Network Anomaly Detection Utilizing Robust Principal Component Analysis

Major Qualifying Project

Advisors:

PROFESSORS LANE HARRISON, RANDY PAFFENROTH

Written By:

AURA VELARDE RAMIREZ
ERIK SOLA PLEITEZ

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.

AUGUST 24, 2017 - MARCH 2, 2018

Abstract

In this Major Qualifying Project, we focus on the development of a visualization-enabled anomaly detection system. We examine the 2011 VAST dataset challenge to efficiently generate meaningful features and apply Robust Principal Component Analysis (RPCA) to detect any data points estimated to be anomalous. This is done through an infrastructure that promotes the closing of the loop from feature generation to anomaly detection through RPCA. We enable our user to choose subsets of the data through a web application and learn through visualization systems where problems are within their chosen local data slice. In this report, we explore both feature engineering techniques along with optimizing RPCA which ultimately lead to a generalized approach for detecting anomalies within a defined network architecture.

TABLE OF CONTENTS

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

1.1 Introduction

It would be difficult to speak on the successes of technology in past years without discussing the recent attacks on user data. The data explosion from technological growth has armed analysts with new strategies for predicting or detecting malicious activity. Taking data from previous attacks can help us prevent or stop future ones from emerging. This method of preventing attacks can be summarized as finding outliers in a data set – or an anomaly. An anomaly is a data point that veers away normal trends in a data set. From credit card fraud detection to cybersecurity attacks, detecting anomalies has become vital in ensuring privacy. Nevertheless, due to differences between data in these domains, a generalized solution for detecting out of the ordinary behavior becomes challenging. In this paper, we will focus on anomaly detection in cybersecurity of the 2011 VAST dataset challenge.

Between the Internet of Things, the rapid advancement of technology and the lack of regulation in the past years, security has become a primary concern for millions of users. In addition, anomaly detection in networks has various layers of mathematical complexity. Deciding which data points seem out of place requires precise analysis of data. This, coupled with the enormous size of data sets, subtle correlation between data points, and potential long system waits for each run cycle makes the process known as feature engineering non-trivial [9].

Security issues have been steadily present in software companies as technology continues to grow in use and complexity. Considering how heavily embedded technology has become in everyday life, cybersecurity is an issue of the highest priority. Damage costs alone from cyber attacks are projected to reach $6 trillion by 2021 and breaches of personal information are occurring with higher frequency as time progresses [48]. Over the past years, government records have been victimized of cyber attacks, for example, the Free Application for Federal Student Aid (FAFSA)

experienced a security breach within its IRS retrieval application. This led to roughly 100,000 taxpayers information being compromised [6]. As time and technology progresses, malicious infiltrations such as the FAFSA breach become increasingly more difficult to predict or detect. To protect user information, avoid denial of service attacks, along with other types of infiltration, it is vital to detect what has gone wrong in the past regarding security. Determining the root cause of these issues assist improving the security of critical information and user privacy.

Anomaly detection has been researched extensively for cybersecurity due to the complexities that it entails. The University of Minnesota's, *Anomaly Detection: A survey*, looked at using network intrusions detection systems to find anomalies. However, detecting anomalous behavior through network data comes with several challenges. This type of dataset is inherently high dimensional and the domain continuously changes over time as intruders adapt and overcome intrusion detections advancements [20].

In this Major Qualifying Project (MQP), we examined ways to diversify a dataset through feature engineering and analyze its relationship with Robust Principal Component Analysis (RPCA). Our contributions were the following:

- Created a user-friendly visualization system.
- Closed the loop between feature generation, mathematical analysis, and visualization.
- Improved overall experience of system administrators by creating a streamlined process through careful construction of sound infrastructure in our code base.

This report explains in depth what anomaly detection is, its process, the different statistical analysis methods that we will use or recommend to use, what feature engineering is, and the impact that the original data set we used had on our project along with the assumptions made.

CHAPTER 2

VAST DATASET CHALLENGE

For this project, we used the 2011 VAST dataset [5]. This dataset included network architecture descriptions, security policy issues, firewall logs, an intrusion detection system (IDS) log, and a Nessus Network Vulnerability Scan Report [24] for the All Freight Corporation (AFC) [5]. The goal of the challenge was to develop situation awareness interfaces that provide insight quickly, clearly, and as efficiently as possible to help manage daily business operations while mitigating network vulnerabilities. In this chapter, we will explain the structure of AFC's network and the associated dataset.

2.1 2011 VAST Dataset

Feature engineering refers to the process of using knowledge of data to create features that can make machine learning algorithms function [47]. More specifically, this refers to attributes that could contribute to the analysis of data to help with its prediction or analysis. Features are characteristics of data that will help distinguish each row of data from each other. For example, a feature would be whether or not a log entry is using a port that has a specific type of vulnerability. This feature becomes relevant as we know that attacks are more likely to come from a computer that might be vulnerable. In general, features surround the balance between finding as much relevant data as possible to ensure that it is unique enough to make an impact in the prediction models.

The dataset includes a folder of firewall logs for four separate days April 13 - 16 of 2011. Each day has one corresponding log file. However, in the instance that the number of rows exceeds Excel's row limit multiple files are created. April 13 is such a case, where there exists more than one log file. There are certain nodes within the dataset that are critical for AFC's network to function properly. It is to be noted that AFC uses virtual machines within their network. In

table 2.1, all nodes are described in the VAST dataset challenge; any node outside 172.x.x.x and 192.x.x.x is external to AFC's network.

IP Address	Node Name	Node Type	Description	Priority
10.200.150.1		Firewall	Firewall interface to the Internet	High
172.20.1.1		Firewall	Firewall interface to External Web Server	High
172.20.1.5		External Web Server	Web server which hosts All Freight's external web site	High
192.168.1.16	Snort IDS	Snort Intrusion Detection System	Snort IDS interface to the network	High
192.168.1.1		Firewall	Firewall interface to data center Virtual Local Area Network (VLAN)	High
192.168.2.1		Firewall	Firewall interface to office VLAN	High
192.168.1.2	DC01	Domain Controller (DC) / Domain Name System (DNS)/ Dynamic Host Configuration Protocol (DHCP) serve	Server running critical network operations: domain controller, domain name server, and dynamic host configuration protocol server	High
192.168.1.3	HRDB01	HR Database Server	Server running the database for employee payroll and benefits	High
192.168.1.4	SRDB01	Shipping / Routing Database Server	Server containing customer data, including shipping requests and routing information	High

192.168.1.5	WEB01	Internal web server	Server that hosts All Freight's corporate intranet, including company news site and policy and procedure manuals	High
192.168.1.5	WEB01	Internal web server	Server that hosts All Freight's corporate intranet, including company news site and policy and procedure manuals	High
192.168.1.6	EX01	Mail server	Server that stores and routes all email that flows into, out of, or internal to All Freight	High
192.168.1.7	FS01	File Server	Server that holds shared files used by workers throughout All Freight	High
192.168.1.14	DC2	DC / DNS server	Server running critical network operations: domain controller and domain name server	High
192.168.1.50		Firewall log	Server that captures system firewall logs	High
192.168.2.10 through 192.168.2.250		Office workstations	Individual workstation computers located in offices or cubicles throughout All Freight	Normal

Table 2.1: *AFC registered network description. [5].*

Important data flow descriptions are:

- Connections outside of the AFC network
  - Web traffic enters with IP address 10.200.150.1 and through port 80.
  - Firewall routes traffic to the external web server on 172.20.1.5 address and through port 80.

- Email from outside AFC network
  - Enter AFC's network with IP address 10.200.150.1 through port 25.
  - Firewall routes traffic to the mail server on IP address 192.168.1.6.
- All AFC staff members use IP addresses 192.168.2.10-250 to browse the internet.

All information above retrieved from VAST dataset challenge description. Fully understanding the structure of a dataset and how data flows in AFC's network is critical to the success of this project and taken into consideration during its execution. In general, a company's policy and security contract should be taken into consideration when creating features.

2.2 Attacks in the VAST Dataset

In addition to the resources mentioned above, the VAST dataset also includes a solutions manual. The answer guide reveals all attacks and steps that led to finding the security vulnerabilities in the VAST dataset. Below are summaries of each attack over the course of four days according to the solution manual.

Type of Attack	Date of Attack	Description
Denial Of Service Attack (DDoS)	04/13/2011 at 11:39 and ended roughly an hour later at 12:51	A DDoS attack aims to disrupt a user's interaction with a system. By using client/server technology an attacker can flood requests of a network, rendering that network useless or experiencing delayed speeds [56]. Throughout this time period there was an attempt to disrupt a corporate web server, most likely to delay network speeds.

Remote Desktop Connection	04/14/2011 at 13:31	Documented violation of corporate policy within the firewall logs. This is part of a socially engineered attack that suggests substantial security risk such as a worm in AFC's network.
Undocumented Computer	04/14/2011 13:23	There was an addition of an undocumented computer in the company internal network. The policy descriptions for AFC said that the workstation computers would be in the range 192.168.2.25-250. The particular IP address for this computer is 192.168.2.251. Although we do not have the background for this computer, the addition of it to the company network is concerning enough that it should be noticed.

Table 2.2: *Attack Descriptions on VAST dataset [5]*

2.3 Avoiding Data Snooping

Data snooping is when an algorithm can cheat through previous knowledge of the answers rather than depend on how data presents itself normally. We used the context of the attacks mentioned above to verify the validity of our features and assist with the iterative feature engineering process. One of our primary concerns of having the answers was the possibility of data snooping. Although we did take inspiration from knowing the attacks and previous MQPs, we limited ourselves to not seeing the answers until we had our first iterations of features. After observing the answers, we analyzed the problems in general to create features that would have the capabilities to solve these issues in a generalized way. For example, when we noted the Remote Desktop attack, we thought about all different types of protocols that could indicate that a computer is being connected from a suspicious place and how ports can hint at suspicious activity.

2.4 Previous Work

In the previous year, there was an MQP that focused on an anomaly detection system. After analysis of their work, we determined that their application was set up in a way that created a disconnected loop of work between mathematicians and computer scientists. From the computer science side, there were matrices of features developed and sent to mathematicians for analysis. These matrices were derived from the 2011 VAST dataset challenge. It was important to determine how their features were produced from this data.

After a code analysis, we determined that the VAST 2011 dataset was ingested into a Python script and then the resulting feature matrix was inserted into a mySQL [10] database. The mySQL tables were statically defined, meaning that the table schema was established during the creation of the feature matrix with a predetermined set of columns. We believe that this posed a problem for a generalized solution. Our initial thought was to research ways of dynamically creating the table schema for the mySQL table. To do this, we chose to optimize the process of running the project and feature matrix creation. Our Python script allowed the feature matrix to be read for mathematical analysis and from here, rows were marked as anomalous or not. This was done with the ground truth from the solution manual and served for cross validation of attacks. The main problem with the previous MQP project was the disconnect between each step in the iterative loop, this needed to be streamlined. With the past MQP's structure of feature generation and math analysis, we chose to close the loop that began with feature engineering, passed to mathematical analysis, and resulted in visualization.

Chapter 3

Anomalies in Cyber Security

The exponential increase of technology in the past years has made malicious intrusions in networks substantially harder to detect anomalies. The heavy integration of technology into our lives has made detecting intrusions all the more important [21]. Anomaly detection involves looking at previous data and searching for behavior that exists out of the norm. Anomaly detection methods create the possibility to find unknown attacks on a system once a normality has been defined. However, false positives are still possible due to the unpredictable nature of network use.

3.1 Anomaly detection methods

Anomaly detection is the process of finding data that is out of the ordinary and does not conform to previous data trends [20]. The difficulty levels to find anomaly corresponds to the layers of complexity a dataset provides.

In Figure 3.1, it is obvious to see that there are points that do not behave as the rest of the data. It is important to note that in this example there are only two dimensions. In a real world example there can be multiple dimensions. Each dimension adds another layer of complexity and can affect or even hide anomalous behavior [20]. One example is network data and traffic. Network data consists of communicating with multiple computers or nodes [40]. Figure 3.2 is an example of how complex network traffic can look in comparison to the simple example above.

In Figure 3.2, we see multiple systems communicating with each other. Each node in this diagram represents a layer of complexity in a network. Finding behavior that is out of the norm with everything taken into consideration very quickly becomes a daunting task. With more data ingested at a fast rate, data becomes noisy and harder to filter through. Regarding network data it is vital to filter through this noise. “Network traffic anomalies reflect the anomalous or malicious

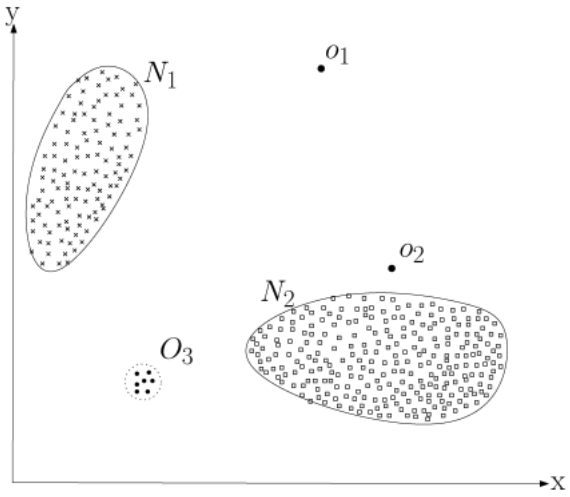


Figure 3.1: *Example of anomaly in two dimensional space [20].*

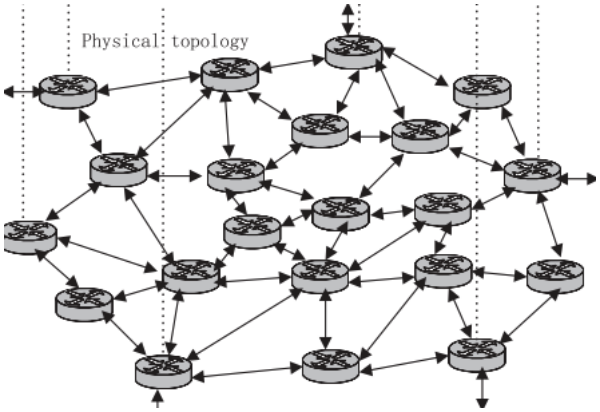


Figure 3.2: *Example of dimensionality in network traffic. Each node represents another layer of complexity. This makes detecting anomalies a non-trivial task [20]*

behaviors that appear in the network. Discovering exact/actual network traffic anomalies is to effectively contain these anomalous or malicious behaviors that damage the network" [69]. These network traffic anomalies communicate a story of attacks in cyber security and it is the responsibility of anomaly detection methods to ensure that they are caught and examined. The question that many experts are posing is which anomaly detection methods are best and how can we better prepare ourselves for unknown attacks [13].

Different project paradigms require different types of anomaly detection methods [20]. Testing different types of anomaly detection methods can result in an increased chance of finding anomalous behavior [13]. In recent years, the different types of anomaly detections methods have skyrocketed. This stems from advancements in technology and an increase in targetable domains [13]. The challenge lies in identifying the correct type of anomaly detection method (ADM) for a

domain. Many anomaly detection methods have different requirements and prerequisites that can makes the process even more difficult. Below is a general architecture for an anomaly detection system.

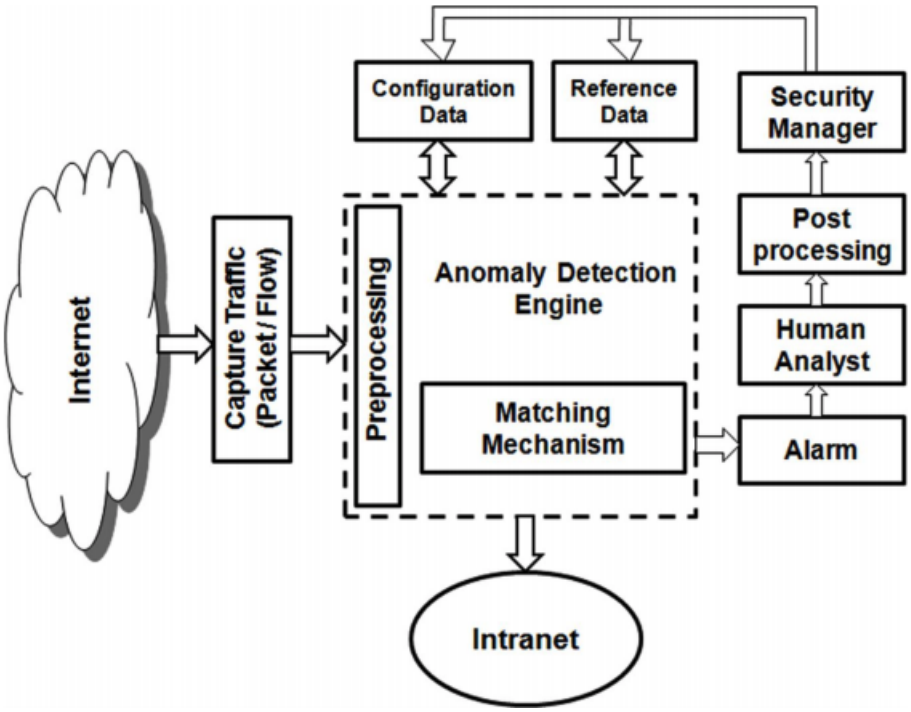


Figure 3.3: *General architecture of anomaly detection system from (Chandola et al., 2009) [20]*

Although Figure 3.3 is a generalized example of an anomaly detection system, it accurately describes the steps that are taken into consideration for examining a dataset. The actual anomaly detection engine is what will normally vary. There are several examples of anomaly detection methods. Monowar H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita's [20] paper mentioned above categorizes network intrusion methods into the following categories: statistical, classification-based, clustering and outlier-based, soft computing, knowledge-based and combination learners. Most of these anomaly detection methods use some form of feature engineering to optimize their anomaly detection algorithms. Bhuyan, Bhattacharyya, Kalita argue that "feature selection reduces computational complexity, removes information redundancy, increases the accuracy of the detection algorithm, facilitates data understanding and improves generalization" [13]. It is therefore, important that the feature selection process is carefully chosen based on a dataset.

CHAPTER 4

FEATURE ENGINEERING

Feature engineering is analyzing data to carefully separate data points in a useful way. The goal is to diversify and create meaningful distances between data points. To be more specific, when analysts first receive data, it might have what seems as redundant or not specific enough information. As stated in chapter 3, feature selection enhances anomaly detection algorithms. In order to choose which data points are anomalies, you have to select which details need to be analyzed, and most importantly, how. One of the challenges about data mining is the different perspective that a user or analyst can have in comparison with a developer. Looking at the problem domain from two different perspectives can derive to a different set of features. [63].

4.1 Feature Engineering Process

The process of feature engineering is an iterative one that ceases when the end goal of a project is met. The process is defined below by Dr. Jason Brownlee [17]:

1. Preprocess Data: Format it, clean it, sample it so you can work with it.
    a) This may involve deleting certain columns after determining any irrelevancies.
    b) Changing the format of certain columns, such as time-stamps or unwanted characters.
2. Transform Data: Feature selection happens here.
    a) The feature library consists of several uniquely identifying characteristics of the data.
    b) Features should not be made for the sake of creation. For example, a timestamp feature would diversify your data; however, that might not help you achieve your goal.
3. Model Data: Create models, evaluate them and fine tune them.

a) After receiving results from the feature selection, how will you visualization or present your findings? How can the transformed data set be modeled to communicate a message?
b) Ensure that your features tell a story within your modeling of data.

Through each iteration of this process more is revealed about a dataset and features are fine tuned until a best fit is found. The best fit varies for each project. Overall, it is important to consider if features are thoughtfully chosen and if through the modeling of features a story is clearly communicated.

4.2 Feature Selection For a Dataset

What specifically defines a feature is what makes feature engineering both challenging, and interesting. Intuitively, one may think of a feature as system functionality that helps characterize the system in the perspective of a human. However, this definition tends to be perceived in different ways, causing confusion and conflicting opinions [63]. Using the time entry of a data point, for example, could be considered a feature because it makes each log entry into a separate data point. Nevertheless, this does not necessarily describe the system or how it works. A separate challenge with feature engineering is the limit of file sizes. Although in an ideal world, you could create a feature for each uniquely identifying characteristic, it's imperative to consider what is possible with the current computational power, and prioritize which features may lead to solving a problem. The selection of features is a task which requires an analytical approach. A dataset must be examined to define which characteristics should be made unique. As mentioned above, choosing the wrong features will not help in the modeling of a dataset and can hinder the problem that feature engineering aims to solve. In the feature selection step is it vital to remove attributes of data that may make a dataset "noisy" and skew any modeling while limiting the number of features to those of higher importance [17].

4.3 Time Series Feature Generation

Although features tend to describe characteristics of a specific dataset, anomaly searching can require an introspective view of your specific domain. Network security has unique attributes that are imperative towards understanding how the network itself is functioning. Osman Ramadan [51], decided to use the number of distinct source and destination ports/IP's and the sum of bytes for each source over a period of time as features since he knew that this could describe a change in the network's normal behavior, like a DDoS attack [51]. Our dataset did not include the exact number of bytes transferred within each request. However, in light of his findings we decided to see if it would be possible to derive a similar concept. How could we analyze when a

source/destination port/ip was experiencing a change in the network? Questions such as these are key to help us derive rules for modeling data and create our anomaly detection system.

4.4 Feature Engineering Generation

In the following sections we will discuss the process of our feature generation on the VAST dataset. The goal was to generate a set of features that increases uniqueness among the dataset in preparation for mathematical analysis, such as RPCA. Over the course of the project's completion we followed the iterative process defined above. Below is an example of the loop that resulted in our feature library. In this section, we also explain the workflow of the project. We will go in depth for each step and the part it played in closing the loop in this MQP.

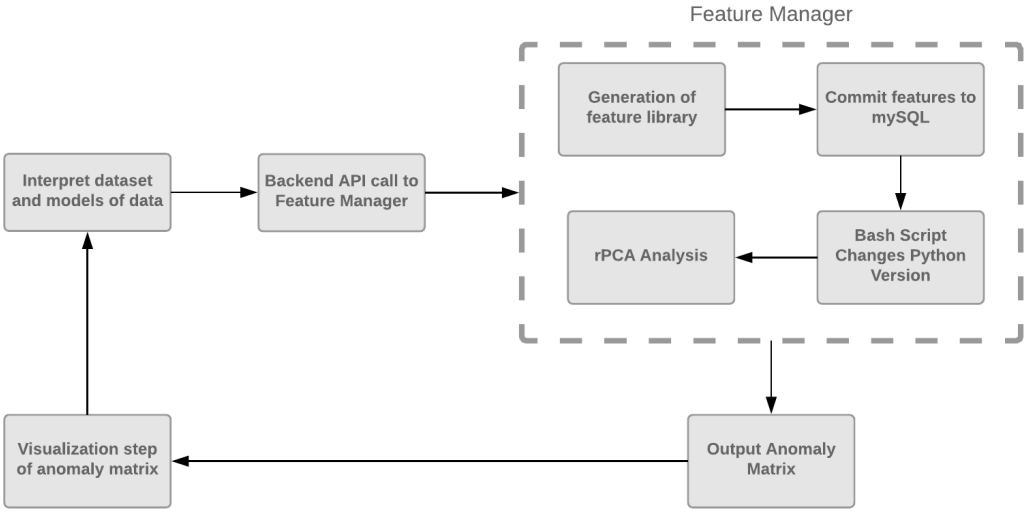


Figure 4.1: *A goal of this project was to create a system that closes the loop between an anomaly detecting system and visualization.*

Once this cycle has converged as a result of an optimal feature set, the next step is visualization. This step will be described in depth in chapter 7.

4.5 Feature Generation Infrastructure

In the following sections, the process of choosing our features for the VAST dataset will be described. This section aims to explain the flow of work that led to the efficient generation of our feature library. As described above, a challenge that lies in feature generation is the size of the dataset. In the VAST dataset, with all 4 days of logs, there are roughly 12.5 million rows. Through trial and error, it was determined that we could not feasibly automate and streamline

the feature generation process with the entire dataset. Processing the data would require very long system waits for feature generation. To mitigate this problem, a Python script functions as a manager for all of the scripts that produce our feature library. Our backend is created using Flask [53], a micro-web framework written in Python to build our web application. This manager is used within our Flask backend explained in chapter 7, and our web application to obtain a desired time slice for a specific log file. This makes it much more manageable to generate features and test different slices.

Our manager also bears the responsibility of running mathematical analysis. It assists with easy experimentation of our algorithm, as explained in chapter 4. It is to be noted that feature generation occurs with Python 2.7 and math analysis uses Python 3.6. There are ways to translate Python 2 to Python 3 with libraries such as Futurize [58] but with several moving components in our code base and not anticipating this problem in the beginning of the project we decided to not convert our code. One could imagine, that with time, a manager could be written entirely in one Python version or one that utilizes the something such as the Futurize library. As with many Python projects, we decided to develop from within a virtual environment using Python's VirtualEnv [14]. We took advantage of this decision and have two Python virtual environments, one for Python 2.7 and one with Python 3.6. All dependencies are clearly defined within our code repository readme [59]. Our manager begins feature generation code in the Python 2.7 virtual environment and a bash script is started once feature generation is complete. This bash script deactivates the Python 2.7 virtual environment, activates the Python 3.6 environment to run our math algorithm on the feature matrix csv file. This process closes the gap between computer science and mathematical analysis. What follows is the visualization step of the results for a user, this is explained in more detail in Chapter 6. A user can use this manager and examine any number of ranges for any log. Another advantage of having a manager script is that adding or removing specific feature sets become infinitely easier.

Since the feature generation process is an iterative one that constantly aims to improve in order to optimize the anomaly detection algorithm, creating a pipeline for this project was necessary. This pipeline allowed us to test several different time slices and assisted with the best fit convergence of our feature library.

4.6 Source and Destination IP Address Feature Sets

The goal output of our features is to create a hot encoded matrix in Excel, or a numerical value to demonstrate if a characteristic applies to the data point [40]. It is difficult to apply anomaly detection methods if the features are not chosen carefully. In our features, every IP address was converted to an integer using Python's struct library. In this section we will describe the trials of deciding how to diversify the IP addresses within the dataset.

4.6.1 Original Feature For Every Unique Source and Destination IP

We used a script in order to find every unique source and destination IP address and we dynamically generated features from this. This script examined a user specified slice of the dataset and generated all of the corresponding distinct IP addresses. From here our feature manager starts another script that will create the feature matrix with respect to these distinct IP addresses. All IP addresses are converted to integers as explained above. Source IP addresses have the prefix "sip_" followed by the integer IP address and destination IP addresses have the prefix "dip_" followed by the integer IP address. Below is an example of how the source IP addresses may be displayed within this specific feature set.

sip_18091791	sip_18091792	sip_18091793	sip_32322361	sip_32322362
0	0	0	0	1
1	0	0	0	0

Table 4.1: *Original IP Address feature example. In this feature set, there is a columns for every distinct IP address.*

This matrix is then inserted into a mySQL table for easy manipulation of data. For example, join in each IP address to a vulnerability scan is one example that can be used in feature generation that would require that flexibility. By having each unique source and destination IP address be its own feature we diversify the dataset and widen our matrix. However, it is important to note the impact that doing this could have on the math analysis and other overall infrastructure of our project. If a user defines a very large time slice of the dataset, the dynamically created features could number in the thousands. This makes it difficult to insert data into mySQL and make queries on it since the table would become so large. For example, if we run our script to generate features for every distinct source and destination IP address, along with our other features, our table would have upwards of a thousand features. In addition, this feature set can hurt the goal of anomaly detection if most of the used IP addresses are unique, since our math algorithm could potentially flag each row as an anomaly. It can be argued that with smaller slices this is a very feasible feature set and can help the diversification of the dataset.

4.6.2 Feature Engineering of the AFC Network IP Addresses

As mentioned in chapter 2, the VAST 2011 dataset challenge included a table of IP addresses that describes the AFC company network architecture.

Figure 3.3, maps the overall transfer and communication of data within AFC's network. All of the IP addresses in this figure are referenced in chapter 2's table. Our next IP address feature

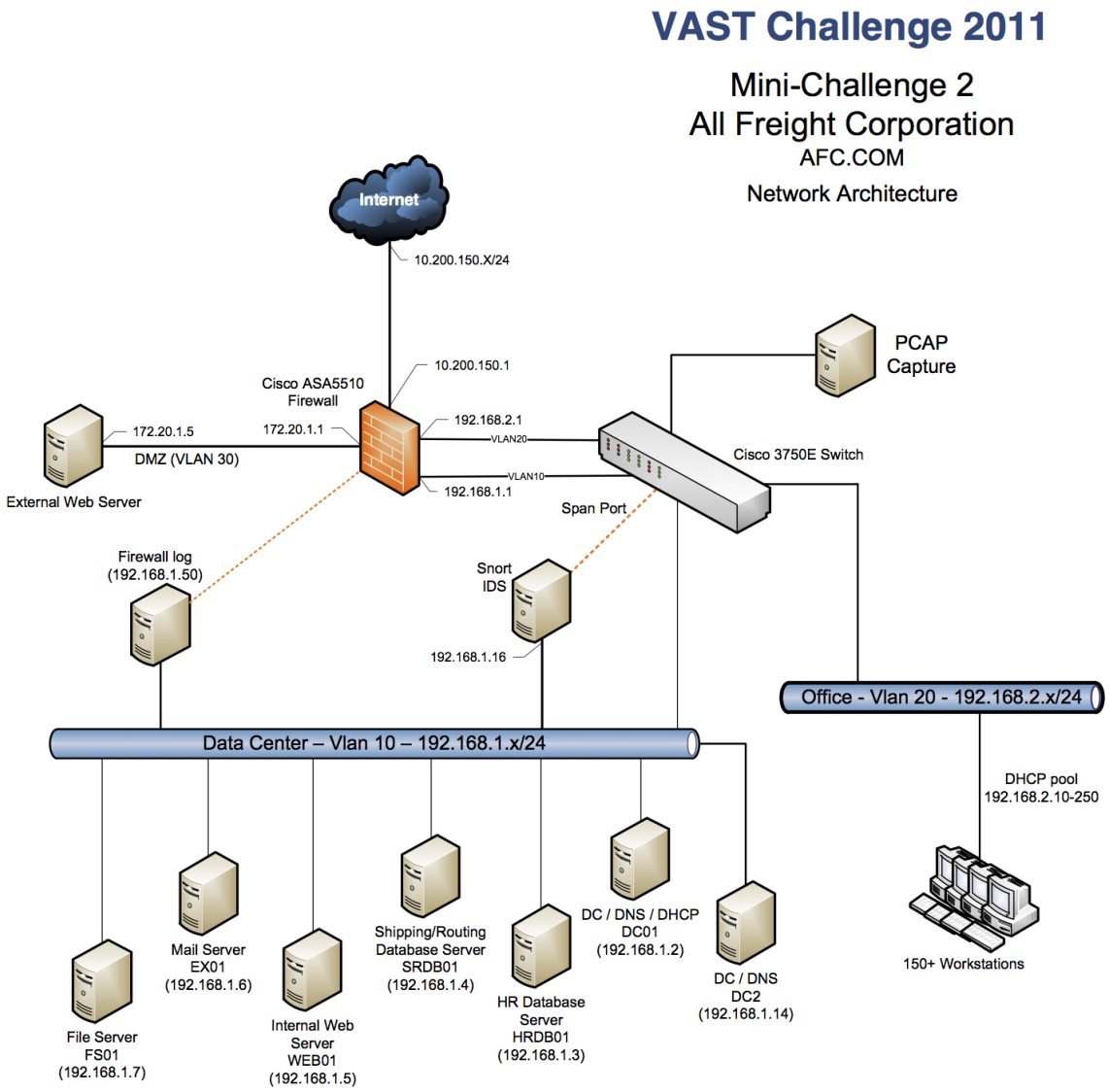


Figure 4.2: *AFC Network Architecture. This outlines the workstation connections in AFC's network and which servers they communicate on. [5]*

set includes the thoughtful consideration of exclusively AFC's network. For each IP address in table 2.1 we have a source and destination feature. Similarly to the previous section's feature set, we prefix each IP address with "sip_" and "dip_" and convert each address to an integer. In addition, there are two features responsible for characterizing any source or destination IP addresses outside of AFC's defined network called "sip_outside" and "dip_outside". Below is an example of how these features look.

sip_outside	sip_18091777	sip_18091793	sip_32322361	sip_32322362
0	0	0	0	1
1	0	0	0	0

Table 4.2: *Individual Column features for each AFC network IP address. With this feature set, there is a column for every IP address defined in AFC's network and there are two columns for source and destination IP addresses outside the defined network.*

There may be several reasons for communication with an outside IP address, however, it is important to mark these IP addresses as they could be the root of malicious activity. In a real world setting, one could imagine speaking with a network analyst at any given company and asking for a similar network architecture diagram or table. By doing this, someone could use our software and place their own IP addresses in a simple table in our Python script and run their own feature generation. This is where our goal of a more generalized solution for anomaly detection is accomplished. In the previous section, the features change from each feature generation run cycle, but in this situation the feature set for IP addresses stay the same. This feature set encompasses the entire AFC network architecture.

By mapping out each IP address in AFC's network as a feature we characterize their company specific architecture. This could make identifying specific workstations or other terminals as a problem quite simple. As for marking outside sources the outside features can alert any source or destination IP address that is external to the AFC network. In the next section we aim to generalize the IP addresses even further.

4.6.3 Examining AFC Network IP Addresses Inside and Outside

This IP address features is a slight variation of the previous section. We wanted to try another version of examining AFC's network architecture. This set includes four features in total: "sip_outside", "sip_inside", "dip_outside", and "dip_inside". Rather than having a feature for

each IP address, we use these features and simply marked whether an IP addresses is in a user defined slice is inside or outside the AFC network. Below is an example of how these features look.

sip_outside	sip_inside	dip_outside	dip_inside
0	1	0	1
0	1	0	1

Table 4.3: *Examining only inside and outside source and destination IP addresses according to AFC's network architecture.*

It was important to test several variations for the IP address features because this attribute in the firewall log can tell an interesting story when it comes to suspicious activity. For the "sip_inside" and "dip_inside" features we still ingest the AFC network architecture similar to the previous section. However, in this set we consolidate all of those features into these two columns. So every IP address defined in AFC's network will be placed within those two columns. The "sip_outside" and "dip_outside" functions the same as the previous section where all IP addresses that are outside of AFC's network will be marked there.

4.7 Ports

Ports can reveal much about abnormalities in network data. For this reason we found it necessary to incorporate them into this project's feature library. We created features for the ports and grouped them according to overall significance to possible malicious attacks. Port numbers below 1024 are known as system or well-known ports. They are specifically meant to be used for certain applications and the system expects them to perform certain tasks. On the other spectrum, ports above 1024 are ports and they can be used by users or are registered. An example is how the 8080 port is registered for websites. For our features, we grouped all port numbers. Unique source ports below 1024 all have their own column with a prefix, "special_source_port_" followed by the port number. All other source ports above 1024 are grouped into a separate column. For the destination ports, we also had two different grouping. Normally, a destination port will be port number 80. Port 80 is used for web servers and "listens" to a client. Since, there is a high chance that port 80 would be used frequently for destination ports we created a column for rows that used port 80. We then classified each destination port that is not 80 as a special destination port. We created a separate column of those ports with prefix, "special_destination_port_" followed by the port number.

In addition,we investigated ports that usually served a program or signaled a type of danger. For example, port 3389 corresponds to the TCP and UDP ports that are used for Microsoft's remote desktop, software that you would not necessarily expect in a non-tech company. In addition, we looked at strategic ports used by hackers or malicious software to have a wider range of ports that could signal an attack. In the table below is a list of the ports that is included in our features.

Port	Description/Threat
3389	Remote Desktop Protocol (RDP) - Ports that allows a user a graphical interface to connect to another computer for Windows.
6783-6785	Remote Desktop Protocol (RDP) SplashTop
53	DNS Exit Strategy and port used to create DDOS attacks.
4444	Trojan to listen in on information
31337	Back Orifice backdoor and some other malicious software programs
6660 - 6669	Internet Relay Chart (IRC) Vulnerabilities & DDoS
12345	Netbus Trojan Horse
514	Explotable Shell

Table 4.4: *Examined ports for feature generation. These ports have been exploited in years past and could indicate a threat [1], [2], [66].*

It was important to tag these ports as they could be critical to alerting a system administrators of possible malicious behavior.

4.8 Averages

An interesting detail about network traffic, is that it is comprised of several layers to make itself efficient and secure. Though normally one would examine packages sent and received to characterize a network flow, that information is not always readily available. The VAST Dataset did not include this; therefore, we needed to emulate how a network would work with the data that we had available.

In order to develop a feature that encompassed anomalous numbers of requests, we had to develop a feature that could encapsulate what a "regular" amount of requests were, and what we considered to be out of that range. We considered this for source IP addresses, destination IP addresses, source ports, and destination ports. We will refer to these as network parameters in this section. To do this, we calculated the average of the network parameter requests per

second. This would be able to detect if a parameter was used more or less than normal during a certain time period. Doing so is critical to detecting attacks such as a DDoS attack. Each second has a certain number of requests per parameter. For example, if you have an IP address as the parameter, 192.0.0.1, and it appears three times in a period of one second, that is three requests per second. Below are the steps that our Python script completes to calculate this feature.

1. Individually examine each second in the dataset as a slice
2. Retrieve the requests per second for each network parameter
3. Calculate average based on the network parameter
4. For each second for a network parameter divide it by the average requests per second

For each of the parameters mentioned above, a mySQL table is defined with the schema depicted in table 4.5.

Field	Type	Null	Key
date_time	varchar(34)	No	Primary
[parameter]	varchar(34)	No	Primary
Appearance per Second	varchar(34)	Yes	
Final_Feature	varchar(34)	Yes	
Final_Work_Feature	varchar(34)	Yes	

Table 4.5: *Average feature set table schema. This feature attempts to emulate the amount of requests each network parameter made.*

Since the table would be nearly identical, except the network parameter, we created a template for the table creation. Similarly to the working set mySQL table, explained in chapter 4 section 5, these tables are defined each time that the feature manager is ran with a different time slice. It is important to note that these slices are not representative of the global dataset, but rather, a just the slice a user chooses. There were a few reasons why we decided to keep these features local rather than global. The last average feature's purpose is to capture a significant change in the network. If we used the global data, a DDoS attack, for example, could skew the data since it will completely change the total work done by a system. Examining the slices locally can imitate a real world system by analyzing a fixed number of logs, and as they come in the last average feature theoretically would be able to capture the change. Finally, there was also a memory component in the generation of the features. If we decided to use global data, in a real time system it would need to be periodically updated when new logs come in, and it would increasingly need more memory. By maintaining the logs locally, we reduce the amount of memory needed and have much more efficient process to creating the features.

In addition, a second feature was generated that looked at the amount of work that a specific address or port generated of the total amount of work done in the system during that time. Although the process is similar to retrieve the current work for the address/port, the difference is that instead of dividing by the average work done, we divide by the total. What this accomplishes is that it contrasts what we consider an above average address/port for the system with what the total work on the system is done. Table 4.6 shows how our features were represented.

Source IP LastAverage	Destination IP LastAverage	Source Port LastAverage	Destination Port LastAverage
1.00	1.28	1.00	0.20
1.83	4.16	1.00	1.00
2.45	2.83	1.00	2.28

Table 4.6: *Average feature example.*

4.9 Vulnerability Scan Analysis

The vulnerability scan used was generated from a Nessus scan [24]. Nessus is a vulnerability scanning platform, used mainly by security analysts. The Nessus vulnerability scan offers information regarding a system's health and if there are areas that malicious hackers could utilize to cause harm. A parser [35] was used on the Nessus scan provided in the VAST dataset to extract information relevant to our feature generation. Among these relevancies are the note, hole, and Common Vulnerability Scoring System (cvss) parameters. Notes represent a security warning and are indicators of behavior that is slightly out of the norm. A hole is representative of a larger security warning, often critical to system health [24]. The cvss value ranges from 0 to 10 and provides further insight into the severity of these health warning indicators. Below is an example of the vulnerability scan used before parsing.

Count Note	Count Hole	Max Severity Note	Max Severity Hole
1	0	1	0
1	0	1	0
51	209	2.6	9.3

Table 4.7: *We use the Nessus scan [24] to validate features. Here we examine the amount of notes and holes per IP address.*

To efficiently query IP addresses that could potentially be the root of a critical vulnerability, the parsed values were stored in a mySQL table. Table 4.8 shows the schema of this table in mySQL:

Field	Type	Null	Key
id	int(11)	No	Primary
ip	varchar(15)	Yes	
count_note	varchar(4)	Yes	
count_hole	varchar(4)	Yes	
max_severity_note	varchar(15)	Yes	
max_severity_hole	varchar(15)	Yes	

Table 4.8: *Nessus Scan table schema. We used this table to join firewall logs and vulnerability scan information.*

Once the table is populated with these values it is joined with the mySQL features table. The join occurs according to the IP address. The result is the features table with the addition of these four vulnerability scan features. With these features it is simple to see exactly which IP addresses are causing problems and will allow us to validate the accuracy of the formerly mentioned features.

Features themselves focus on describing the data set rather than finding anomalies [31]. To find anomalies using features, we depend on learning algorithms, which are used as a part of machine learning to process information and determine patterns for a given dataset [3]. In the next chapter, we will discuss different learning algorithm techniques and how they can affect anomaly detection [3].

Chapter 5

Learning Algorithms

Once feature generation was completed, the data could be exported for mathematical analysis. The extracted information was analyzed through statistical methods in an attempt to analytically detect anomalies in the dataset. Various parameters were tested and produced results that accurately detected anomalies when comparing the mathematical results to the ground truth network data.

5.1 Supervised and Unsupervised Learning

There are many statistical methods that could be used as we are looking for relationships between characteristics of network logs to find discrepancies and anomalies. Many statistical methods were evaluated to help determine what would fit best for the network data being analyzed, including different types of learning algorithms. Supervised learning includes statistical approaches where possible results are already known, and the data being analyzed is labeled with correct answers [28]. On the contrary, unsupervised learning algorithms can analyze data to find patterns and relationships that might not have been known or examined before [27]. Principal Component Analysis (PCA) is an unsupervised statistical approach used for data dimension reduction and processing [72]. This is done by taking a data set of observations and converting them to a set of linearly uncorrelated principal components for further analysis. By using a Singular Value Decomposition (SVD), a given matrix M $\in \mathbb{R}^{m \times n}$ can be broken down as follows: [23]

$$M = U\Sigma V^T$$

where U $\in \mathbb{R}^{m \times n}$ is a unitary matrix, $\Sigma \in \mathbb{R}^{n \times n}$ which is a positive definite diagonal matrix with zeros elsewhere, and V $\in \mathbb{R}^{m \times n}$, also a unitary matrix [23]. The diagonal entries $\sigma_i \in \Sigma$ are known

as the singular values of M, are similar but not always the same as eigenvalues. Singular values are only equivalent to eigenvalues when the given matrix is real, symmetric, square, positive definite matrix. As stated, PCA can be used for dimension reduction and to project matrices onto a lower dimensional subspace, which would work well for finding anomalies within network data. PCA unfortunately suffers from each principal component being a linear combination of all original observations, making it difficult to interpret results and susceptible to outliers.

5.2 Singular Values

For the first subset of network data that was analyzed after preprocessing what was done, the resulting matrix of network data logs was $\mathrm{M} \in \mathbb{R}^{5034\times 34}$. There were initially four logs in the matrix that returned undefined for the IP addresses, which resulted in these rows being deleted from the matrix as to not interfere with the analysis. Then, for an SVD to be produced for this matrix, the first four columns of the matrix had to be ignored. These columns were id, destination IP, source IP, and time log, and were not of values that could be computed for the new matrices. This resulted in the matrix being trimmed down to a size of 5034 by 34, at which point the matrix was factorized as an SVD producing the results below.

The Σ matrix contained 30 unique singular values (σ_i) [19], corresponding to the 30 columns in our adjusted matrix . These σ_i values declined in weight as they were expected to, and only the first 13 σ_i values returned coefficients above zero. This means that with the largest 13 singular values, we have the ability to predict the other 17 singular values for any log we're given, allowing us to predict the remaining information and details of an individual log if we know less than half of it's information. The 30 σ_i values can also be seen below in descending order. After the results were produced, more analysis was done on the logs and columns of the network data that were being produced. The matrices of network data were of size 1000000 by 34 for the first 12 csv files that were exported, and the 13th csv file contained slightly fewer logs of data, meaning there were slightly less than 13 million logs of data to iterate through. The 13 unique network data csv files were read and concatenated into a singular data matrix, with the first four columns being ignored, which was then z-Transformed and processed into an SVD the same way our subset sample of data had been. The z-Transformation process is described later in section 5.4 of this paper.

Using the first two singular values of the Σ matrix, every log in the concatenated csv files were plotted by using the first two singular values of the Σ matrix as the x and y axis. The columns of the csv files were also plotted by the same convention, using the first two singular values of the Σ as the axis for plotting. The first two singular values were selected above the rest because the results of the Σ matrix are returned in descending order, so the first two singular values represent the two values that scale the U and V^T unitary matrices of the SVD to recreate the original matrix.

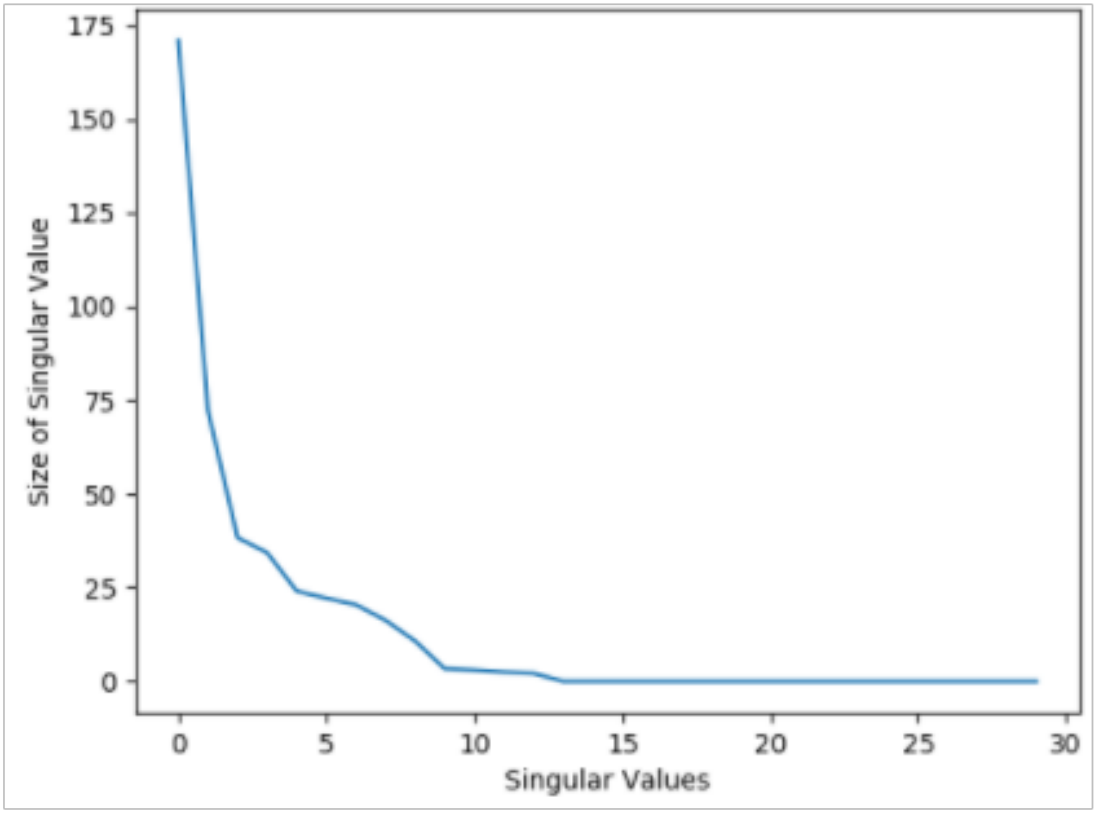


Figure 5.1: *Visual representation of the singular values of the first time slice of data. Singular values appear in descending order due to the construction of the SVD and Σ [19].*

The first two singular values were then multiplied by the U unitary matrix, returning a vector of slightly less than 12 million rows by 2 columns in size. This was also done to the V^T unitary matrix, of which the results were transposed to be in the same form as the other product. The columns of the two products of U and V^T respectively were then separated into arrays, the values were sent to lists, and the lists were then zipped together to create the data points that would be plotted. Once the values for the U and V^T by the first two singular values plots were generated, they were produced with the results below.

The singular values by U graph displayed all logs within the concatenated matrix of all 13 network data csv files, which displays a somewhat linear relationship between the data, with individual data points falling above and below a general trend line. The singular values by V^T graph displayed the columns of the concatenated matrix of all 13 network csv files, with each point in the plot representing a different column. These columns were expected to be less linearly dependent on one another than the log data, because the columns of the matrices represent the features of each individual data point, where there was clearly disparity between many of the logs and respective features.

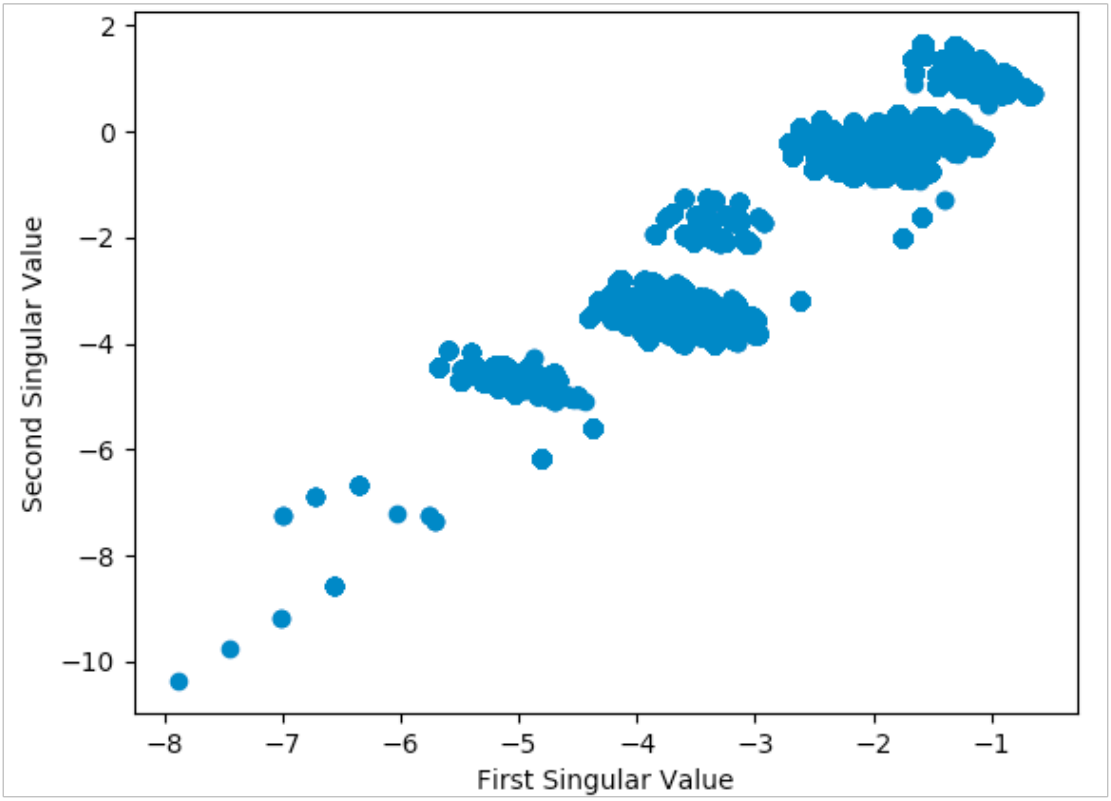


Figure 5.2: *Singular Values by U. This displays a loose linear relationship between all data points. The first two singular values were chosen to be the axes because those are the two most dominant values to help predict other features of an individual log [19].*

5.3 Robust Principal Component Analysis

Robust Principal Component Analysis (RPCA) is an adjusted statistical approach of PCA which works with corrupted observations and outliers [25]. While PCA is susceptible to outliers as previously stated, RPCA can detect a more accurate low dimensional space to be recovered, and that is why RPCA is necessary over standard PCA for anomaly detection in the network data. RPCA works to recover a low-rank matrix L and a sparse matrix S from corrupted measurements, which in the case of network data would be the anomalies of the data set. Robust PCA works in conjunction with the Singular Value Decomposition (SVD) factorization method of separating matrices into distinct unitary and diagonal matrices, giving an optimization problem as follows: [25]

$$min(L,S)||L||_* + \lambda||S||_1$$

$$\text{subject to, } |M-(L+S)| \leq \varepsilon$$

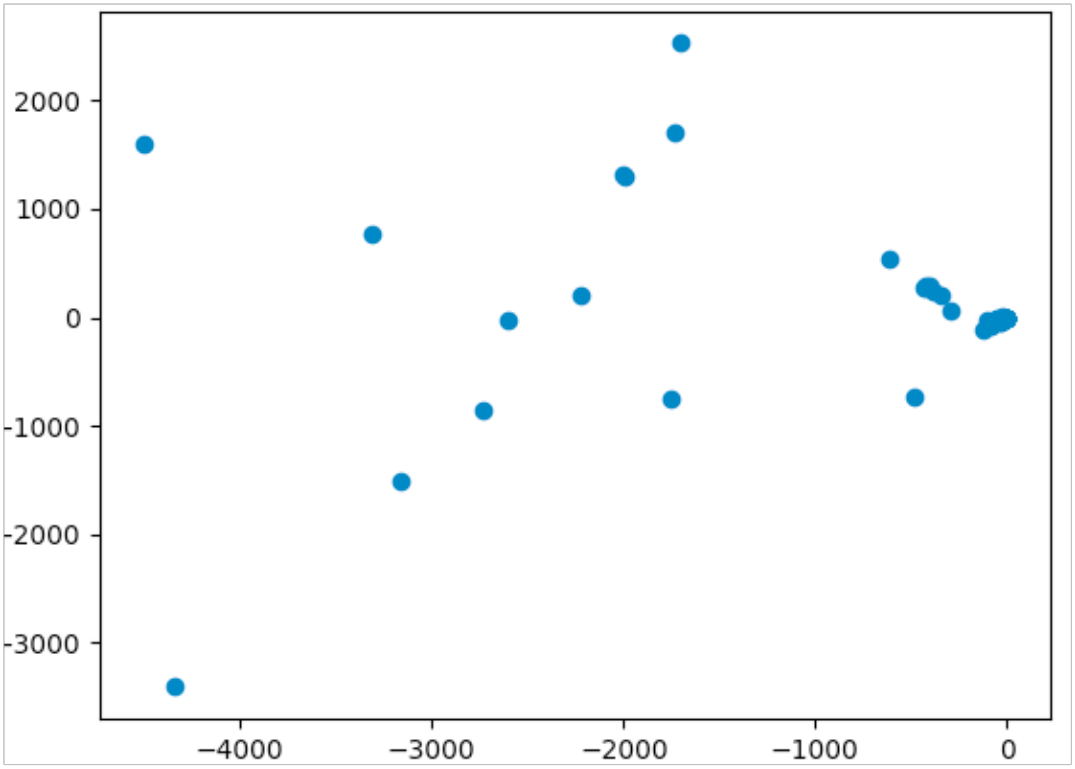


Figure 5.3: *Singular Values by V^T. The spareness of the plot shows how there is no apparent linear relationship between the columns or features of the dataset [19]. This is logical because features are linearly independent of each other. For example, IP addresses and ports do not depend on each other.*

In this equation, L is the low rank matrix which can be factorized as an SVD, $||L||_*$ is the nuclear norm of L (the sum of the singular values), λ is the coupling constant between L and S, $||S||_1$ is the sum of the entries S, and ε is the matrix of point-wise error constants that improve the noise generated from real world data. Due to the nature of the network data, traditional PCA would be too receptive to outliers and would prove to be ineffective and anomaly detection. As a result, RPCA was the statistical method chosen for anomaly detection.

5.4 Z-Transform

Due to having various different types of information that were created through the feature generation process, data normalization was necessary. Between the source and destination ports, source and destination IP addresses, and averages that were calculated for when certain IP addresses and ports came up throughout the dataset, a z-Transformation of the data was

necessary as a preprocessing method before further analysis could be done. This was in each column of the matrices that were generated, and was done by using the z-score statistical method. [46]

$$z = \frac{(x - \mu)}{\sigma}$$

For each column in the data set, the mean μ and standard deviations σ of a given column were calculated. If the σ of a given column was not zero, each individual data point in the column was taken, had the μ of the column subtracted from the given point, and that sum was then divided by the σ of the total column. If the σ of a given column was zero, this would have resulted in division by zero for all points in the column and would have produced NaNs (not a number), denoting infinities and numbers that could not be read for further analysis. This was done for all columns of data in the matrices that were generated, and these new z-Transformed matrices were exported as new pandas dataframes for future use. After the z-Transformation was completed, the new data matrix was evaluated as a numpy SVD (Singular Value Decomposition) to generate the singular values of the z-Transformed data of the original given matrix.

5.5 λ

One of the most crucial variables to RPCA is λ, the coupling constant between the low dimension L and S matrices, where L and S sum together to create the original matrix used for analysis [25]. The L matrix in this case is an SVD of the non-anomalous data, and the S matrix is a sparse matrix that represents the data points from the original matrix that are found by RPCA to be anomalous. λ is the variable that differentiates what is considered anomalous, what is not, and controls what is allowed into the S matrix. If λ is small, more data points will be allowed to move from the L matrix to the S matrix, and as λ increases fewer data points will be considered anomalous. The theoretical value of λ is equal to the following:
[42]

$$\lambda = \frac{1}{\sqrt{max[m,n]}}$$

where m and n represent the rows and columns of the original matrix. For this project, λ was altered from the default value in an attempt to improve the capabilities of RPCA in finding anomalous entries within network data. This was done by changing the numerator value of one to a integer variable, and then changing the variable many times to see the differences between the L and S matrices as lambda was changed. This was done on a chosen small slice of data from the network data set and was analyzed as a 5000 by 30 matrix. This equation can be seen below, where H is the integer variable in change λ:
[42]

$$\lambda = \frac{H}{\sqrt{max[m,n]}}$$

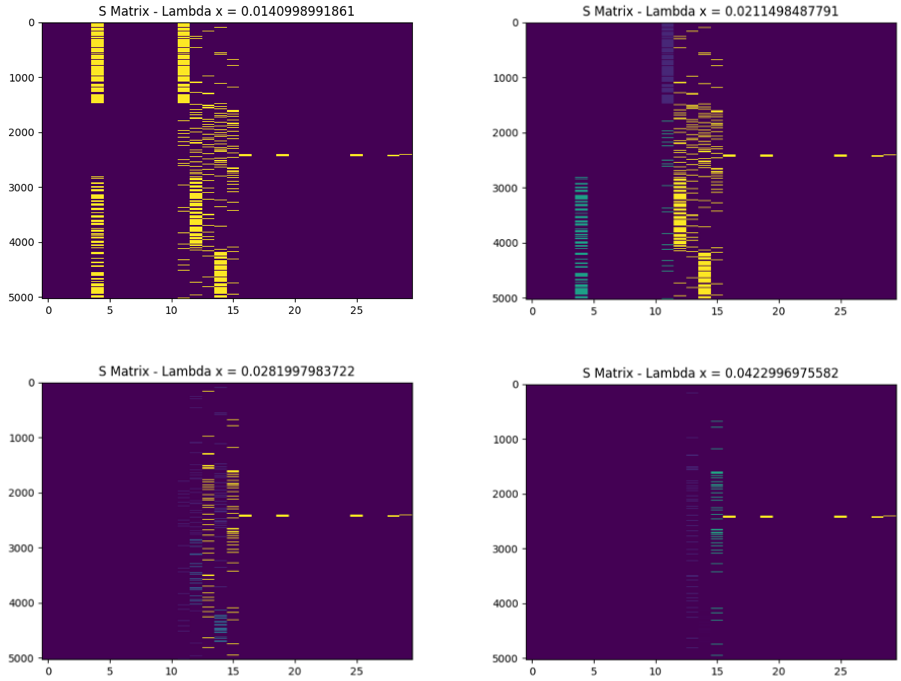


Figure 5.4: *imShow visualizations. These four plots depict sparse S (anomaly) matrices that change as the value of λ increases. This shows how the coupling constant alters which data points are classified as anomaly and which are normal. As λ is increased the sparse matrices lose entries in their matrices, thus the plots appear to have less data [42].*

By using the matplotlib [26] imShow function, the following plots were produced to provide visuals of how the entries in S were changing as λ was altered. A few examples of the imShow plots can be seen below, where it can be seen that fewer entries appear in the S Matrix imShow plots as the value of λ is increased.

5.6 μ and ρ

Two other variables used in RPCA are mu μ and rho ρ, which are used in the augmented Lagrangian parameter that are used for convergence testing. The theoretical μ and ρ are taken from *The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices* [41] and are used in conjunction with given epsilon ε values to act as the stopping criterion to help determine whether the RPCA code is converging based on the matrix it was given. The default μ and ρ are respectively:
[41]

$$\mu = \frac{1}{||D||_2}$$

$$\rho = 1.2172 + 1.858\rho_s$$

$$\text{where, } \rho_s = \frac{|\Omega|}{mn}$$

Where $||D||_2$ represents the spectral norm (largest singular value) of the given matrix D, and ρ_s acts as the sampling density between ρ and ρ_s. For this project, μ and ρ were altered from their default value in an attempt to improve the capabilities of RPCA and improve the convergence rate of given matrices being tested. Two time slices were chosen from the network data of size 5000 by 30 and 50000 by 526 respectively after features generation. The tables below show how the convergence rate of a given matrix was changed after μ and ρ were changed, using the theoretical λ value for all tests run. As seen in Figure 5.5 and 5.6, the μ and ρ values of 0.05 and 1.05 respectively greatly improved the number of iterations needed for RPCA to meet it's given stopping criterion. These values were deemed to be an improvement over the theoretical values [41] as a result.

5000 x 30	Rho	1.001	1.01	1.05	1.1	1.15
Mu	Iterations to convergence					
0.01		1000+	270	110	1000+	1000+
0.05		458	194	88	98	1000+
0.075		393	191	1000+	1000+	1000+
0.1		332	177	1000+	1000+	1000+
0.15		250	150	1000+	1000+	1000+

Figure 5.5: *Initial μ and ρ testing*

50000 x 526	Rho	1.001	1.025	1.05
Mu	Iterations to convergence			
0.01		1000+	310	134
0.05		498	247	102
0.075		427	239	1000+
0.1		364	224	1000+
0.15		280	182	1000+

Figure 5.6: *μ and ρ testing after feature generation*

5.7 Implementing RPCA

After tests were done on the first subset of data, a larger subset of data was chosen to test on. The second subset of data contained network logs from the DDoS attack, and after ignoring the non-discrete columns of the matrix the size was 50000 by 526. This matrix was z-Transformed and factorized as an SVD, and its singular values were calculated accordingly [25]. However, this matrix was also run through RPCA, which produced different singular values due to the matrix being projected to a lower dimensional space. The matrix was decomposed into the form $M = L + S$, where $L = U\Sigma V^T$ is factorized as an SVD and S represents the sparse anomaly matrix of the given matrix M [25]. The plots of the original singular values, the RPCA singular values, and the imshow plot of the sparse anomaly S matrix, which depicts what cells are regarded as anomalous, are shown in Figures 5.7. 5.8, and 5.9.

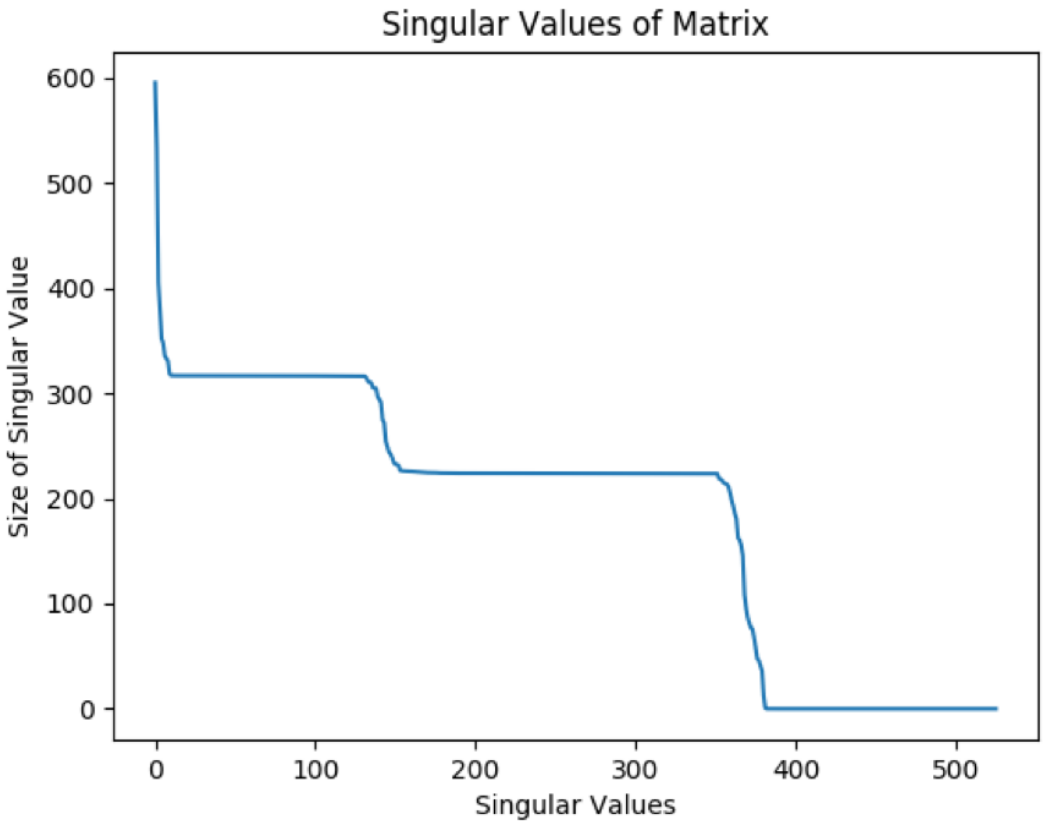


Figure 5.7: *Singular values of new time slice [19].*

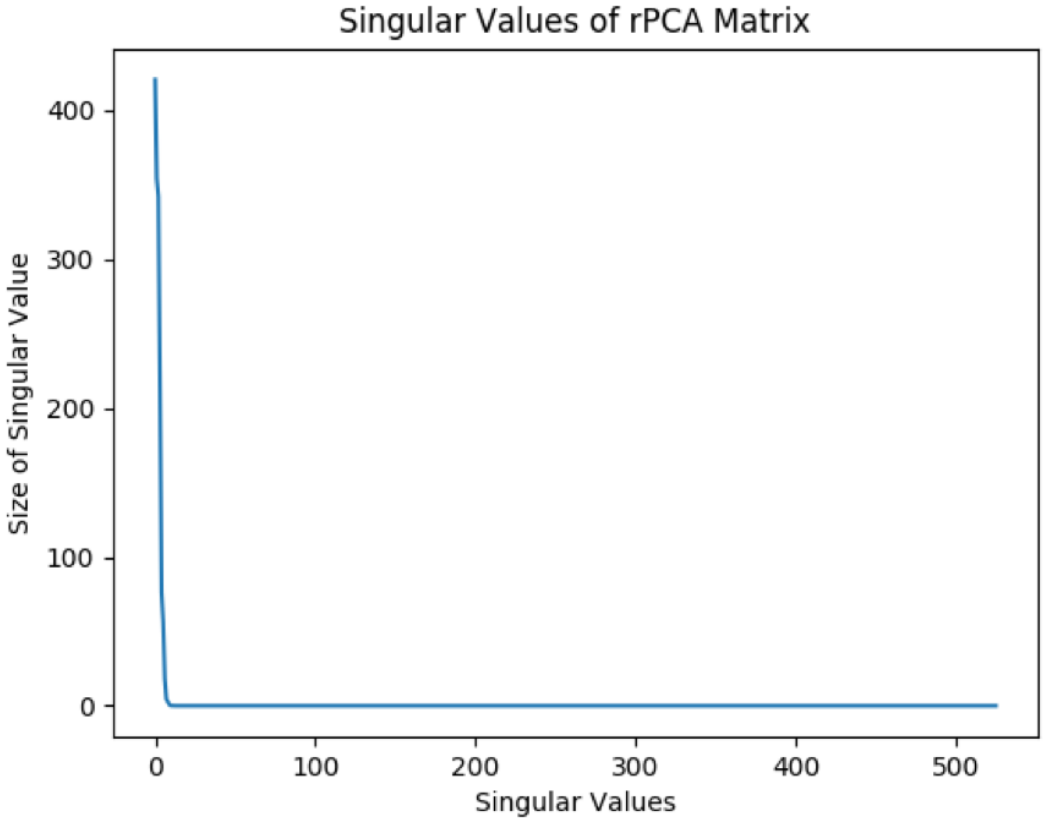


Figure 5.8: *Singular values of S matrix from RPCA [19]. This plot has a steep downward trend which is due to the S matrix being sparse and therefore having few entries greater than 0. The result of this is a matrix that has very few dominant singular values which influences the data points in the anomaly matrix [25].*

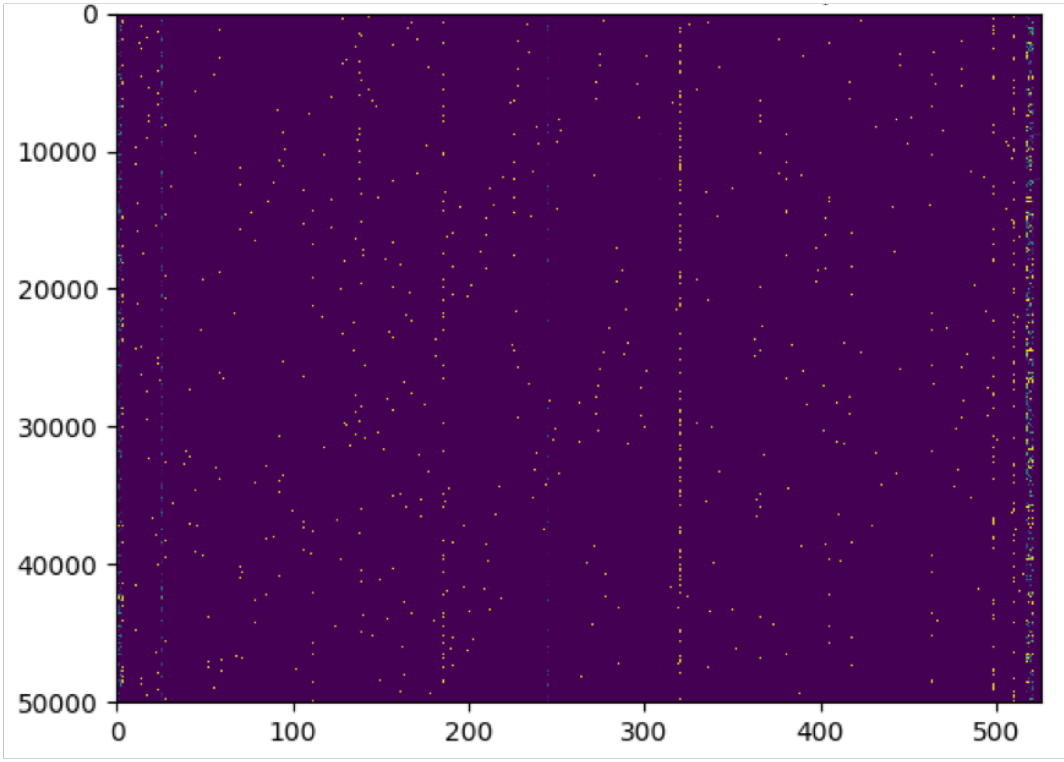


Figure 5.9: *S matrix imShow plot after RPCA. Visualize representation of anomaly S matrix. The spareness of this results in few data points being represented in this plot [25].*

After the manager, described in section 4.5 (Feature Infrastructure) was fully set up, mathematical analysis tests became much easier to run for different feature sets and time slices from the network data, as well as λ, μ, and ρ value tests for RPCA. This manager allowed editing capabilities in the terminal using the nano command for parameters in RPCA, changing the range of network log data to be analyzed, and streamlined the process for joining the results of an anomalous S matrix with the ground truth of the network data. However, before the results of these S matrices could be compared with the ground truth information, the entries of the S matrix needed to be reverse z-Transformed to de-normalize the data back to the original relative scales it entered with. This process was the reverse of the normalization method of the z-score in Section 5.4 (z-Transform), where all columns of the S matrix underwent the following:
[4]

$$x = z\sigma + \mu$$

where z is the z-Transformed value of each cell in the column, and σ and μ respectively represent the standard deviation and mean of the given column. Similar to the forward z-Transform

performed earlier, if the σ of a column was 0, the column was skipped over in the reverse z-Transform process since it would not have been normalized in the first place. The data in these columns would be of the same relevant scale as it was when it entered the iterative process. If the value of σ was not zero, the entries in that column went through the above process and returned to their original relative scale to be compared with the ground truth network data. After the reverse z-Transformation was completed for every column in the S matrix, the new reverse anomaly matrix was joined with the ground truth network data for cross validation purposes.

5.8 λ Tests

Once preprocessing had been done to properly reverse z-Transform our S matrices and cross validated them with the ground truth network data, tests were run on a select number of time slices for the features that had been generated. For the time slices described above in chapter 2, ten different λ values were tested, and the number of iterations to convergence was record for each test run. For each test, the non reverse z-Transformed S matrix, the reverse z-Transformed S matrix, and the joined S matrix were all exported as csv files. The table below depicts the results of the tests run on these four time slices:
[42]

$$\lambda = \frac{H}{\sqrt{max[m,n]}}$$

	H	Time Slice 1	Time Slice 2	Time Slice 3	Time Slice 4
Lambda 1	1	1000+	28	20	1000+
Lambda 2	1.2	1000+	29	28	1000+
Lambda 3	1.4	108	29	34	1000+
Lambda 4	1.6	1000+	26	83	1000+
Lambda 5	1.8	1000+	24	49	1000+
Lambda 6	2	1000+	25	37	1000+
Lambda 7	2.2	1000+	26	352	1000+
Lambda 8	2.4	1000+	27	183	1000+
Lambda 9	2.6	1000+	26	89	1000+
Lambda 10	2.8	1000+	25	38	1000+

Table 5.1: *Initial λ Testing*

As described in Section 4.1 (λ), λ acts as the coupling constant between returned the low dimensional L matrix and sparse anomalous S matrix from RPCA, where the two matrices

combine to create the original matrix M. In all of these tests, the default values for μ and ρ from Lin-Chen-Ma [41] were used, as these tests attempted to examine the parameter λ and its relation to anomaly detection. As the integer value H was increased, the value of λ did not linearly impact the number of iterations required for convergence of RPCA, but the entries of S changed between each λ tested.

5.9 Training Set

These 40 tests were done to experiment with viability of the manager for future tests with different times slices. After the tests were completed, two additional sets of features were generated as highlighted in chapter 6, and three time slices were chosen as the training set of the network data. The feature sets are denoted originalFeatures, sipdipIndividualColumns, and sipDipIn. Similar to the original 40 tests that were run, the default values for μ and ρ from Lin-Chen-Ma [41] were used, as these tests attempted to examine the parameter λ and its relation to anomaly detection. For each test, the non reverse z-Transformed S Matrix, the reverse z-Transformed S Matrix, and the joined S Matrix were all exported as csv files, and the resulting tables show the results of the 90 tests run [42]:

Original	H	Time Slice 1	Time Slice 2	Time Slice 3
Lambda 1	1	1000+	1000+	1000+
Lambda 2	1.2	1000+	1000+	1000+
Lambda 3	1.4	108	1000+	1000+
Lambda 4	1.6	1000+	1000+	1000+
Lambda 5	1.8	1000+	25	1000+
Lambda 6	2	1000+	25	1000+
Lambda 7	2.2	1000+	1000+	1000+
Lambda 8	2.4	1000+	36	228
Lambda 9	2.6	1000+	47	1000+
Lambda 10	2.8	1000+	53	133

Table 5.2: *Lambda Testing with Original feature set*

SipDipIndividual	H	Time Slice 1	Time Slice 2	Time Slice 3
Lambda 1	1	1000+	1000+	1000+
Lambda 2	1.2	1000+	1000+	1000+

Lambda 3	1.4	188	1000+	1000+
Lambda 4	1.6	1000+	1000+	1000+
Lambda 5	1.8	1000+	1000+	1000+
Lambda 6	2	1000+	1000+	1000+
Lambda 7	2.2	1000+	33	1000+
Lambda 8	2.4	1000+	1000+	1000+
Lambda 9	2.6	1000+	1000+	1000+
Lambda 10	2.8	1000+	1000+	1000+

Table 5.3: *Lambda Testing with AFC individual columns feature set*

SipDipIn	H	Time Slice 1	Time Slice 2	Time Slice 3
Lambda 1	1	1000+	1000+	1000+
Lambda 2	1.2	1000+	1000+	1000+
Lambda 3	1.4	108	1000+	1000+
Lambda 4	1.6	1000+	1000+	1000+
Lambda 5	1.8	1000+	1000+	1000+
Lambda 6	2	1000+	1000+	1000+
Lambda 7	2.2	1000+	1000+	1000+
Lambda 8	2.4	1000+	1000+	1000+
Lambda 9	2.6	1000+	1000+	1000+
Lambda 10	2.8	193	1000+	1000+

Table 5.4: *Lambda Testing with inside and outside feature set*

For each joined S Matrix that was exported throughout this process that was compared with ground truth network data for cross validation, confusion matrices were produced to compare the True Positive, False Positive, True Negative, and False Negative results of our anomaly detection, where positive and negative results represented anomalous and non-anomalous data respectively. The results of these confusion matrices can be seen in chapter 6, where the increase in the value of λ coincided with better anomaly detection results for finding True Negatives and eliminating False Positives. In the next chapter, we focus on interpreting these results to help data analysts prevent attacks.

Chapter 6

Results

6.1 μ and ρ

As discussed in chapter 5 section 6 (μ and ρ), the default values for the RPCA parameters μ and ρ, defined in Lin-Chen-Ma, were altered in this project in an attempt to improve the rate of convergence for anomaly detection. The tables in chapter 5 show how RPCA fared when μ and ρ were set to their default values, and as a means of comparing the default values with those calculated in chapter 5 section 6, the same 90 tests were run on the same feature sets and time slices described in chapter 5. The values of μ and ρ were set to 0.05 and 1.05, respectively, and the results of the new μ and ρ values are shown in the tables below:

SipDipIndividual	H	Time Slice 1	Time Slice 2	Time Slice 3
Lambda 1	1	141+	81+	1000+
Lambda 2	1.2	1000+	85	1000+
Lambda 3	1.4	143	150	1000+
Lambda 4	1.6	148+	285	1000+
Lambda 5	1.8	1000+	1000+	1000+
Lambda 6	2	1000+	1000+	1000+
Lambda 7	2.2	304+	329	1000+
Lambda 8	2.4	1000+	1000+	1000+
Lambda 9	2.6	1000+	129	1000+
Lambda 10	2.8	1000+	1000+	1000+

Table 6.1: *μ and ρ testing on AFC's individual column IP address feature*

SipDipIn	H	Time Slice 1	Time Slice 2	Time Slice 3
Lambda 1	1	1000+	77	1000+
Lambda 2	1.2	1000+	1000+	1000+
Lambda 3	1.4	1000+	1000+	1000+
Lambda 4	1.6	1000+	1000+	1000+
Lambda 5	1.8	1000+	1000+	1000+
Lambda 6	2	1000+	1000+	1000+
Lambda 7	2.2	1000+	1000+	1000+
Lambda 8	2.4	1000+	1000+	1000+
Lambda 9	2.6	1000+	1000+	1000+
Lambda 10	2.8	454	1000+	1000+

Table 6.2: *μ and ρ testing on inside and outside features*

The changed μ and ρ values only slightly altered the number of iterations for the third feature set (sipDipIn) for all ten λ values that were tested. However, for the second feature set (sipdipIndividualColumns) there was noticeable improvement between the default μ and ρ values from Lin-Chen-Ma and the ones tested in this project, particularly in the DDoS and rdp time slices.

6.2 λ

The coupling constant ρ was prominent throughout this project, acting as the parameter that allowed experimentation with different S matrices to improve the anomaly detection capabilities of RPCA. As stated in chapter 5, the tests that were run on our three Feature Sets and three Time Slices, using ten λ values per combination, produced 90 unique joined S matrices for us to compare to the ground truth network data. Confusion matrices, as explained in chapter 5 section 9, were produced to determine the rate of success for detecting anomalies within the network data, of which the best results can be seen below for each unique Time Slice.

6.3 Final Features

To help with the final selection of our features, we created a confusion matrix for each type of attack and specific time time slice for each of our IP addresses choices. These confusion matrices are produced using the S matrix from the output of RPCA. As explained in chapter 4, we chose to

create a feature for every distinct port, similar to the first iteration of our IP address feature set. An important note, is that after much deliberation, we realized that our destination port feature that listed each port below 1024 was causing several false positives, and we decided to omit it during the creation of these confusion matrices. Given more time we would have created more iterations of different port feature set in order to avoid omitting the S values in the resulting confusion matrices.

6.3.1 Detecting Denial of Service Attack

As explained in chapter 2, DDoS attacks occur when one machine attempts to disrupt services through means of enormous amounts of calls to a network. The VAST dataset challenge outlines this attack in the solution manual. The attack occurred on the first day of the dataset, April 13th, 2011 at 11:39:00 and ended at 12:51:00 that same day. The time slice chosen for this attack was from 11:33:47 to 11:39:57, which accounted for roughly one minute of the attack. This slice ensures that we have enough log data that is considered normal and enough data that is out of the ordinary and can be malicious. As with the other attack results, we ran the time slice and tested three different feature sets and produced confusion matrices for each run cycle. In order to help us solidify our feature choices.

Original Features		
	True	False
Positive	145	841
Negative	0	644

AFC Columns		
	True	False
Positive	145	841
Negative	0	644

Inside and Outside Ranges		
	True	False
Positive	145	226
Negative	615	644

Table 6.3: *Confusion Matrices of Original Features for DDoS*

From the tables above, we see that the first two feature sets functioned the same and returned the same results. The reasoning for this is that this time slice is relatively small and as a result of the nature of the two feature sets they created similar feature matrices. The first set has a distinct column for each IP address and the second has a column for each feature in the AFC network and two columns for any outside source or destination IP addresses. As stated before, this time slice is small relative to the data set and may only include one IP address that is outside of AFC's network and as a result, would create a nearly identical feature matrix.

The third feature set, however yields different results. With this set there were the same number of true positives but it also detects 615 true negatives. This eliminates several of the false positives that the other two feature sets included. There are still however, the same number of false negatives. This could be a result of the time slice being too small and not having enough rows to accurately determine what is a true negative and what is a false negative. Regardless, we see an increase in correct results with the third feature set. Therefore it is determined that the third feature set is more accurate in detecting denial of service attacks.

6.3.2 Detecting Socially Engineered Attack

A socially engineered attack occurs when humans curiosity are to enable malicious activity. In the solutions manual, a socially engineered attack that results in an unauthorized remote desktop protocol begins on April 14th, 2011 13:31. As stated in chapter 2, this attack comes from a computer that is outside of AFC's network and uses the common port 3389 in order to make the unauthorized connection that can be used to harm AFC's network. This port is among security concern ports we examined since in the past several companies have been victimized by this exploit. For a full table of ports examines in this feature and reasons why, refer to the ports section in chapter 4. Similar to before, we chose another time slice to have a proof of concept that our feature functions properly. Our time slice begins at April 14th, 2011 at 13:30:55 and ends at 13:32:00 a few minutes after. Again, below are the confusion matrices for our three main feature sets.

Original Features		
	True	False
Positive	2	5506
Negative	0	0

AFC Columns		
	True	False
Positive	2	300
Negative	5206	0

Inside and Outside Ranges		
	True	False
Positive	2	236
Negative	5270	0

Table 6.4: *Confusion Matrices of AFC individual column Features*

For detecting attacks that can happen among ports, our features were progressively more accurate. In the first feature set we detect the attack; however, there are an unacceptable amount of false positives. These false positives most likely are derived from the amount of IP addresses it detected as anomalous. The second feature set had much more true negatives rather than false positives. This feature set was much more accurate at determining what was an attack and what was considered in the realm of normal in the slice. Finally, the last set was the most accurate with finding attacks and reducing the amount of false positives. Again, the third feature set gives us more reasoning for choosing it into our final feature library.

Although false positives are not as critical as false negatives, a good anomaly detection system should reduce that amount of false positives wherever possible. Our third feature set does this, and with this time slice it achieved a 95.7% success rate at determining anomalous behavior.

6.3.3 Detecting Undocumented Computer Attack

As mentioned in the previous sections, we ran three different set of features for detecting the undocumented computer attack. The time slice chosen for this feature was from April 15th, 2011 14:05:00 to April 15th, 2011 14:08:00. The feature which detected the undocumented computer relied on knowledge of the system architecture that the company outlined in the VAST dataset description. This helped outline how critical it is to consider the security contracts a company has in place while detecting attacks. Violation of these rules are an indication that something in the system is off, or that an employee does not understand them clearly. In this particular time

slice, there were two features that highlighted the undocumented computer attack within our three features.

The Original and SIP and DIP individual columns features identified computers that breached the security contract. However, the column that identified each unique IP address highlighted every row in the S matrix as an anomaly. This signaled that this feature was the wrong data for RPCA since every point was considered an outlier. The source IP and destination IP feature that classified an IP address as inside or outside the contract significantly decreased the amount of false negatives, while still being able to find the attack.

Original Features		
	True	False
Positive	4	983
Negative	0	0

AFC Columns		
	True	False
Positive	4	983
Negative	0	0

Inside and Outside Ranges		
	True	False
Positive	4	2
Negative	981	0

Table 6.5: *Confusion matrices of inside and outside Features*

In summary, we believe that detecting an undocumented computer attack depends upon a security contract that must be established by the company. Overly describing the network caused issues because every IP address was unique enough to cause RPCA to evaluate them as an anomaly. Even though it accurately describes the data, it had adverse consequences – similarly to the effect of using a timestamp as a feature. This pattern shows promise towards future analysts as learning the specific domain of their company can be used to generate features.

6.4 Case Study: Reducing Number of False Positives

One of the dilemmas regarding our security concert port, was the sheer number of false positives that it could generate. How does an analyst distinguish between a port that can be used for a daily function to when that port is being exploited? What can characterize an attack differently than its proper use?

We investigated using AFC's Network on top of the security concern port to capture the idea that an exploited port would most likely come from outside the internal network. This helps us distinguish workers using a port for a task, such as transferring files, to an external user attempting to exploit it, such as steal information from the file system. With this in mind, we only considered a port as concerning if either the destination or source port is not in the expected network.

SIP and DIP Ranges		
	True	False
Positive	145	226
Negative	615	644

AFC and Port Join Feature		
	True	False
Positive	145	155
Negative	686	644

Table 6.6: *The table to the right shows a decrease in false positive due to considering ports that are used commonly but can be used maliciously*

As shown in the above tables, the number of false positives for this slice was reduced by 5%. In a bigger time slice, this could significantly impact the number of concerns that the security analyst needs to be aware of. This emphasizes that there is a need to have a structured security contract that can be applied to customize features that will describe the system.

CHAPTER 7

PROTOTYPE TO EXPLORE AND EVALUATE ANOMALY DETECTION

Visualizing any domain in an easily understood way is a challenge in itself, additional dimensions of data inherently makes this much more difficult. Fisseha Gidey and Charles Awono Onana argue in their paper on computer visualization, High Dimensional Data Visualization: Advances and Challenges that, "...the ever increasing dimensions of datasets, the physical limitations of the display screen (2D/3D), and the relatively small capacity of our mind to process complex data at a time pose a challenge in the process of visualization" [33].

There are various options for visualizing anomaly detection. The Nokia Group Research Center models their anomaly detection system using a self-organizing map which functioned as as a way to detect when a feature is acting abnormally while reducing the amount of data that needs to be monitored [36]. Another example includes generalizing the anomaly detection system as a time-series monitoring system. This allows an analyst to notice alarming data points in real time and functions for all different types of domains [65]. Some visualizations include utilizing machine learning techniques to classify and cluster anomalies in their visualizations [20].

In this MQP, we had the challenge of visualizing high dimensional data while perceiving and detecting anomalies. For this reason, we found it best to experiment with several different visualization technologies and techniques. Among the technologies used are d3 [15], and MatPlotLib's Singular Value Decomposition (SVD) graphs [26].

7.1 Requirements for Visualization

To develop our visualization, we chose requirements that would encompass a flexible system to run different analyses on the dataset, compute different features dynamically, and validate the different anomalies found in the VAST dataset challenge. Our requirements are as follows:

1. Viewing the Regular Log Data
    a) Upload the original firewall logs data with server-side pagination in order for an analyst to observe what is happening in the system.
    b) Allow the analyst themselves to use their own knowledge of networks to be able to observe if any logs appear curious or out of the normal.
2. Allow Flexibility in Choosing Time Slices to Analyze and Run RPCA Dynamically
    a) Allowing a user to select which time-slices to analyze by providing a way to select which logs to examine given a specific date and time.
    b) Computes features dynamically to allow for less memory storage being used and incorporate the ability to compute new features if more data is added.
3. Ability to Observe Ground Truth
    a) Since the answers for this particular challenge is available, the ability to cross-validate the answers in the front-end will allow to receive live-feedback on both our feature generation and the mathematical analysis being ran.
    b) Ensure that your features tell a story within your modeling of data.
4. Ability to Quickly Detect Abnormal Behavior.
    a) Alert a user if a log is anomalous, it is critical to include a way to warn the user that something requires their immediate attention.
    b) This feature will also reduce the amount of analysis an analyst needs to do, as they can focus their primary attention to those values that we highlight as potential threats.
5. Exploring More Information About RPCA Values
    a) Provide the full-picture of the mathematical analysis by providing a way to recover more detailed information on the results from RPCA.
    b) It is imperative to balance showing enough information for an analyst to comfortably observe data and allowing further exploration without it becoming overwhelming.

Finalizing these requirements lead us to think critically about the different visualization tools available to create this product. In our implementation section, we consider the different aspects of libraries, user interactivity, and how to frame our web application in an effective and user-friendly manner.

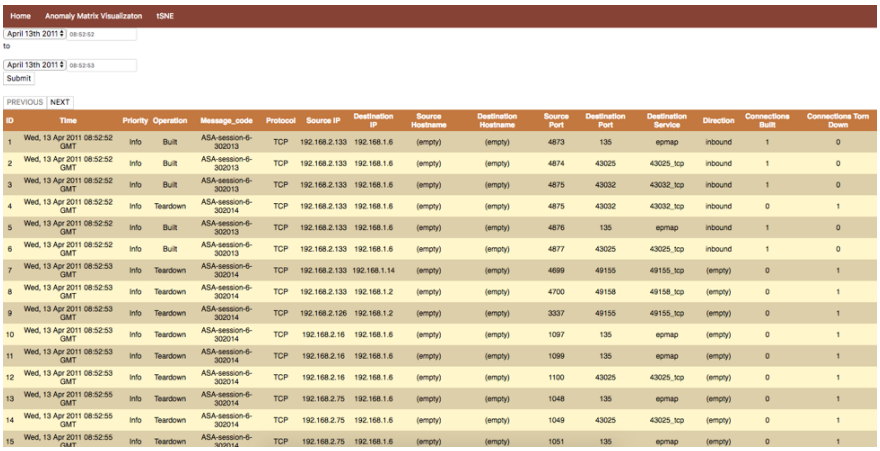
7.2 Overview of Web Application and Implementation

Our visualization system is comprised of a simple interface where users can choose a time slice of the VAST dataset to run anomaly detection. The firewall logs are displayed for a user to examine any raw logs before choosing a slice. Following the selection, the application displays an

anomaly matrix visualization, and a tSNE component which depicts our features with and without RPCA. In this section, we focus on the primary function of the web application – dynamically running features on time-slices and the anomaly matrix visualization and its implementation.

7.2.1 Web Application Components

1. Home Page

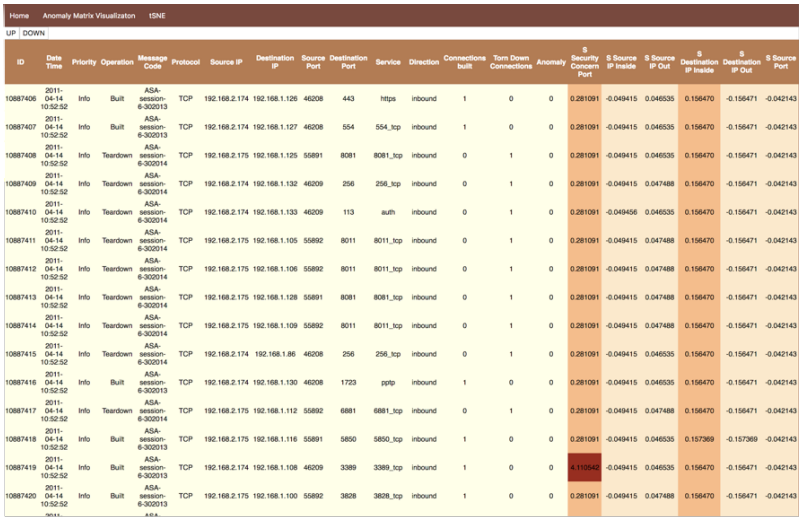


ID	Time	Priority	Operation	Message_code	Protocol	Source IP	Destination IP	Source Hostname	Destination Hostname	Source Port	Destination Port	Destination Service	Direction	Connections Built	Connections Torn Down
1	Wed, 13 Apr 2011 08:52:52 GMT	Info	Built	ASA-session-6-302013	TCP	192.168.2.133	192.168.1.6	(empty)	(empty)	4873	135	epmap	inbound	1	0
2	Wed, 13 Apr 2011 08:52:52 GMT	Info	Built	ASA-session-6-302013	TCP	192.168.2.133	192.168.1.6	(empty)	(empty)	4874	43025	43025_tcp	inbound	1	0
3	Wed, 13 Apr 2011 08:52:52 GMT	Info	Built	ASA-session-6-302013	TCP	192.168.2.133	192.168.1.6	(empty)	(empty)	4875	43032	43032_tcp	inbound	1	0
4	Wed, 13 Apr 2011 08:52:52 GMT	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.133	192.168.1.6	(empty)	(empty)	4875	43032	43032_tcp	inbound	0	1
5	Wed, 13 Apr 2011 08:52:52 GMT	Info	Built	ASA-session-6-302013	TCP	192.168.2.133	192.168.1.6	(empty)	(empty)	4876	135	epmap	inbound	1	0
6	Wed, 13 Apr 2011 08:52:52 GMT	Info	Built	ASA-session-6-302013	TCP	192.168.2.133	192.168.1.6	(empty)	(empty)	4877	43025	43025_tcp	inbound	1	0
7	Wed, 13 Apr 2011 08:52:53 GMT	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.133	192.168.1.14	(empty)	(empty)	4699	49155	49155_tcp	(empty)	0	1
8	Wed, 13 Apr 2011 08:52:53 GMT	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.133	192.168.1.2	(empty)	(empty)	4700	49158	49158_tcp	(empty)	0	1
9	Wed, 13 Apr 2011 08:52:53 GMT	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.126	192.168.1.2	(empty)	(empty)	3337	49155	49155_tcp	(empty)	0	1
10	Wed, 13 Apr 2011 08:52:53 GMT	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.16	192.168.1.6	(empty)	(empty)	1097	135	epmap	(empty)	0	1
11	Wed, 13 Apr 2011 08:52:53 GMT	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.16	192.168.1.6	(empty)	(empty)	1099	135	epmap	(empty)	0	1
12	Wed, 13 Apr 2011 08:52:53 GMT	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.16	192.168.1.6	(empty)	(empty)	1100	43025	43025_tcp	(empty)	0	1
13	Wed, 13 Apr 2011 08:52:55 GMT	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.75	192.168.1.6	(empty)	(empty)	1048	135	epmap	(empty)	0	1
14	Wed, 13 Apr 2011 08:52:55 GMT	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.75	192.168.1.6	(empty)	(empty)	1049	43025	43025_tcp	(empty)	0	1
15	Wed, 13 Apr 2011 08:52:55 GMT	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.75	192.168.1.6	(empty)	(empty)	1051	135	epmap	(empty)	0	1

Figure 7.1: *Home page that display original raw logs.*

Upon entering the web application, a user will see the raw firewall logs in the VAST dataset. We used a React data mapping to display the firewall logs on the home page through server side pagination. A user can start our anomaly detection system here after pciking a time slice.

2. Anomaly Matrix Table



ID	Date Time	Priority	Operation	Message Code	Protocol	Source IP	Destination IP	Source Port	Destination Port	Service	Direction	Connections built	Torn Down Connections	Anomaly	S Security Concern Port	S Source IP Inside	S Source IP Out	S Destination IP Inside	S Destination IP Out	S Source Port
10887406	2011-04-14 10:52:52	Info	Built	ASA-session-6-302013	TCP	192.168.2.174	192.168.1.126	46208	443	https	inbound	1	0	0	0.281091	-0.049415	0.046535	0.156470	-0.156471	-0.042143
10887407	2011-04-14 10:52:52	Info	Built	ASA-session-6-302013	TCP	192.168.2.174	192.168.1.127	46208	554	554_tcp	inbound	1	0	0	0.281091	-0.049415	0.046535	0.156470	-0.156471	-0.042143
10887408	2011-04-14 10:52:52	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.175	192.168.1.125	55891	8081	8081_tcp	inbound	0	1	0	0.281091	-0.049415	0.046535	0.156470	-0.156471	-0.042143
10887409	2011-04-14 10:52:52	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.174	192.168.1.132	46209	256	256_tcp	inbound	0	1	0	0.281091	-0.049415	0.047488	0.156470	-0.156471	-0.042143
10887410	2011-04-14 10:52:52	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.174	192.168.1.133	46209	113	auth	inbound	0	1	0	0.281091	-0.049456	0.046535	0.156470	-0.156471	-0.042143
10887411	2011-04-14 10:52:52	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.175	192.168.1.105	55892	8011	8011_tcp	inbound	0	1	0	0.281091	-0.049415	0.047488	0.156470	-0.156471	-0.042143
10887412	2011-04-14 10:52:52	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.175	192.168.1.106	55892	8011	8011_tcp	inbound	0	1	0	0.281091	-0.049415	0.047488	0.156470	-0.156471	-0.042143
10887413	2011-04-14 10:52:52	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.175	192.168.1.128	55891	8081	8081_tcp	inbound	0	1	0	0.281091	-0.049415	0.047488	0.156470	-0.156471	-0.042143
10887414	2011-04-14 10:52:52	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.175	192.168.1.109	55892	8011	8011_tcp	inbound	0	1	0	0.281091	-0.049415	0.047488	0.156470	-0.156471	-0.042143
10887415	2011-04-14 10:52:52	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.174	192.168.1.86	46208	256	256_tcp	inbound	0	1	0	0.281091	-0.049415	0.046535	0.156470	-0.156471	-0.042143
10887416	2011-04-14 10:52:52	Info	Built	ASA-session-6-302013	TCP	192.168.2.174	192.168.1.130	46208	1723	pptp	inbound	1	0	0	0.281091	-0.049415	0.046535	0.156470	-0.156471	-0.042143
10887417	2011-04-14 10:52:52	Info	Teardown	ASA-session-6-302014	TCP	192.168.2.175	192.168.1.112	55892	6881	6881_tcp	inbound	0	1	0	0.281091	-0.049415	0.047488	0.156470	-0.156471	-0.042143
10887418	2011-04-14 10:52:52	Info	Built	ASA-session-6-302013	TCP	192.168.2.175	192.168.1.116	55891	5850	5850_tcp	inbound	1	0	0	0.281091	-0.049415	0.046535	0.157369	-0.157369	-0.042143
10887419	2011-04-14 10:52:52	Info	Built	ASA-session-6-302013	TCP	192.168.2.174	192.168.1.108	46209	3389	3389_tcp	inbound	1	0	0	4.110542	-0.049415	0.046535	0.156470	-0.156471	-0.042143
10887420	2011-04-14 10:52:52	Info	Built	ASA-session-6-302013	TCP	192.168.2.175	192.168.1.100	55892	3828	3828_tcp	inbound	1	0	0	0.281091	-0.049415	0.047488	0.156470	-0.156471	-0.042143

Figure 7.2: *Anomaly matrix table produced by RPCA.*

After a user has ran our anomaly detection system on a specified time slice, the anomaly matrix will be displayed. The matrix has is comprised of S values from RPCA. By hovering over cells a tool tip with M and M - S values will appear. Along with RPCA relevant values are the raw firewall log information. In doing so, if a row is determined to be anomalous an analyst can quickly see information regarding that entry.

3. tSNE Exploration

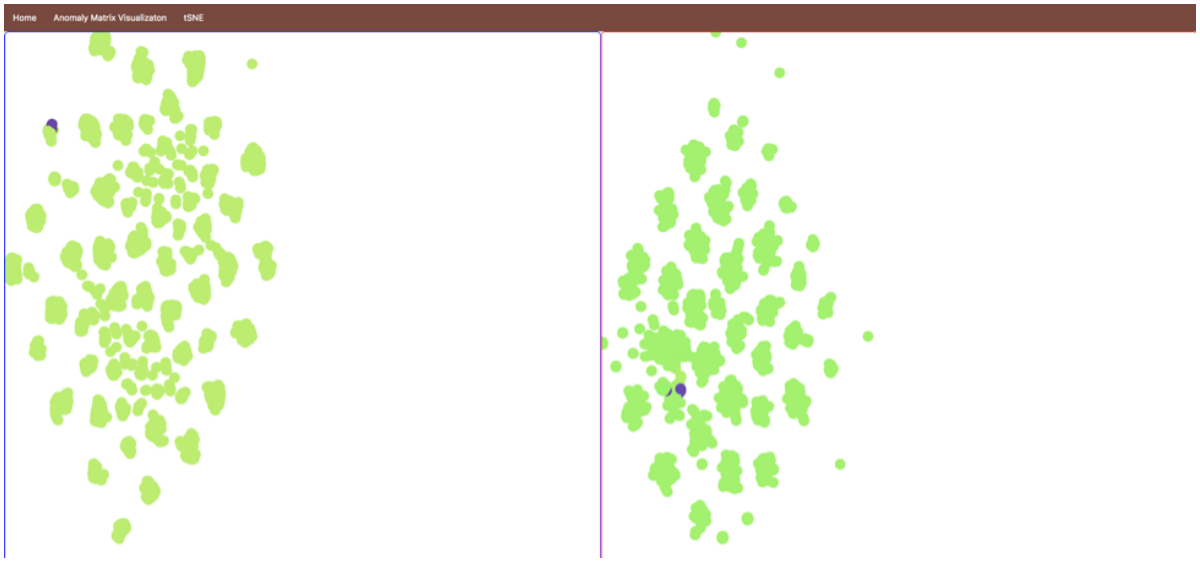


Figure 7.3: *tSNE visualizations. Left is tSNE ran on our feature set. Right is tSNE ran on the S matrix produced by RPCA.*

By navigating to the tSNE tab, a user can run our tSNE visualization. To the left is tSNE ran on our feature matrix and to the right is tSNE ran with the RPCA anomaly matrix. Our results with tSNE are preliminary and we explain how tSNE can be leveraged in chapter 8.

7.2.2 Choosing a Visualization Library

The first step on creating a visualization system was examining how much customization was needed to create the different perspectives we wanted to highlight in our web application. Although several libraries exist to create graphs, such as c3 [62], recharts [71], nvd3 [61], etc, they tend to limit the flexibility of creating unique and specific purpose visualizations. Our final decision rested tbetween d3 and Recharts due to their ability to create compelling results. Table 7.1 depicts our final analysis on these libraries.

We concluded that d3 would fit our application the best since d3 is a flexible and powerful visualization system. Although it does have a steep learning curve, there are several examples that we could follow along and it has a big online community available to ask questions. In addition, its flexibility makes it easy to adapt its visualizations if the set of features change. For

d3		Recharts	
Pros	Cons	Pros	Cons
• Customization • Large community and several resources • Interactive components	• Steep learning curve	• Renders Quickly and Smoothly • Fairly extensible library • Responsive • Intuitive	• Depends on React.js • Limited examples available

Table 7.1: *Pros and cons of d3 and Recharts [7], [71]. These were the last two libraries that we were evaluating for our visualization system.*

our application, we created a table using d3 and color coded it based on the values in the S-Matrix produced by rPCA. We highlighted anomalies in red, and allowed user analysts to explore the M and M-S values by using a tooltip to avoid the cluttering of data.

7.2.3 Web Framework

With the creation of each visualization it was important to consider the presentation of the graphs, or techniques. We designed a React application, using Flask as the backend framework. Flask's microframework nature allowed us to quickly set up the skeleton of our application while focusing more on the front end aspects of the React app.

In order to create an easy to use and intuitive application we start the web application loading the original user logs with server-side pagination. After visualizing the log activity, in order by date, the user can select a date range that includes the date, hour, minutes and even seconds in which they want to run the feature analysis. In a real world system, one could imagine current log activity display in a similar fashion. After a user examines the data set and have chosen a time slice for feature analysis, the Flask backend sets in motion a series of backend API calls to our manager. Our web app takes advantage of quick mySQL queries and creates a table called working set which includes the specific log files selected by the user. The working set table is redefined whenever a user chooses a time slice. Then, we create features based on that slice, and finally, we ran RPCA to create an anomaly csv that depicts the features and the original data. The csv is hosted and read with d3. As mentioned in the previous section, we display a table with the anomaly csv information and highlight those anomalies that have an S value greater than one with different shades of red to indicate how severe we think it is and include interactive tooltips for the M and M-S matrix.

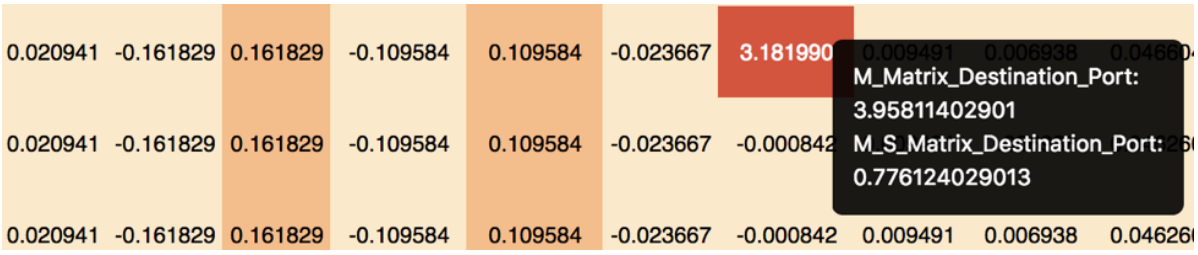


Figure 7.4: *Depicts the tooltip for M and M-S Values. M is the original values and M-S is the predicted value of the cell examined.*

7.2.4 Data for Table

As outlined in chapter 4, we had upwards of 100 columns in our features. A table that contains that much information would completely overwhelm the user as they would have to scroll horizontally to be able to see them, and would never be able to get the full scope of each row. Therefore, we decided to consolidate the destination and source ip/port features into one column to reduce the amount of columns. To do this, we paired up each row's actual port or address to the feature that was provided in RPCA. In other words, we took the maximum value in the S matrix corresponding to each row and feature pair and displayed it to the user. This made a significant impact in the number of columns as it reduced it to around 30.

In addition, we wanted to display the original data (M) and the predicted value (M-S matrix) so the analyst could explore the original data and compare RPCA's prediction to what the actual values were. At first, we played with idea of displaying them in the same table cell if S was greater than a certain threshold that signaled an anomaly. However, that made each cell inconsistent and it seemed to display more information that was necessary. We created a tooltip for each cell in the row that was related to an S matrix. This would allow an analyst to decide which values he or she is curious about and selectively view the M and M-S values as depicted in Figure 7.4.

Finally, we created a color scale to highlight S matrix values according to the level of severity. Currently, the scale darkens as the number gets bigger towards a dark red. We picked red since it unequivocally has a sense of urgency attached to it. We used ColorBrewer [16] to find other colors that would meld nicely with the red. Figure 7.5, is the table with its full ranges, with light yellow representing a neutral value.

7.3 Case Studies

In order to explain how our system would work if an analyst used it. We created case studies of how the work flow of the application functions and highlight the different attacks that our anomaly detection system was able to find. This section illustrates the specific steps a system analyst can do to find potential anomalies and how it is currently cross validated.

Torn Down Connections	Anomaly	S Security Concern Port	S Source IP Inside	S Source IP Out	S Destination IP Inside	S Destination IP Out	S Source Port	S Destination Port
0	0	0.020941	-0.161829	0.161829	-0.109584	0.109584	-0.023667	4.933151
0	0	0.020941	-0.161829	0.161829	-0.109584	0.109584	-0.023667	3.240892
0	0	0.020941	-0.161829	0.161829	-0.109584	0.109584	-0.023667	6.437453
0	0	0.020941	-0.161829	0.161829	-0.109584	0.109584	-0.023667	4.933151
0	0	0.020941	-0.161829	0.161829	-0.109584	0.109584	-0.023667	3.240892
0	0	0.020941	-0.161829	0.161829	-0.109584	0.109584	-0.023667	6.437453
0	0	0.021022	-0.161829	0.161829	-0.109584	0.109584	-0.023667	-0.005361
0	0	0.021022	-0.161829	0.161829	-9.247493	9.247493	-0.023667	-0.005361
0	0	0.021022	-5.717763	5.717760	-0.109584	0.109584	-0.023667	-0.005361

Figure 7.5: *Color scale that portrays warnings, here the larger the S value, the darker the background color.*

7.3.1 System Analyst Explores Web Framework to Detect Remote Desktop Protocol

1. Viewing Log Data

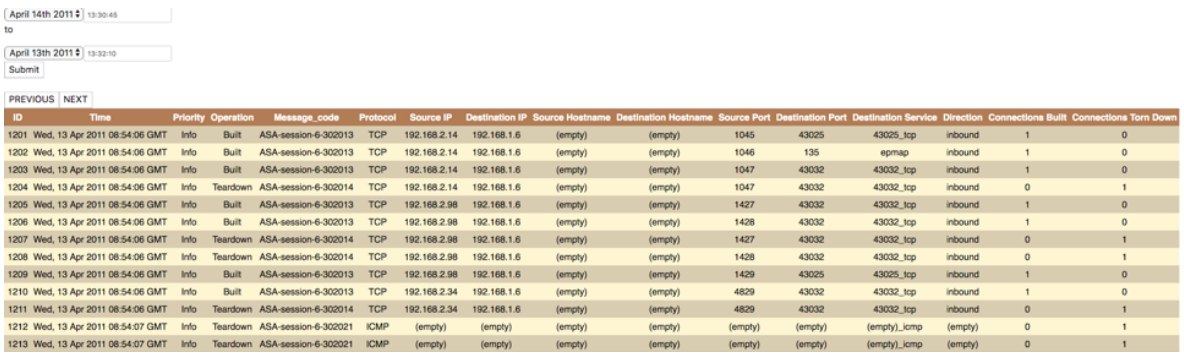


Figure 7.6: *Viewing VAST firewall log data in our web application*

The process of analyzing data would begin with our home screen depicting the firewall logs. In a real time environment this could update as different firewall logs come in.

2. Using Date Selector

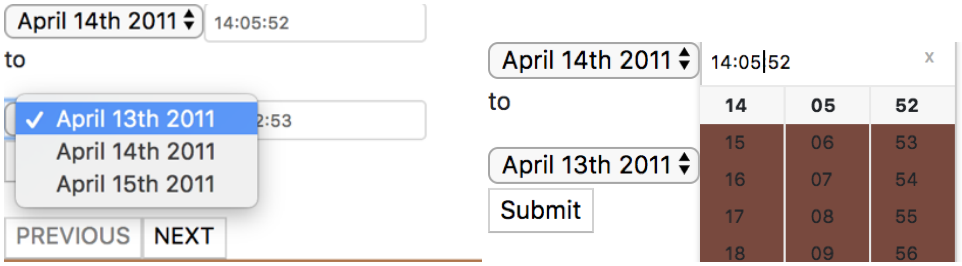


Figure 7.7: *Date Selectors in application. Here you would choose dates to run RPCA on a specified time slice.*

After the user decides which data to run feature analysis on, they select a date, hour, minutes, and seconds in which to generate the features. To view the Remote Desktop Protocol attack, the selected time would be April 14th 13:30:45, and April 14th 13:32:10.

3. Observing the Anomaly Matrix

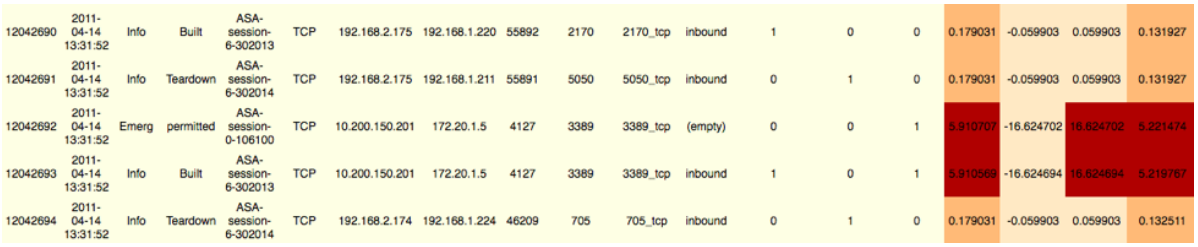


Figure 7.8: *Visualization of RPCA anomaly matrix in web application.*

We used our visualization system in order to cross validate attacks with the ground truth data. As mentioned in chapter 2, the remote desktop protocol attack was socially engineered from an external IP address of AFC's network. In Figure **??** we examined a subset of data with both anomalous and normal data.

4. Exploring M and M-S Values

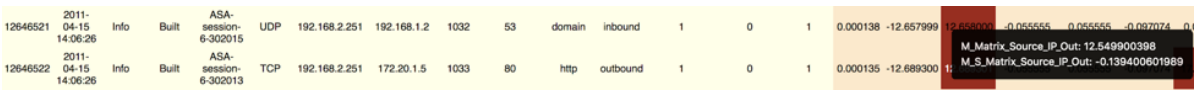


Figure 7.9: *Example of how M and M-S values are depicted.*

If an analyst is further interested in the M and M - S values it is easily accessible by hovering over the S values. This will display a tooltip that shows both values and could be

examined further. Although in this MQP we did not explore these values, it could expand the anomaly detection process.

The features that captured this attack were the security concern port and the source and destination outside features. Our cross validation was simple because the solution manual provided all anomalous data. However, one could imagine a network analyst using our visualization system, noticing the red scale color warning, and making an informed observation of this attack.

7.3.2 System Analyst Explores Web Framework to Detect UDC

Another example of using our visualization system to detect attack is the undocumented computer attack. As mentioned in chapter 2, the attack stemmed from suspicious activity from a computer outside of AFC's network. To generate this result, the start and end time entered were April 15th 14:05:52 and April 15th 14:06:53, respectively. The feature that flagged the attack here is the destination outside network feature. Again, we used the ground truth data to cross validate this attack, but in a real world system the red color scale warning would notify a system analyst to further examine the log and make an informed observation.

12646620	2011-04-15 14:06:38	Info	Deny	ASA-session-6-106015	TCP	192.168.1.2	192.168.2.116	49158	3136	3136_tcp	(empty)	0	0	0	0.055490	-0.078892	0.078892	-0.055550	0.055550
12646621	2011-04-15 14:06:38		Deny	ASA-session-6-106015	TCP	192.168.1.14	192.168.2.116	49155	3133	3133_tcp	(empty)	0	0	0	0.055490	-0.078892	0.078892	-0.055550	0.055550
12646622	2011-04-15 14:06:38	Info	Teardown	ASA-session-6-302014	TCP	172.20.1.5	192.168.2.251	80	1033	1033_tcp	outbound	0	1	1	0.055490	-0.078892	0.078892	-17.892403	17.892403
12646627	2011-04-15 14:06:38	Info	Built	ASA-session-6-302013	TCP	192.168.2.46	192.168.1.14	1422	445	microsoft-ds	inbound	1	0	0	0.055490	-0.078892	0.078892	-0.055550	0.055550

Figure 7.10: *Undocumented computer attack shown in anomaly matrix. Here we cross validated that our features found this attack by referencing the ground truth.*

7.4 High Dimensional Visualization with tSNE

Beyond validation, we went further and explored the use of an algorithm that could handle high-dimensional data. In particular, we decided to explore the use of tSNE [44], an award winning technique to visualize these types of data sets.

In Figure 7.11, tSNE is ran on two different sets of data. The left most visualization incorporates the final features created in our back-end as outlined in chapter 4, while the rightmost shows the S values for the same feature set to compare the clustering results. This allows further analysis into the possibilities of using tSNE or other high visualization algorithms combined with RPCA to find anomalies promptly. An interesting result to note was that in the tSNE visualization without RPCA, we observe that the ground-truth anomalies (in purple) are very distinctly in two different clusters – yet in the visualization with RPCA, the anomalies (pictured in red) are considered less different. This highlights the impact tSNE and RPCA could have in an anomaly detection process. It is important to note that in order to confirm that the anomalies are in the

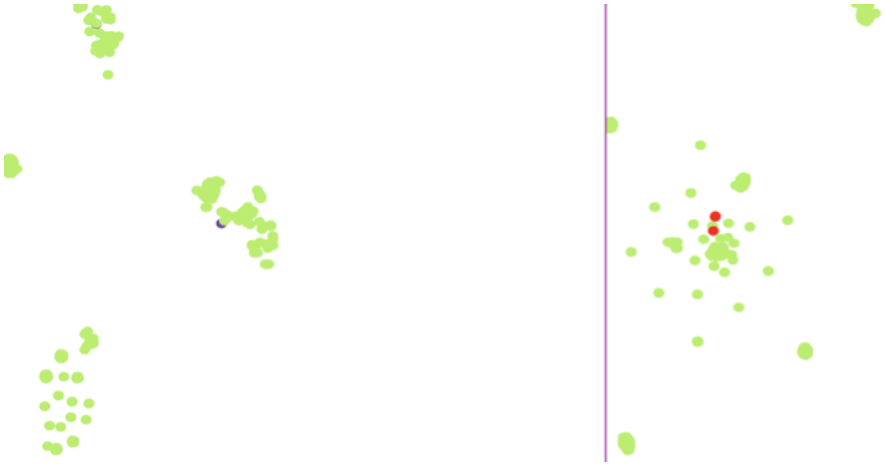


Figure 7.11: *tSNE visualization using our features. To the left is the visualization with our features and to the right is post-RPCA*

clusters mentioned above, it would require a clustering algorithm such as K-means [49], which is highlighted in our recommendations.

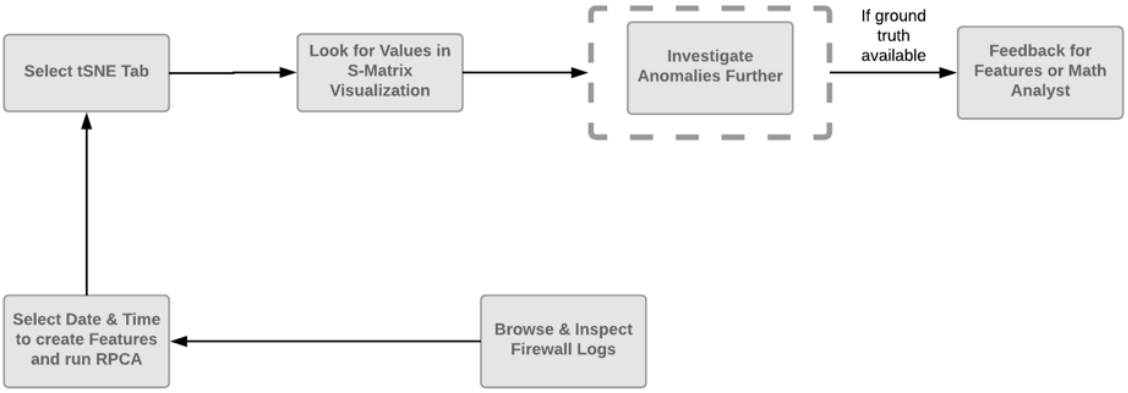


Figure 7.12: *Flowchart of tSNE result creation. One of our goals in closing the loop in an anomaly detection system and running tSNE dynamically is another example of this.*

Detailed in Figure 7.12 is the process that an analyst would follow to recreate the example in Figure 7.11. To continue providing exploratory grounds for the analyst, the tSNE visualization is ran in the user-defined time-slice. The workflow of an analyst would include observing which data-points or clusters are distinguishable from regular data and further investigate in the S matrix values table if those data points show a concerning value. This would provide a way to further close the loop of anomaly detection.

CHAPTER 8

FUTURE WORK/RECOMMENDATIONS

8.1 Limitations

Our project had several, important to note limitations. In our case, we were examining a limited 4-day dataset. In one hand, it contained a significant amount of data (12 million rows) because of the DDoS. Nevertheless, in a real-world system that tracked cyber security logs, this would steadily increase and would create difficulties in both memory and performance. If a project in the future continued using time-slices to calculate features, it would be feasible to manage more data. However, analyzing it would take a more scalable system.

Due to the time-constraints in the length of the project, there are certain characteristics of the current project that could not be generalized. For example, when we run our RPCA code, we provide every row in our feature generation, which includes some data that must be ignored. We currently ignore it explicitly in our code, but in the future a solution that can provide analysis and be able to identify which columns are anomalies without having to hard code those columns would be extremely valuable.

Currently, the code that has been written for our application is both in Python 2.7 and 3.6. We have a script that depends upon certain environments existing (as detailed in our readme) [59], which switches between Python versions. In an ideal world, both of these applications would be written in the same version of Python.

8.2 Build Application on one Python Version

Throughout this project we continued feature engineering work that was written in Python 2.7. Given time, we would try to port our code base to exclusively using Python 3. Although, there is no limit to the documentation of Python 2 code, it would be better for efficiency and flexibility

to build the anomaly detection system exclusively in Python 3 [54]. This would eliminate the necessity of having two virtual environments and using a bash script to change Python versions in the middle of a run cycle. If a team were to continue our work, a first step would be to ensure that the code repository is translated into Python 3 or start from scratch in Python 3 and use our code and paper as a starting point. Python 3 should be used for the infrastructure of this MQP considering that most Python 2 code is now legacy code.

If the next team would like to use our code and not rewrite the entire repository it could be worthwhile to translate the code with Python's Futurize library, as stated in chapter 5.

8.3 Explore clustering with tSNE to visualize features

We created a tSNE visualization using d3 to explore how reducing the dimensionality of our feature set would translate in our anomaly detection system and its usability to further explore the concerns. In an interview with David Beach [12], we discussed the meaning of Figure 7.11 and the possibilities of using tSNE for this project. In tSNE, it is critical to understand the feature space, since it focuses on clusters and distances become meaningless. In Beach's work, he used the firewall logs to detect connections between machines and calculate the probability that a node will communicate with another. With his work, he was able to visualize the different instances in which attacks were occurring in the dataset. To be able to use these features, think in terms of the manifold, and define what it means for two points to be close to each other could help increase the analysis of a dataset to help prevent attacks [12]. An interesting angle to explore would be to classify the clusters using an algorithm such as k-means [49]. This would allow color coding on the front-end and exploring different parameters in tSNE to observe the different results and highlight clusters that might be anomalous. Finally, it would be interesting to add more interactivity so that an analyst can decide which tSNE parameters they would like to run themselves and analyze the outliers more carefully.

8.4 Data Smoothing

One could imagine some form of data smoothing being applied to the dataset to reduce noisy information and allowing outliers to become more apparent. Data smoothing has not been applied in this project, but from a mathematical perspective, it would be interesting to see how smoothing would affect the anomaly detection system for this dataset. In financial investment software smoothing has been used to explain the behavior of stocks and determine patterns in the market [39].

8.5 Rule Generation Interface

In this report, we have outlined several feature sets to detect attacks in the VAST dataset. These feature sets could be modified and made into rules that could be toggled. In the future, it would be interesting to apply rule generation for this dataset challenge, and extend the generalization solution we have proposed. A network administrator could use rules with a firewall log file to detect certain instances of possibly malicious behavior. There can be a system where a set of rules exist for a data set. These rules could then be generated in a web platform by a user depending on what they are interested in. For example, a network administrator may be interested in detecting IP addresses from untrusted clients. In this instance, they may only want to apply a rule set that corresponds to that case. The problem with this is an increasing amount of rule sets that would require constant updating. However through a learning algorithm, there could be an anomaly detection system that generates rules for a set. Such an algorithm is explored in *Learning Rules for Anomaly Detection of Hostile Network Traffic* by Matthew V. Mahoney and Philip K. Chan [45].

Chapter 9

Conclusion

Anomaly detection is devised of several components that make it complex and hard to standardize. Engineering features that can be applied in various situations and ensuring that a data analyst can generate features on demand is a stepping stone towards facilitating the process. In this Major Qualifying Project, we used feature engineering and Robust Principal Component Analysis to detect anomalous behavior in the 2011 VAST dataset challenge. Throughout this project we were able to detect multiple attacks in the dataset including: denial of service, socially engineered remote desktop infiltration, and undocumented computers appearing in a company's network. Through our web application, we connected the loop between a system analyst and mathematician. A user can quickly query the dataset and choose exactly which times they would like to examine and run features on. From here, it is simple to see which rows can be out of the normal trend. Through several iterations of the feature engineering process we found a feature library that RPCA agreed with and is also generalized. Without much effort we could ingest another company's network architecture, mainly IP address and ports information and find behavior out of the realm of that network.

BIBLIOGRAPHY

[1] Common windows exploit port list.
https://support.zen.co.uk/kb/knowledgebase/common-windows-exploit-port-list.

[2] Trojans & backdoors index.
https://www.acunetix.com/vulnerabilities/trojans/.

[3] What is learning algorithm.
https://www.igi-global.com/dictionary/learning-algorithm/16821.

[4] Using the z-table in reverse: from the regions to the z-values, 2008.
http://norsemathology.org/wiki/index.php?title=Using_the_Z-Table_in_Reverse:_from_the_regions_to_the_Z-values.

[5] Vast challenge 2011 - mini-challenge 2 computer network operations at all freight corporation„ 2011.

[6] Up to 100,000 taxpayers compromised in fafsa tool breach, i.r.s. says, Apr 7, 2017.
http://scholar.aci.info/view/146d5be147f0bc303ec/15b45f5402e00019c57125b.

[7] Slant: D3 vs recharts, 2018.
https://www.slant.co/versus/10577/21007/~d3-js_vs_recharts.

[8] A. M. Alvarez, M. Yamada, A. Kimura, and T. Iwata.
Clustering-based anomaly detection in multi-view data.
page 1545–1548, New York, NY, USA, 2013. ACM.

[9] M. R. Anderson, D. Antenucci, V. Bittorf, M. Burgess, M. J. Cafarella, A. Kumar, F. Niu, Y. Park, C. Ré, and C. Zhang.
Brainwash: A data system for feature engineering.
In *CIDR*, 2013.

[10] D. Axmark and W. M.
Mysql 5.7 reference manual, 2015.
http://dev.mysql.com/doc/refman/5.7/en/index.html.

[11] K. Bansal and E. M. Bansal.
Data clustering and visualization based various machine learning techniques.
International Journal of Advanced Research in Computer Science, 7(6), Nov 1, 2016.

[12] D. Beach.
Private interview, February 23 2018.

[13] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita.
Network anomaly detection: Methods, systems and tools.
IEEE Communications Surveys & Tutorials, 16(1):303–336, 2014.

[14] I. Bicking.
Virtualenv, 2011.
`https://virtualenv.pypa.io`.

[15] M. Bostock.
d3 library, 2017.
`https://github.com/d3`.

[16] C. Brewer.
Colorbrewer, 2015.
`http://colorbrewer2.org/#type=sequential&scheme=BuGn&n=3`.

[17] J. Brownlee.
Discover feature engineering, how to engineer features, 2014.
`https://machinelearningmastery.com/discover-feature-engineering\`
`-how-to-engineer-features-and-how-to-get-good-at-it/`.

[18] A. Buja, D. F. Swayne, M. L. Littman, N. Dean, H. Hofmann, and L. Chen.
Data visualization with multidimensional scaling.
Journal of Computational and Graphical Statistics, 17(2):444–472, Jun 1, 2008.

[19] J.-F. Cai, E. J. Candès, and Z. Shen.
A singular value thresholding algorithm for matrix completion.
SIAM Journal on Optimization, 20(4):1956–1982, 2010.

[20] V. Chandola, A. Banerjee, and V. Kumar.
Anomaly detection: A survey.
ACM Computing Surveys (CSUR), 31(3), 2009.

[21] S. Chen and V. Janeja.
Human perspective to anomaly detection for cybersecurity.
Journal of Intelligent Information Systems, 42(1):133–153, Feb 2014.

[22] L. Cranor.
A framework for reasoning about the human in the loop.
https://www.usenix.org/legacy/event/upsec/tech/full_papers/cranor/cranor_html/.

[23] L. De Lathauwer, B. De Moor, and J. Vandewalle.
A multilinear singular value decomposition.
SIAM journal on Matrix Analysis and Applications, 21(4):1253–1278, 2000.

[24] R. Deraison.
Nessus vulnerability scanner, 2010.
https://www.tenable.com/products/nessus/nessus-professional.

[25] X. Ding, L. He, and L. Carin.
Bayesian robust principal component analysis.
IEEE Transactions on Image Processing, 20(12):3419–3430, 2011.

[26] M. Droettboom, E. Firing, D. Dale, and J. Hunter.
Matplotlib library.
https://matplotlib.org/.

[27] J. G. Dy and C. E. Brodley.
Feature selection for unsupervised learning.
Journal of machine learning research, 5(Aug):845–889, 2004.

[28] M. A. Figueiredo.
Adaptive sparseness for supervised learning.
IEEE transactions on pattern analysis and machine intelligence, 25(9):1150–1159, 2003.

[29] H. Flores-Huerta, J. Link, and C. Litch.
Cyber security network anomaly detection and visualization.
2017.

[30] Y. Gao, X. Miao, G. Chen, B. Zheng, D. Cai, and H. Cui.
On efficiently finding reverse k-nearest neighbors over uncertain graphs.
The VLDB Journal, 26(4):467–492, Aug 2017.

[31] P. Garcia-Teodoro, E. Vazquez, G. Macia-Fernández, and J. Diaz-Verdejo.
Anomaly-based network intrusion detection: Techniques, systems and challenges.
28:18–28, 02 2009.

[32] D. Geer.
Securing risky network ports, Apr 24, 2017.
https://search.proquest.com/docview/1891377793.

[33] G. F. Gidey and C. A. Onana.
High dimensional data visualization: Advances and challenges.
International Journal of Computer Applications, 162(10), Jan 1, 2017.

[34] R. A. Grimes.
Danger: Remote access trojans, September 2002.
`https://technet.microsoft.com/en-us/library/dd632947.aspx`.

[35] L. Harrison.
Github repository of vulnerability scan parser.
`https://github.com/ornl-sava/nv/blob/master/source/js/parser/src/parser.js`.

[36] A. J. Hoglund, K. Hatonen, and A. S. Sorvari.
A computer host-based user anomaly detection system using the self-organizing map.
In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 5, pages 411–416. IEEE, 2000.

[37] A. J. Hoglund, K. Hatonen, and A. S. Sorvari.
A computer host-based user anomaly detection system using the self-organizing map.
volume 5, page 416 vol.5, 2000.

[38] A. Holzinger.
Interactive machine learning for health informatics: when do we need the human-in-the-loop?
Brain Informatics, 3(2):119–131, Jun 2016.

[39] Investopedia.
Data smoothing.
`https://www.investopedia.com/terms/d/data-smoothing.asp`.

[40] D. Jiang, Z. Xu, P. Zhang, and T. Zhu.
A transform domain-based anomaly detection approach to network-wide traffic.
Journal of Network and Computer Applications, 40:292–306, Apr 2014.

[41] Z. Lin, M. Chen, and Y. Ma.
The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices.
arXiv preprint arXiv:1009.5055, 2010.

[42] J.-X. Liu, Y.-T. Wang, C.-H. Zheng, W. Sha, J.-X. Mi, and Y. Xu.
Robust pca based method for discovering differentially expressed genes.
In *BMC bioinformatics*, volume 14, page S3. BioMed Central, 2013.

[43] Y. Liu, H. Xu, H. Yi, Z. Lin, J. Kang, W. Xia, Q. Shi, Y. Liao, and Y. Ying.
Network anomaly detection based on dynamic hierarchical clustering of cross domain data.
pages 200–204. IEEE, 2017.

[44] L. Maaten.
tSNE, 2008.
https://lvdmaaten.github.io/tsne/.

[45] M. V. Mahoney and P. K. Chan.
Learning rules for anomaly detection of hostile network traffic.
In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 601–604. IEEE, 2003.

[46] D. S. Mare, F. Moreira, and R. Rossi.
Nonstationary z-score measures.
European Journal of Operational Research, 260(1):348–358, 2017.

[47] S. Mehdi and J. Khalid.
Revisiting Traffic Anomaly Detection Using Software Defined Networking, volume 6961 of *Recent Advances in Intrusion Detection*, pages 161–180.
Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[48] S. Morgan.
Cybersecurity facts, figures and statistics for 2018, 2018.
https://www.csoonline.com/article/3153707/security/top-5-cybersecurity-facts\
-figures-and-statistics.html.

[49] A. P. Muniyandi, R. Rajeswari, and R. Rajaram.
Network anomaly detection by cascading k-means clustering and c4.5 decision tree algorithm.
Procedia Engineering, 30:174–182, 2012.

[50] 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, 12:2825–2830, 2011.

[51] O. Ramadan.
A data science approach to network security using machine learning and time series analysis, 2016.
https://www.linkedin.com/pulse/data-science-approach-network-security-\
using-machine-learning-osman/.

[52] S. Rastegari, P. Hingston, and C.-P. Lam.
Evolving statistical rulesets for network intrusion detection.
Applied Soft Computing, 33:348–359, Aug 2015.

[53] A. Ronacher.
Flask, 2017.
`http://flask.pocoo.org/`.

[54] J. Rouse.
Python 3 vs python 2: It's different this time.
`https://www.activestate.com/blog/2017/01/python-3-vs-python-2-its-different-time`.

[55] C. Ré, A. A. Sadeghian, Z. Shan, J. Shin, F. Wang, S. Wu, and C. Zhang.
Feature engineering for knowledge base construction.
Jul 23, 2014.

[56] S. S. Sahu and M. Pandey.
Distributed denial of service attacks: A review.
International Journal of Modern Education and Computer Science, 6(1):65, Jan 1, 2014.

[57] S. Saravanakumar, Umamaheshwari, D.Jayalakshmi, and R.Sugumar.
Development and implementation of artificial neural networks for intrusion detection in computer network .
International Journal of Computer Science and Network Security, 10, 2010.

[58] E. Schofield.
Futurize, 2013.
`http://python-future.org/futurize.html`.

[59] E. Sola, A. Velarde Ramirez, and K. Guth.
Mqp anomaly vis 2017.
`https://github.com/wpivis/mqp-anomalyvis-17`.

[60] R. Sommer and V. Paxson.
Outside the closed world: On using machine learning for network intrusion detection.
pages 305–316, 2010.

[61] D. Souther.
nvd3 library.
`https://github.com/novus/nvd3`.

[62] M. Tanaka.
C3 library.
`http://c3js.org/`.

[63] C. R. Turner, A. Fuggetta, L. Lavazza, and A. L. Wolf.
A conceptual basis for feature engineering.
Journal of Systems and Software, 49(1):3–15, December 15, 1999.

[64] L. J. P. van der Maaten and G. E. Hinton.
Visualizing high-dimensional data using t-sne.
Journal of Machine Learning Research, 9(nov):2579–2605, 2008.

[65] L. Wei, N. Kumar, V. N. Lolla, E. J. Keogh, S. Lonardi, and C. A. Ratanamahatana.
Assumption-free anomaly detection in time series.
In *SSDBM*, volume 5, pages 237–242, 2005.

[66] B. Wippich.
Detecting and preventing unauthorized outbound traffic.
`https://www.sans.org/reading-room/whitepapers/detection/detecting-preventing-unauthorized-outbound-traffic-1951`.

[67] C. Xiong, J. Zhang, and X. Luo.
Ridge-forward quadratic discriminant analysis in high-dimensional situations.
Journal of Systems Science and Complexity, 29(6):1703–1715, 2016.

[68] Y. Yan, E. Ricci, R. Subramanian, G. Liu, and N. Sebe.
Multitask linear discriminant analysis for view invariant action recognition.
IEEE Transactions on Image Processing, 23(12):5599–5611, 2014.

[69] W. Yu, N. Zhang, X. Fu, and W. Zhao.
Self-disciplinary worms and countermeasures: Modeling and analysis.
IEEE Transactions on Parallel and Distributed Systems, 21(10):1501–1514, 2010.

[70] T. Zhang, Q. Liao, and L. Shi.
Bridging the gap of network management and anomaly detection through interactive visualization.
pages 253–257. IEEE, 2014.

[71] H. Zhou.
Recharts, 2015.
`https://github.com/recharts/recharts`.

[72] H. Zou, T. Hastie, and R. Tibshirani.
Sparse principal component analysis.
Journal of computational and graphical statistics, 15(2):265–286, 2006.