a FPR of 0. These $\lambda$'s were 0.178, 0.183, and 0.234, with $\beta$'s of 4.482, 4.4, 4.408 using the data snooping trick explained in the previous section. Then $\lambda$ = .175, .18, and .235 were tested with several different $\beta$'s. For $\lambda$ = .175, only a $\beta$ of 4.48 produced a FPR of 0. For $\lambda$ = .18, only a $\beta$ of 4.4 produced a FPR of 0. For $\lambda$ = .235, the best $\beta$ of 4.4 produced a FPR of .0006. Therefore $\lambda$ of .175 was chosen with $\beta$ = 4.48 as the variables for the detection algorithm.

Once $\lambda$ and $\beta$ were determined, the detection algorithm was tested on the DoS and undocumented computer slices of data used to test the threshold detection algorithm. The result is in figure 7.3.

This result for the undocumented computer attack is not surprising since threshold detection algorithm was also unable to predict the undocumented computer attacks. Further explanation on the undocumented computer attack is given in section 9.1.

## 7.3 Random Forest Classifier

Random Forests can also be utilized to determine which columns of $S_1$ are anomalous. This method is more complex than the previous two methods and is explained in section 6.4. Random Forest Classification was implemented using scikit-learn in python [18]. Random Forests was trained on the RDP attack and tested on the DoS and undocumented computer added to the network attacks. On the DoS attack the Random Forest Classifier predicted with 100% accuracy the 4963 anomalies and 37 not anomalous firewall logs. When running cross-validation the mean

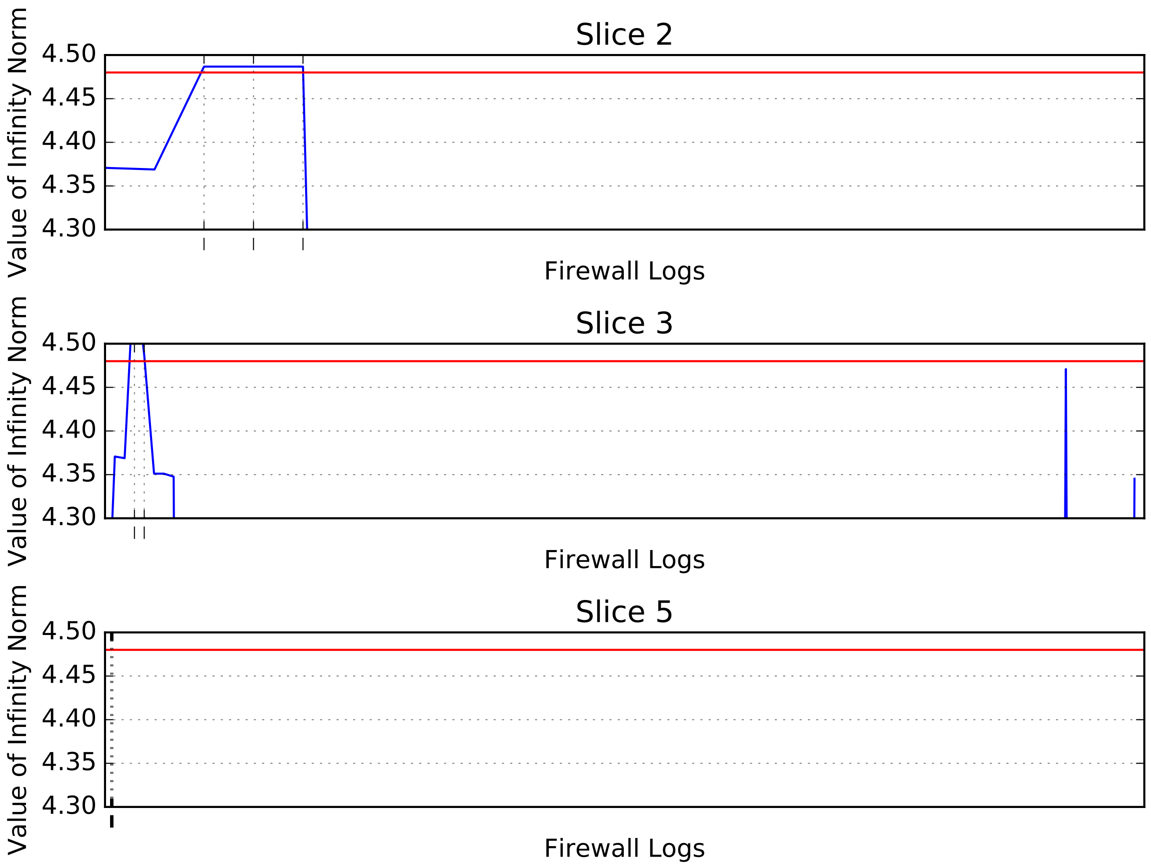


**Figure 7.3:** *In this figure the results of Top Two element Detection trained on the RDP attack with $\lambda$ = .175 and $\beta$ = 4.48 are presented. The red line is the threshold ($\beta$). The dashed lines indicate the locations of true attacks. The blue line is the sum of the max two elements in each column of S. RDP attack has a known TRP of 1 and FPR of 0, since the detection algorithm was trained on RDP attack. The detection algorithm is able to accurately predict the three anomalies in the DoS attack and was unable to predict any of the anomalies in the undocumented computer attack. This result is identical to the result of the threshold detection algorithm in the previous section.*

cross-validation score showed the random forest classifier had a 99% accuracy on predicting anomalies in the DoS attack slice. For undocumented computer, the cross-validation accuracy was .997709920993 although there were no true positive anomalies found. When applying the random forest classifier to undocumented computer attack, all four anomalies were marked as not anomalies. Additionally, when looking at the percent of trees in the random forest that predicted the anomalies in the undocumented computer attack to actually be anomalous, there were only 1%. Therefore 99% of the trees in the random forest indicated that the four anomalies in the undocumented computer attack would be not anomalous.

Random forest classification was tested on 76 possible $\lambda$'s between .025 and .2. Of these 76 only 6 did not accurately predict the anomalies in the DoS attack slice. These $\lambda$'s were 0.025, 0.027, 0.03, 0.032, 0.116, and 0.135. Every other $\lambda$ between .025 and .2 accurately predicted the 4963 anomalies in the DoS attack slice. Unfortunately none of these $\lambda$'s were able to accurately predict any of the 4 anomalies in the undocumented computer attack slice. This result on the undocumented computer is identical to the previous detection algorithms and will be explained further in section 9.1.

# Chapter 8

# Affect of $\lambda$ on each Detection Algorithm

After depicting the results of each of the detection algorithms separately, this section will analyze the relationships between results of the threshold detection algorithm and the random forest algorithm. Top two element detection algorithm was ignored for this section since the results for this detection algorithm were identical to the results from threshold detection. Additionally, the effects of $\lambda$ on each detection algorithm will be analyzed.

## 8.1 Detection Algorithms trained on DoS attack

In figures 8.1 and 8.2, the accuracies for varying $\lambda$'s for the two detection algorithms are recorded. The smallest $\lambda = .025$ is equivalent to the best theoretical $\lambda$ for the recovery problem which is defined in section 5.2.

The theoretical best $\lambda$ for the recovery problem has 100% FPR for the slice of data the detection algorithm was trained on. This means that even after training the threshold algorithm to have the best $\alpha$, the threshold detection was still unable to predict the anomalies. In this particular situation, every firewall log in the slice of data was marked as anomalous. With the best theoretical $\lambda$ (.025) and the $\alpha$ selected from training Threshold detection on the DoS attack, the threshold detection algorithm was tested on the RDP attack and the undocumented computer attack. For the theoretical best $\lambda$, the FPR was again 100% in the RDP attack slice. This indicates that the detection algorithm predicted all of the firewall logs to be anomalies. When the threshold detection algorithm was applied to the undocumented computer slice, not a single firewall logs was marked as anomalous.

Alternatively, if the best $\lambda$ on the DoS attack was chosen with the corresponding $\alpha$, the accuracy of predictions greatly increased. If $\lambda = .175$ was selected, the FPR and FNR were both 0 indicating 100% accuracy in predicting anomalies. When these values were in the threshold detection algorithm to detect anomalies in RDP attack, the results were not as accurate. The FPR for $\lambda$ = .175 on the RDP attack was still 0, yet the FNR was 50%. Since there were only two true attacks in the RDP slice of data, the detection algorithm was only able to predict one of the attacks. Again, the undocumented computer attack detection predicted no anomalies as originally shown in section 7.1.

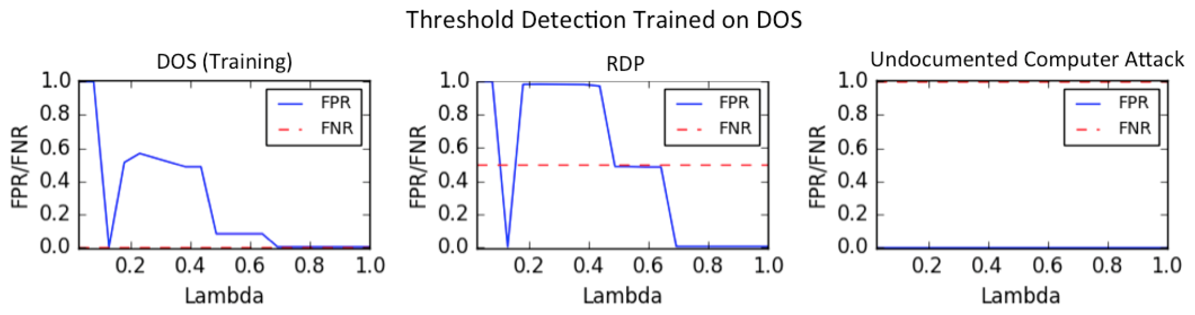


**Figure 8.1:** *In this figure is the detection accuracy of the threshold detection algorithm trained on the DoS attack. For 20 $\lambda$'s between .025 and .1, the FPR and FNR's were marked on the plot. A lot FPR and low FNR is the best accuracy in detection of anomalies in the given slice of data.*

Compared to the threshold detection algorithm trained on DoS attack, the random forest detection algorithm trained on the DoS attack greatly improved the accuracy of predicting anomalies. Figure 8.2 represents the results of the random forest detection algorithm trained on the DoS attack. When the random forest was trained on the DoS attack, the forest could accurately predict every anomaly in the DoS attack. Additionally the random forest fairly accurately predicted the RDP attack. Even the $\lambda$ which produced the worst testing result on the RDP attack (the theoretical best $\lambda$) had a FPR significantly lower than the equivalent FPR of the threshold detection algorithm. The worst detection accuracy for the random forest had a FNR of 0 and a FPR of .002, indicating that all of the attacks were correctly detected and a few additional firewall logs which were not attacks were marked as anomalous by the detection algorithm. This accuracy is an improvement on the best accuracy of the threshold detection algorithm which was a FPR of 0 but a FNR of .5. Therefore, in the problem of anomaly detection, the significance of not predicting an anomaly was greater than claiming additional firewall logs are anomalous. The random forest detection algorithm had a better overall prediction accuracy and the theoretical $\lambda$ was not the best $\lambda$ for the detection algorithm. Finally, selecting the largest $\lambda$, which is equivalent to PCA, provided the best accuracy on the RDP testing slice. Unfortunately the undocumented computer attack was unable to be predicted.

## 8.2 Detection Algorithms trained on the RDP attack

The detection algorithms trained on the RDP attack were analyzed with similar methods as the detection algorithms trained on the DoS attack in section 8.1. The threshold detection

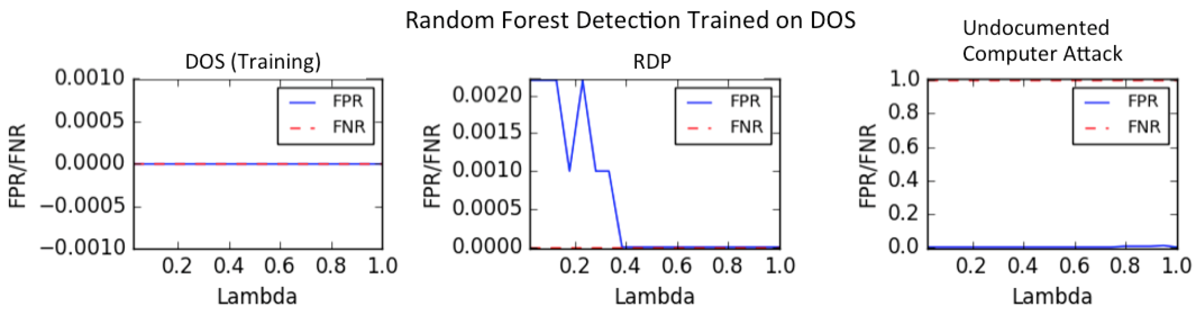


**Figure 8.2:** *In this figure is the detection accuracy of the random forest detection algorithm trained on the DoS attack. For 20 $\lambda$'s between .025 and .1, the FPR and FNR's were marked on the plot. A lot FPR and low FNR is the best accuracy in detection of anomalies in the given slice of data.*

algorithm trained on RDP attack is represented in figure 8.3. In this figure, the theoretical best $\lambda$ ($\lambda$ = .025) represents the worst accuracy in detection. This result is repeated from the detection algorithms trained on the DoS attack. When $\lambda$ = .025, the FPR = 1. This indicated that all firewall logs in the RDP slice were marked as anomalous. The same FPR occurred for the testing of the algorithm on the DoS attack. Marking every firewall log is not useful for predicting which of the firewall logs are true anomalies. Therefore the theoretical best $\lambda$ was not the best $\lambda$ for predicting anomalies in the threshold detection algorithm trained on the RDP attack. If $\lambda$ was trained on the RDP slice, the $\lambda$ chosen for the detection algorithm would be .7 since the FPR and FNR of the RDP attack were both 0. This $\lambda$ produces a 100% accuracy on the RDP attack and, when testing on the DoS attack, the accuracy is also 100%. Therefore training $\lambda$ on RDP attack performed significantly better than the $\lambda$ given by theory.

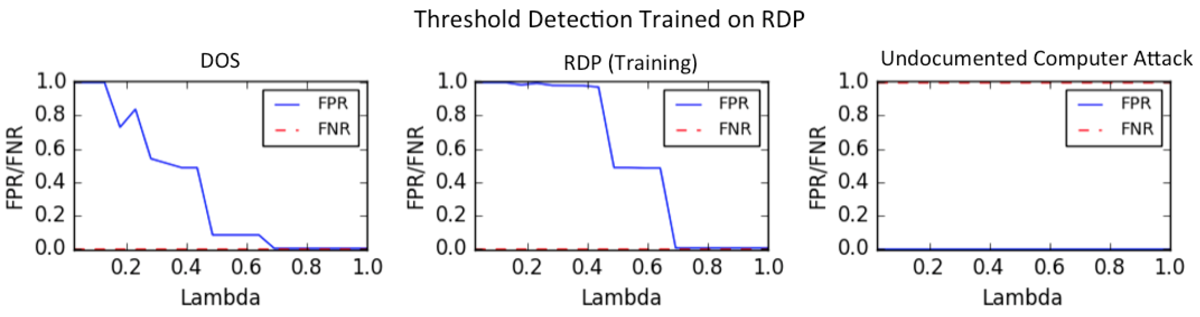


**Figure 8.3:** *In this figure is the detection accuracy of the threshold detection algorithm trained on the RDP attack. For 20 $\lambda$'s between .025 and .1, the FPR and FNR's were marked on the plot. A lot FPR and low FNR is the best accuracy in detection of anomalies in the given slice of data.*

In Figure 8.4, the random forest detection algorithm trained on the RDP attack is represented. As with the random forest detection trained on the DoS attack, the random forest classifier was able to correctly predict every attack firewall log in the trained slice of data. The accuracy of predictions in the RDP attack slice was 100%. Additionally, selecting any $\lambda$ above .025 performed with a 100% accuracy on the testing data in the DoS attack. This selection of the best $\lambda$ include the largest $\lambda$ which was equivalent to PCA. The large $\lambda$ produced the best detection accuracy on the training and testing slices. Again the worst selection of $\lambda$ was the $\lambda$ provided by theory. As with the detection algorithms trained on DoS attack, the random forest classifier trained on RDP attack always had a higher accuracy. Even when $\lambda$ = .025, the FNR of the testing data was only 0.14. This FNR was significantly better than the FPR of 1 in the threshold detection algorithm trained on RDP attack. Therefore the best $\lambda$ was not the $\lambda$ provided by theory and the random forest detection algorithm performed better than the threshold detection algorithm on both the testing and training data. These results were identical to the results found when the detection algorithms were trained on DoS attack. Unfortunately the undocumented computer attack was again unable to be predicted by either detection algorithm.

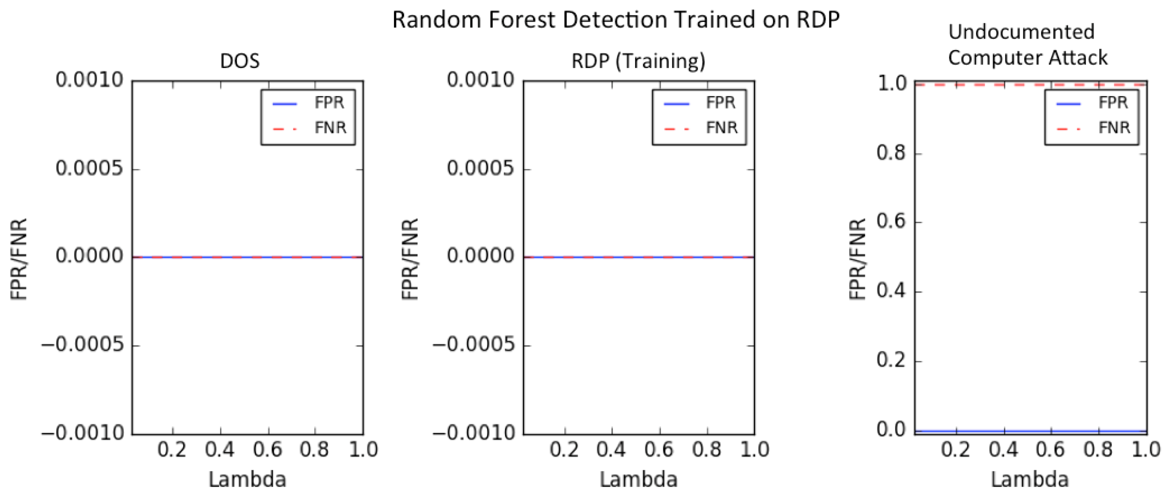


**Figure 8.4:** *In this figure is the detection accuracy of the random forest detection algorithm trained on the RDP attack. For 20 $\lambda$'s between .025 and .1, the FPR and FNR's were marked on the plot. A lot FPR and low FNR is the best accuracy in detection of anomalies in the given slice of data.*

## 8.3 Detection Algorithms trained on Undocumented Computer attack

The final comparison between the detection algorithms was for the detection algorithms trained on the undocumented computer attack. Figure 8.5 represented the threshold detection algorithm trained on the undocumented computer attack. Again the theoretical best $\lambda$ produced the worst accuracy for the detection algorithm. Additionally selecting the best $\lambda$ ($\lambda$ = .8) from the undocumented computer trained set was able to accurately predict the DoS testing slice with

100% accuracy, but the RDP attacks were not able to be predicted.

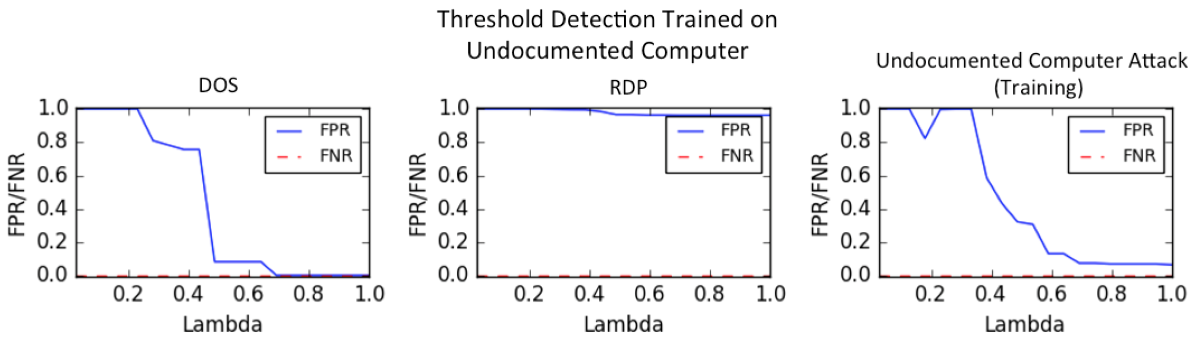


**Figure 8.5:** *In this figure is the detection accuracy of the threshold detection algorithm trained on the undocumented computer attack. For 20 $\lambda$'s between .025 and .1, the FPR and FNR's were marked on the plot. A lot FPR and low FNR is the best accuracy in detection of anomalies in the given slice of data.*

Figure 8.6 represents the random forest detection algorithm trained on the undocumented computer attack. Again the theoretical best $\lambda$ had the worst prediction accuracy, but selecting the high $\lambda$ equivalent to PCA did not provide an improved detection accuracy.

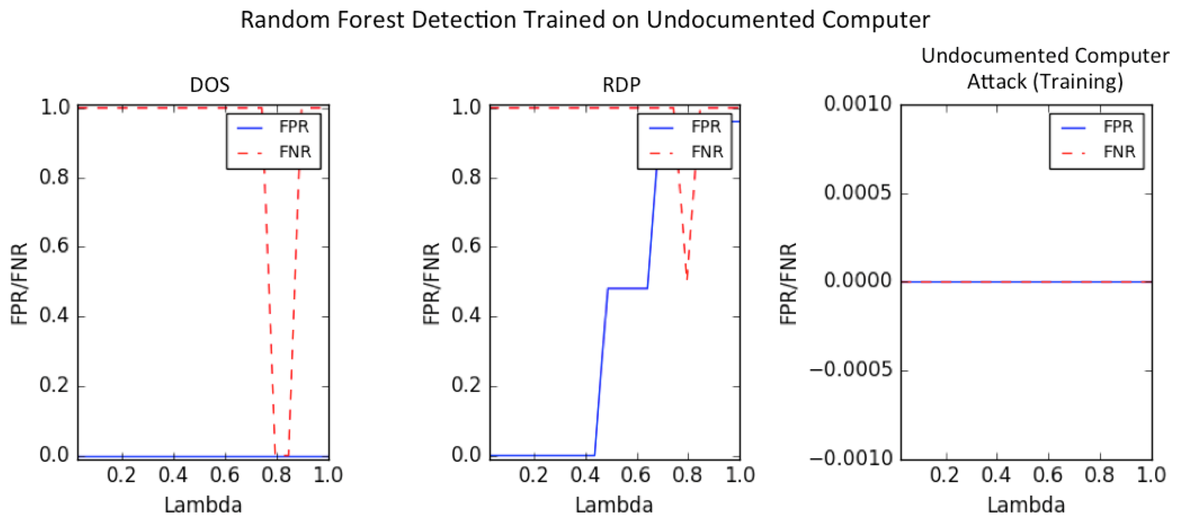


**Figure 8.6:** *In this figure is the detection accuracy of the random forest detection algorithm trained on the undocumented computer attack. For 20 $\lambda$'s between .025 and .1, the FPR and FNR's were marked on the plot. A lot FPR and low FNR is the best accuracy in detection of anomalies in the given slice of data.*

# Chapter 9

# Anomaly Detection Conclusions

## 9.1 Future Work on Anomaly Detection

For each of the detection algorithm's investigated, no detection algorithm was able to accurately predict the undocumented computer attack. An interesting observation about the undocumented computer attack can be found in the difference in its number of large singular values in Figure 9.1. The DoS and RDP attacks both had a small number of singular values within the given range of $\lambda$s. The maximum number of singular values above 10 for the largest $\lambda$, for both the DoS and RDP attacks, was less than 60. The undocumented computer had over 200 singular values that were over 10. This indicates that a large majority of the variance in the undocumented computer attack occurs within 200-dimensional space as opposed to the other attacks where the majority of the variance is in less than 60-dimensional space. This indicated that the majority of the data from the undocumented computer attack lies within the L matrix described in section 4.2. From this observation, there is an increased likelihood that the attacks in the undocumented computer slice actually occurred in the L matrix, instead of the S matrix. Since the detection algorithms were tested on the S matrices, the undocumented computer attack would not be able to be predicted. Therefore, further research into the where the attacks lay within the undocumented computer slice could lead to better detection accuracy of the undocumented computer attacks.

## 9.2 Anomaly Detection Conclusions

1. RPCA anomaly detection was accurate on sparse anomalies. In each of the types of anomalies, only 2-4 of the unique firewall logs were actually anomalies. Therefore an interesting

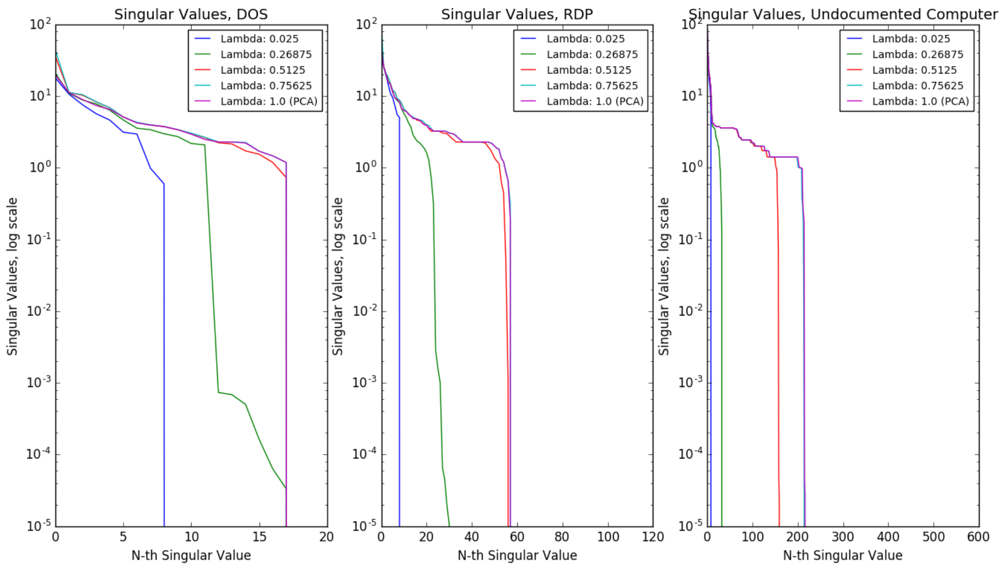


**Figure 9.1:** *In this figure is a plot of the singular values for each type of attack for a range of $\lambda$s. The smallest $\lambda$ is equivalent to the theoretical best $\lambda$ for the prediction problem and the largest $\lambda$ is equivalent to PCA.*

result of the anomaly detection is actually being able to predict these anomalies when they are so rare in the data.

2. The theoretical best $\lambda$ for the recovery problem was not the best $\lambda$ for the anomaly detection problem in this paper. Each of the detection algorithms proved that a $\lambda$ trained on the data provided more accurate detection of anomalies than the theoretical $\lambda$.

3. Random forest classification provided better accuracy of anomaly detection than the threshold detection algorithm. For each slice the detection algorithm was trained on, the random forest classification resulted in lower FPR and FNR than the threshold detection algorithm.

# Chapter 10

# Visualization System

With the anomaly detection portion of this report completed, we then focused on the development of a novel visualization system. Our visualization system supports several goals related to anomaly analysis. Users of the system can inspect a familiar overview of network traffic and computed features, select individual anomalies for follow-up, and analyze an anomaly's context to assess whether those anomalies should be escalated. We strove to meet these goals by providing familiar overviews of network features and allowing users to sort and filter traffic by selections while maintaining the context of the general overview in order to aid in pattern matching and help users separate anomalous traffic from regular traffic. We took inspiration from existing visualizations that have been done in this design space, and designed a visualization system comprised of multiple elements. Our system consists of a network overview, a feature view, and a detail view, that will allow users to view traffic and attributes over time and sort logs by attributes in order to analyze specific elements of log files, as well as view anomalies in the context of general network traffic, and see which of the features of anomalous firewall logs implicated the log as anomalous.

## 10.1 Related Work

We began by looking at publications that also used VAST 2011 MC2 data in their work. One of these papers, *University of Buenos Aires – svc+canvas+processing heatmaps*, utilized heatmaps and parallel coordinates to analyze the VAST data and detect attacks by means of visual analysis [19]. Using parallel coordinates visuals one can quickly glean much information about a specific selection, especially with interactivity such as brushing and filtering. However, this visual can quickly become overburdened with large traffic flows. One such example is attempting to inspect an anomaly embedded within a DoS attack. Without proper filtering based on DoS specific knowledge for that situation, this visualization becomes a messy and hard to navigate. This naturally led us to find work done with parallel sets as a scalable alternative. However, in switching to parallel sets one loses temporal information and the ability to inspect traffic per log or packet.

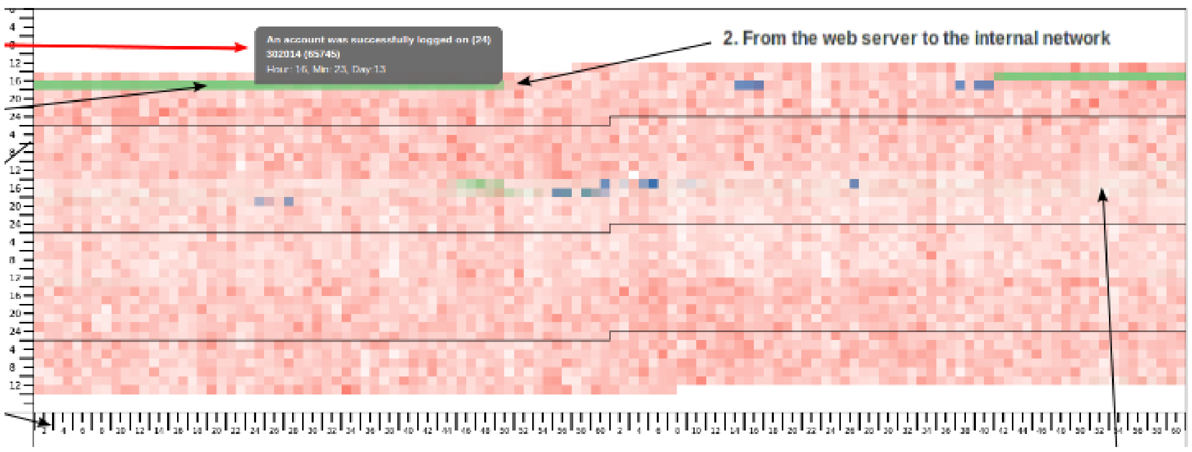


**Figure 10.1:** *Heatmap used in University of Buenos Aires – svc+canvas+processing heatmaps. Each horizontal line represents a new day, with each square of the heatmap representing one minute in the day. The green line represents the DoS attack on the Web Server in day 1 of the VAST data[19].*

The author of *University of Buenos Aires – svc+canvas+processing heatmaps*, Walter Lamagna synchronized the multiple data sources provided by the VAST 2011 MC2: IDS logs, PCAP files, Firewall, and Security logs. Lamagna then used a heatmap, seen in Figure 10.1, to visualize this data, although he doesn't talk about what led him to this choice. With the heatmap in Figure 10.1, Lamagna was able to easily detect the DoS attack in the first day of the VAST data. He used this heatmap in conjunction with parallel coordinate graphs to provide a more close-up view of each square of the data.

One of the papers that we drew the most inspiration from was *Understanding the Contexts of Network Traffic Alerts* (CoNTA), by Bram Cappers and Jarke van Wijk. As the title suggests, this paper focused on providing users with context when they received network traffic alerts from an IDS. This context allowed users to compare and contrast anomalies with normal network traffic while also provided with message level information in order to help make decisions on whether a group of alerts are are associated and whether or not to escalate the alerts or mark them as a false alarm [20]. Visualization this information also helps network security analysts make visually identify patterns to between alerts and the messages that caused them.

CoNTA utilizes coordinated multiple views to allow users to refine selections based on attributes present in the network overview, something that we also chose to incorporate in our visualization. Their coordinated multiple views allowed users to look at a broad overview of network traffic and refine, or filter, that view based on interactions with attributes presented as

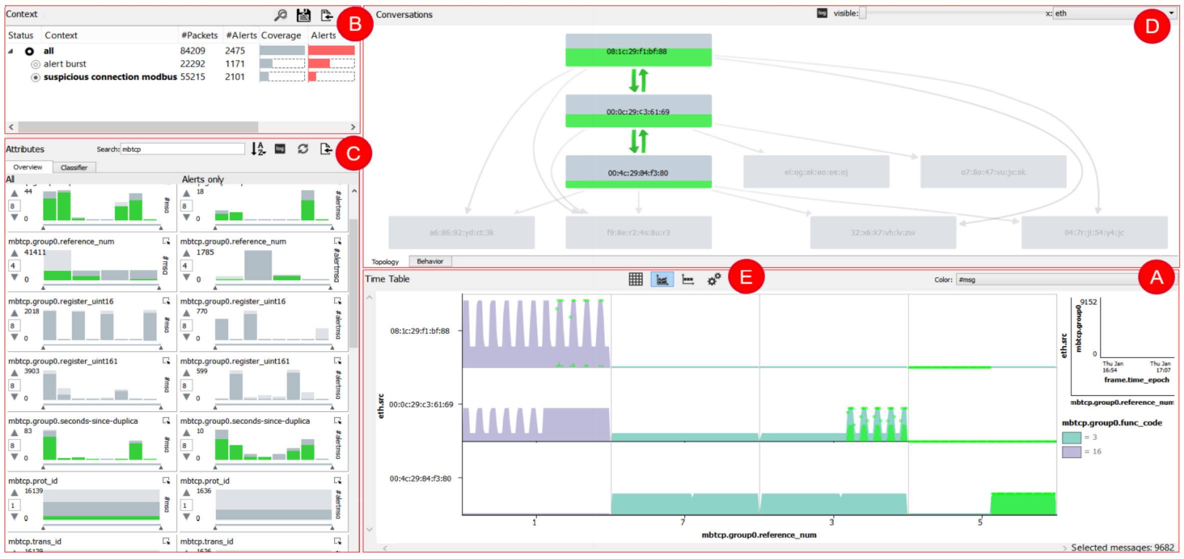


**Figure 10.2:** *Graphical user interface of the prototype in Understanding the Contexts of Network Traffic Alerts. Histograms are used to display attribute frequencies for both normal and anomalous traffic[20].*

scented widgets. The overview could then be modified to show all alerts pertinent to that view. Changing how alerts are viewed and selecting alerts allowed users to sort by attributes relevant to the investigation. From this point a user could inspect individual alerts to begin drilling down on the root cause of any sequence alerts and make selections to view in a wider context . The histograms on the left side in Figure 10.2, in the view labeled "C", show data for an already selected time range. The histograms on the far left displays the attributes frequencies of regular network data and the ones on the right of those show attributes attribute frequencies of traffic that their IDS alerted on [20].

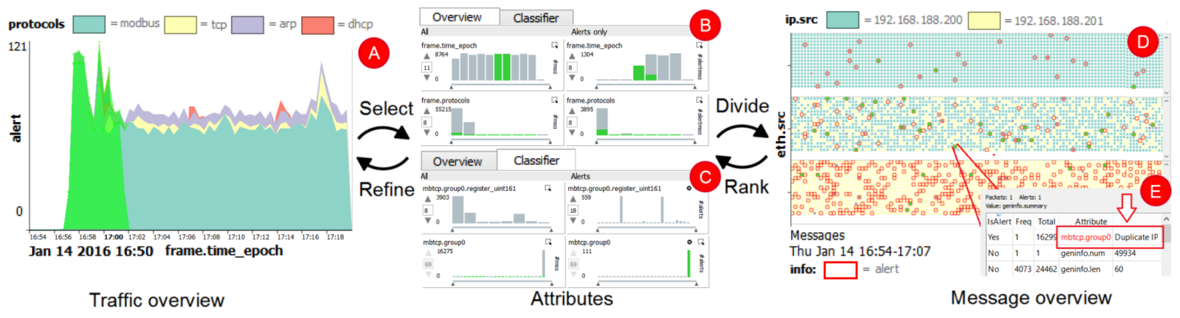


**Figure 10.3:** *Prototype system used to detect "Man-in-the-Middle behavior using selection-based attribute ranking." [20] Users select a time range from the timeline on the left. Then, they can view the attributes and see how they are divided up. They can rank attributes, and then view them again, allowing them to refine their selections.*

One of the examples from the paper, shown in Figure 10.3, is using the prototype system to detect a "Man-in-the-Middle" attack. To accomplish this, users follow the workflow shown in the figure: select a time range, divide it up, rank attributes, refine selection, and repeat. This reinforces the idea that interaction is one of the central elements in CoNTA [20].

In their study, Cappers and van Wijk used two different data sets, one artificial set involving 5 hosts, 80,000 messages, and 170 attributes. The other was a real-world data set consisting of 3 days of recorded internal SMB2 network traffic from a university. For the second data set, they had no ground truth [20].

By contrast, our study, on the VAST 2011 MC2 data, involved over 248 internal hosts, and over 12,942,561 lines of firewall logs. We created 555 attributes, and we did have ground truth for the entire dataset. The studies also differ in that we examined firewall data, whereas Cappers and van Wijk used IDS alerts instead. IDS alerts are something that we will discuss in Future Work.

Another visualization that we took some inspiration from is the INFUSE system. The paper that introduces this system, *INFUSE: Interactive Feature Selection for Predictive Modeling of High Dimensional Data* gave us both an idea for our own visualization, as well as some ideas of the things that we wanted to avoid in our visualization.

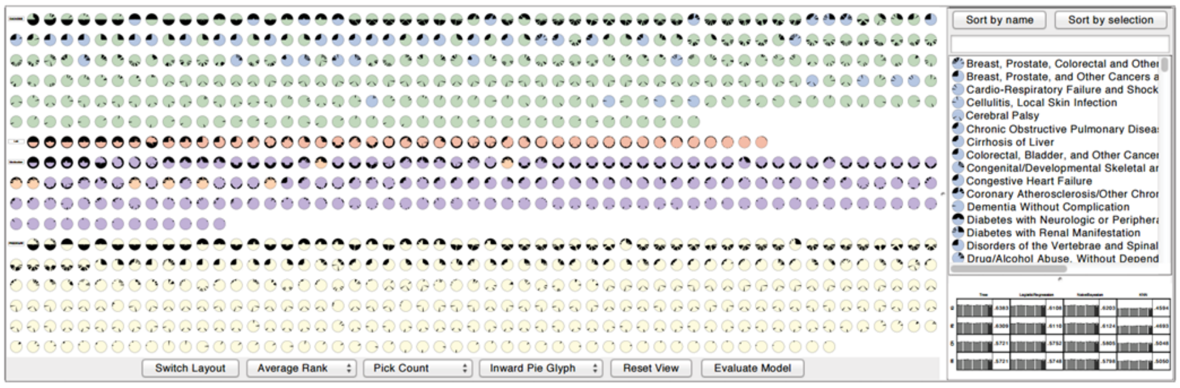

Sort by name
Sort by selection
Breast, Prostate, Colorectal and Other
Breast, Prostate, and Other Cancers a
Cardio-Respiratory Failure and Shock
Cellulitis, Local Skin Infection
Cerebral Palsy
Chronic Obstructive Pulmonary Disea
Cirrhosis of Liver
Colorectal, Bladder, and Other Cance
Congenital/Developmental Skeletal ar
Congestive Heart Failure
Coronary Atherosclerosis/Other Chron
Dementia Without Complication
Diabetes with Neurologic or Periphera
Diabetes with Renal Manifestation
Disorders of the Vertebrae and Spinal
Drug/Alcohol Abuse, Without Depend
Switch Layout
Average Rank
Pick Count
Inward Pie Glyph
Reset View
Evaluate Model

**Figure 10.4:** *An overview of the INFUSE system [21]. Users can view the impact of different attributes, which aids in feature engineering. Each icon represents a feature, divided into 4 quadrants. The amount of black in each quadrant specifies a low heuristic value, meaning that the more black present overall in an icon, on average, the less useful the feature is.*

Figure 10.4 shows the INFUSE system, a system that is designed to aid users in feature selection, by showing which features are the most influential in their algorithms. Looking at this system, we noticed that it is very crowded: lots of small, dense data is represented. The authors of the paper also notes, in their conclusion, the need for clutter reduction in their system [21]. We did like the idea of showing which features are the most useful after anomaly detection has already been run, and we incorporated this element in the Feature View component of our visualization. We also determined that our system should be easier to use for end-users and have a simpler interface.

Both of these systems contributed to our overall design. From CoNTA, we derived a general structure for our system, and the idea of multiple views. From INFUSE, we designed the Feature View component of our system, and saw ways in which we could make our visualization system more user-friendly.

## 10.2 Visualization Design Space

Visualizations have been built from network data since the late 1990's. The field has come a long way since then. Many earlier visualizations focused on providing a broad overview of network traffic for exploration of an entire network trace. Others focus on showing an enumeration of network traffic, in an attempt to give context for pattern matching, while allowing users to inspect points of interest. Most of these exploratory visualizations work towards some common goals. They mainly attempt to meet three objectives: gain awareness through visualization of some enumeration of network attributes, provide sorted selections or allow users to refine traffic by their own selections, and allow users to inspect traffic in order to get the fine details. We believe these are important goals to strive for in a visualization for cyber security.

Many of the earlier visualizations that we've seen would be very outdated by today's standards, and there hasn't been much previous work done that attempts to integrate anomaly detection into the visualization as well as feature engineering. A lot of the systems we saw were for visualizing network data in general, with no real anomaly detection involved, or for visualizing features for feature selection algorithm creation [21]. One of the earliest papers that we looked at, *Data Mining Approaches for Intrusion Detection* discussed the need for a support environment driving pattern recognition, feature engineering, and the evaluation of detection models [22]. The authors brought up the need to be able to assist users in pattern recognition related to anomalous network traffic, as well as integrate the feature engineering process and evaluation of an anomaly detection model into a cohesive system. Earlier systems, like PixelCarpet, used mostly static views, and ran into issues scaling to data with hundreds of features [23], whereas newer systems coordinate multiple views to help users link different data sources together, allowing them to filter results and quickly reduce the workspace down to a selection relevant to an anomaly or alert.

In the paper *Bridging the Gap of Network Management and Anomaly Detection through Interactive Visualization*, many different types of visualizations are discussed. The paper covers Area Charts, Gantt Charts, Treemaps, Network Graphs, and the concept of linking [24]. The authors of this paper put forth the idea for a web-based visualization tool that utilizes these different views to show different aspects of the data. These views would be linked together, so that users would be able to investigate the diverse elements of whichever part of the data they wanted to. Linking views together can yield much deeper insight into the data, and can make finding causal relationships easier, as well as uncover unforeseen connections. Linking multiple views together is a practice called Coordinated Multiple Views [25]. The authors worked with the VAST 2013 data set to create their visualization system [24].

Many projects used timeline overviews of a network capture to show the frequency of traffic over time[26], and some also used multiple views to allow users to make a selection and inspect specific traffic[20]. Figure 10.5 shows one example of this. It shows Botnet traffic capture, broken up by port, effectively showing traffic frequencies over a certain time range. This allows users of this visualization to effectively look at traffic density over time, and has the added benefit of narrowing this down to specific port traffic.

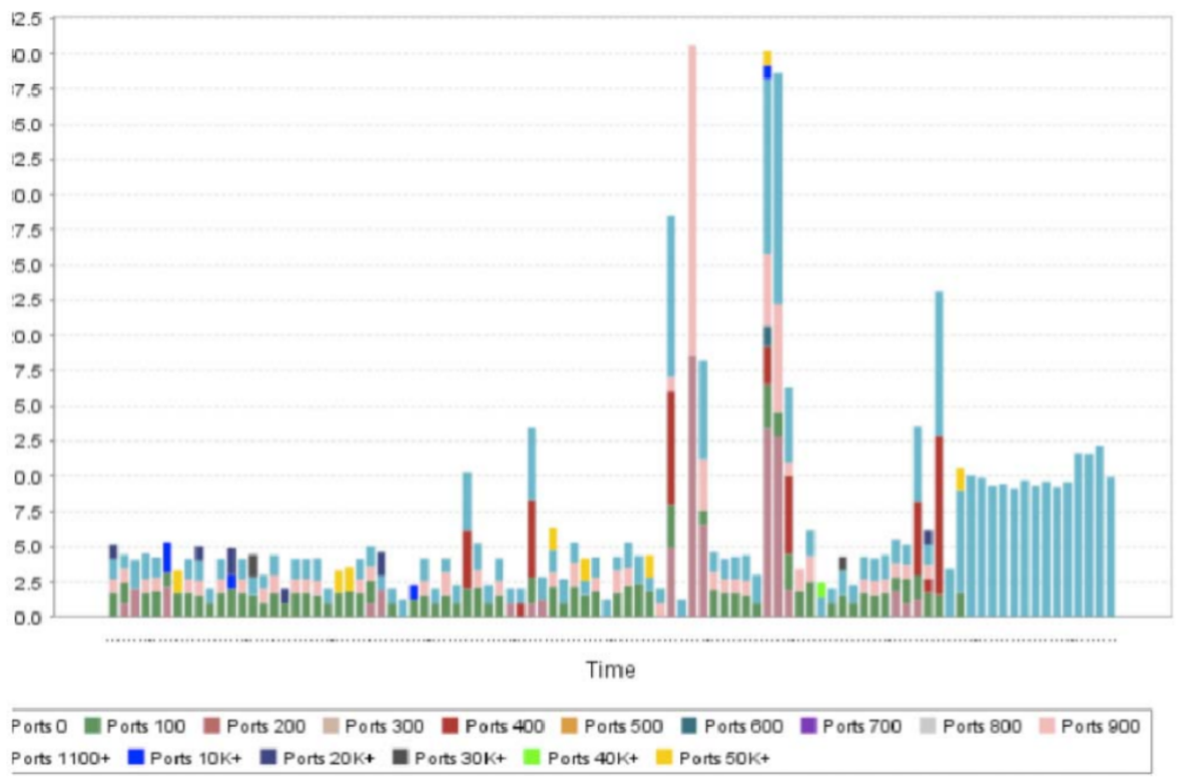


**Figure 10.5:** *Botnet traffic capture cube-root scale histogram[26]. This graph shows frequency of network traffic over time. The traffic here is broken up according to port.*

The network overviews allowed cyber security analysts to view network representations in a familiar format in order to quickly gauge traffic fluctuations based on any selection of attributes within the network. Multiple views could be utilized coordinate selections on various features and sort out traffic down to a specific sub-selection of packets or sensor alerts to begin a deep inspection and while still being able to connect that selection back to a broader overview where a user can gain context for their selection. Another common theme of modern network anomaly visualizations is the notion of preserving network context, even when looking at specific packet data [27]. Both of these elements, broad overviews with the ability to sort by network feature, are crucial for allowing a cyber security analyst to filter large network captures down to selections that are relevant to a current investigation while maintain the context of the rest of the traffic. The context of how a packet or firewall log appears in a network can be invaluable for cyber security analysts, both in determining if the packet is anomalous, as well as for the purpose of aiding in the process of pattern recognition. A system that allows a user to customize views such that their selections begin to reveal recognizable patterns is an invaluable feature that any visualization system should strive to produce.

With this project's specific focus on feature engineering and cyber security anomaly detection, we decided that rather than implement a visualization with just one view, or focused on one specific goal, we would implement a system of visualizations that utilized multiple coordinated views, such as the visualization prototype in *Understanding the Contexts of Network Traffic Alerts*[20]. This would allow us to show how the features that we engineered impacted the anomaly detection, as well as provide users with useful context about the network and detailed information about each firewall log that they were interested in.

## 10.3 Design Process

The process of planning a new visualization system involved looking at previous visualization for cyber security work. We began with an iterative process of taking ideas from other works and creating one-off visualizations in order to gain a sense of what works and what doesn't work in terms of information gained, ease of use, and scalability. This work flow also gave us the opportunity to approach our final design modularly by swapping out our pre-made visualizations in the final system before settling on a combination that best served our goals.

Our visualization system went through multiple iterations before we settled on our final layout and component views. We started off creating simple visualizations in python, and eventually reached our current web-based interactive system, with multiple views and ways for users to investigate the data.

1. **Python**

The earliest versions of our visualization system were in python, running simple PCA on raw data values from the .csv PCAP and Firewall files. This was more to get accustomed to working with the VAST data than it was to achieve meaningful visualization progress. One of the major things we learned during this time was the usefulness of feature engineering vs. running algorithms on raw data. When making these visualizations, we colored data according to the ground truth that we had from the VAST solution document which detailed the attacks. At the time of the visualization seen in Figure 10.6, we only had a feature set of around 20. None of these features were raw data from the firewall .csv files, these features were the more uniform type of features that we implemented later in the project.

One thing these early visualizations also helped us to determine was which attacks we wanted to focus on. Seen here in Figure 10.6, green dots representing the SMTP email attack

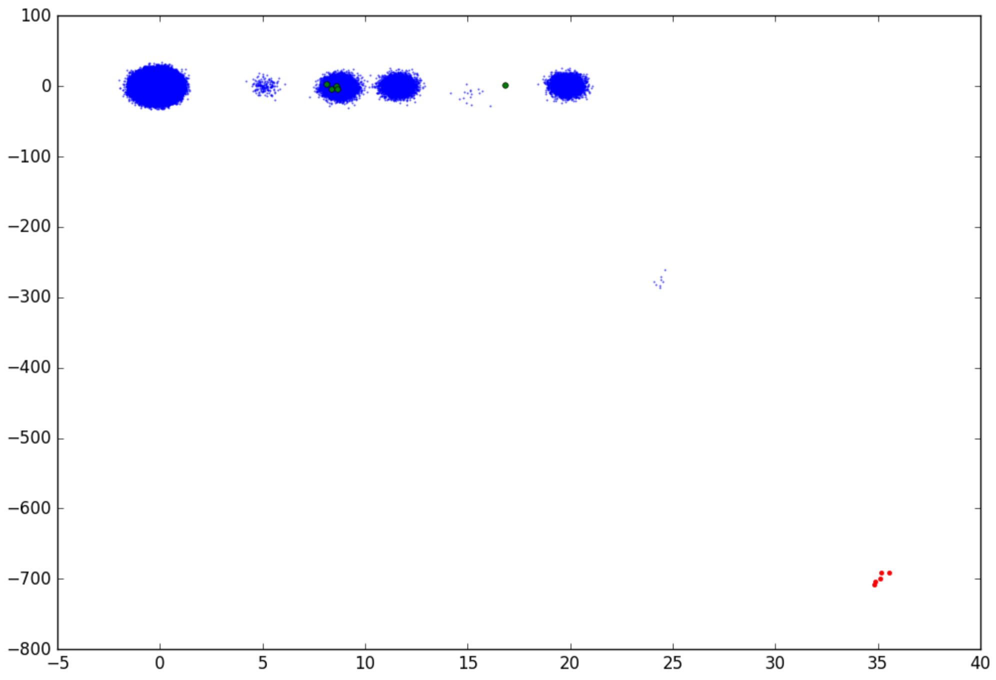


**Figure 10.6:** *One of our last python plots using just PCA. Jitter was introduced to this plot to show the difference between dense clusters and more sparse clusters. Green dots represent the SMTP email attack (one we later chose not to pursue), and red represent the RDP attack during day 2 of the VAST 2011 data set. Blue dots represent all other firewall logs.*

did not stand out from regular network data because the contents of that email are what make it malicious but we cannot not analyze message contents with firewall logs, which only provide connection data. This was using matplotlib's implementation of PCA in python, not the more advanced algorithms that were implemented later, but it did get us to take another look at the data.

The red dots in Figure 10.6, representing the RDP attack, did stand out from the rest, even though our feature set at the time was much more limited than our final feature set. This was the only attack that noticeably stood out from other traffic and was thus marked as anomalous by way of visual inspection.

2. **Web-Based**

We then transitioned into building web-based visualizations and created our first traffic overview using C3, a D3-based reusable chart library. C3's simple JSON chart definitions and chart API allowed us to quickly build interactive charts with coordinated subviews. The overview timeline, which was populated from the firewall log, not the packet capture data, allowed users to select a time range and get a "zoomed-in" view of the range that they selected with an auto-adjusting x-axis. This feature, while originally simply used for viewing the frequency of packets during a certain time range, eventually became useful to us to provide users with additional detail on the selected time range (something we will discuss in the Components section). The chart API also allowed us to define our own call-back functions for user interactions, simplifying the task of providing a table of a sub-selection upon user request.

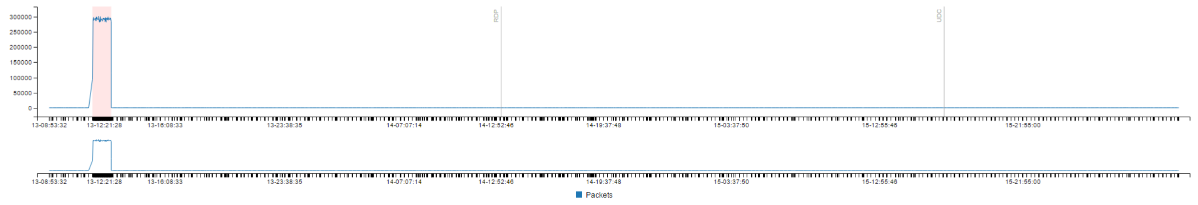
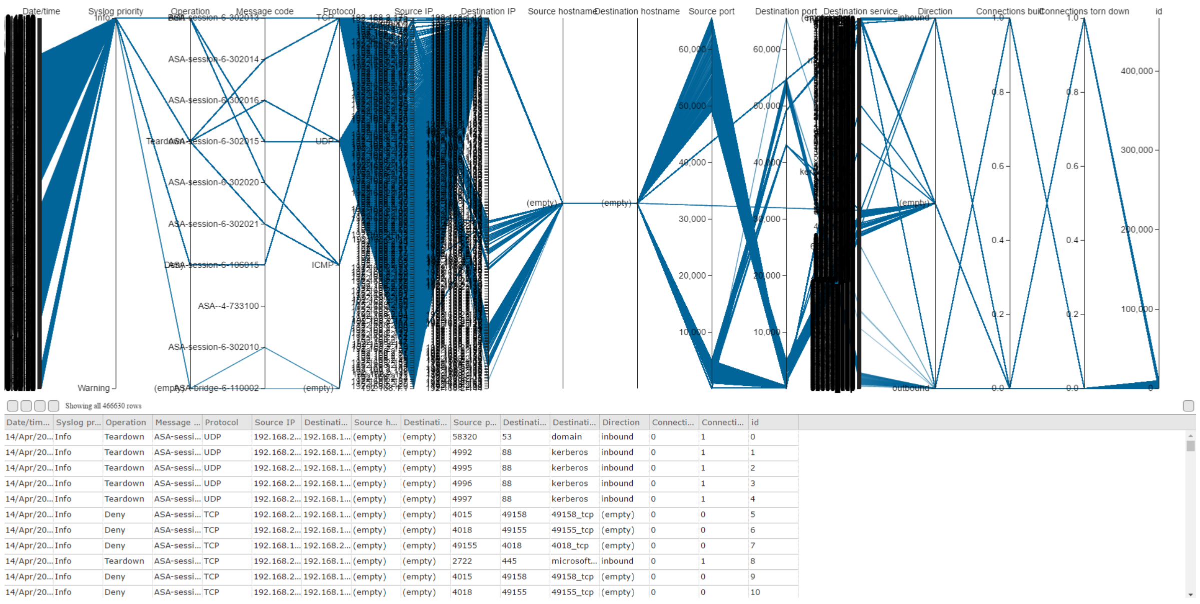


**Figure 10.7:** *The first complete timeline in our web-based implementation. The red highlighted area on the left represents the DoS attack. The RDP and Undocumented Computer Added (UDC) are also labeled. No region has been selected for zoom-in here.*

The visualization in Figure 10.7 was our first complete Overview module. This was later improved for the final version of our system. The final version was used to view output from the anomaly detection algorithm vs ground truth in the context of the full network traffic.

3. **Parallel Coordinates**



Showing all 466630 rows

| Date/tim... | Syslog pr... | Operation | Message ... | Protocol | Source IP | Destinati... | Source h... | Destinati... | Source p... | Destinati... | Destinati... | Direction | Connecti... | Connecti... | id |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 14/Apr/20... | Info | Teardown | ASA-sessi... | UDP | 192.168.2... | 192.168.1... | (empty) | (empty) | 58320 | 53 | domain | inbound | 0 | 1 | 0 |
| 14/Apr/20... | Info | Teardown | ASA-sessi... | UDP | 192.168.2... | 192.168.1... | (empty) | (empty) | 4992 | 88 | kerberos | inbound | 0 | 1 | 1 |
| 14/Apr/20... | Info | Teardown | ASA-sessi... | UDP | 192.168.2... | 192.168.1... | (empty) | (empty) | 4995 | 88 | kerberos | inbound | 0 | 1 | 2 |
| 14/Apr/20... | Info | Teardown | ASA-sessi... | UDP | 192.168.2... | 192.168.1... | (empty) | (empty) | 4996 | 88 | kerberos | inbound | 0 | 1 | 3 |
| 14/Apr/20... | Info | Teardown | ASA-sessi... | UDP | 192.168.2... | 192.168.1... | (empty) | (empty) | 4997 | 88 | kerberos | inbound | 0 | 1 | 4 |
| 14/Apr/20... | Info | Deny | ASA-sessi... | TCP | 192.168.2... | 192.168.1... | (empty) | (empty) | 4015 | 49158 | 49158_tcp | (empty) | 0 | 0 | 5 |
| 14/Apr/20... | Info | Deny | ASA-sessi... | TCP | 192.168.2... | 192.168.1... | (empty) | (empty) | 4018 | 49155 | 49155_tcp | (empty) | 0 | 0 | 6 |
| 14/Apr/20... | Info | Deny | ASA-sessi... | TCP | 192.168.1... | 192.168.2... | (empty) | (empty) | 49155 | 4018 | 4018_tcp | (empty) | 0 | 0 | 7 |
| 14/Apr/20... | Info | Teardown | ASA-sessi... | TCP | 192.168.2... | 192.168.1... | (empty) | (empty) | 2722 | 445 | microsoft... | inbound | 0 | 1 | 8 |
| 14/Apr/20... | Info | Deny | ASA-sessi... | TCP | 192.168.2... | 192.168.1... | (empty) | (empty) | 4015 | 49158 | 49158_tcp | (empty) | 0 | 0 | 9 |
| 14/Apr/20... | Info | Deny | ASA-sessi... | TCP | 192.168.2... | 192.168.1... | (empty) | (empty) | 4018 | 49155 | 49155_tcp | (empty) | 0 | 0 | 10 |

**Figure 10.8:** *Our parallel coordinates implementation. The blue lines each represent one firewall log, and the locations they cross on the vertical axes represent their values for those fields. The table below shows the same data, but in the original table format.*

Next, we experimented with parallel coordinates graphs, coupled with a table, both co-reactive to interactions with the other element. This view has the ability to provide a large amount of information in a format that is quick to understand. Unfortunately, when approaching large traffic flows this representation quickly becomes cluttered and hard to digest, even when interactive brushing renders only a user's selection.

We rendered a parallel coordinates chart from a small subset of the first day of firewall logs. The chart was slow to load and created a messy layout. This test showed the need for more work in user filtering and customization. Should a user choose a smaller subset of traffic from a separate coordinated view and choose which attributes to render on the parallel coordinate chart, they may be able to achieve a suitable visualization through their own work. This required deeper front and back end work on a visualization without a promising future. For this reason we chose to leave further exploration of this visualization for future work and continued elsewhere.

4. **Final Iteration**

Our final system consists of 3 main component views: the Network Overview, the Feature View, and the Detail View. A final Attribute view We found that by incorporating multiple views, we could provide users with both specific information on their selected firewall logs, as well as context about the overall network traffic surrounding those logs.

## 10.4 Components

1. **Network Overview**

The overview module of our visualization allows users to look at a wide range of firewall logs over time. This element of the visualization helps to serve as the first line of detection for anomalies such as DoS attacks, where the network is flooded with packets, as that stands out easily in a frequency graph.

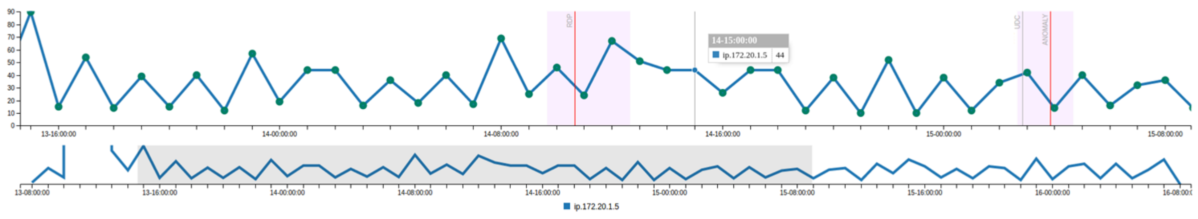


**Figure 10.9:** *Overview of all firewall logs over time. Users can select a range from the bottom portion of this view, and see a zoomed-in look at that range in the top portion. Areas shaded red indicate where we ran anomaly detection. Detected anomalies are marked with red lines.*

In Figure 10.9, which was mocked up with our prototype, the red regions are where the anomaly detection algorithm was run on the data. Red lines are detected anomalies, and gray lines represent ground-truth attacks. The subgraph below shows a small overview of the entire time period. A subsection of the overall range has been selected, showing a zoomed-in view of the region above where anomaly detection was run.

Users may brush over time ranges within the view by clicking and dragging, which allows them to see a zoomed-in view with more detail. The selected time range sets the time boundaries for the views in the rest of the visualization system.

2. **Detail View**

The detail view presents users with more information on specific time ranges in the log file. The detail view is populated with all of the log files in the range that users select in the overview. The detail view displays a table where users can see the low-level firewall log data, as well as a series of graphs which show frequencies of features over the selected time range. The user can select a row from the table to see a feature view for that specific log.

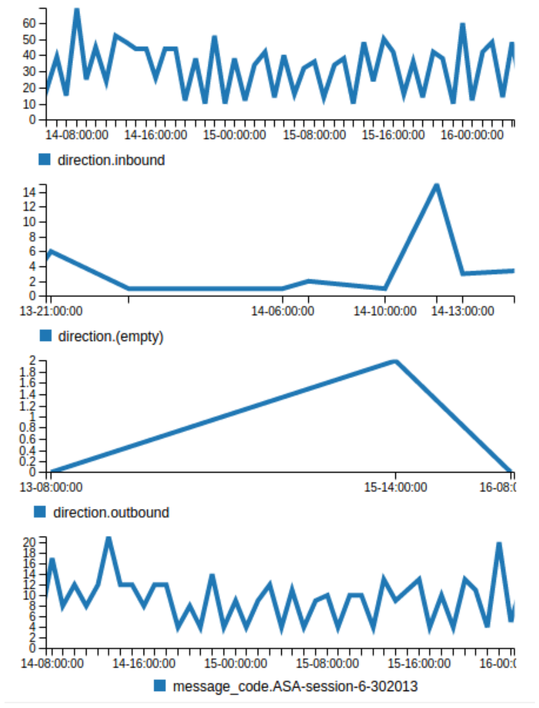


**Figure 10.10:** *Detail View (Graph Component). This view shows feature frequencies over a time range specified by the user.*

The detail view in Figure 10.10 shows four features and their frequencies over time during a time range specified by a user. Not pictured in this figure is the table where users can select a row from the firewall log to see specific data in the feature view.

3. **Feature View**

The feature view presents users with information on the features of whichever log they select. These features are from the feature set that we created earlier on, and users can see which of the 555 attributes were present in this row of the log file, as well as where this file differed from the norm, and why it was marked as an anomaly, if it was. Features are shown in blue if they are present, and if a feature is anomalous it is marked in orange. Small orange tick marks denote that the lack of a feature was anomalous. All features are labeled with the name of that specific feature, so that users can determine which elements caused that row to be marked as anomalous or not by the algorithm.

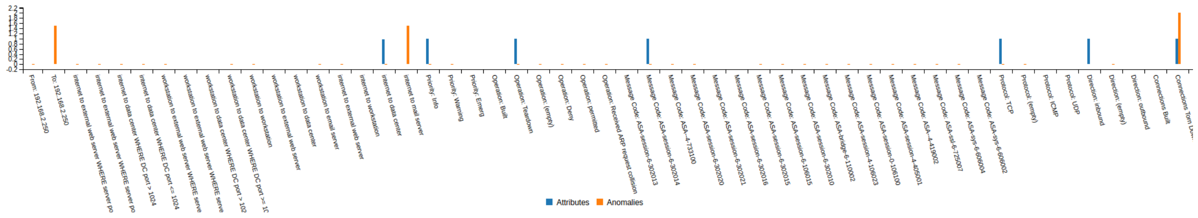


**Figure 10.11:** *Feature View in our visualization system. Orange lines indicate which features were implicated by the algorithm as anomalous. Blue lines indicate whether or not a feature is present.*

The feature view in Figure 10.11 shows a range of features (all 555 would not be legible in one page width) for a line selected from the firewall log by a user within the detail view.

## 10.5 Case Study: RDP

In order to evaluate the performance of our visualization system we staged a user in the position of conducting an investigation and the subsequent incident response after the anomaly detection algorithm marked a selection of firewall logs as anomalous . For our purposes we will only be using scans from the same regions that our anomaly detection algorithm was tested on. Therefore, the user in this case will have three days of firewall logs along with anomaly indicators representing output from the anamoly detection output run on a couple slices of the entire three days.

1. **Using the Overview**

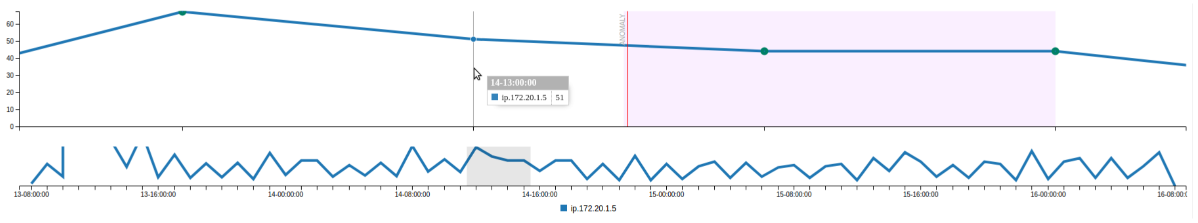


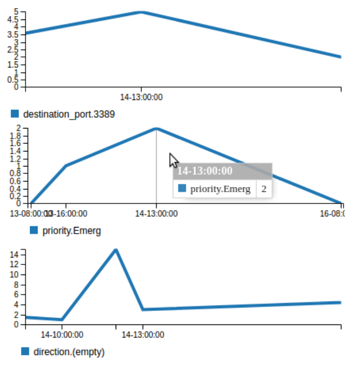
**Figure 10.12:** *Here we select a tight range around the detected anomaly in our Network Overview.*

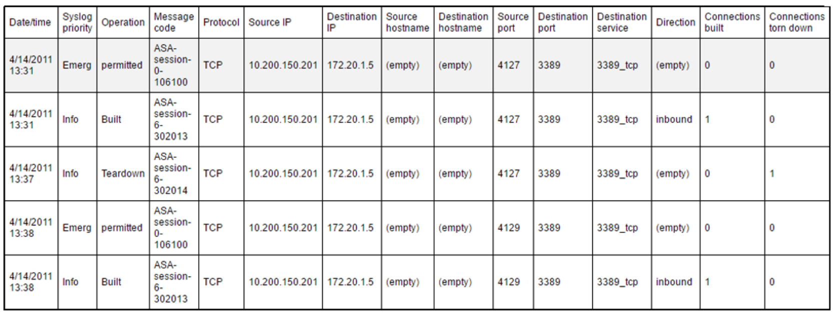
We first select the range that we would like to investigate with the Overview tool (see Figure 10.12. Looking at the timeline, we can select the range immediately surrounding the RDP attack, shown to the user as "ANOMALY", which has been detected by our anomaly detection system.

In the overview, we can now see the traffic immediately around the detected anomaly. Within that time period of the detected anomaly there are only 51 firewall logs. There are no obvious patterns in the frequency of traffic to indicate that there is malicious intent here. Therefore, the user takes away the number of firewall logs present in the time period and continues down to the other views.

2. **Using the Detail View**

With the graph component of the Detail View, seen in Figure 10.13, we can see the frequencies of features over the selected time range we selected in the Overview. From here, we can quickly identify the features that were unique to the RDP attack, as the graphs will depict flat lines of few occurrences with sudden spikes during the attack. We can select these features to sort the table element of the Detail View to display logs with these features first. Beginning with a visual inspection, there aren't many attributes that stand out during that time period. Despite lack of detail available from firewall logs, we still see three obvious peaks in frequency. Combing through the graphs we begin to find attributes with activity in our selected time period. We found three attributes here that were sparsely distributed and had a visible peak of activity during the time period of the anomaly.





| Date/time | Syslog priority | Operation | Message code | Protocol | Source IP | Destination IP | Source hostname | Destination hostname | Source port | Destination port | Destination service | Direction | Connections built | Connections torn down |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4/14/2011 13:31 | Emerg | permitted | ASA-session-0-106100 | TCP | 10.200.150.201 | 172.20.1.5 | (empty) | (empty) | 4127 | 3389 | 3389_tcp | (empty) | 0 | 0 |
| 4/14/2011 13:31 | Info | Built | ASA-session-6-302013 | TCP | 10.200.150.201 | 172.20.1.5 | (empty) | (empty) | 4127 | 3389 | 3389_tcp | inbound | 1 | 0 |
| 4/14/2011 13:37 | Info | Teardown | ASA-session-6-302014 | TCP | 10.200.150.201 | 172.20.1.5 | (empty) | (empty) | 4127 | 3389 | 3389_tcp | (empty) | 0 | 1 |
| 4/14/2011 13:38 | Emerg | permitted | ASA-session-0-106100 | TCP | 10.200.150.201 | 172.20.1.5 | (empty) | (empty) | 4129 | 3389 | 3389_tcp | (empty) | 0 | 0 |
| 4/14/2011 13:38 | Info | Built | ASA-session-6-302013 | TCP | 10.200.150.201 | 172.20.1.5 | (empty) | (empty) | 4129 | 3389 | 3389_tcp | inbound | 1 | 0 |

**Figure 10.13:** *Selecting the first feature from the graph view results in our anomalous log appearing as top row of the table.*

The top graph in Figure 10.13 shows traffic to port 3389, which happens to be the remote desktop protocol port. Only ports of known services that are actively used inside our time period will appear in these views. Zooming out from the time period in the chart will show that the use of this service across the network is sparse, therefore this attribute is of value to us so we select it to pin the view and continue investigating.

Next, we came across the second graph showing a priority of "Emerg". This one contains a more predominent spike. We adjust the view to see there are only two logs in our time range and only three over all three days. This implies that our firewall may have picked up on the anomaly as well. The final graph shows a direction of empty, which further investigation will show that it is a side effect of the "Emerg" priority. This chart is the most obvious sign of an anomaly as it both shows a spike in activity solely inside of our time range and indicates that the firewall is logging "empty" rows meaning that there is a cause for a serious investigation in this area. At this point we would be ready to escalate the anomaly to an "attack" or network violation and begin a deeper investigation.

With the table element of the Detail View, we can inspect individual lines of the firewall log. From the selections that we made in the graph component, we know the logs that we are looking for will be at the top of the table view. We can sort the table by any of the three attributes discovered from the feature views. However, we will continue the use of the table view without the knowledge gained from the detail view described above.

An analyst that continues straight from the overview to the table view can sort the table by anomaly and arrive directly at the rows that were implicated by the anomaly detection algorithm. Assuming knowledge of the rules set by the network administrator, we immediately realize that port 3389 or the remote desktop protocol service, along with a source IP originating from outside of our network means this connection is violating the company's network policy. At this point we escalate the anomaly and respond to this incident most likely by revising our firewall policy to block those connections instead of allowing them with a warning.

3. **Inspecting the Features**

Selecting a firewall log from the table view above brings us to the feature view of that log. In the Feature View, we can see all of the features for the log that we selected in the Detail View. We can view all of the features that the log does or does not possess, and can compare these with the features of regular data by selecting other lines from the table element of the Detail View. This way, we can see why the log was marked as anomalous, as well as how it compares to regular network traffic. In this case, we confirmed that the anomaly was implicated due to the attributes we pinned. In a different scenario we may arrive at this view and find no reason for it to be marked as anomalous. In this case we would begin investigating the reason it was marked by the anomaly detection algorithm in order to make changes and improve the algorithm as necessary. In this way we can include feedback from investigations as part our analysis of anomaly detection and use it as a feedback loop from which to better fit the algorithm to our particular network.

## 10.6 Uses

In our case studies with the VAST 2011 MC2 data, we found that our visualization helped to more easily show the results of the anomaly detection algorithm. By highlighting the anomalies that were detected with the algorithm users could easily brush over sections of firewall data and drill down to see which attributes are active in the time period of their selection and use that knowledge to sort a table of logs to inspect line by line alongside knowledge of which specific rows were marked as anomalous.

We believe that the overall system can be helpful to network analysts conducting incident response by helping to determining where attacks have occurred on their network, even when those attacks are not previously defined. If the anomaly detection algorithm is run on areas where the user thinks that there has been an attack, and the algorithm has been trained on previous attacks on the network, then our case studies have shown that it has been able to detect attacks, albeit only from our limited dataset. The visualization component should help users to not only identify attacks, but determine where they came from and how they differ from regular network traffic.

# CHAPTER 11

# FUTURE WORK IN DATA VISUALIZATION

The time-frame of our project as well as our group size meant that we were limited in what we could accomplish, and as such we were able to get a satisfactory protoype which works with the anomaly detection algorithm, but there were still other avenues that we would have liked to explore.

## 11.1 IDS and PCAP Data

Using only firewall logs constricted our analysis to network policy violations due to the nature of this data source as meta-data about connections as opposed to message level contents. While the VAST 2011 MC2 data is full of useful data sources, and contains useful representations of anomalies and attacks, none of these attacks we targeted for our anomaly detection algorithm are present in the PCAP data. We would like to work with either real-world packet captures or curated datasets where packet capture was done on regions of the network where anomalies existed.

Our project's time-frame also caused us to choose not to work with IDS logs. Visualizing IDS logs and presents another difficulty for our final visualization. IDS logs would be an interesting addition to the project, either if used in conjunction with firewall and PCAP data to create additional features or used alone to create a different feature set. It would, however, introduce a problem of trying to fit in this second source of network alerts in a distinct manner from anomaly detection output.

Something that would also be interesting to implement would be using vulnerability scans to create features based on specific vulnerabilities within the network. An example of this would be if a vulnerability scan reported that a machine with IP:X had a vulnerability on port:Y. To engineer useful features for this specific threat, we could create a feature to be something like

"DestinationIP:X and DestinationPort:Y" to isolate this specific vulnerability. This would make traffic that was implicated by this feature to stand out slightly from the rest.

## 11.2 Real-Time System

Another feature that we would like to implement would be to make the visualization, and also anomaly detection, available in real-time, or close to real time. At this point, the visualizations take a few seconds each to load, and the anomaly detection takes a few seconds to run on each 5000-line slice of firewall data. The visualizations would be able to work in a real time system that updated only every few seconds, but the anomaly detection would end up slowing things down in a larger network or if used on broader sections of data. In a small network with fewer firewall log lines per minute, it could be feasible to implement our entire system, albeit with some changes to our firewall log processing pipeline.

Currently, our visualization runs off of an SQL database, and the anomaly detection is run on .csv files that are generated by running our feature-engineering code on the original .csv firewall log files. In a real-time system, the saving of these .csv files and loading them, which is done in python and is not involved with our visualization web interface at this time, would cause significant slowdowns beyond just the processing time for each slice of data. With enough time to develop however, it would be possible to streamline this process into one workflow, and create a data pipeline to introduce concurrency into the process and achieve near real-time analysis.

## 11.3 Deep Packet Inspection

Deep packet inspection features would also be an element that we would be interested in adding to this project. With our current data source, we are limited in the depth of detail we can provide and inspect in our visualization system. By definition, this means we cannot detect APTs (Advanced Persistent Threats), malicious users that are using knowledge of our network architecture to hide their activity. However, if we were able to work with PCAP data instead, including packet features and message level contexts would help this system take the next step towards being a fully integrated incident response system for all types of attacks. Currently, a user would have to match firewall logs to packets in a separetly open application (i.e. Wireshark) in order to achieve the level of depth necessary to investigate APTs.

# Chapter 12
# Visualization Conclusions

Our visualization system ended up being comprised of multiple, fairly simple views. The views work in conjunction with each other to provide users with network context, as well as low-level firewall log details. Our system is able to incorporate the results of our anomaly detection algorithm in such a way that end users are able to easily spot and investigate identified anomalies.

# Bibliography

[1] “Vast challenge 2011 - mini-challenge 2 computer network operations at all freight corporation,” 2011.

[2] K. K. Randy Paffenroth and L. Servi, “Robust pca for anomaly detection in cyber networks,” 2016.

[3] C. Eckart and G. Young, “The approximation of one matrix by another of lower rank,” *Psychometrika*, vol. 1, no. 3.

[4] E. J. Candès, X. Li, Y. Ma, and J. Wright, “Robust principal component analysis?,” *J. ACM*, vol. 58, pp. 11:1–11:37, June 2011.

[5] J. E. Beck and B. P. Woolf, *High-Level Student Modeling with Machine Learning*, pp. 584–593.
Berlin, Heidelberg: Springer Berlin Heidelberg, 2000.

[6] H. Abdi and L. J. Williams, “Principal component analysis,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[7] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*, vol. 6.
Springer, 2013.

[8] E. J. Candès and T. Tao, “The power of convex relaxation: Near-optimal matrix completion,” *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2053–2080, 2010.

[9] D. Gross, “Recovering low-rank matrices from few coefficients in any basis,” *CoRR*, vol. abs/0910.1879, 2009.

[10] E. J. Candes and Y. Plan, “Matrix completion with noise,” *Proceedings of the IEEE*, vol. 98, no. 6, pp. 925–936, 2010.

[11] E. Candès and B. Recht, “Exact matrix completion via convex optimization,” *Commun. ACM*, vol. 55, pp. 111–119, June 2012.

[12] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, “Rank-sparsity incoherence for matrix decomposition,” *SIAM Journal on Optimization*, vol. 21, no. 2, pp. 572–596, 2011.

[13] G. W. Milligan and M. C. Cooper, “A study of standardization of variables in cluster analysis,” *Journal of Classification*, vol. 5, no. 2, pp. 181–204, 1988.

[14] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[15] L. Breiman, “Bagging predictors,” *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[16] J. R. Quinlan *et al.*, “Bagging, boosting, and c4. 5,” in *AAAI/IAAI, Vol. 1*, pp. 725–730, 1996.

[17] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, pp. 5–32, Oct. 2001.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[19] W. M. Lamagna, “University of buenos aires – svc+canvas+processing heatmaps,” 2011.

[20] B. C. M. Cappers and J. J. van Wijk, “Understanding the context of network traffic alerts,” in *2016 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pp. 1–8, Oct 2016.

[21] J. Krause, A. Perer, and E. Bertini, “Infuse: Interactive feature selection for predictive modeling of high dimensional data,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, pp. 1614–1623, Dec 2014.

[22] W. Lee and S. J. Stolfo, “Data mining approaches for intrusion detection,” in *Proceedings of the 7th Conference on USENIX Security Symposium - Volume 7*, SSYM’98, (Berkeley, CA, USA), pp. 6–6, USENIX Association, 1998.

[23] J. Landstorfer, I. Herrmann, J. E. Stange, M. Dörk, and R. Wettach, “Weaving a carpet from log entries: A network security visualization built with co-creation,” in *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 73–82, Oct 2014.

[24] T. Zhang, Q. Liao, and L. Shi, “Bridging the gap of network management and anomaly detection through interactive visualization,” in *2014 IEEE Pacific Visualization Symposium*, pp. 253–257, March 2014.

[25] J. C. Roberts, “State of the art: Coordinated multiple views in exploratory visualization,” in *Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2007)*, pp. 61–71, July 2007.

[26] H. Shiravi, A. Shiravi, and A. A. Ghorbani, “A survey of visualization systems for network security,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, pp. 1313–1329, Aug 2012.

[27] J. R. Goodall, W. G. Lutters, P. Rheingans, and A. Komlodi, “Preserving the big picture: visual network traffic analysis with tnv,” in *IEEE Workshop on Visualization for Computer Security, 2005. (VizSEC 05).*, pp. 47–54, Oct 2005.