Master Thesis: Detecting Exploit kits using Machine Learning

Author: **Pallavi Jagannatha (pallavijagannatha@gmail.com)**

MSc: EIT Digital Master on Security and Privacy

Graduation Committee:

Dr. Zhiyuan (Thomas) Tan (Edinburgh Napier University)

Dr. Andreas Peter (University of Twente)

Dr. Alexandr Vasenev (University of Twente)

Christian Prickaerts (FOX-IT)

Maarten Van Dantzig (FOX-IT)

Martin van Hensbergen (FOX-IT)

Abstract

The job of a security analyst is often similar to the task of finding a needle in a haystack when the clock is ticking. It is not about having the knowledge of computing and networking on a level of how it functions, but it is about finding something that is present underneath the covers. A security analyst has to go through a lot of data manually to learn/identify about a security incident. However, it is not only time-consuming, but also difficult to perform a thorough analysis of large data manually. This work focuses on providing an efficient methodology for a two-layer detection scheme, which is not only lightweight but also effective in attack detection and clustering.

Exploit kits are a serious cyber threat today as they are responsible for a big percentage of malware infections worldwide. Everyone from inexperienced hacker to a sophisticated cybercriminal can use an Exploit kit to spread malware. An Exploit kit is a tool which detects and Exploits the vulnerabilities present in a system to spread malware (banking malware, ransomware, etc.). An Exploit kit typically has a management console, add-on functions and a set of vulnerabilities that can Exploit different applications (which have security-related vulnerabilities) to launch an attack. The whole process is automated which makes it very effective. Exploit kits contribute to a significant portion of web-based threats on the internet today, creating an effective staging ground for criminal activities. Through the extensive history of Exploit kit attacks, it is evident that it is important to have an effective system in place to detect and defend users from such attacks.

This research project has developed a new approach for the automatic detection of a network security incident, namely Exploit kits in this work. This research uses machine learning to detect Exploit kits with the help of supervised and unsupervised learning approach. The feature selection based on information gain allows data itself to explain the importance of individual features for the detection. This helps human experts avoid tedious manual feature selection.

Keywords: *Exploit kits, Supervised learning, Unsupervised learning, Feature selection, Naive Bayes, K-means clustering, Vectorization, Information Gain.*

Dedication

I would like to dedicate my thesis to my **Appa** (Father), **Jagannatha K.N**, my **Amma** (Mother), **Padma B.K**, my eldest sister, **Dr. Pushpa Jagannatha**, my brother-in-law, **Dr. Vinay K.S**, my elder sister, **Ramya Jagannatha**, my niece, **Abhigna Sanvi**, my Ma1, **Ingrid Muller-Farny**, my Ma2, **Roetie Adams**, and my dog, **Tippu**. I would like to thank them for their unconditional love, support and encouragement.

Acknowledgements

I would like to use this opportunity to express my gratitude to my supervisors from the University of Twente, Dr. Zhiyuan (Thomas) Tan, Dr. Andreas Peter and Dr. Alexandr Vasenev for their guidance, advice and support in my research and academic developments.

I would like to specially thank my supervisor, Dr. Zhiyuan (Thomas) Tan for motivating me to build my machine learning knowledge. Without that knowledge and his guidance, this research would not have been possible. I appreciate that he believed in me to provide his time and knowledge to make this possible. Dr. Zhiyuan (Thomas) Tan is a great person and a scholar; without him, this thesis may not have been written.

I would like to thank my supervisors from FOX-IT, Barry Weymes, Christian Prickaerts, Maarten Van Dantzig and Martin Van Hensbergen for providing me with an intellectual environment, constant support and the long discussions about my research.

I would like to thank Renato Fontana, Dirk-Jan Korpershoek, Yonne de Bruijn, Lennart Haagsma, Antonios Terefos, Leon Buikstra, Erik Schamper, Max de Bruijn, Gina Doekhie, Mike Stokkel, Matthijs Gielen, Pepijn Westen**,** Bradley Prent from FOX-IT for their valuable insights on my research.

I would like to thank my best friends at FOX-IT, Valentijn Harmers, Maarten van Dantzig Paolo Schipani, Juno Teeuw, Claudia, Paul Platanias and Rolf Don for their support, encouragement, and making my research experience at FOX-IT a memorable journey.

Finally, I would also like to thank David Fincher, a movie director, who inspired me to pursue my masters in Cyber Security through his movie "The Girl with the Dragon Tattoo".

Contents

List of Figures

Abbreviations

SOC	Security Operation Centre
SSE	Sum Squared Error
IDS	Intrusion Detection System
NB	Naive Bayes
TFIDF	Term Frequency Inverse Document Frequency

Chapter 1 Introduction

The internet was originally developed when computers were huge and expensive. Only a few organizations, such as universities, business organizations and governments could afford them. We had a network of computers when these computers started communicating with each other and it grew up as Personal Computers (PCs) which emerged in 1980s. People started using it for more than just communication. Other devices, such as phone, cell tower and so on started connecting with them. Today's world is made of computers, laptops, tablets and smart-phones. We use them for communication, work, news and entertainment. Furthermore, we connect to a global network that brings the entire world to us via them. We extend ourselves to the online world through our devices. We are always connected to a global network; this means we could be vulnerable just by being connected. The ease at which all these devices could connect with each other has come at the expense of security.

The widespread use of multiple new platforms gives rise to new points of attack for cyber criminals. Exploits are malicious programs that contain an executable code with an intent to take advantage of one or more vulnerabilities present in the software running on a remote device. An *Exploit kit* is a toolkit that contains Exploits to spread malware. The Exploit kits have become powerful and easily available, enabling even less technically proficient cybercriminals to take advantage of them. Therefore, common devices such as computers, laptops, tablets and smart-phones, now need methods to monitor and handle the security breaches.

Before considering the methods to handle security breaches, it is important to understand where we stand from the security point of view. A security analyst has to go through a lot of data to learn/identify about a network security incident. As of now, the security analysts are investing their time and efforts to do their best by investigating the security incidents in the data manually. However, they can do a lot better with tools that can optimise their work. There are many security events generated by many devices which detect abnormal behaviour in the network. This has led to the obvious problem of generation of enormous amount of data. Added to this, some events reported could also be a case of false alarm at times.

Despite having enormous data filled with all the valuable information, it is still a lot of work to obtain the needed information in order to detect, prevent and respond to network security incidents. It is only logical to build upon the knowledge obtained from understanding the past incidents that can help to respond to incidents effectively. This is where machine learning can help to address the issue and automate the process.

Applying well known machine learning algorithms to build and train the machine to detect a network security incident allows a security analyst to choose a fitting solution based on the requirements specification. This research focuses on detecting Exploit kits using both supervised machine learning approach and unsupervised machine learning approach. This research can make a significant positive impact to security companies as it can help them with their Security Operation Centers (SOC).

1.1 Research question

Exploit kits are currently the most powerful weapon used to spread malware on the Internet. This is a big threat to Internet security as it allows cybercriminals to easily infect a large volume of machines Exploiting various software vulnerabilities. The impact of Exploit kits is dramatic since a cyber criminal can perform several kinds of attacks, like identity theft or banking fraud using them.

This thesis revolves around the threat of Exploit kits, in particular to provide an efficient methodology to detect and cluster different types of Exploit kits. This can reduce the manual efforts that has to be made by security analysts in detecting this network security incident.

How to design an efficient and automated approach to detect Exploit kits and cluster them further into different types?

1.1 Contribution

By combining the supervised and the unsupervised machine learning approaches, this work takes the idea of a two-layer detection scheme which provides an effective classification and clustering of the Exploit kits.

In supervised approach, instances, namely data objects, are assigned with pre-defined category labels. The malicious data (i.e. Exploit kit) and legitimate data are the pre-defined labels in this work. The given new instance is assigned a label based on the likelihood suggested by the training set of labelled instances. This method requires manual labelling of the data; and in unsupervised approach, there is no need for labelled instances or human intervention at any point in the whole process but poses the difficulty to determine the number of clusters in advance. Therefore, significant research efforts have been invested to bridge the gaps.

The combined efforts of these two techniques can reduce the manual efforts that has to be made by security analysts in detecting this network security incident. This research provide an efficient methodology of two-layer detection scheme to classify and cluster the Exploit kits of all different types.

This two layer detection scheme can be used when clients have HTTP logs, which in this research project are BRO HTTP logs. In general, these HTTP logs could also come from a proxy or firewall log. Many companies do have HTTP logs available rather than PCAP files, which is commonly used in network-based IDS. Knowing the type of Exploit kits is useful, because the types of Exploits served by various Exploit kits can help an analyst in assessing the possible damage the Exploit kit could to the end users of a certain network. For example if Exploit D only employs Flash Exploits, while all end users in the attacked network do not have Flash installed, there is little damage to be done by this specific Exploit kit.

Contributions to FOX-IT:

This research was conducted in cooperation with **Fox-IT**, a Netherlands-based security company. **Fox-IT** prevents, solves and mitigates the most serious cyber threats. The Security incidents are now more sophisticated, specific and adaptable. This makes the detection and prevention of such attacks harder. Security analysts are urged to reduce cyber criminals' free time within the network, to reach root causes quicker, and to gain knowledge from recent incidents to reduce future attacks. With the time sensitive nature of security incidents, it is important to have automated tools to help an analyst investigate the network security incidents as it can save time. This research project presents FOX-IT a proof of concept that automates the detection of various types of Exploit kits.

Contributions to Society:

The cyber security landscape is getting complex. The potential cyber threats and the increase in sophistication of attacks urges the security engineer to be technically proficient enough to fight as innovation in the digital world happens much faster than that in the physical world. Thus, this research is intended to address this problem. Moreover, with Internet of Things (IoT) getting popular, there are going to be many security incidents to be examined. The research achievements of this project could help to reduce the manual efforts to be made to detect the Exploit kits.

Contributions to Science:

This research project will contribute optimised methods to detect Exploit kits effectively and to reliably identify various kinds of Exploit kits without demanding expensive time and efforts. The outcomes of this project will serve as the foundation for the future work to be done in this field.

1.2 Limitations and scope

As with any research, there are limitations to be taken into consideration. The malicious data of Exploit kits used to set up the training data does not have the name of the respective Exploit kit family/group each Exploit kit belongs to. The second stage of the work involves clustering of the data which indicates the possible number of different Exploit kits' family/group present in the data. It is usually very difficult to determine the accurate number of clusters.

Following the above considerations, our aim is to demonstrate the effectiveness of the introduced methodology and as a generic example, this work will apply it to the data derived from the post classification stage. Also, when it comes to results of clustering, this work does not focus on deep analysis of the clusters formed, that is the different families/groups of Exploit kits formed, as it beyond the scope of this research due to time constraint.

The aim of this study is to evaluate the performance of the supervised machine learning algorithm and the unsupervised machine learning algorithm used in this work. The work proposes that the unsupervised machine learning can be used to give feedback to the security analyst to aid in the task of identifying the type of Exploit kits used to target the network and/or customers when signature based Intrusion Detection System is not in place. This study will form the basis for developing a hybrid approach of supervised and unsupervised paradigm to the domain of network security by effectively classifying and clustering the Exploit kits.

1.3 Structure of the report

The rest of this report is structured in the following fashion. The prior work related to analysis and detection of Exploit kits is presented in Chapter 2. Thereafter, the main concepts and the terminologies used throughout the report is introduced in Chapter 3. The concepts of machine learning used in this work is also included in this chapter. The intersection of machine learning and network security concepts used to answer the research question is presented in Chapter 4. The methodology of this work is also included in this chapter . The results and significance of this work is presented in Chapter 5. The conclusion of this work is presented in Chapter 6.

Chapter 2 Related Work

Before going through this work, it is important to take a look at the existing work related to detection of Exploit kits, specifically paying attention to the most advanced techniques and strategies used by security researchers. There are basically two techniques, namely static analysis and dynamic analysis, to detect malicious activities on the web pages. The static analysis uses some static features and the source code of the web pages to characterize the malicious payload. The dynamic analysis focuses on analysis of the behaviour when a web page is loaded and executed in a browsing environment. A strategy common to both static and dynamic analysis is that they extract features to capture the patterns that can identify malicious payloads in web pages, based on which classification method is decided, generally using machine learning techniques.

Exploit kits are not an easy threat and cannot be tackled easily as they have different software components. Hence, this requires a diversified approach to detect and defend. Only by combining different types of solutions one can build an effective detection system which can identify Exploit kits and prevent the Exploitation of vulnerabilities.

In recent years, many behaviour-based strategies based on sandboxes and script analyzers that execute scripts and code snippets in a protected environment are used for the detection of Exploit kits. There are also web-based solutions to analyse URLs that is usually implemented in browsers. They help add these compromised websites to blacklists to prevent users from visiting these web pages. Then there are file-based solutions which can help in the post-infection stage, that is after the malicious payload is delivered to the victim's system. They look for the malicious executable in the file system and block the same to not to allow it to run on the victim's s system anymore.

2.1 Analysis of Exploit kits

This further section presents work related to analysis of Exploit kits.

2.1.1 PExy

De Maio et al. [44] provide analysis of the source code of multiple Exploit kits.

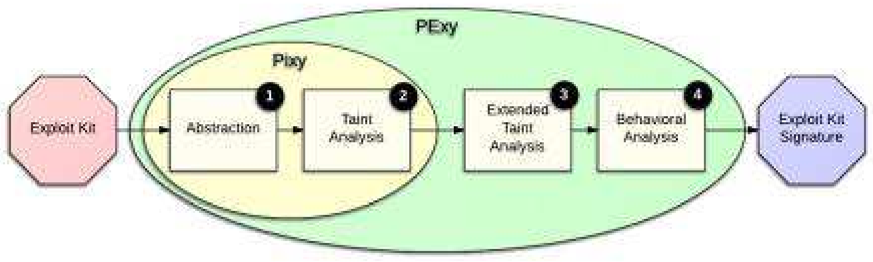


Figure 2.1: Architecture of PExy . Adapted from [44])

PExy presents detailed overview of the server-side part of the Exploit kits, performing static analysis, De Maio et al. have discovered the important coding practice followed, interesting detection mitigation techniques and the methods used by the Exploit-kit authors to decide on the type of Exploit kits to be served.

The static analysis of the source code to extract as many Exploits as possible aims to provide the entire set of checks that an Exploit kit performs to decide on the right kind of Exploit to deliver to the victim. The difficult part is to generate the configuration that can extract all the possible Exploits associated with a kit. The study also presents the common practices followed by Exploit kit developers and shows that the signs of similarities are found among the different families/kinds of Exploit kits. when some detection system identifies an Exploit sent by an Exploit kit, the URL is normally flagged as malicious. Since each Exploit kit has several Exploits, flagging a web site as malicious based on this detection is not enough. This is because one can easily miss out on updating signatures present in the antivirus for all the Exploits present in the detected Exploit kit. The PExy uses the source code of an Exploit kit to find and narrow down set of URL parameters and user agents to extract all the possible Exploits from a particular Exploit kit. The set of parameters extracted from an Exploit kit also work very well with an Exploit kit from another family since many Exploit kits from different families share a certain amount of code in common. The figure 2.1 presents the architecture of PExy. The goals of this research are 1) to find HTTP parameters dependant branches in the code 2)to find the set of parameters that these branches depend on. This can characterize an Exploit kit with a collection of HTTP parameters, and a set of values for those parameters that can cover the largest portion of source code possible. The results of this study show that in spite of complex analysis on the server-side components of an Exploit kit, it is not a simple task to trigger a complete Exploit kit execution (involving all the Exploits) even with big set of URL parameters.

2.1.2 Anatomy of Exploit Kits, Preliminary Analysis of Exploit Kits as Software Artefacts

Kotov et al. [57] present a comprehensive analysis performed on the source code of more than 30 different Exploit kits. The surprising revelation from this research is that the number of vulnerabilities targeted by the tested Exploit Kit is very limited, and modes of attack are in fact very simple. The strength of the Exploit kits lies in the possibility of choosing suitable Exploits and way to monitor traffic. The analysis is based on source code obtained from more than 30 different Exploit kits. They focused their investigation on functional features of the Exploit kits, their delivery mechanisms, the evasion techniques used by them, code being re-used, and code protection mechanisms used. This leads to a conclusion that Exploit kit developers' primary concern is to improve the experience of their customers, and the functionalities common to all Exploit kits are user-agent validation, code obfuscation, IP blocking, use of commercial code protection tools and a management board for the customization of the kit.

2.1.3 MalwareLab

Allodi et al. [58] Present MalwareLab, an isolated environment where Exploit kits were installed and analysed. Their experimental approach examined the features and functionalities of these different Exploit kits. They tested the strength of these Exploit kits in terms of their resilience against change of software in time and the results showed for how long they could effectively perform. The Exploit kits collected were from the period of 2007 to 2011.

Their first experiment in this direction failed because of the fact that the different Exploits in an Exploit kit were engineered well enough to work regardless of the state of the memory on the victim's machine. Figure 2.2 shows the results for the resilience of the Exploits kits against software configuration updates. This graph basically shows three types of Exploit kits. The first group contains Exploit kits that show similar trends in infecting effectiveness. These were poor quality Exploit kits. The curves are identical for these Exploit kits which indicate that these kits have the same Exploits. The second group of Exploit kits exhibit significantly high resiliency. In the figure 2.2, Shaman exhibits better performance in comparison to Eleonore. The third group contains Exploit kits that were designed to be effective for a specific period of time. In the figure 2.2, Eleonore shows high infection rate till 2008 and drops significantly in the following years. These are time-specific Exploit kits. Through the experiment, Allodi et al. have demonstrated that some Exploit kit developers put in great efforts in writing resilient code and this can be clearly seen in high quality Exploit kits. The higher quality kits are effective for a longer period of time with lower peaks and low quality kits infect more number of victims in a short period of time but have lower resilience. This work demonstrates the performance of Exploit kits over time in terms of "quality".

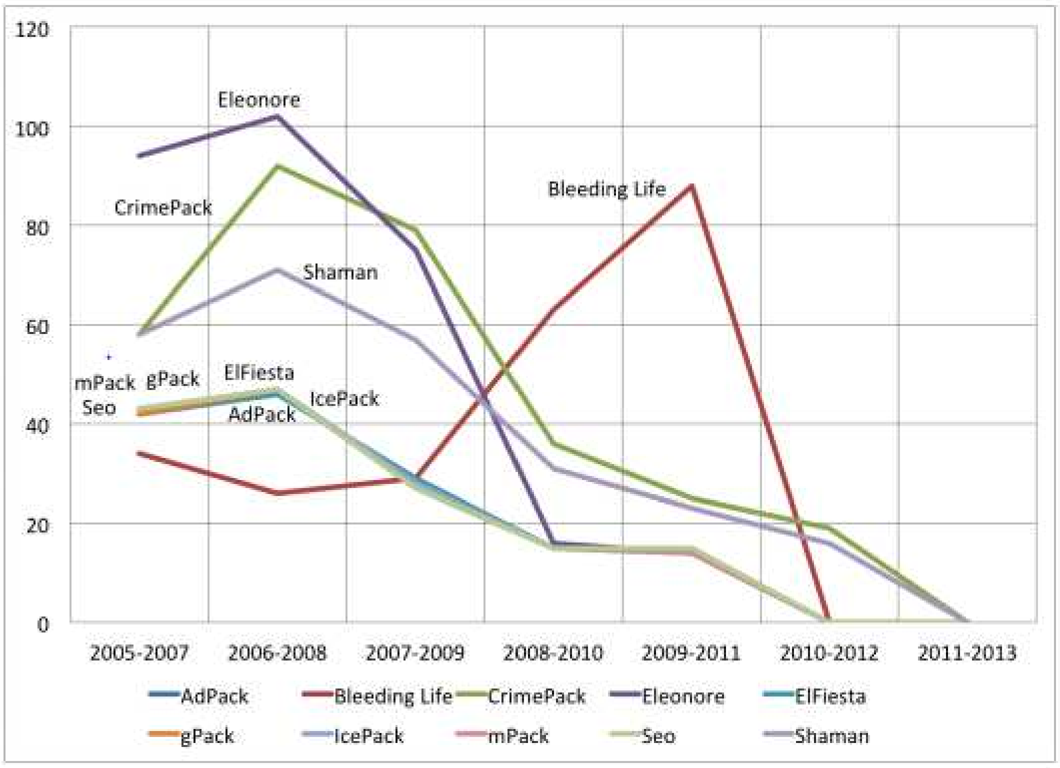


Figure 2.2:Infections per time window per Exploit kit. Adapted from [58]

2.1.4 Manufacturing Compromise: The Emergence of Exploit-as-a Service

Grier et al. [59] present the concept of Exploit-as-a-service which is one of the major threats for internet security. This research examines the topics such as the amount of traffic generated in the malicious website after the malware is successfully installed on the victim's machine, possibility to support the kit with custom malware, overview of the prevalence and statistics of malware binaries. These malware binaries are divided into families to portray the characteristics of each family along with the respective preferred channel used for their delivery. This research has collected data of 77.000 malicious URLs provided by Google Safe Browsing and a crowd-sourced blacklist. This work considers DNS traffic from real network, provides statistics on the popularity of the malware families, and lifetime and popularity of the domains that are part of drive-by-download infections. This work describes the overview of the malware landscape with its various delivery channels such as droppers, email, drivebys, and warez. This work shows that due to the short lived domains hosting Exploit kits and cloaking techniques, the Crawler-based detection systems are not very effective. Therefore, in-browser defensive strategies prove to be an efficient approach as it can mitigate the difficulties about locating and decloaking.

2.2 Detection of Exploit kits

This further section presents work related to detection of Exploit kits.

2.2.1 Kizzle

Stock et al. [43] present the first prevention technique to find Exploits kits. This is named Kizzle which can recognise the JavaScript code used by Exploit kits. With their analysis of the JavaScript code used by Exploit kits, they found out that the infection code obtained after deobfuscation does not vary much from one Exploit kit instance to another. This fact is used to build a detection system which is resilient to superficial changes in JavaScript code. Kizzle can detect Exploit kits delivering JavaScript code with signatures which can reduce the manual work of a security analyst. This work also shows detailed description of evolution of Exploit kits with time. This work analyses four families of Exploit kits. This work is restricted to JavaScript code to derive signatures by using unsupervised learning method for different families of Exploit kits. Kizzle generates anti-virus signatures and can generate new signatures within hours to find Exploit kits. Figure 2.3 shows the architecture of KIZZLE. This malware signature compiler has false-positive rate under 0.03% and false-negative rate under 5%.

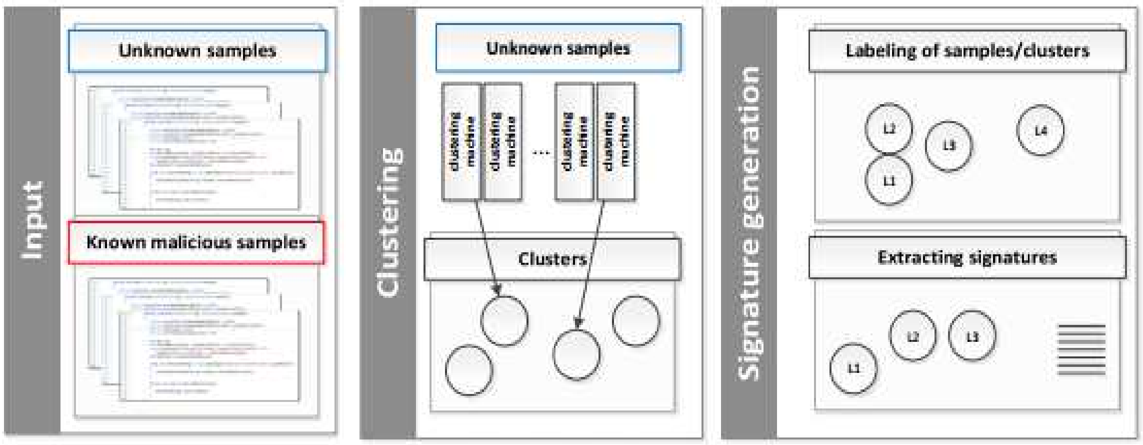


Figure 2.3: Architecture of Kizzle. Adapted from [43]

2.2.2 EkHunter

Eshete et al. [60] present an effective counter-offensive model against the threat of Exploit kits after surveying 30 real-world Exploit kits. They adopted an offensive approach to find vulnerabilities in Exploit kit source code. They obtained good results with their approach as they found more than 180 vulnerabilities in the Exploit kits surveyed. Their experiment showed that Exploit kits did not implement protection mechanisms to resist counter-offensive attacks. Eshete et al. have designed a tool, EkHunter which can detect vulnerabilities in Exploit kit's code. This information can be used to find the identity of the kit user and to compromise an Exploit kit. Their strategy can accelerate the process of detecting and taking down the botnets which are big threat to Internet security today. They explain the problem of kit infiltration and the steps that led to the deployment of the vulnerability analysis system. They also discuss the ethical and legal implications of their approach.

2.2.3 WebWinnow

Eshete et al. [11] present WebWinnow to detect URLs hosted by Exploit kits. They have built a detection system based on features extracted from over 40 Exploit kits. It uses machine learning techniques to train a classifier based on both the attack-centric behaviour and the self-defence behaviour exhibited by the Exploit kits. Attack-centric features are those features involved in the actual infection, such as fingerprinting, obfuscation, etc. The features such as evasion techniques and IP blocking are the features related to self-defence behaviour. when an unknown URL is presented to the detection system, WebWinnow analyses the URL under test to check if it matches with the pattern of known Exploit kits' URLs. The performance of the classifier has a true positive rate of about 99% on a separate testing sets except for the Random Tree classifier.

2.2.4 Revolver

Kapravelos et al. [61] present a novel approach to automatically identify evasive behaviour in JavaScript code with Revolver. Revolver is a very efficient tool for evasion detection which can compare the code under test with a large database of JavaScript programs in terms of their similarities and differences to detect the evasive code. Revolver dynamically analyzes the code, executing and monitoring it to examine any code generated at runtime as well as static code. The analysis made by Revolver consists of several phases. During the first phase, Revolver compares the different scripts and detects syntactic-level differences in each cluster. In the next phase, Revolver then figures out the effects of the differences on page execution. In the final phase, it tries to detect presence of evasion routines. The main contribution of their work is an efficient code similarity detection technique and an evasive code detection engine. The code similarity detection technique is based on Abstract Syntax Tree analysis and dynamic analysis techniques that are resilient to polymorphism, dynamic code generation and obfuscation. The evasive code detection engine is based on different classification of similar scripts.

2.2.5 JSAND

Cova et al. [62] present a detection approach that employs both anomaly detection and emulation techniques to uncloak any hidden behaviour of the JavaScript code. This work focuses on malicious applications of the JavaScript code such as attacks on browsers and their plug-ins that can enable drive-by-download attack. They present a tool called JSAND (JavaScript Anomaly-based Analysis and Detection). This has been tested over 140,000 web pages and can be used to analyze a suspicious URL and files. This tool can detect zero-day drive-by downloads as it has both anomaly detection and machine learning techniques which can be used to compare the behaviour of the emulated JavaScript code under test with a learned model of benign JavaScript code execution. The detection system then generates a set of signatures that can be implemented in a signature-based system.

The system uses an instrumented browser to visit a webpage to analyze the events that would be triggered during the HTML elements interpretation and JavaScript code execution. The system extracts one or more features for each event and these features are classified as either anomalous or non-anomalous using anomaly detection techniques. The system also provides the attack's details such as Exploits used.

Cova et al. have identified ten features that efficiently identify drive-by download attacks with their intrinsic characteristics. This work also generates signatures that can deobfuscate obfuscated JavaScript code.

2.2.6 Cujo

Rieck et al. [63] present a system, CUJO which is embedded in a web proxy to efficiently detect drive-by-downloads with emphasis on malicious JavaScript. Rieck et al. have made use of both static and dynamic analysis to examine the JavaScript code. They have built and trained a binary classifier with JavaScript features to detect drive-by-downloads using Support Vector machines. This classifier provides a detection rate of 95% with an average analysis time of 0.5 seconds per a web page and low false alarms.

2.2.7 Hidost

Šrndić et al. [45] provide the first static machine-learning-based malware detection system, Hidost, which operates on multiple file formats such as PDF and SWF (Flash). The results show that Hidost has outperformed most of the antivirus engines deployed by the website virus total in detecting highest number of malicious PDF files.

2.2.8 Learning to detect and classify malicious executables in the wild

Kolter et al. [46] provide a method to detect and classify malicious executables using machine learning and data mining. They set up a training set with benign and malicious executables using n-grams to characterise the features. They have used Naive Bayes, Decision tree, and support vector machines algorithms. They have achieved the accuracy of 97.11 % with Naive Bayes algorithm.

2.2.9 Unsupervised anomaly-based malware detection using hardware features

Tang et al. [47] have presented a detector, which works based on unsupervised machine learning to find malware Exploits. They have carefully selected micro architectural features to build baseline models of benign program execution and then use the same profiles to detect deviations that occur due to malware Exploitation. The approach works better along with signature-based detectors to improve the security as it can serve as complementary to signature based detectors.

2.2.10 Next-generation of Exploit kit detection by building simulated obfuscators

Luo et al. [1] provide a rather structured approach towards Exploit kit detection, but it is case specific (JavaScript samples of 5 Exploit kit families). They have aimed to minimize the manual efforts when reproducing the obfuscator (reverse engineering the obfuscator from obfuscated samples) based on JavaScript normalization technique and hierarchical cluster algorithm. They have opted for agglomerative approach to measure the proper threshold to best classify samples depicting the timeline of Exploit kits from the perspective of the evolution of its obfuscator.

There have been many different efforts to provide an effective approach for the automation of detecting different kinds of Exploit kits. The understanding about identifying the different types of Exploit kits present in the network traffic is the key. This in turn can be used to generalise which can be used for the automation.

Unlike the others' work that has opted for either supervised learning technique or unsupervised learning technique and not both in combination, this work makes use of both. This work relies on the features that are extracted from HTTP log file to characterize the behaviour of the Exploit kits. These derived features are used for the detection. In the first layer, this work uses Multinomial Naive Bayes algorithm to detect the Exploit kits as Multinomial Naive Bayes is a fast algorithm with low storage requirement. This is a supervised machine learning algorithm and is robust to irrelevant features as they tend to cancel each other out without affecting the results. This algorithm works best when many equally important features are to be considered as it is the case in this work. It provides a good dependable baseline for classification involving text data.

This work opts for clustering of the data in the next layer of the work. Clustering is the type of unsupervised learning which groups the data points that share common characteristics. To measure the similarity between the data points, the distance measurement is used. This algorithm is used to group Exploit kits belonging to different families/groups.

Chapter 3 Background and main concepts

This chapter presents the main concepts and relevant terminologies to this research work.

3.1 Bro Network Security monitor logs

This section mentions about the log files that are considered as input for this research.

Network security monitoring provides us with a picture of the state of a network via security log files. However, manual analysis of these log files can be labour intensive. One of the potential solutions is to structure the data from these logs and to mine the most valuable hidden information.

Although there are many good Intrusion Detection Systems (IDS) available on the market, most of them are proprietary products and provide very limited room for conducting academic research. Thus, IDS products from open-source communities are better fit into my research project. Open-source IDSes, such as Bro or SNORT [14], could be potential platforms to carry out the research.

Bro IDS has features such as the ability to run in high speed environments, use sophisticated signatures, and can be flexibly configured, making it the more suitable choice in comparison with SNORT. Bro is also polished enough with more than 15 years of research, has bridged a gap between academia and industry, and has been continuously supported. Bro is in use in many scientific environments and widely used by major universities, research labs, supercomputing centres, and open-science communities making it the best choice for my project [14].

Bro is an open source Unix-based network monitoring framework. It performs at high speed, can handle large volume monitoring, no packet drops, real time notification and its logging mechanism is separated from the alerting policies. It provides network security monitor, traffic inspection, attack detection, log recording, distributed analysis, full programmability and several other features which are useful for security analysts. Bro creates high level flow of the events and stores them in multiple separate files. The data can be parsed and mined for the crucial information about the malicious activities in the network monitored. These logs include complete record of every connection seen along with application-layer transcripts such as all HTTP sessions with their requested URIs, key headers, MIME types, server responses, DNS requests with replies, SSL certificates, key content of SMTP sessions, and much more. This can display the network's activity in detail with the flexibility for production deployment. [15]

3.2 Exploit kits

Exploit kits are a serious cyber threat today as they are responsible for big percentage of malware infections worldwide. [65] Exploit Kits are powerful cyber weapons that carry out a malicious campaign targeting innocent victims to leverage them for various methods of profit. Everyone from inexperienced hackers to sophisticated cybercriminals can use the Exploit kits to spread malware through Exploitation of client-side vulnerabilities, usually targeting browsers and plug-ins that a website can access through the browser exposed APIs. Exploit kits have pre written code which make use of outdated software applications running on a computer. The drive-by-downloads, spam and denial-of-service are some of the examples of attacks that are launched through Exploit Kits. They have been around in the cyber world for the past 10 years at least. They have evolved through the years and have become modular and sophisticated enough with capabilities to evade detection, thwart analysis and deliver suitable Exploit kits.

3.1.1 Background and origin:

The history of the Exploit kit attacks dates back to the creation of WebAttacker kit [1] found in Russian underground market in 2006. WebAttacker kit Exploits vulnerabilities found in Microsoft™ Windows®, Mozilla Firefox®, and Java applications through spam and compromised websites.

Later 2006, another Exploit kit, named Mpack, [66] was spotted. It is more sophisticated than WebAttacker kit in terms of information gathering. Mpack provides detailed statistics of the victim including its location. This kit consists of a collection of PHP scripts that targets security holes in programs like Mozilla Firefox or Apple Quicktime [1].

Then several Exploit kits were released in 2007, namely, NeoSploit, Phoenix, Tornado, and Armitage Exploit kits. They were followed by more Exploit kits, namely Fiesta, AdPack, and FirePack Exploit kits in 2008. Then the Blackhole Exploit Kit which surfaced in 2010 was widely used Exploit kit in 2012 and 2013, and posed a major threat to users. Then there were several other Exploit kits, namely, Fiesta, Nuclear, SweetOrange, Styx, FlashPack, Neutrino, Magnitude, Angler, and Rig which were still in use in 2014. [1]

The ability of an Exploit kits to launch attacks depends on the vulnerabilities they can Exploit and these vulnerabilities have be updated in timely fashion. If a vulnerability is new and not been patched by a user, this could be of great value to an attacker as he can spread malware more efficiently. The key for an attacker is to constantly update their Exploit kits with new vulnerabilities to Exploit.

Timeline:

Year	***Exploit Kits***
2006	MPack and WebAttacker kit
2007	Armitage Exploit Pack, IcePack Exploit Kit, NeoSploit Exploit Kit 1.0, Phoenix Exploit Kit, Tornado Exploit Kit
2008	AdPack, Fiesta Exploit Kit, FirePack Exploit Kit
2009	CrimePack 1.0, Eleonore Exploit Kit, Fragus Exploit Kit, Just Exploit Kit, Liberty Exploit Kit, Lucky Sploit, MyPoly Sploit, Neon Exploit System, SPack, Siberia Exploit Pack, Unique Sploits Exploit Pack, Yes Exploit Kit 1.0/2.0
2010	Blackhole Exploit Kit 1.0, Bleeding Life Exploit Kit 1.0/2.0, Dragon Pack, Nuclear Exploit Kit 1.0, Papka Exploit Pack, SEO Sploit Pack
2011	Best Pack G01, Pack Exploit Kit, Katrin Exploit Pack, OpenSource Exploit Kit, Sava Exploit Kit
2012	Alpha Pack, CK Exploit Kit, Cool Exploit Kit, CrimeBoss Exploit Kit ,CritXPack, GrandSoft Exploit Kit, Impact Exploit Kit, KaiXin Exploit Pack, Kein Exploit Pack, NucSoft Exploit Pack, ProPack, RedKit Exploit Kit, Sakura Exploit Kit, Serenity Exploit Pack, Sibhost/Glazunov Exploit Kit, Styx Exploit Kit 2.0, SweetOrange Exploit Kit, Techno XPack Yang Pack, ZhiZhu Exploit Kit

Table 3.1: Timeline of Exploit kits

The table 3.1 shows the timeline of different Exploit kits from 2006 to 2012. [46]

3.1.2 How an Exploit kit works

An Exploit kit is a tool which detects and Exploits the vulnerabilities presenting in the system to spread malware, such as banking malware, ransomware, etc. This action can be described in three steps, namely scanning the system for vulnerabilities, Exploiting the discovered vulnerabilities by injecting malicious code into the system and executing the malicious code. [3]

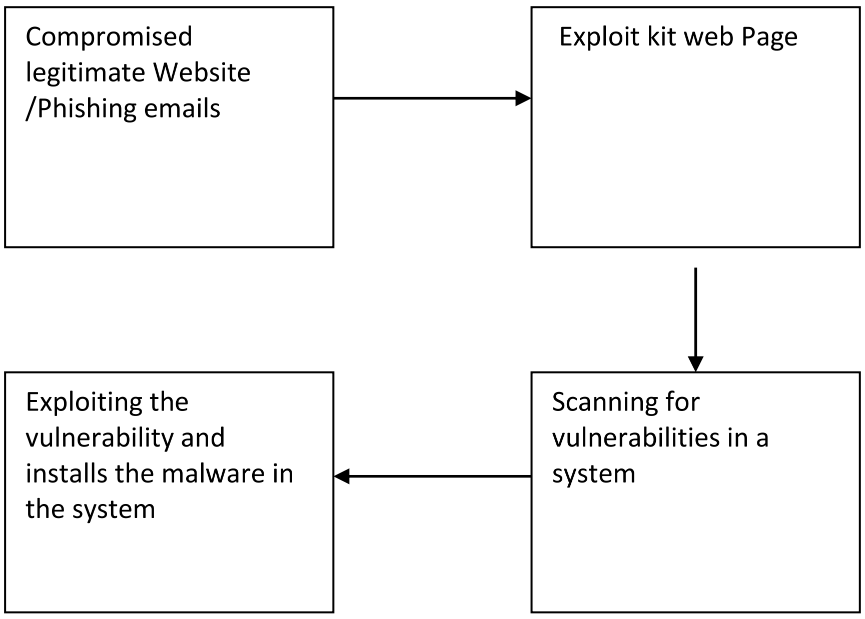

Compromised legitimate Website /Phishing emails
Exploit kit web Page
Scanning for vulnerabilities in a system
Exploiting the vulnerability and installs the malware in the system

Figure 3.1: Overview of a typical Exploit kit attack

When a user visits a legitimate website that has been compromised, the user's browser is redirected to a malicious website that hosts an Exploit kit. This is called **a drive-by attack**. [64] Then the Exploit kit scans the browser to look for vulnerabilities/security holes, such as out-of-date browser plug-ins, and delivers the payload, which is typically a malicious software. Finally, the malware proceeds to function as it is programmed to. This can be seen in Figure 3.1.

In a typical Exploit kit attack scenario, contact is the first step of the infection. During this step, an attacker tries to lure users to access the link of an Exploit kit server. This is often done through spammed emails, compromised websites or malicious advertisements [3]. In the case of spammed emails, an attacker usually tries to trick the users into clicking the link through social engineering. In the case of malicious advertisements, the advertisements redirects the users of website to Exploit kit server.

In the second step, an attacker screens the victims using a traffic direct system based on certain set criteria. This involves the use of traffic distribution system (TDS), whose primary function is to intelligently route web traffic that arrives at the TDS. The redirection of traffic is not necessarily monitored by an attacker but by the person who sells the same in Underground markets. An attacker can specify the target and this is achieved by filtering out victims based on the mentioned requirements. [3]

The next steps would be Exploitation and infection. Once the attacker is successful in getting a victim past steps 1 and 2, a victim will be in Exploit kit's landing page. The landing page would profile the environment of a victim, and to determine which vulnerabilities can be best used in the attack [3].

The page will send requests to the Exploit kit server to download the Exploit files based on the vulnerable application identified. The usual targeted vulnerabilities by Exploit kits are those found in Web browsers, Java, Adobe Flash Player, and Adobe Acrobat and Reader. The attacker then delivers and executes the malware in the victim’s environment after successfully exploiting the vulnerability. There are many types of malware that are associated with Exploit kit attacks, such as ransomware, online banking malware, etc. [3].

The Exploit kits can now obfuscate the payload with encryption which makes detection almost impossible as the payload will be later decrypted in victim's machine in memory by shellcode after it has been successfully downloaded in victim's machine. The Exploit kits obfuscate the Exploit file using different techniques to avoid detection. For example, jar files can be compressed using Pac200 [3] which is a compacting archive format, angler Exploit kit makes use of the same for its java Exploits to avoid the detection. Another example would be use of the tool called DoSWF [3] by Exploit kits, namely FlashPack [3] and Magnitude, to hide Flash player Exploit files [1].

Exploit kits do far more sophisticated things like using techniques such as domain shadowing which is compromising registered accounts, hosting the malicious activities under sub domains for those legitimate domains, 302 cushioning which is use of HTTP 302 redirects to use as another tier of redirection (another way for them to hide in the noise), custom encrypted payloads where the victim system is posting out data, payload is custom encrypted to every single victim [3].

3.3 Machine Learning

Machine learning has grown from a field of statistics to a broad discipline that has produced fundamental statistical-computational theories of learning processes and has designed various learning algorithms that can be used in commercial systems for applications like speech recognition, computer vision, medical diagnosis, bio-surveillance, recommender systems, sales prediction, fraud detection, robot control and so on [4]. These algorithms can learn from the data itself regardless of the volume and complexity of the data without relying on the rules-based programming for various reasons like tasks are too complex to be programmed or tasks are beyond Human Capabilities or tasks require adaptivity. Machine learning uses real data to train a model for solve various problems, such as classification, clustering, prediction, anomaly detection, etc. The figure 3.2 shows the general framework for machine learning approach.

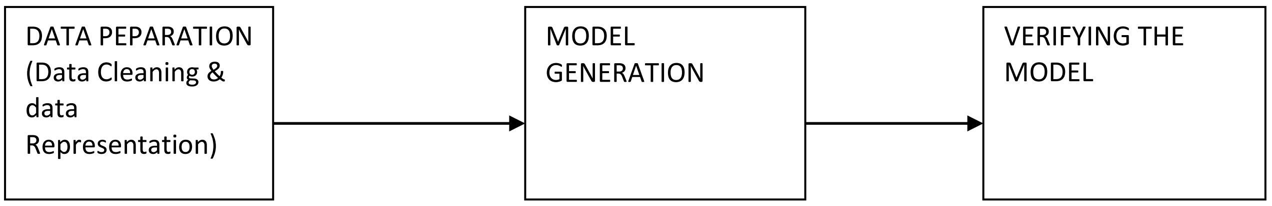


Figure 3.2: A general framework for the machine learning approach

Data pre-processing:

The different types of data available are binary/discrete, continuous values, time series, natural language text, images, sound and so on. Data pre-processing involves data cleaning and representation. Real-life data is usually noisy. The different possible reasons are missing data, out of sync fields, misspellings, special values, spikes, and data from different sources (old/new) with slightly different meaning, inconsistent data or irrelevant data. [7] This step is done to present the data in a form that could be used to extract information which helps in understanding the data with its importance in the context of relevance and details. The exact choice of learning method and the choice of representation of features are very important. With the wrong representation, no method would be successful and a good representation guarantees success with almost any method chosen. The process of finding a suitable representation requires much domain knowledge and problem understanding.

Model generation:

The different types of machine learning algorithms can be used to build and train a model based on the requirement specifications. This model is then tested with the some part of the data set used for training using methods such as k-fold validation method for example. This can be repeated multiple times to check the accuracy of the algorithm. The different machine learning methods include

- ✓ Case based methods
  This is based on the fact that similar patterns belong to the same class. This is generally easy to train (every pattern is saved but sometimes similar patterns may not be identical), but takes longer during recall, to find the similar patterns. Model size increases with the number of seen examples. This requires specification of a distance measure. Examples of this are Table lookup, Nearest neighbour, k-Nearest neighbour. [49]

- ✓ Logical Inference
  This is based on constructing logical expressions that characterizes the classes. This typically considers one feature at a time. Examples of this are Inductive logic, Decision trees, Rule based systems [50].

- ✓ Artificial Neural Networks
  This is inspired by the neural structure of the brain. The neural units are connected with others and each connection is weighted. The weights are adjusted to produce the best mapping. The 'Deep' architectures require much data to train. Examples of this are multilayer perceptron, Self-Organizing Maps, Bolzmann machines, deep neural networks [51].

- ✓ Statistical methods
  There are large number of methods, from simple to complex. The fundamental idea is to calculate the probability of each class(c) given a feature vector(x), $P(c|x)$. The Parametric versus nonparametric methods are considered depending on whether the forms of the class distributions are known or not. Examples of this are Naive Bayes, Mixture models, Hidden Markov models, Bayesian networks, MCMC, Kernel density estimators, Particle filters. [52]

- ✓ Heuristic search. Examples of this are Genetic algorithms, Reinforcement learning, simulated annealing, Minimum Description Length. [53]

Validation of the model:

The created model is validated using previously unseen data samples, which are also known as test data. "Validation" demonstrates how the model performs while being put into practise.

3.2.1 Supervised learning

Supervised machine learning algorithms learn from supplied data instances to produce general hypotheses, which then make predictions about unknown data instances. In the predictive or supervised learning approach, it learns a function from a set of labelled data, which is also known as training data, containing a set of inputs X_i mapped to set of outputs (labels/classes) Y_i, The learnt function is then used to map a new given data sample (also known as test data) to its corresponding label/class.

An input, X_i, from the training set could be a complex structured object, such as an image, a sentence, an email message, a time series, a molecular shape, a graph, etc. Its corresponding output, Y_i, could in principle be a categorical or a nominal variable [7]. The goal of supervised learning is to build an accurate model describing the distributions of particular categories of data [7].

Since supervised algorithms assume that the labels of data items are known, the need for knowing the category structure in advance and generation of correctly labelled training set can be challenging at times in case of large and dynamic datasets.

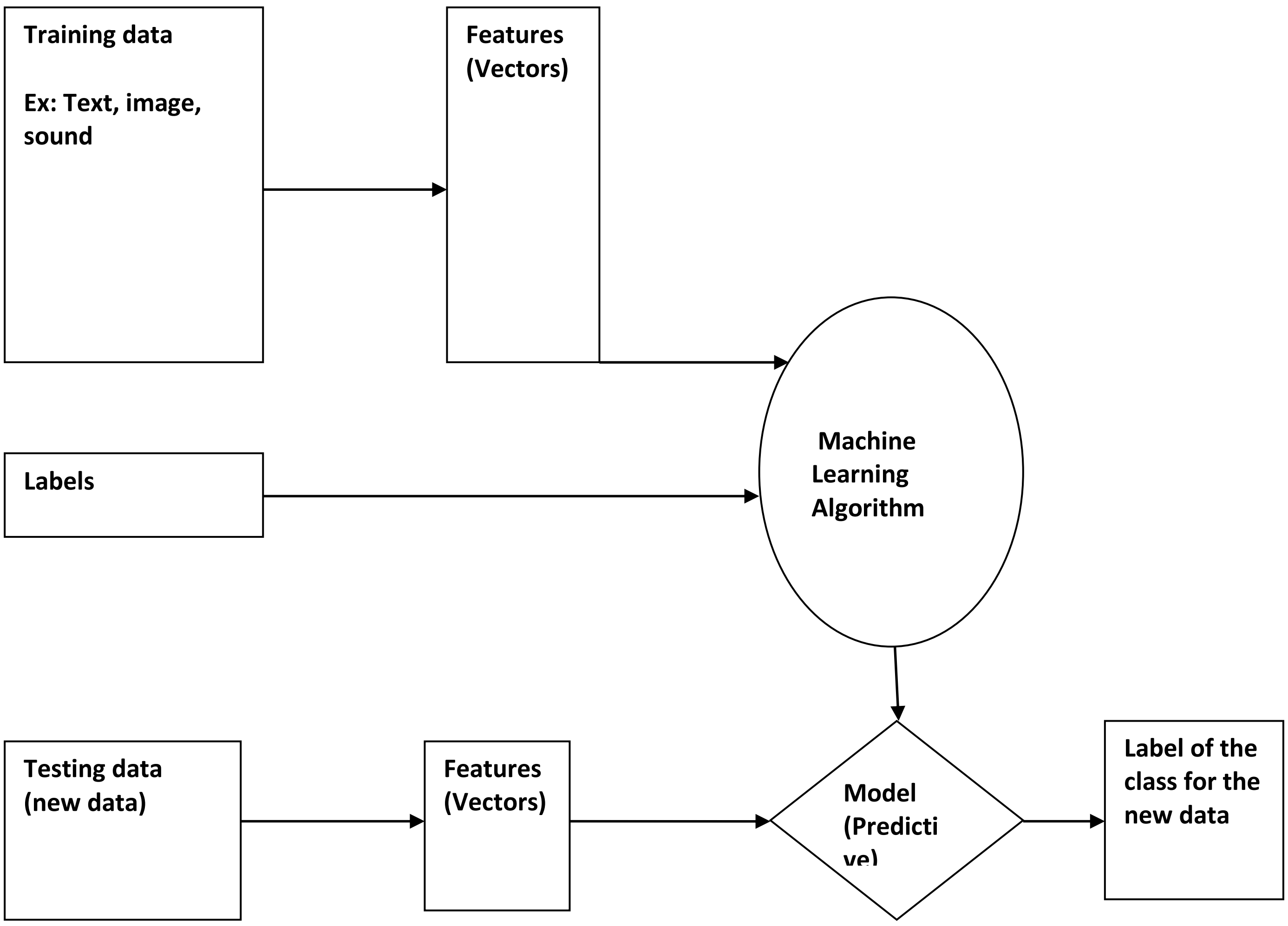


Figure 3.3: A general framework for the Supervised Machine Learning approach

Figure 3.3 presents a general framework for the supervised machine learning approach. The data in the training set are labelled with pre-defined classes. It guides the machine learning algorithm to find the corresponding label for the given training data. Every data instance in training set and test set used by machine learning algorithms is represented using the same set of features. These features can be continuous, categorical or binary. The trained classifier can assign class labels to test data instances. [5]

In supervised approach, instances (data) are assigned with pre-defined category labels based on the likelihood suggested by a training set of labelled but requires the manual labelling of data.

3.2.2 Unsupervised learning

The second main type of machine learning is the descriptive or unsupervised learning approach. Here we are only given inputs, and the goal is to find the pattern that exists in the data which can be an answer to a question. This is called knowledge discovery. [7]

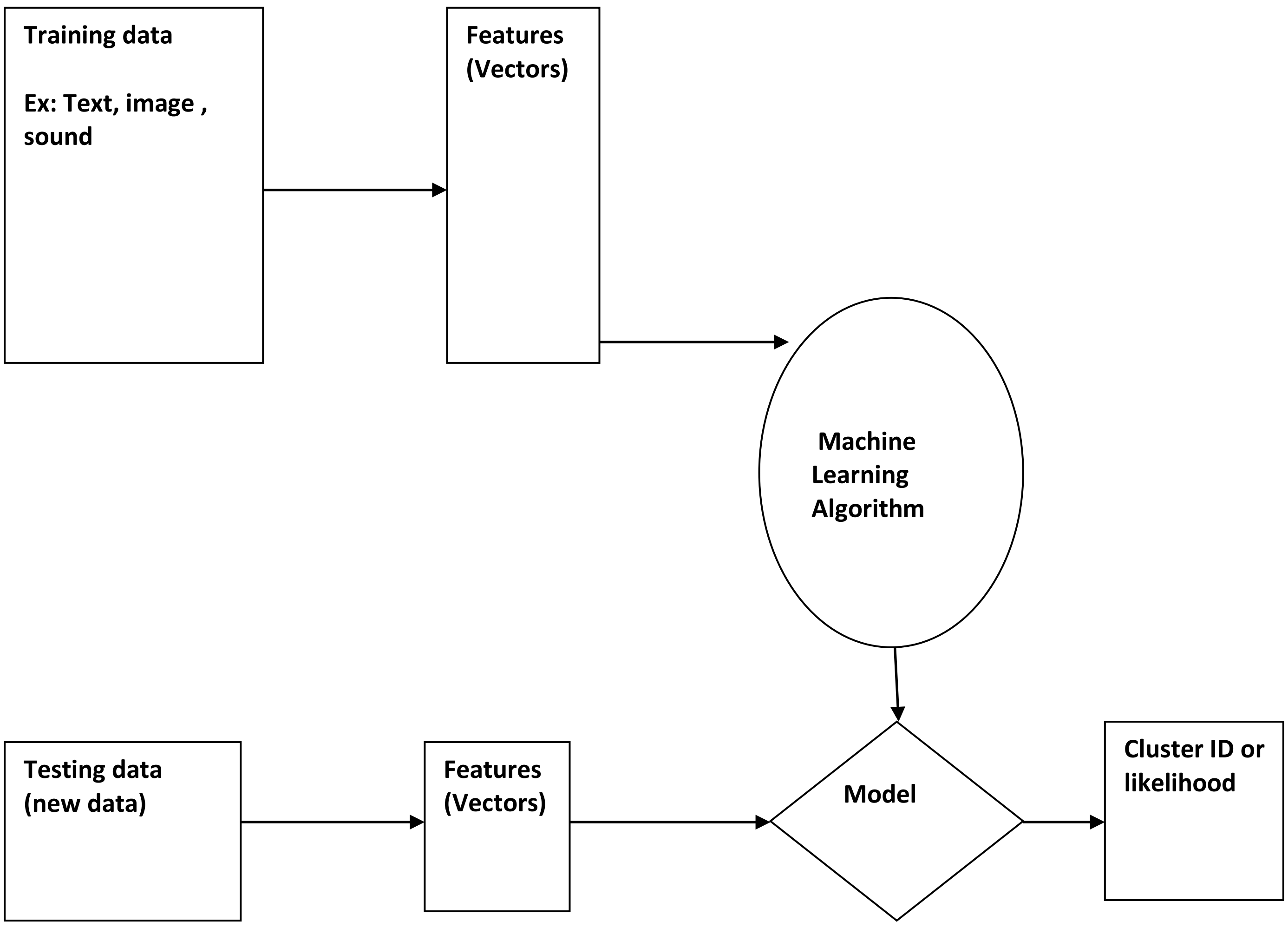


Figure 3.4: A general framework for the Unsupervised Machine Learning approach

Figure 3.4 gives a general framework for the Unsupervised Machine Learning approach. Here, depending on the algorithm used, a pattern is found in the given data which is used to classify new data presented.

In unsupervised approach, there is no need for labelled instances or human intervention at any point of the whole process but at the same time poses the difficulty such as determining the number of clusters for clustering in advance.

3.4 Multinomial Naive Bayes

This section mentions about the supervised machine learning algorithm used in layer 1 of this work.

For an input instance X, the Naive Bayes classifier determines the probability of *X* belonging to each different class given and then one with the highest probability is assigned to the instance. The features are assumed to be mutually independent in Naive Bayes and probability of class *Ci* is calculated using Bayes' Theorem.

$$P(C_i|\vec{X}) = \frac{P(X_1|C_i)P(X_2|C_i)P(X_3|C_i)\ldots\ldots\ldots P(X_n|C_i).P(C_i)}{P(\vec{X})} \qquad \textbf{(3.1)}$$

Equation 3.1 provides mathematical formula to calculate the probability of class C_i given *X*. The Naive Bayes (NB) classifier uses the joint probabilities of terms and respective categories to estimate the probabilities of categories given a test data making it a is a probabilistic model. [36] This is sometimes called naive because of the assumption that all terms involved in the data are conditionally independent of each other given a category. Due to this, the parameters for each term can be learned separately and this speeds the computation operations compared to other classifiers. There are two common event models for NB which involves text classification, multinomial model and multivariate Bernoulli model. In these models classification of test data is performed by applying the Bayes' rule [18]

$$\boldsymbol{P(c_j|d_i)} = \frac{\boldsymbol{P(c_j)\cdot P(d_i|c_j)}}{\boldsymbol{P(d_i)}} \qquad \textbf{(3.2)}$$

where d_i is a test data and c_j is a category. The equation 3.2 gives mathematical formula for Naive Bayes. The posterior probability of each category c_j given the test data d_i, i.e. $P(c_j|di)$, is calculated and d_i, data is assigned to the category with the highest probability. The $P(c_j)$ and $P(d_i|c_j)$ are estimated from the training set data to calculate $P(c_j|d_i)$. The $P(d_i)$ can be eliminated from the computation as it is same for each category. The category prior probability, $P(c_j)$, can be estimated as follows:

$$\frac{\boldsymbol{\hat{P}(c_j)=\sum_{i=1}^{N} y(d_i,c_j)}}{\boldsymbol{N}} \qquad \textbf{(3.3)}$$

where, N is number of training data and y(d_i|c_j) is defined as follows:

$$y(d_i, c_j) = \begin{cases} 1\ if\ d_i \in c_j \\ 0\ otherwise \end{cases} \tag{3.4}$$

The equation 3.3 and 3.4 provides mathematical formula to calculate prior probability for each category. The prior probability of each category c_j is estimated by the fraction of data in the training set belonging to that category c_j . The multinomial model and multivariate Bernoulli model use different ways to estimate $P(d_i \mid c_j)$ .The multinomial model is explained below:

In the multinomial model a document d_i is an ordered sequence of term events, drawn from the term space T. The probability of each term event does not depend on position in the document, term's context and length of the document as it is assumed to be independent in this model. So, each document d_i is drawn from a multinomial distribution of terms with number of independent trials equal to the length of *di*. The Equation 3.5 provides the mathematical formula for probability of document d_i belonging to a category in multinomial model

The probability of a document di given its category cj can be approximated as:

$$P(d_i|c_j) \approx \prod_{i=1}^{|d_i|} P(t_i, c_j) \tag{3.5}$$

where $|d_i|$ is the number of terms in data d_i; and t_i is the i_{th} term occurring in document d_i. Thus the estimation of $P(d_i|c_j)$ is reduced to estimating each $P(t_i|c_j)$ independently. The following Bayesian estimate is used for $P(t_i|c_j)$:

$$\hat{P}(d_i|c_j) = \frac{1 + TF(t_i, c_j)}{|T| + \sum_{t_k \in T} TF(t_k, c_j)} \tag{3.6}$$

The Equation 3.6 provides mathematical formula for Bayesian Estimate applied to $P(t_i|c_j)$. Here, *TF(ti,cj)* is the total number of times term *ti* occurs in the training set documents belonging to category *cj* . The summation term in the denominator stands for the total number of term occurrences in the training set data belonging to category *cj* . This estimator is called Laplace estimator and assumes that the observation of each word is a priori likely [17].

Multinomial model: an instance is represented by a feature vector with integer elements whose value is the frequency of that word in the instance.

Thus given a training set of instances (with labelled class) and a set of K classes, multinomial text classification model is as follows:

1. The number of words in the vocabulary defines the dimension of the feature vectors which is represented by Vocabulary V.

2. In the training set, the total number of instances and frequency of the word are calculated. The total number of instances is N. The number of instances labelled with class C=k, for each class k=1, . . . , K, N_k. The frequency of word w_t in instance D_i, x_{it} is computed for every word w_t in V.

3. The likelihoods $P(w_t|C=k)$ is estimated using

$$\hat{P}(w_t|C = k) = \frac{\sum_{i=1}^{N} x_{it} z_{ik}}{\sum_{s=1}^{|v|} \sum_{i=1}^{N} x_{is} z_{ik}} \quad \textbf{(3.7)}$$

where z_{ik} is an indicator variable which equals 1 when *Di* has class C=k, and equals 0 otherwise

4. The priors *P(C=k)* are estimated using $\hat{P}(C = k) = \frac{N_k}{N}$ **(3.8)**

5.The unlabeled instance *Dj* is classified by estimating the posterior probability for each class using

$$P(C/D_j) \propto P(C/x_j) \propto P(C) \prod_{i=1}^{|V|} P(w_t|C)^{x_{it}} \quad \textbf{(3.9)}$$

3.5 K-means clustering

This section mentions about the machine learning algorithm used in the layer 2 of the work.

Clustering is a type of unsupervised machine learning algorithms, which do not require any labelled data for training. Clustering algorithms group data points according to their similarities. In other words, how close these data points are. Give a real-life example, Assuming that we have a dataset containing human data samples and dog data samples but we are not sure how humans and dogs look like. Each sample described with two features, namely "walking with four legs" and "having a tale". The two features of each sample have Boolean values. A clustering algorithm groups all the samples according their feature values. If the data samples collected properly, we should see identical patterns in the two clusters, namely the human cluster and the dog cluster. Thus, we can use the patterns to describe and distinguish humans and dogs.

It is important to know the of number of clusters to be formed, then clusters can be found using the kmeans algorithm.

Let k be the number of clusters. The algorithm is as follows:

1. k data points from the data set are selected randomly as the initial centroids

2.Iteratively data points are assigned to their nearest centroid and centroids are updated incrementally, i.e., after each assignment of a document to its nearest centroid and this stops the clusters are stable

K-means is a simple but powerful technique for identifying grouping data with similar features in.

The centroid vector c of cluster C of instances is given by the equation 3.10 :

$$c = \frac{\sum_{d=c} d}{C} \tag{3.10}$$

So, *c* is obtained by averaging the weights of the terms of the instances in *C*. The similarity between a instance *d* (data) and a centroid vector *c* is given by cosine similarity measure equation 3.11.

$$cos(d, c) = \frac{d \bullet c}{|d|.|c|} \tag{3.11}$$

3.6 Input data

This section mentions about the format of the data considered as input for this research.

There are several websites with databases collecting information on various types of security incidents, and one such place is Security Repo, an open source containing data of various security incidents and is available at http://www.secrepo.com/#. The data considered for this research is meta data of bro logs of various Exploit kits collected and generated in 2012.

The data for training is set up with 2000 instances which includes 1000 instances of legitimate data and 1000 more instances of malicious data (Exploit kit in this case). The malicious data for training is obtained from the meta-data of bro logs of Exploit kits found in secrepo.com. The legitimate data for training is obtained by generating a PCAP file which can be done using, for example, Wireshark or tcpdump.

3.7 Textual data to numeric format

The data under consideration for the research is textual data. This data has to be converted to numeric format to use it in machine learning. This can be done by using following methods:

3.7.1 Word count vector

This is based on Vector Space Model [64]. Each instance is treated as a vector of its words. There is one co-ordinate for every possible word *w*. The similarity between vectors is calculated by their dot product. Given the instances I_1 and I_2, their dot product is given by equation 3.12

$$I_1 \cdot I_2 = \sum_w I_1(w) \cdot I_2(w) \tag{3.12}$$

The obvious problem is that dot product is not scale invariant. This is solved by normalising them by words using equation 3.13.

$$\frac{\boldsymbol{I_1 \cdot I_2}}{||\boldsymbol{I_1}||\cdot||\boldsymbol{I_2}||} \tag{3.13}$$

When the angle between vectors given by $\boldsymbol{\theta(I_1, I_2) = arccos} \frac{I_1 \cdot I_2}{||I_1||\cdot||I_2||}$ is 0, then the instances are identical and is not identical when it is 90°.

The algorithm takes into account the number of occurrences of a given term in training instances from its class , including multiple occurrences. [35]

The algorithm reads the different instances. Then the algorithm splits each instances (for each character in the line, if it is not alphanumeric, previous word is added to the list and then it starts with a new word)into words. Then it counts the word frequencies (the word list is sorted, for each word in word list, counter is incremented if it same as last word else last word and its counter is added to the list and counter is reset to 0). Then it computes the dot products (it starts with first words of each instance if the words are equal, word frequencies are multiplied and added to the total, and is repeated until the instances run out of words). When the dot product is 0, it implies that the instances are same and when the dot product is 90°, it implies that instances are not identical. [35]

3.8 Feature selection

This section mentions about the method to validate the strength of the features considered for machine learning.

The goal of feature selection is to pick the strong relevant features which can help solve the problem at hand by ignoring redundant and noisy features which do not give much information about the data.

Information gain assign a scoring to each feature by applying a statistical measure to the features. The features are ranked by the score and the suitable features can be selected based on their score. This is univariate and considers all the features independently, or with regard to the dependent variable. This makes it suitable for this work.[56]

Information Gain:

Information gain measures the number of bits of information gained in a data for category prediction considering the presence and absence of a term to better represent the data. When the set of possible categories is {*d1, d2, ..., dm*}, the IG for each unique term t is calculated as follows [16]:

$$\mathbf{IG(t) = -\sum_{i=1}^{m} P(d_i).\log(P(d_i)) + P(t).\sum_{i=1}^{m} P(d_i|t).\log(P(d_i|t)) + P(\bar{t})\sum_{i=1}^{m} P(d_i|\bar{t}).\log(P(d_i|\bar{t}))}$$

(3.14)

Equation 3.14 provides mathematical formula to calcualte information gain for each term. For the two cases of when the feature is present and when the feature is absent, decrease in the entropy is calculated. $P(d_i)$ is the prior probability of category d_i It can be estimated from the fraction of instances in the training set belonging to category d_i. *P(t)* is the prior probability of term t. It can be estimated from the fraction of instances in the training set in which term t is present. $P(\bar{t})$ can be estimated from the fraction of instances in the training set in which term t is absent. The Terms whose Information gain are less than some predetermined threshold which depends on the problem at hand are removed from the feature space. [16]

3.9 Scikit- library

This section mentions about the functions made use of in the implementation of this work.

Scikit Learn provides a wide range of supervised learning algorithms and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license. It is distributed under many Linux distributions, encouraging academic use and commercial use. [68]

Some popular groups of models provided by Scikit Learn include Clustering, Cross Validation, Datasets, Dimensionality Reduction, Ensemble methods, Feature extraction, Feature selection , Parameter Tuning, Manifold Learning, Supervised Models (Naive Bayes, discriminate analysis, neural networks, support vector machines and decision trees). [68]

This work has made use of following scikit-library functions:

1) class sklearn.naive_bayes.MultinomialNB

This is Naive Bayes classifier used for multinomial models (multinomially distributed data). This is usually used when discrete features (for example word count vectors) are considered for classification. [68]

2) sklearn.feature_extraction.text import CountVectorizer

This imports the countvectorizer from sklearn.feature_extraction.text, then creating the vectorizer to feed the data in the form of strings. It takes each word(anything with more than

2 letters as a word) as token and assigns an integer and counts the number of times it occurs in a instance(data). [68]

3) class sklearn.cluster.KMeans

This kmeans algorithm groups the data by trying to separate data in n clusters of equal variance, minimizing a criterion known as the inertia (measure of how internally coherent clusters are). This requires the number of clusters to be mentioned in advance. [68]

Chapter 4 Method

This research uses machine learning to detect the Exploit kit attacks with the help of supervised learning and unsupervised learning. The procedure is explained below.

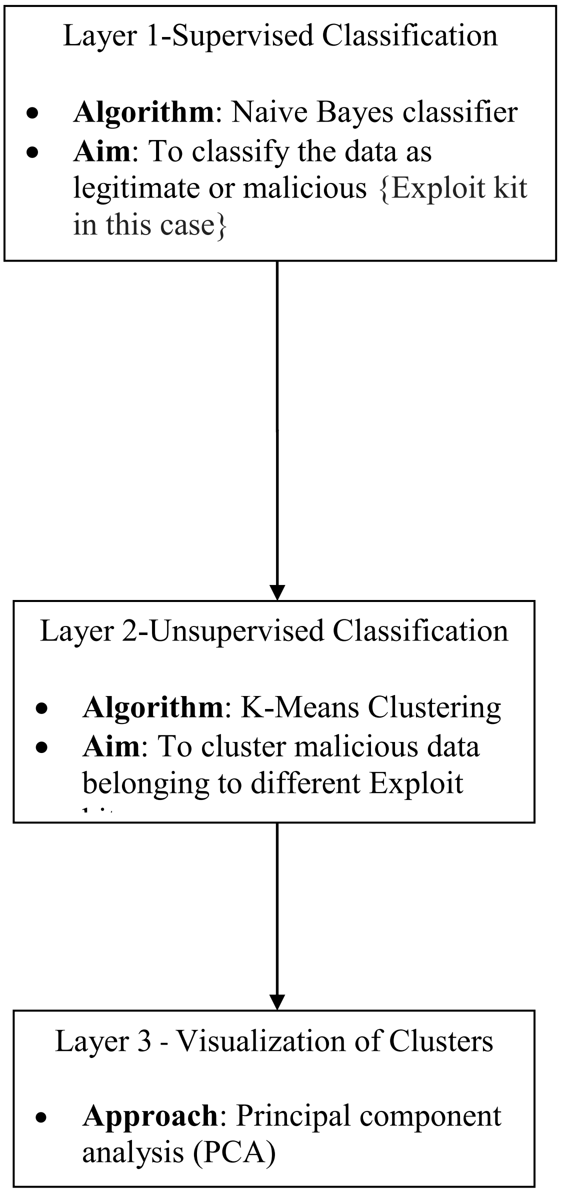

Layer 1-Supervised Classification
Algorithm: Naive Bayes classifier
Aim: To classify the data as legitimate or malicious {Exploit kit in this case}
Layer 2-Unsupervised Classification
Algorithm: K-Means Clustering
Aim: To cluster malicious data belonging to different Exploit
Layer 3 - Visualization of Clusters
Approach: Principal component analysis (PCA)

Figure 4.1: Overview of the methodology

Figure 4.1 represents the overview of the methodology used in this work.

Layer 1:

This stage involves supervised machine learning approach by using multinomial Naive Bayes algorithm. The Naive Bayes algorithm classifies the given data as either legitimate data or malicious data which is Exploit kit in this research project.

The information gain algorithm [16] is implemented to select the best features for machine learning. Based on this algorithm and relevance of features for detection, the features are chosen and used in detection of Exploit kits.

Layer 2:

In the second stage of the work, the output of the first stage is clustered using an unsupervised learning algorithm, namely K-means clustering [7]. This algorithm clusters the data into different groups of Exploit kits.

The data is textual and it is converted to numeric format using unigram Characterization [67]. Each alphabet within a feature, which is a string in context of Exploit kit detection, is mapped to a number to build a distinct profile. A 36-dimensional numeric vector represents the URL and domain name of the malicious data. The value of each element in the vector is a digit, and each dimension stands for a distinct alphabet or a number.

Then the data is clustered into groups of different Exploit kits based on the similarity between the profiles of malicious instances. The value of *k*, namely the number of clusters, is determined based on the knowledge of the data and using an Elbow method [54]. Then the data is clustered into *k* different groups of Exploit kits.

Layer 3:

The clusters are visualised after being reduced to two dimensions using Principal Component Analysis (PCA) [63] as the data is 36 in dimension at layer 2.

4.1 Feature selection through information gain

The different parameters typically present the HTTP bro log are id.orig_h, id.orig_p, id.resp_h, id.resp_p, trans_depth, method, host, uri, referrer, user_agent, request_body_len, response_body_len, status_code, status_msg, info_code, info_msg, filename, tags, username, password, proxied, orig_fuids, orig_mime_types, resp_fuids, resp_mime_types. As currently all popular Exploit kits use the HTTP protocol, not HTTPS, this work will consider the HTTP log file as input.

It is expected that an increasing amount of Exploit kits will start favouring the use of HTTPS in the future, as the encryption of the traffic will be advantageous to the criminals behind these kits. Using encryption their attacks might stay under the radar longer, because intrusion detection systems will have more difficulties detecting their patterns.

The goal of an Exploit Kit is to infect users with malware and compromise the victims' environment to stage further criminal activities. To accomplish the goal, Exploit Kits exhibit several attack-centric characteristics such as profiling of victim (system configuration), redirection, content being served and so on. From the point of view of detecting Exploit Kits, the workflow of a typical Exploit Kit gives useful insights about the characterization of Exploit Kit behaviours, From the analysis, the following attack-centric features of Exploit Kits make the best features for detection.

1)Host: Host is the server on which the Exploit kit is hosted. Therefore, by examining the domain name present in the feature host, one can assess the likelihood of it being legitimate or not(for example by using TLD)

Ex: bfdshfhdssdhbksbksbkfbgkfbgkf.pallavi.ru, where sub domain bfdshfhdssdhbksbksbkfbgkfbgkf is unusual and long.

2)Referer: Referer gives information about the normal webpage which leads the user to the Exploit landing page. This can inform about the fact that the normal webpage(which referred a user to a Exploit landing page) has been compromised.

Ex: Normal website -> Exploit landing page-> Exploit, in this example one can conclude that normal website has been compromised.

3)User-Agent: This can give information about the configuration of the system user has including operation system, browser's version and so on. This can of be use to know the vulnerabilities present in the browser and likelihood of that user getting infected with a malware from an Exploit kit.

4)URL: This can be a good indicator to know if a given URL points to an Exploit Kit by examining it.

5)Status code: This contains a list of Hypertext Transfer Protocol (HTTP) response status codes. For Example 301 indicates moved permanently, 302 indicates found, 303 indicates See other, 403 indicates Forbidden, and so on.

6)Content-Type: This can indicate the type of content served by the Exploit kit.

As mentioned in the earlier chapter, Feature selection can be used for eliminating irrelevant and redundant features by selecting a subset of features which describe the samples better to produce a better classification performance.

Information gain algorithm measures the number of bits of information gained for detecting Exploit kits in this case when the presence or absence of a term in the data is known. This is done by Calculating the information gain (reduction in entropy) that would result by splitting the data on chosen attribute, then Calculating the frequency of each of the values in the target attribute, then calculating the sum of the entropy for each subset of records weighted by their probability of occurring in the training set, and then subtracting the entropy of the chosen attribute from the entropy of the whole data set with respect to the target attribute. [13]

The code used to calculate information gain is mentioned below:

```
# Calculates the entropy of the training set data set for the target attribute.
def entropy(data, target_attr):

    val_freq = collections.defaultdict(int)
    data_entropy = 0.0
    # Calculate the frequency of each of the values in the target attribute
    for item in data:
        if item == target_attr:
            for val_ in data[ item ]:
                val_freq[ val_ ] += 1.0


    # Calculate the entropy of the data for the target attribute
    for freq in val_freq.values():
        data_entropy += (-freq/len(data)) * math.log(freq/len(data), 2)
    return data_entropy


    # Calculates the information gain (reduction in entropy) that would result by splitting the
data on the chosen attribute (attr).
def gain(data, target_attr):
    val_freq = collections.defaultdict(int)
    subset_entropy = 0.0

    # Calculate the frequency of each of the values in the target attribute
    for value in data:
        if value == target_attr:
            for val_ in data[value]:
                val_freq[val_] += 1.0
    for value in val_freq:
        freq = val_freq[value]
        val_prob = freq / sum(val_freq.values())

        # Calculate the sum of the entropy for each subset of records weighted by their
probability of occuring in the training set.
        subset_data = [ ]
        for attr in data:
                for val_ in data[attr]:
                    if val_ == value:
                        subset_data.append(val_)
        subset_data = {target_attr : subset_data}
        subset_data = pd.DataFrame(subset_data,columns=[target_attr])
        subset_entropy += val_prob * entropy(subset_data, target_attr)


    # Subtract the entropy of the chosen attribute from the entropy of the whole data set with
respect to the target attribute (and return it)
    return (entropy(data, target_attr) - subset_entropy)
```

The information gained from the features which represent attack-centric behaviour of the Exploit kits are good enough to classify the new data as either legitimate data or malicious data. This can be seen in the results section. Based on the output of the program and relevance of features for detection, Host, URI, referrer, user-agent and status-code which are all connection specific, as they are all part of the request and MIME which is content specific, as it describes the content of the requested page are the features selected for detecting Exploit kits. These features are interesting and they provide the information about the steps involved in the Exploit kit attack. This work has discarded time, uid, id.orig_p, id.resp_h, id.resp_p, trans_depth, request_body_len, response_body as these features provide information only about a specific incident that cannot be used to detect other Exploit kits with machine learning.

4.2.1 Malicious/Bad data for training

The data considered for the research is the meta data of bro logs of various Exploit kits collected and generated in 2012. This is obtained from Security Repo, an open source containing data of various security incidents and is available at http://www.secrepo.com/#.

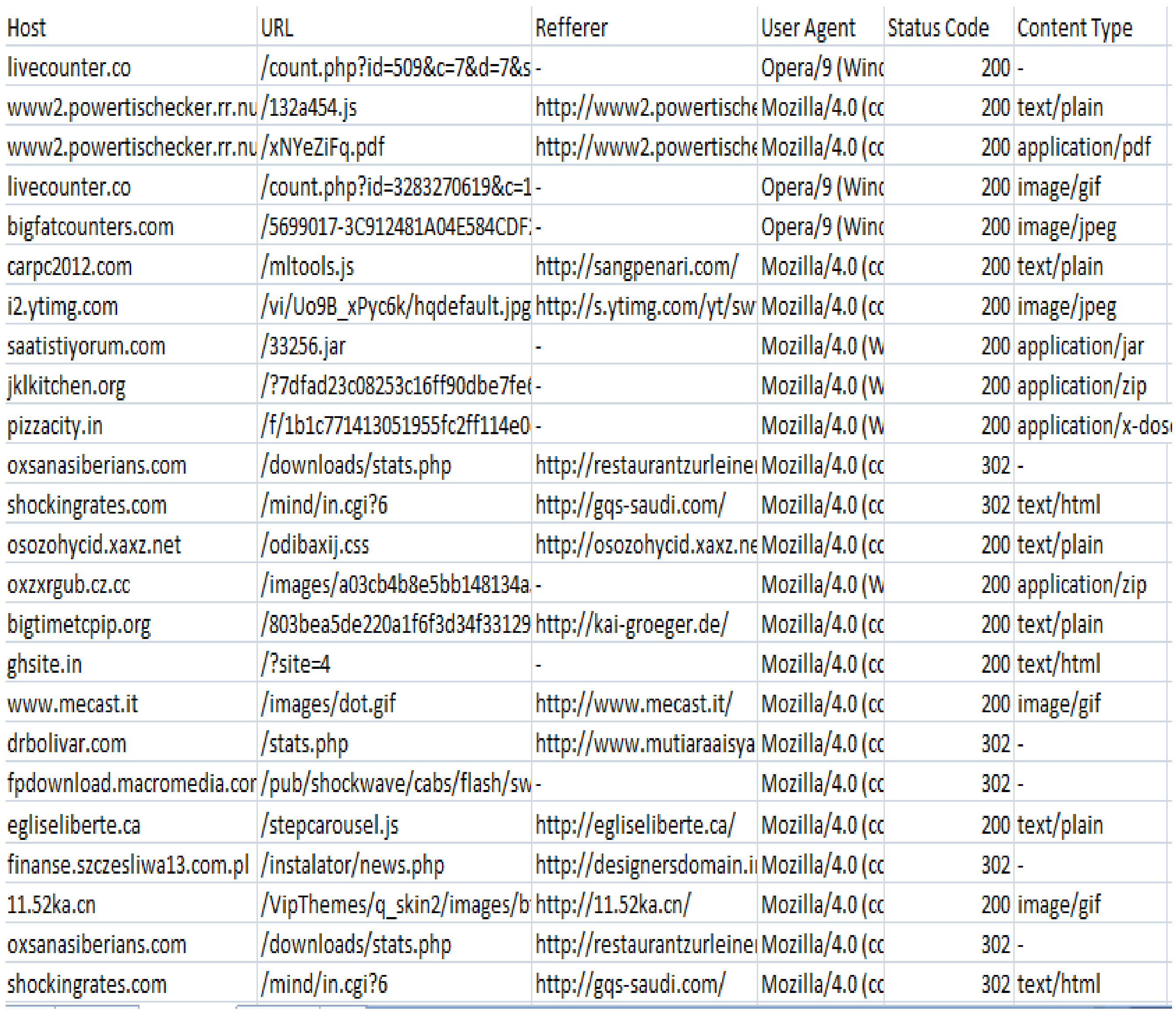
As mentioned in the figure 3.1, the first step of an Exploit kit attack begins with the redirection with an intention to send a user from a site a user visits to the malicious landing web page. The Exploit kit landing page then launches the suitable Exploits. Therefore, this is an important step without which there would be no traffic going to an Exploit kit landing page.

When a web site has been compromised, in this case (due to Exploit kits) it suggests that a redirect link would have been placed on it. This basically sends a small part of the web page, called an iframe, from the compromised site to the a landing page that will perform the actual attack. The user will have no knowledge of this as they do not witness anything unusual.

The process of redirect page to landing pages to malware servers to command and control (**C&C**) is series of independent phases which allows an attacker to set up the infrastructure as needed for each phase using different systems and techniques

The redirect page itself is not usually seen but the redirect site is discovered by the referer in the http protocol while looking at the initial calls to the landing page. The transition of the attack from the redirect page to the landing page give the attacker an advantage to design a redirect page with the sole function of tunnelling attacks to the Exploit servers. There can be multiple redirect pages scattered across the Internet acting as a funnel to send traffic to the landing page for Exploitation. The Exploit kit landing page will contain a malicious javascript to collect the relevant information about the configuration of the victims that visit the infected website. This information related to browser plugins and can include JAR files, PDFs and flash, is used to launch suitable attacks. Then the malicious program is downloaded and installed which often places itself in a number of places on the victim's system as to maintain itself on the system on a reboot or deletion of some of its components. Then the attacker gets further control of the victim's system once the malware then connects back to the attacker.

This research has considered all these stages of an Exploit kit attack to represent the malicious data set for the machine learning training. This work has considered about 1000 http log files of various Exploit kits to set up the data for training. Each instance taken from a different Exploit kits in the training file represent one of the different stages of the attack explained above. The malicious data is represented by 0 in the data set in this work.

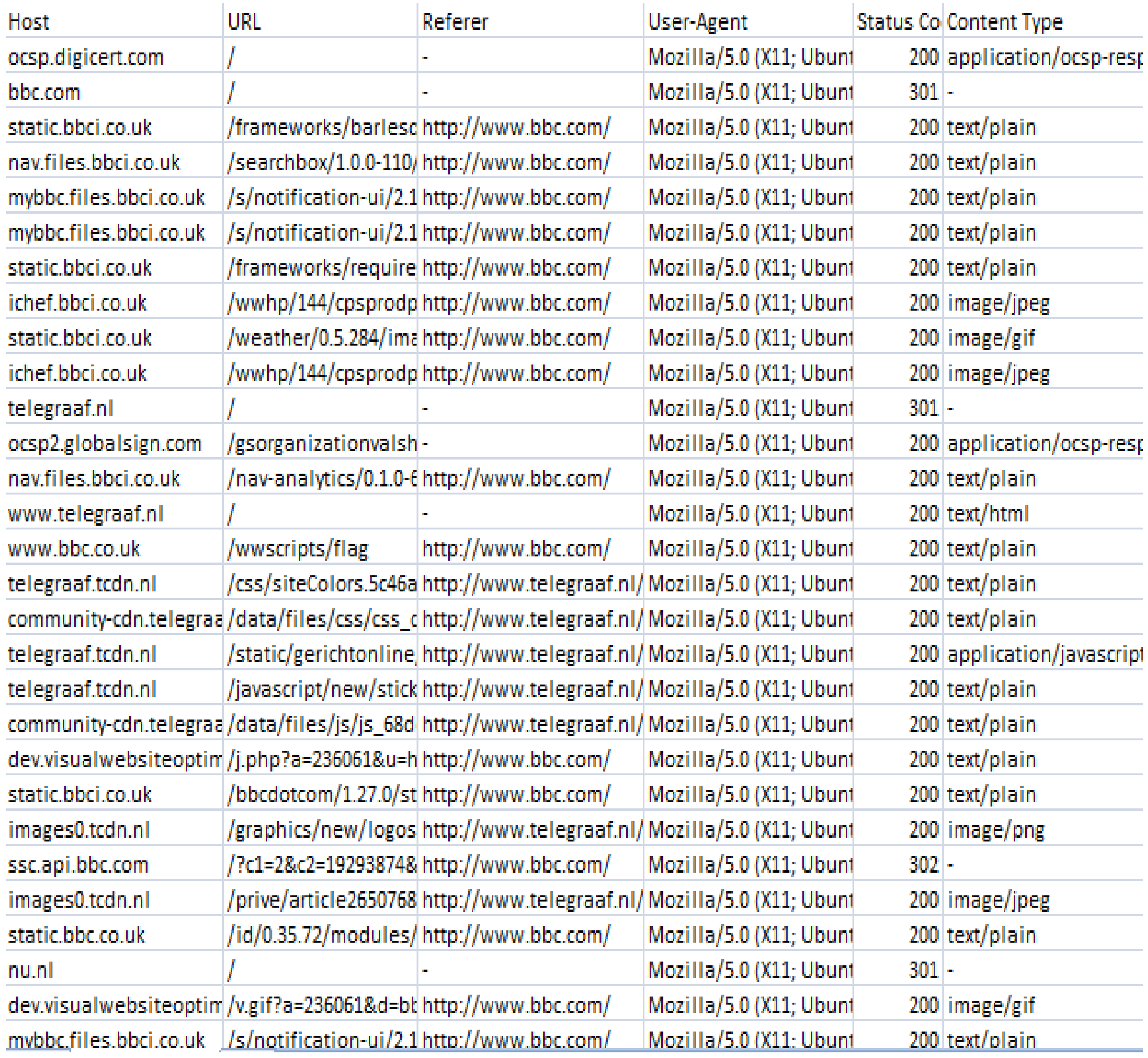


Host	URL	Refferer	User Agent	Status Code	Content Type
livecounter.co	/count.php?id=509&c=7&d=7&s	-	Opera/9 (Winc	200	-
www2.powertischecker.rr.nu	/132a454.js	http://www2.powertische	Mozilla/4.0 (cc	200	text/plain
www2.powertischecker.rr.nu	/xNYeZiFq.pdf	http://www2.powertische	Mozilla/4.0 (cc	200	application/pdf
livecounter.co	/count.php?id=3283270619&c=1	-	Opera/9 (Winc	200	image/gif
bigfatcounters.com	/5699017-3C912481A04E584CDF:	-	Opera/9 (Winc	200	image/jpeg
carpc2012.com	/mltools.js	http://sangpenari.com/	Mozilla/4.0 (cc	200	text/plain
i2.ytimg.com	/vi/Uo9B_xPyc6k/hqdefault.jpg	http://s.ytimg.com/yt/sw	Mozilla/4.0 (cc	200	image/jpeg
saatistiyorum.com	/33256.jar	-	Mozilla/4.0 (W	200	application/jar
jklkitchen.org	/?7dfad23c08253c16ff90dbe7fe(	-	Mozilla/4.0 (W	200	application/zip
pizzacity.in	/f/1b1c771413051955fc2ff114e0	-	Mozilla/4.0 (W	200	application/x-dos
oxsanasiberians.com	/downloads/stats.php	http://restaurantzurleinei	Mozilla/4.0 (cc	302	-
shockingrates.com	/mind/in.cgi?6	http://gqs-saudi.com/	Mozilla/4.0 (cc	302	text/html
osozohycid.xaxz.net	/odibaxij.css	http://osozohycid.xaxz.ne	Mozilla/4.0 (cc	200	text/plain
oxzxrgub.cz.cc	/images/a03cb4b8e5bb148134a	-	Mozilla/4.0 (W	200	application/zip
bigtimetcpip.org	/803bea5de220a1f6f3d34f33129	http://kai-groeger.de/	Mozilla/4.0 (cc	200	text/plain
ghsite.in	/?site=4	-	Mozilla/4.0 (cc	200	text/html
www.mecast.it	/images/dot.gif	http://www.mecast.it/	Mozilla/4.0 (cc	200	image/gif
drbolivar.com	/stats.php	http://www.mutiaraaisya	Mozilla/4.0 (cc	302	-
fpdownload.macromedia.cor	/pub/shockwave/cabs/flash/sw	-	Mozilla/4.0 (cc	302	-
egliseliberte.ca	/stepcarousel.js	http://egliseliberte.ca/	Mozilla/4.0 (cc	200	text/plain
finanse.szczesliwa13.com.pl	/instalator/news.php	http://designersdomain.i	Mozilla/4.0 (cc	302	-
11.52ka.cn	/VipThemes/q_skin2/images/b	http://11.52ka.cn/	Mozilla/4.0 (cc	200	image/gif
oxsanasiberians.com	/downloads/stats.php	http://restaurantzurleinei	Mozilla/4.0 (cc	302	-
shockingrates.com	/mind/in.cgi?6	http://gqs-saudi.com/	Mozilla/4.0 (cc	302	text/html

Table 4.1: Malicious data set up for training

4.2.2 Legitimate/Good data for training

The good data/legitimate data is obtained by generating a PCAP file which can be done using, for example Wireshark or tcpdump. The data is generated by browsing online while Wireshark would record the same and generate a file which later is parsed into different log files using BRO Network Security Monitor. The HTTP log file of these PCAPS are used to set up good data for machine learning training. Each instance taken from these http logs represent legitimate traffic. The legitimate data is represented by 1 in the data set in this work.



Host	URL	Referer	User-Agent	Status Co	Content Type
ocsp.digicert.com	/	-	Mozilla/5.0 (X11; Ubunt	200	application/ocsp-resp
bbc.com	/	-	Mozilla/5.0 (X11; Ubunt	301	-
static.bbci.co.uk	/frameworks/barlesc	http://www.bbc.com/	Mozilla/5.0 (X11; Ubunt	200	text/plain
nav.files.bbci.co.uk	/searchbox/1.0.0-110/	http://www.bbc.com/	Mozilla/5.0 (X11; Ubunt	200	text/plain
mybbc.files.bbci.co.uk	/s/notification-ui/2.1	http://www.bbc.com/	Mozilla/5.0 (X11; Ubunt	200	text/plain
mybbc.files.bbci.co.uk	/s/notification-ui/2.1	http://www.bbc.com/	Mozilla/5.0 (X11; Ubunt	200	text/plain
static.bbci.co.uk	/frameworks/require	http://www.bbc.com/	Mozilla/5.0 (X11; Ubunt	200	text/plain
ichef.bbci.co.uk	/wwhp/144/cpsprodp	http://www.bbc.com/	Mozilla/5.0 (X11; Ubunt	200	image/jpeg
static.bbci.co.uk	/weather/0.5.284/ima	http://www.bbc.com/	Mozilla/5.0 (X11; Ubunt	200	image/gif
ichef.bbci.co.uk	/wwhp/144/cpsprodp	http://www.bbc.com/	Mozilla/5.0 (X11; Ubunt	200	image/jpeg
telegraaf.nl	/	-	Mozilla/5.0 (X11; Ubunt	301	-
ocsp2.globalsign.com	/gsorganizationvalsh	-	Mozilla/5.0 (X11; Ubunt	200	application/ocsp-resp
nav.files.bbci.co.uk	/nav-analytics/0.1.0-6	http://www.bbc.com/	Mozilla/5.0 (X11; Ubunt	200	text/plain
www.telegraaf.nl	/	-	Mozilla/5.0 (X11; Ubunt	200	text/html
www.bbc.co.uk	/wwscripts/flag	http://www.bbc.com/	Mozilla/5.0 (X11; Ubunt	200	text/plain
telegraaf.tcdn.nl	/css/siteColors.5c46a	http://www.telegraaf.nl/	Mozilla/5.0 (X11; Ubunt	200	text/plain
community-cdn.telegraa	/data/files/css/css_c	http://www.telegraaf.nl/	Mozilla/5.0 (X11; Ubunt	200	text/plain
telegraaf.tcdn.nl	/static/gerichtonline_	http://www.telegraaf.nl/	Mozilla/5.0 (X11; Ubunt	200	application/javascript
telegraaf.tcdn.nl	/javascript/new/stick	http://www.telegraaf.nl/	Mozilla/5.0 (X11; Ubunt	200	text/plain
community-cdn.telegraa	/data/files/js/js_68d	http://www.telegraaf.nl/	Mozilla/5.0 (X11; Ubunt	200	text/plain
dev.visualwebsiteoptim	/j.php?a=236061&u=h	http://www.bbc.com/	Mozilla/5.0 (X11; Ubunt	200	text/plain
static.bbci.co.uk	/bbcdotcom/1.27.0/st	http://www.bbc.com/	Mozilla/5.0 (X11; Ubunt	200	text/plain
images0.tcdn.nl	/graphics/new/logos	http://www.telegraaf.nl/	Mozilla/5.0 (X11; Ubunt	200	image/png
ssc.api.bbc.com	/?c1=2&c2=19293874&	http://www.bbc.com/	Mozilla/5.0 (X11; Ubunt	302	-
images0.tcdn.nl	/prive/article2650768	http://www.telegraaf.nl/	Mozilla/5.0 (X11; Ubunt	200	image/jpeg
static.bbc.co.uk	/id/0.35.72/modules/	http://www.bbc.com/	Mozilla/5.0 (X11; Ubunt	200	text/plain
nu.nl	/	-	Mozilla/5.0 (X11; Ubunt	301	-
dev.visualwebsiteoptim	/v.gif?a=236061&d=bb	http://www.bbc.com/	Mozilla/5.0 (X11; Ubunt	200	image/gif
mybbc.files.bbci.co.uk	/s/notification-ui/2.1	http://www.bbc.com/	Mozilla/5.0 (X11; Ubunt	200	text/plain

Table 4.2: Legitimate data set up for training

4.3 Layer 1 - Textual data to numeric format

4.3.1 Term frequency and inverse document frequency (TFIDF)

It is a product of term frequency and inverse document frequency. The term frequency indicates the number of time a term appears in a document of the corpus. This has an obvious problem of terms being considered prominent beyond their value. This can be solved by calculating inverse document frequency which is a measure of how important each term is in a document.

Documents	Method	Host
Document1	GET	www.morango.it
Document2	GET	www.samistream.com
Document3	GET	a.adorika.net

Documents	Uri	Referrer
Document1	/wp-content/plugins/nextgen-gallery/css/nggallery.css?ver=1.0.0	http://www.morango.it/
Document2	/thumbs/geosuper.PNG	http://samistream.com/
Document3	/c/banner_s?tenant=AD&selection=3367&size=300x250&di=1&skin=iframe	http://www.samistream.com/728x90.html

Documents	User Agent	Status Code
Document1	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)	200
Document2	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)	404
Document3	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)	200

Documents	resp_mime_types
Document1	text/plain
Document2	text/html
Document3	text/html

Table 4.3: set of documents used to set up as an example for TFIDF vectorisation

Term frequency is nothing but number of times a term appears in each document/instance.

Inverse document frequency (Idf) is numerical measure of how much information each term provides. This can be calculated by the formula log (N/n), where N is total no of documents and n is the no of documents which contain the term.

TFIDF=Term frequency * inverse document frequency

Document 1
1)Tf (Get, d1) = (1/7) and Idf (Get, D) =log (3/3) and TFIDF= (1/7) * 0 = 0
2)Tf(www.morango.it, d1) = (1/7) and Idf (www.morango.it, D) =log (3/1)= 0.48 and TFIDF= (1/7) * 0.48 = 0.07
3)Tf(/wp-content/plugins/nextgen-gallery/css/nggallery.css?ver=1.0.0, d1) = (1/7) and Idf ((/wp-content/plugins/nextgen-gallery/css/nggallery.css?ver=1.0.0, D) =log (3/1)= 0.48 and TFIDF= (1/7) * 0.48 = 0.07
4)Tf(http://www.morango.it/, d1) = (1/7) and Idf (http://www.morango.it/, D) =log (3/1)= 0.48 and TFIDF= (1/7) * 0.48 = 0.07
5)Tf(Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1), d1) = (1/7) and Idf (Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1), D) =log (3/1)= 0.48 and TFIDF= (1/7) * 0.48 = 0.07
6)Tf(200, d1) = (1/7) and Idf (200), D) =log (3/2)= 0.18 and TFIDF= (1/7) * 0.48 = 0.03
7)Tf(text/plain, d1) = (1/7) and Idf (text/plain), D) =log (3/1)= 0.48 and TFIDF= (1/7) * 0.48 = 0.07
Document 2
1)Tf(Get, d2) = (1/7) and Idf (Get, D) =log (3/3) and TFIDF= (1/7) * 0 = 0
2)Tf(www.samistream.com, d2) = (1/7) and Idf (www.samistream.com, D) =log (3/1)= 0.48 and TFIDF= (1/7) * 0.48 = 0.07
3)Tf(/thumbs/geosuper.PNG, d2) = (1/7) and Idf ((/thumbs/geosuper.PNG, D) =log (3/1)= 0.48 and TFIDF= (1/7) * 0.48 = 0.07
4)Tf(http://samistream.com/,d2) = (1/7) and Idf (http://samistream.com/,D) =log (3/1)= 0.48 and TFIDF= (1/7) * 0.48 = 0.07
5)Tf(Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)), d2) = (1/7) and Idf (Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0), D) =log (3/2)= 0.18 and TFIDF= (1/7) * 0.48 = 0.03
6)Tf(404, d2) = (1/7) and Idf (404, D) =log (3/1)= 0.48 and TFIDF= (1/7) * 0.48 = 0.07
7)Tf(text/html, d2) = (1/7) and Idf (text/html), D) =log (3/2)= 0.18 and TFIDF= (1/7) * 0.48 = 0.03
Document 3
1)Tf(Get, d1) = (1/7) and Idf (Get, D) =log (3/3) and TFIDF= (1/7) * 0 = 0
2)Tf(a.adorika.net, d1) = (1/7) and Idf (a.adorika.net, D) =log (3/1)= 0.48 and TFIDF= (1/7) * 0.48 = 0.07

3)Tf(/c/banner_s?tenant=AD&selection=3367&size=300x250&di=1&skin=iframe, d1) = (1/7) and Idf ((/c/banner_s?tenant=AD&selection=3367&size=300x250&di=1&skin=iframe, D) =log (3/1)= 0.48 and TFIDF= (1/7) * 0.48 = 0.07
4)Tf(http://www.samistream.com/728x90.html,d1) = (1/7) and Idf (http://www.samistream.com/728x90.html, D) =log (3/1)= 0.48 and TFIDF= (1/7) * 0.48 = 0.07
5)Tf(Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0),d1) = (1/7) and Idf (Mozilla Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0),D) =log (3/2)= 0.18 and TFIDF= (1/7) * 0.48 = 0.03
6)Tf(200, d1) = (1/7) and Idf (200), D) =log (3/2)= 0.18 and TFIDF= (1/7) * 0.48 = 0.03
7) Tf(text/html, d2) = (1/7) and Idf (text/html), D) =log (3/2)= 0.18 and TFIDF= (1/7) * 0.48 = 0.03

Table 4.4: Term frequency and inverse document frequency calculation for the documents

The vector formed using TFIDF for the three documents is given below:

Vector:
[[0,0,0]
[0.07,0.07,0.07]
[0.07, 0.07,0.07]
[0.07,0.03,0.07]
[0.07, 0.03, 0.03]
[0.03, 0.07, 0.03]
[0.07,0.03,0.03]]

Table 4.5: Vector from of the documents calculated using Term frequency and inverse document frequency calculation

The features available in http bro log are with contents in it are ts, uid, id.orig_h, id.orig_p, id.resp_h, id.resp_p, trans_depth, method, host, uri, referrer, user_agent, request_body_len, response_body_len, status_code, resp_mime_types, resp_fuids.

From Exploit kit detection method, host, uri, referrer, user_agent, status_code, ,resp_mime_types are interesting as they give information about the steps involved in Exploit kit attack but method feature is discarded but there is not much information gain as it is present in all the documents which implies lack of its importance.

But the Term frequency and inverse document frequency calculation (TFIDF) fails in this work because of lack of consistency (there were different number of words in each instance which led to a vector that could not be used in Naive Bayes classfier) in the data and TFIDF is more suitable for similarity measurements in the data.

4.3.2 Word count vector

Word count vector fixes a dictionary containing n different words each instances found in the training set. It associates word count vector "a" with each instance where a_i = number of times word *i* appears in each instance. Each word in an instance is formed after it is separated by "/" and " " (space) in this approach.

Example:

Dictionary	Instance 1	Instance 2	Instance n
bratwurst.de	0	1	0
2.html	1	1	1
http:alyanaa.fr	0	0	1
Mozilla4.0	1	1	1
302	1	1	1
Texthtml	0	0	0

Table 4.6: An example for work count vector

Document-term matrix is given by collection(Dictionary) of N instances, with word count n-vectors $a_1, \ldots, a_N$, document-term matrix is N × n matrix A, with A_{ij} = number of times word j appears in instance i , rows of matrix A are $a_1^T, \ldots, a_N^T$ and jth column of A shows occurrences of word j across Dictionary.

when there are two instances with word count vectors a_1, a_2 and histogram vectors h_1, h_2 , the distance measure (of dissimilarity) is given by $||h1 - h2||$. This is expected to be small when the instances tend to be similar which can help to classify the data as legitimate or malicious.

4.4 Layer 1 - Multinomial Naive Bayes.

Multinomial Naive Bayes is a fast algorithm with low storage requirement. It is robust to irrelevant features as they tend to cancel each other out without affecting the results. It is good in domain with many equally important features as in this research. The features selected through information gain in this research such as host, URI, referrer, user-agent, status code, mime-type are equally important .It is optimal if the independent assumption hold true making it good dependable baseline for classification involving text data.

An example:

Set up	Instances	Words	Class
Training	**1**	**adregmedia.com /ga.js adregmedia.com**	**Good**
Training	**2**	**adregmedia.com adregmedia.com 200**	**Good**
Training	**3**	**adregmedia.com 302**	**Good**
Training	**4**	**google-analytics.com /IC/XvidSetup.exe adregmedia.com**	**Bad**
Testing	**1**	**adregmedia.com adregmedia.com adregmedia.com google-analytics.com /IC/XvidSetup.exe**	**Predict**

Table 4.7: An example for multinomial Naive Bayes Classification

$$\hat{P} = \frac{N_c}{N} \qquad \textbf{(4.1)}$$

where $\hat{P}$ is probability of a class, N_c is the number of instances belonging to that class and N is total number of instances. The Equation 4.1 presents Prior for each class/category

$$\hat{P}(w|c) = \frac{Count(w,c) + 1}{Count(c) + |V|} \qquad \textbf{(4.2)}$$

where $\widehat{\boldsymbol{P}}(\boldsymbol{w}|\boldsymbol{c})$ is likelihood of a word given a class, $\boldsymbol{Count}(\boldsymbol{w},\boldsymbol{c})$ is count of the that word occurring in that class, $\boldsymbol{Count}(\boldsymbol{c})$ is count of all the words occurring in that class, V is total number of unique words in present in all the instances. The Equation 4.2 provides mathematical formula for Conditional probability.

Prior for class good P(good)= 3/4 [class good appears 3 times out of 4 instances in the table]

Prior for class bad P(bad)= 1/4 [Class bad appears once out of 4 instances in the table]

Conditional Probabilities

P(adregmedia.com | Good) = (5+1)/(6+8)= 6/14 =3/7

[the term adregmedia.com appears 5 times in the class Good plus 1 which is divided by the total number of words in class good plus the number of unique words in the entire set (vocabulary)]

P(google-analytics.com | Good) = (0+1)/(6+8)= 1/14 [the term google-analytics.com does not appear in the class Good plus 1 which is divided by the total number of words in class good plus the number of unique words in the entire set (vocabulary)]

P(/IC/XvidSetup.exe| Good) = = (0+1)/(6+8)= 1/14 [the term (/IC/XvidSetup.exe does not appear in the class Good plus 1 which is divided by the total number of words in class good plus the number of unique words in the entire set (vocabulary)]

P(adregmedia.com | Bad) = (1+1)/(3+6)= 2/9 [the term adregmedia.com appears once in the class Bad plus 1 which is divided by the total number of words in class Bad plus the number of unique words in the entire set (vocabulary)]

P(google-analytics.com | Bad) = (1+1)/(3+6)= 2/9 [the term google-analytics.com appears once in the class Bad plus 1 which is divided by the total number of words in class Bad plus the number of unique words in the entire set (vocabulary)]

P(/IC/XvidSetup.exe| Bad) = (1+1)/(3+6)= 2/9 [the term /IC/XvidSetup.exe appears once in the class Bad plus 1 which is divided by the total number of words in class Bad plus the number of unique words in the entire set (vocabulary)]

Choosing the class for the test instance:

$$P(Good \mid test) \propto \widehat{\boldsymbol{P}} * \widehat{\boldsymbol{P}}(\boldsymbol{w}|\boldsymbol{c}) = \widehat{\boldsymbol{P}} * \widehat{\boldsymbol{P}}(\boldsymbol{w}|\boldsymbol{Good}) = \mathbf{3/4 * 3/7 * 3/7 * 3/7 * 1/14 * 1/14 = 0.0003}$$

$$P(Bad \mid test) \propto \widehat{\boldsymbol{P}} * \widehat{\boldsymbol{P}}(\boldsymbol{w}|\boldsymbol{c}) = \widehat{\boldsymbol{P}} * \widehat{\boldsymbol{P}}(\boldsymbol{w}|\boldsymbol{Bad}) = \mathbf{1/4 * 2/9 * 2/9 * 2/9 * 2/9 * 2/9 = 0.0001}$$

In this example, the chosen class would be class Good as a result of the calculation. If the number of instances are equal in training set for each classes, then the effect of words being repeated will increase the chance of that instance belonging to that class and unique words would imply otherwise.

4.5 Feature selection for k-means clustering

The features selected for k-means clustering are domain name (Connection-Specific) and URL (Connection-Specific). This can help to identify and group the instances of given URL and domain which belong to a particular type of Exploit Kit.

Ex 1: XXX.com/php?id=11, XXX.com/ php?id=12, XXX.com/ php?id=13, in this particular example, Exploit kit owner use php?id=11, php?id=12, php?id=13 to identify different criminals who use the same Exploit kit.

Ex 2: abc.bet.com/123, bcd.bet.com/123, ced.bet.com/123, in this particular example, Exploit kit owner use abc, bcd and ced to identify different criminals who use the same Exploit kit.

These features can create a quite distinctive profile, making it easy for similarity measurements and therefore help in grouping the same type of Exploit kits together. This is because when a cyber criminal sells a particular Exploit kit (sells it to many other criminals), he identifies them with different URI (for example php?id=11, php?id=12, php?id=13 for different criminals) or uses different subdomains to identify them. This information is useful in grouping several Exploit kits of same type together.

4.6 Layer 2-Textual data to numeric format

The features selected for clustering is textual and it is converted to numeric format using unigram characterisation.

4.6.1 Uni-gram characterisation

The two features namely domain name and URL are combined together. Each alphabet of the string in the features are mapped to a number to build a distinct profile. This is done by creating a 36 dimensional vector, values in the vector are digits and each dimension stand for a distinct alphabet and numbers and the data is normalised in the end to compensate for the different lengths of the string. All the other special characters (for example -, . and so on) are excluded. The code mentioned below creates a profile from the features of each instance.

```
def _calc_vector(cls, vector_str):

    temp = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

    array1 = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't, 'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

    array2 = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']

    for letter in vector_str:
            # print(letter)
            if letter in array1:
                temp[array1.index(letter)] += 1
            elif letter in array2:
                temp[array2.index(letter)] += 1
        return [x/float(sum(temp)) for x in temp]
```

Example:

Host	**Url**	**Feature for clustering**
www.couchtarts.com	**/templates/yoo_pure/warp/js/dropdownmenu.js**	**www.couchtarts.com/templates/yoo_pure/warp/js/dropdownmenu.js**
21.cedrictherealtor.com	**/news/faults-ending.php**	**21.cedrictherealtor.com/news/faults-ending.php**
www1.smart-checkerfl.net.tf	**/31202e33.js**	**www1.smart-checkerfl.net.tf/31202e33.js**
libocai.com	**/directly/persuades-output_compiler.php**	**libocai.com/directly/persuades-output_compiler.php**
adf.ly	**/js/entry.js**	**adf.ly/js/entry.js**
bilsoierwer.tk	**/50610006.html**	**bilsoierwer.tk/50610006.html**
sausagesments.com	**/hole/games/java_trust.php?f=103**	**sausagesments.com/hole/games/java_trust.php?f=103**
ads.clicksor.	**/newServing/searchTrack.php?**	**ads.clicksor.com/newServing/searchTra**

com	**nid=1&random=1523964293**	**ck.php?nid=1&random=1523964293**
allocatzds.aelita.fr	**//index.php?d4bdab7f2856dbb1991b9a8d6d905b6e**	**allocatzds.aelita.fr//index.php?d4bdab7f2856dbb1991b9a8d6d905b6e**
laisvai.lt	**/jstools.js**	**laisvai.lt/jstools.js**

Table 4.8: Input data for Uni-gram Characterisation

The output for the features from uni-gram characterisation is

Host vector
[0.05, 0.0, 0.075, 0.05, 0.125, 0.025, 0.025, 0.05, 0.05, 0.0, 0.0, 0.05, 0.025, 0.075, 0.05, 0.05, 0.0, 0.075, 0.05, 0.075, 0.025, 0.0, 0.025, 0.0, 0.0, 0.0, 0.0, 0.025, 0.025, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.030303030303030304, 0.0, 0.06060606060606061, 0.0, 0.12121212121212122, 0.06060606060606061, 0.0, 0.030303030303030304, 0.0, 0.030303030303030304, 0.030303030303030304, 0.030303030303030304, 0.030303030303030304, 0.030303030303030304, 0.0, 0.0, 0.0, 0.06060606060606061, 0.06060606060606061, 0.09090909090909091, 0.0, 0.0, 0.09090909090909091, 0.0, 0.0, 0.0, 0.030303030303030304, 0.06060606060606061, 0.06060606060606061, 0.09090909090909091, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.045454545454545456, 0.022727272727272728, 0.09090909090909091, 0.045454545454545456, 0.09090909090909091, 0.0, 0.0, 0.022727272727272728, 0.09090909090909091, 0.0, 0.0, 0.06818181818181818, 0.045454545454545456, 0.0, 0.09090909090909091, 0.11363636363636363, 0.0, 0.06818181818181818, 0.045454545454545456, 0.06818181818181818, 0.06818181818181818, 0.0, 0.0, 0.0, 0.022727272727272728, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.07142857142857142, 0.0, 0.0, 0.07142857142857142, 0.07142857142857142, 0.07142857142857142, 0.0, 0.0, 0.0, 0.14285714285714285, 0.0, 0.07142857142857142, 0.0, 0.07142857142857142, 0.0, 0.0, 0.0, 0.07142857142857142, 0.14285714285714285, 0.07142857142857142, 0.0, 0.0, 0.0, 0.0, 0.14285714285714285, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.04, 0.0, 0.0, 0.08, 0.0, 0.0, 0.04, 0.08, 0.0, 0.04, 0.08, 0.04, 0.0, 0.04, 0.0, 0.0, 0.08, 0.04, 0.08, 0.0, 0.0, 0.04, 0.0, 0.0, 0.0, 0.16, 0.04, 0.0, 0.0, 0.0, 0.04, 0.08, 0.0, 0.0, 0.0]
[0.12195121951219512, 0.0, 0.024390243902439025, 0.0, 0.0975609756097561, 0.024390243902439025, 0.04878048780487805, 0.04878048780487805, 0.0, 0.024390243902439025, 0.0, 0.024390243902439025, 0.07317073170731707, 0.024390243902439025, 0.04878048780487805, 0.04878048780487805, 0.0, 0.024390243902439025, 0.14634146341463414, 0.07317073170731707, 0.04878048780487805, 0.024390243902439025, 0.0, 0.0, 0.0, 0.0, 0.024390243902439025,

0.024390243902439025, 0.0, 0.024390243902439025, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.06896551724137931, 0.0, 0.08620689655172414, 0.05172413793103448, 0.05172413793103448, 0.0, 0.017241379310344827, 0.034482758620689655, 0.05172413793103448, 0.0, 0.034482758620689655, 0.017241379310344827, 0.034482758620689655, 0.06896551724137931, 0.05172413793103448, 0.034482758620689655, 0.0, 0.08620689655172414, 0.06896551724137931, 0.017241379310344827, 0.0, 0.017241379310344827, 0.017241379310344827, 0.0, 0.0, 0.0, 0.0, 0.034482758620689655, 0.034482758620689655, 0.034482758620689655, 0.017241379310344827, 0.017241379310344827, 0.017241379310344827, 0.0, 0.0, 0.034482758620689655]
[0.10344827586206896, 0.10344827586206896, 0.017241379310344827, 0.1206896551724138, 0.05172413793103448, 0.034482758620689655, 0.0, 0.017241379310344827, 0.034482758620689655, 0.0, 0.0, 0.05172413793103448, 0.0, 0.017241379310344827, 0.017241379310344827, 0.034482758620689655, 0.0, 0.017241379310344827, 0.017241379310344827, 0.034482758620689655, 0.0, 0.0, 0.0, 0.017241379310344827, 0.0, 0.017241379310344827, 0.017241379310344827, 0.034482758620689655, 0.017241379310344827, 0.0, 0.017241379310344827, 0.034482758620689655, 0.05172413793103448, 0.017241379310344827, 0.034482758620689655, 0.06896551724137931]
[0.1111111111111111, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.1111111111111111, 0.1111111111111111, 0.0, 0.16666666666666666, 0.0, 0.0, 0.1111111111111111, 0.0, 0.0, 0.0, 0.2222222222222222, 0.1111111111111111, 0.0, 0.05555555555555555, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

Table 4.9: Output of Uni-gram Characterisation

4.7 Layer 2 - K-means clustering

Clustering is a type of unsupervised machine learning algorithms, which do not require any labelled data for training. Clustering algorithms group data points according to their similarities. In other words, how close these data points are. Give a real-life example, Assuming that we have a dataset containing human data samples and dog data samples but we are not sure how humans and dogs look like. Each sample described with two features, namely “walking with four legs” and “having a tale”. The two features of each sample have Boolean values. A clustering algorithm groups all the samples according their feature values. If the data samples collected properly, we should see identical patterns in the two clusters, namely the human cluster and the dog cluster. Thus, we can use the patterns to describe and distinguish humans and dogs.

The value of k, number of clusters can be assumed based on the knowledge of the data. This work has implemented a program which calculates the ideal number of clusters based on Elbow method.

Elbow method computes the sum of squared error (SSE) for different values of k starting from 0. The SSE is defined as the sum of the squared distance between each member of the cluster and its centroid. The equation 4.3 provides mathematical formula for Sum of squared Error:

$$SSE = \sum_{I=1}^{k} \sum_{x \in C_i} dist(m_i, x)^2 \quad \textbf{(4.3)}$$

can be used as a measure of the quality of the clustering, where the error is the distance from a data point to the centroid of the cluster it belongs to. In the equation 3.9, for each point, the error is the distance to the nearest cluster ,SSE is obtained by squaring these errors and summing them up, x is a data point in cluster C_i and m_i is the representative point for cluster C_i . when two clusters are given, the one with the smallest error is chosen. [54]

The value of k is plotted against the SSE calculated, one can see that error decreases as K increase. The idea is to choose that k at which SSE decreases abruptly because when the number of clusters increases, distortion becomes smaller.

This is done by using the code below

```
def do_KMeans(vec):
   "" This function takes a list of 36 dimensional samples and finds clusters """

   SSE = []
   prev = 10000000000
   THRESH = 0.1
    for i in range(len(vec)):
      fitted = KMeans(n_clusters=i + 1, n_init=10).fit(vec)
      errors = fitted.transform(vec)
      curr = sum([min(err) ** 2 for err in errors])
      SSE.append(curr)
      if abs(prev - curr) < THRESH:
         break
      prev = curr
   print "======="
   print SSE
   plt.plot(SSE)
   plt.xlabel('K-number of clusters')
   plt.ylabel('Sum Squared Error')
   plt.title('SSE Curve')
   plt.show()
   plt.clf()
   print "Ideal number of clusters is: " + str(i + 1)
   returnfitted
```

Based on the value of k obtained from the elbow method, cluster centres are calculated for the k clusters, then each point is assigned to the nearest clusters, then the new cluster centres are calculated and process is repeated until the clusters are stable.

4.7.1 Visualisation and Principal Component Analysis

Visualisation of the clusters formed is done using a dimensionality reduction technique , Principal Component Analysis (PCA) which can bring out strong patterns in the dataset and emphasize the variation. This is used to obtain a new 2- dimensional coordinate system (basis) from a 36-dimensional data after considering only the axes that contain as much of the dataset's variance as possible and ignoring the axes with very little variance. The two dimensional clusters are plotted.

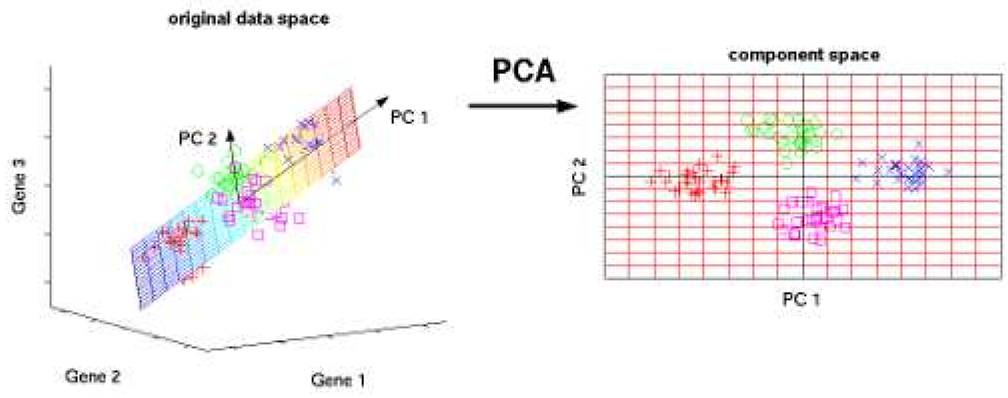


Figure 4.2: Principal Component Analysis . Adapted from [63]

The figure 4.2 shows an example that contains gene expression data which are mainly located within a three-dimensional subspace. Principal Component Analysis is used to visualize these data points by reducing the dimensionality of the data. The three original variables are reduced to a lower number of two new variables termed principal components by identifying the two-dimensional plane that optimally describes the highest variance of the data. This two-dimensional subspace can then be presented as a two-dimensional component space by rotating the same. [63]

Chapter 5 Results

5.1 Analysis of Features

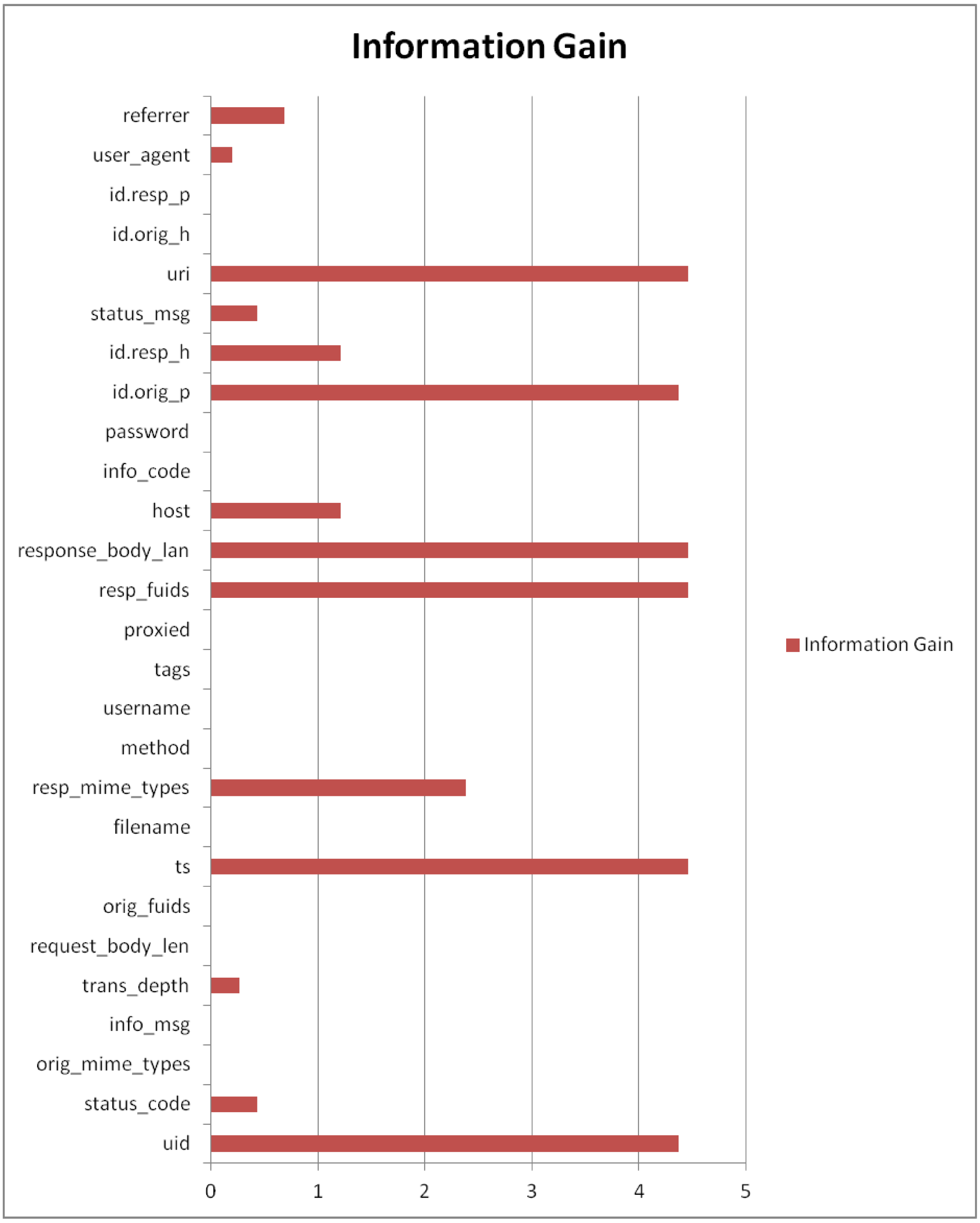


Figure 5.1: Analysis of features using Information gain

Based on the features used in this work, we now discuss the significance of the features from the machine learning point of view using information gain.

The figure 5.1 shows the information gain value provided by each feature in the HTTP bro log file. The graph shows higher value of gain for the features uid, status code, tans_dept, ts, mime_type, resp_fluids, response_body_len, host, id_orig, id.resp_h, status_msg, uri and refferer. Based on the domain knowledge, the features selected for effective detection of Exploit kit in this work are host (Connection-Specific), uri (Connection-Specific), referrer (Connection-Specific), user_agent (Connection-Specific), status_code (Connection-Specific) and mime_type (Content-Specific).These aggregate values of features, in fact, are fairly discriminative in putting apart Exploit kit and non Exploit kits. The connection-Specific and content-Specific features used in the work are backed up by the statistical variations in the graph which are indicators of the discriminative power of these features in practice. The features selected guarantee good detection rate. The other features such as time, uid, id.orig_p, id.resp_h, id.resp_p, trans_depth, request_body_len, response_body are discarded as they only provide information about a specific attack which cannot be used to generalise in order to detect other Exploit kits.

5.2 Accuracy of the Classifier on the Training Set

The metrics to evaluate the classifiers is Detection Rate (DR) on the training set using 10-fold cross validation. The detection rate is percentage of correctly classified data.

5.2.1 Cross Validation

This approach uses the entire transformed dataset to train and test a given the algorithm, Naive Bayes in this work. This involves separating the dataset into a number of equally sized groups of instances (called folds). This work opts for 10-fold (K-fold) validation. Then the model is then trained on 9 folds with an exception of one fold which is used for the test. The process is repeated K times so that each fold get's an opportunity at being left out used as the test dataset. Finally, the performance measures are averaged across all folds to estimate the capability of the Naive Bayes algorithm in this work.

The original sample of 2400 instances (1200 instances of legitimate data and 1200 ore instances of malicious data)is randomly partitioned into k equal size subsamples of 240 instances (120 instances of malicious data and 120 more instances of legitimate data in this work). Of the k subsamples, a single subsample(240 instances) is retained as the validation data for testing the model, and the remaining k-1 (2160 instances) subsamples are used as training data. The cross-validation process is then repeated 10 times (the folds), with each of the k subsamples used exactly once as the validation data.

10-fold cross validation involved training and testing a model 10 times:

- ✓ #1: Train on folds 1+2+3+4+5+6+7+8+9, test on fold 10
- ✓ #2: Train on folds 1+2+3+4+5+6+7+8+10, test on fold 9
- ✓ #3: Train on folds 1+2+3+4+5+6+7+9+10, test on fold 8
- ✓ #4: Train on folds 1+2+3+4+5+6+8+9+10, test on fold 7
- ✓ #5: Train on folds 1+2+3+4+5+7+8+9+10, test on fold 6

- ✓ #6: Train on folds 1+2+3+4+6+7+8+9+10, test on fold 5
- ✓ #7: Train on folds 1+2+3+5+6+7+8+9+10, test on fold 4
- ✓ #8: Train on folds 1+2+4+5+6+7+8+9+10, test on fold 3
- ✓ #9: Train on folds 1+3+4+5+6+7+8+9+10, test on fold 2
- ✓ #10: Train on folds 2+3+4+5+6+7+8+9+10, test on fold 1

The Performance of classifier on a separate testing set using 10-fold method is shown below:

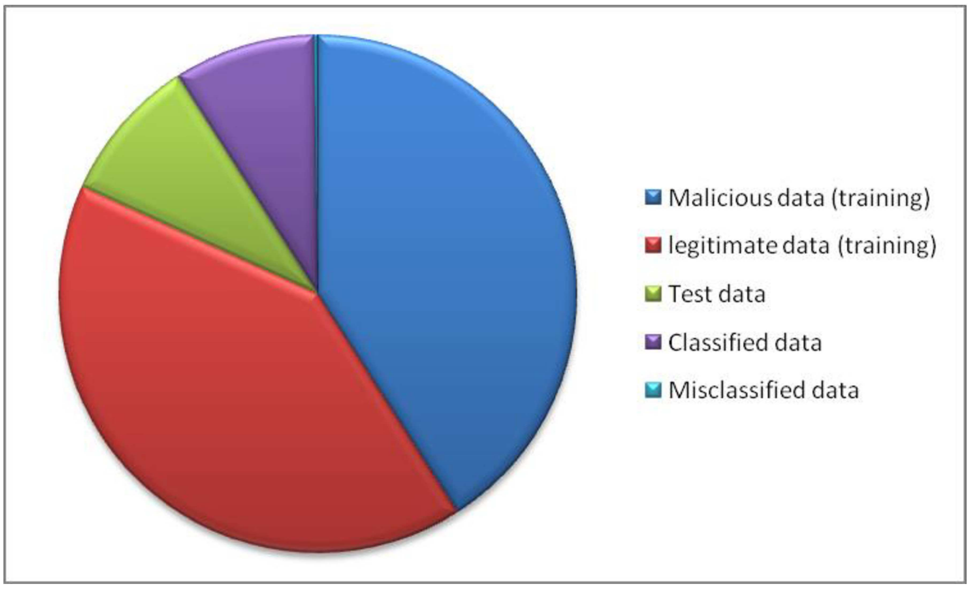


Figure 5.2: Performance of the classifier for K=1

From the Figure 5.2, the number of instances of malicious data used for the training is 1080, the number of legitimate data instances used for the training is 1080, the number of instances of test data (120 malicious data instances and 120 legitimate data instances) is 240. The number of correctly classified data is 234 in this case with 6 misclassified data instances. The accuracy of this validation is 97.5%.

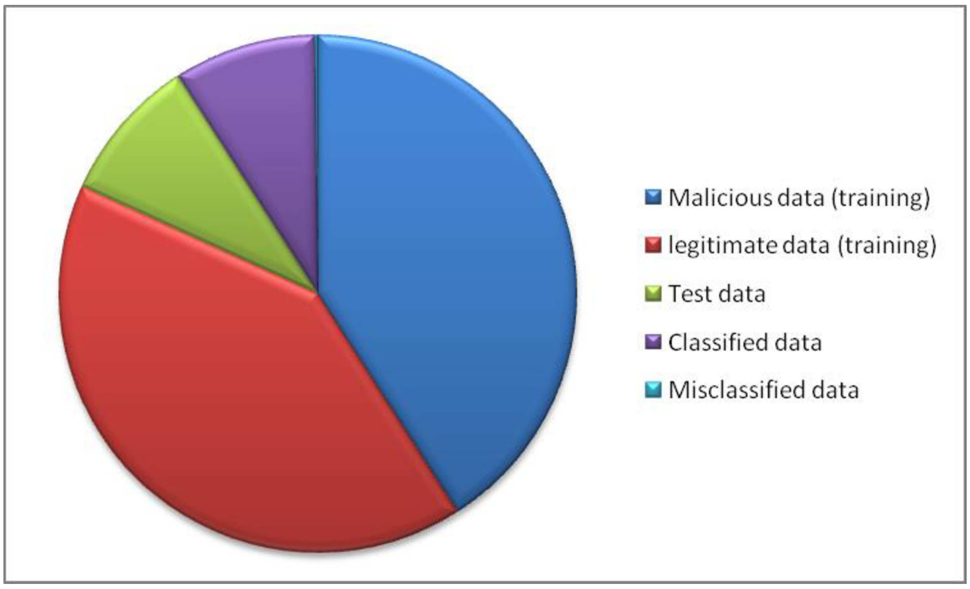


Figure 5.3: Performance of the classifier for K=2

From the Figure 5.3, the number of instances of malicious data used for the training is 1080, the number of legitimate data instances used for the training is 1080, the number of instances of test data (120 malicious data instances and 120 legitimate data instances) is 240. The number of correctly classified data is 235 in this case with 5 misclassified data instances. The accuracy of this validation is 97.9%.

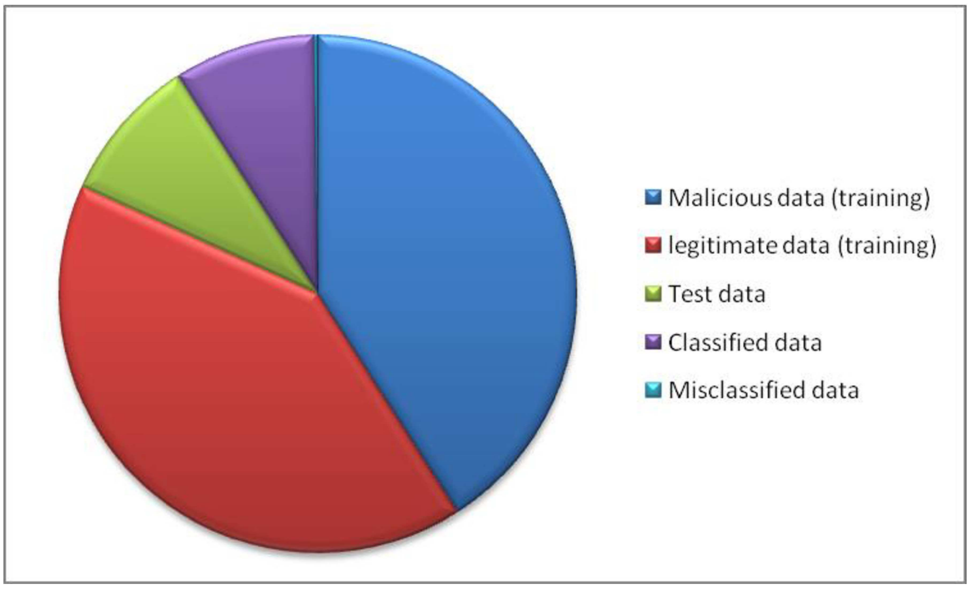


Figure 5.4: Performance of the classifier for K=3

From the Figure 5.4, the number of instances of malicious data used for the training is 1080, the number of legitimate data instances used for the training is 1080, the number of instances of test data (120 malicious data instances and 120 legitimate data instances) is 240. The number of correctly classified data is 234 in this case with 6 misclassified data instances. The accuracy of this validation is 97.5%.

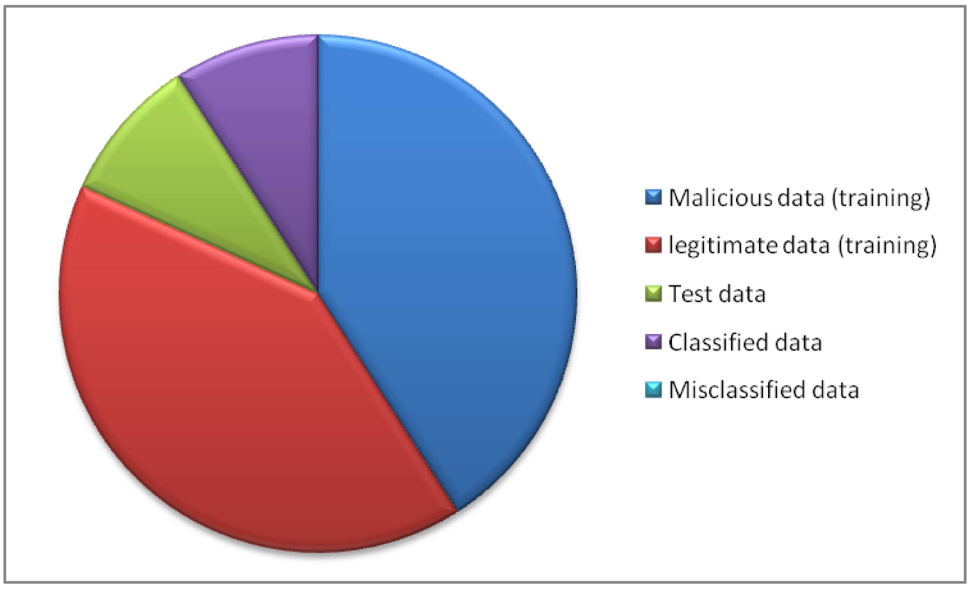


Figure 5.5: Performance of the classifier for K=4

From the Figure 5.5, the number of instances of malicious data used for the training is 1080, the number of legitimate data instances used for the training is 1080, the number of instances of test data (120 malicious data instances and 120 legitimate data instances) is 240. The number of correctly classified data is 240 in this case with 0 misclassified data instances. The accuracy of this validation is 100%.

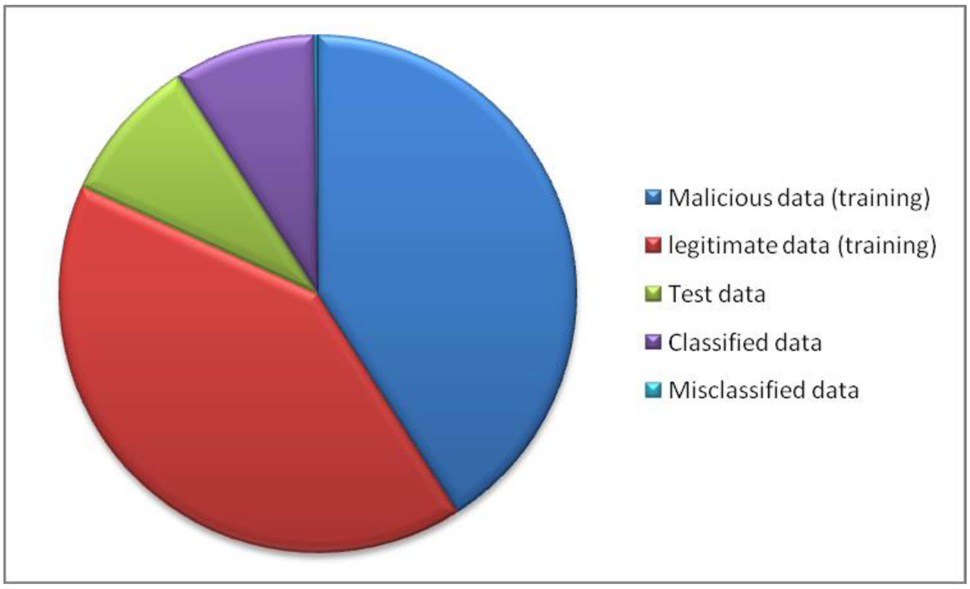


Figure 5.6: Performance of the classifier for K=5

From the Figure 5.6, the number of instances of malicious data used for the training is 1080, the number of legitimate data instances used for the training is 1080, the number of instances of test data (120 malicious data instances and 120 legitimate data instances) is 240. The number of correctly classified data is 234 in this case with 6 misclassified data instances. The accuracy of this validation is 97.5%.

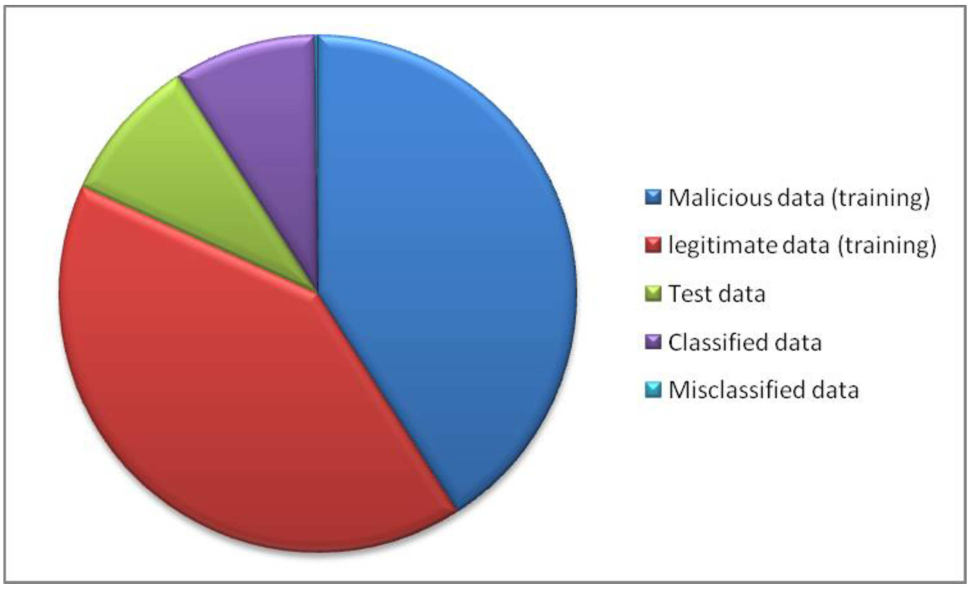


Figure 5.7: Performance of the classifier for K=6

From the Figure 5.7, the number of instances of malicious data used for the training is 1080, the number of legitimate data instances used for the training is 1080, the number of instances of test data (120 malicious data instances and 120 legitimate data instances) is 240. The number of correctly classified data is 235 in this case with 5 misclassified data instances. The accuracy of this validation is 97.9%.

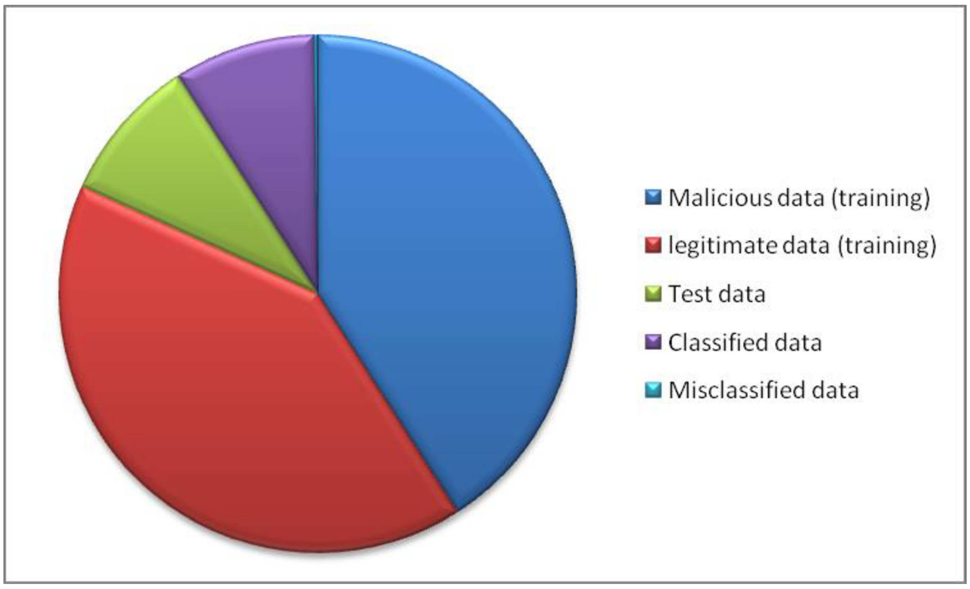


Figure 5.8: Performance of the classifier for K=7

From the Figure 5.8, the number of instances of malicious data used for the training is 1080, the number of legitimate data instances used for the training is 1080, the number of instances of test data (120 malicious data instances and 120 legitimate data instances) is 240. The number of correctly classified data is 234 in this case with 6 misclassified data instances. The accuracy of this validation is 97.5%.

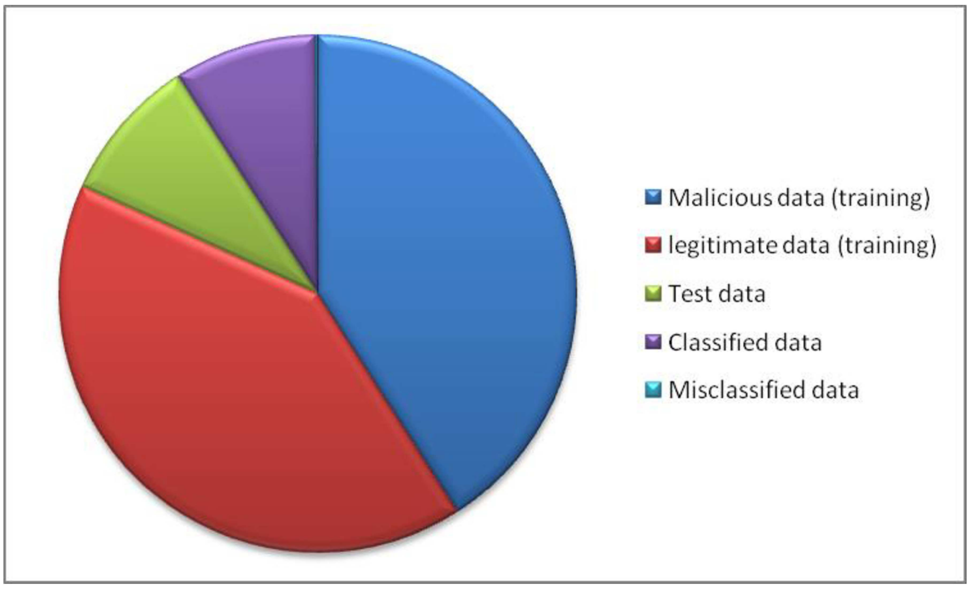


Figure 5.9: Performance of the classifier for K=8

From the Figure 5.9, the number of instances of malicious data used for the training is 1080, the number of legitimate data instances used for the training is 1080, the number of instances of test data (120 malicious data instances and 120 legitimate data instances) is 240. The number of correctly classified data is 236 in this case with 4 misclassified data instances. The accuracy of this validation is 98.3%.

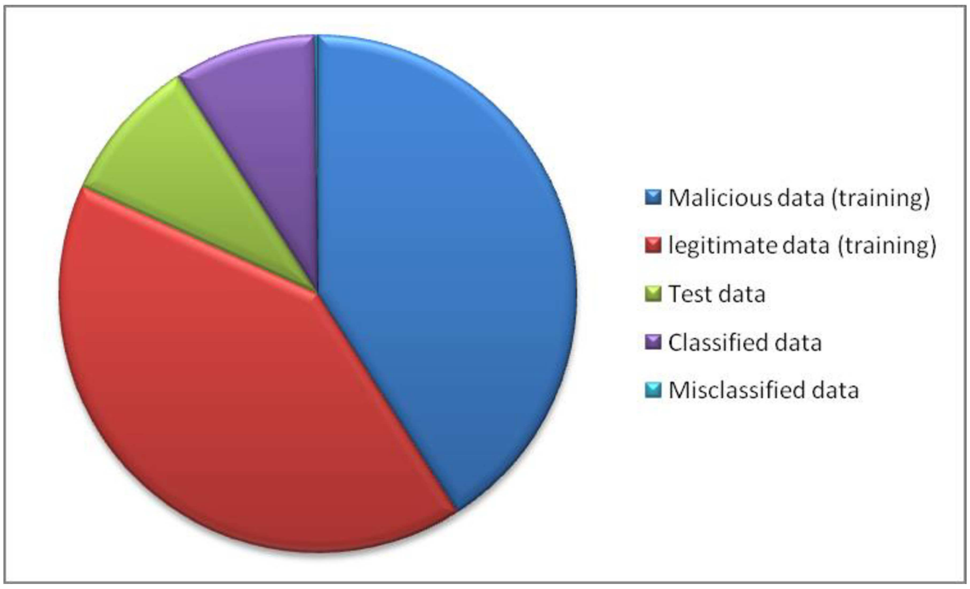


Figure 5.10: Performance of the classifier for K=9

From the Figure 5.10, The number of instances of malicious data used for the training is 1080, the number of legitimate data instances used for the training is 1080, the number of instances of test data (120 malicious data instances and 120 legitimate data instances) is 240. The number of correctly classified data is 235 in this case with 5 misclassified data instances. The accuracy of this validation is 97.9%.

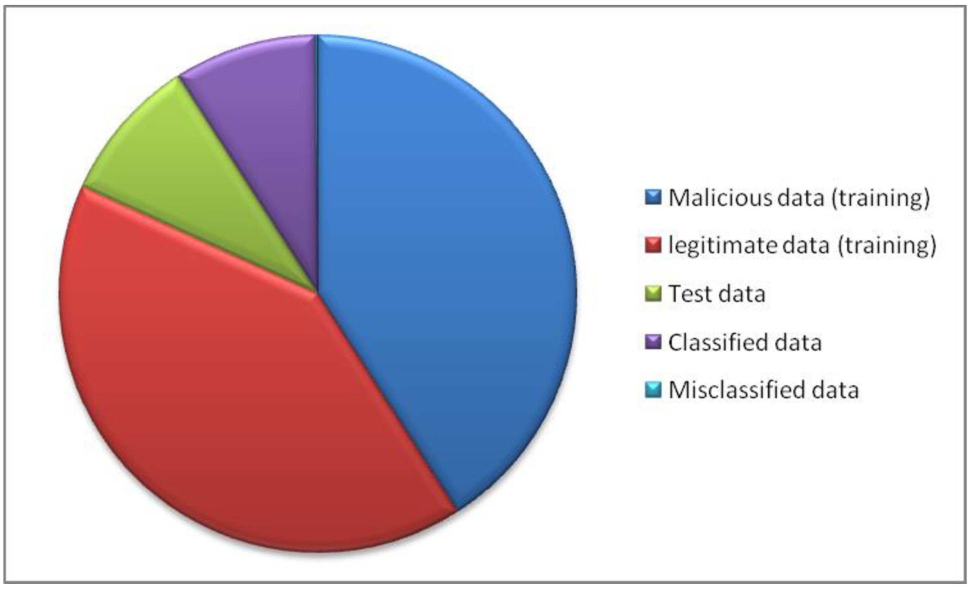


Figure 5.11: Performance of the classifier for K=10

From the Figure 5.11, the number of instances of malicious data used for the training is 1080, the number of legitimate data instances used for the training is 1080, the number of instances of test data (120 malicious data instances and 120 legitimate data instances) is 240. The number of correctly classified data is 236 in this case with 4 misclassified data instances. The accuracy of this validation is 98.3%.

K-fold training	Detection Rate (DR)
Fold 1	97.5%
Fold 2	97.9%
Fold 3	97.5%
Fold 4	100%
Fold 5	97.5%
Fold 6	97.9%
Fold 7	97.5%
Fold 8	98.3%
Fold 9	97.9%
Fold 10	98.3%

Table 5.1: Performance of the classifier on the training data using K-fold validation

From the table 5.1, the Average performance measure of the Naive Bayes Classifier in this work is [(97.5*4) +(97.9*3)+(98.3*2)+(100*1)]/10 = 98.03%**.** Using 10-fold cross validation, the classifiers is quite precise which indicates the discriminative power of the features used based on attack-centric behaviour of the Exploit kits. The meanings of DR in this context is the percentage of correctly classified data. The advantage of this approach is high accuracy, ability to modify the classifier (features) and very little time to classify the data.

5.3 Accuracy of the classifier on the Test Set

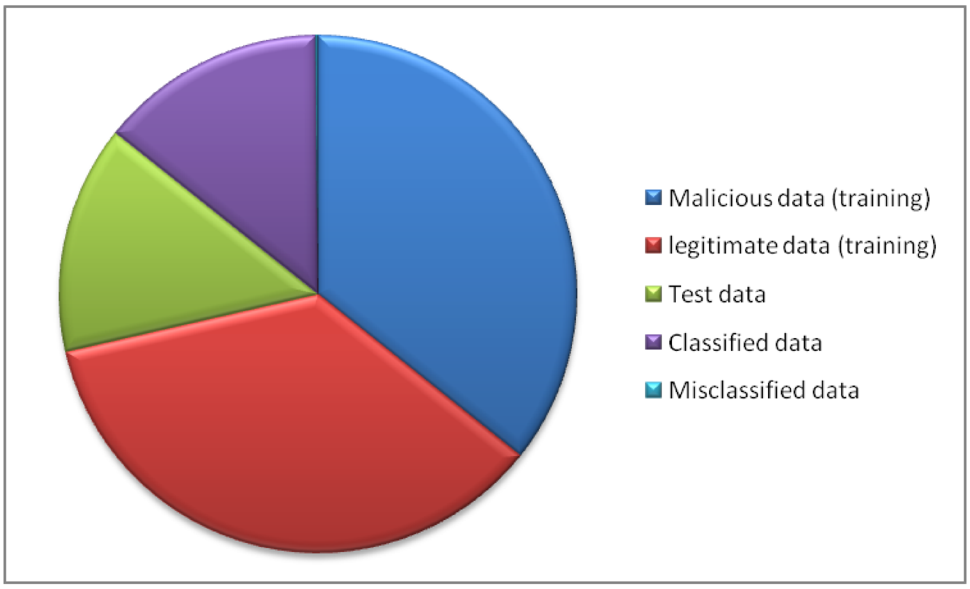


Figure 5.12: Performance of the classifier on the test data

From the Figure 5.12, the number of instances of malicious data used for the training is 1000, the number of legitimate data instances used for the training is 1000, the number of instances of test data (211 malicious data instances and 189 legitimate data instances) is 400. The number of correctly classified data is 396 in this case with 4 misclassified data instances. The accuracy of this validation is 99%.

Naive Bayes Classifier	Detection Rate (DR)
Test data	99%

Table 5.2: Performance of the classifier on the training set

The accuracy of Classifiers on the Testing Set shows the accuracy of the classifiers on an independent test set. Overall, the classifier has achieved about 99% accuracy which can be seen from table 5.2. As shown in the results, the classifier is precise enough to correctly classify samples disjoint with the training set. The meanings of DR in this context is the percentage of correctly classified data.

5.4 Accuracy of Clustering

The performance of the K-means clustering algorithm on the training set:

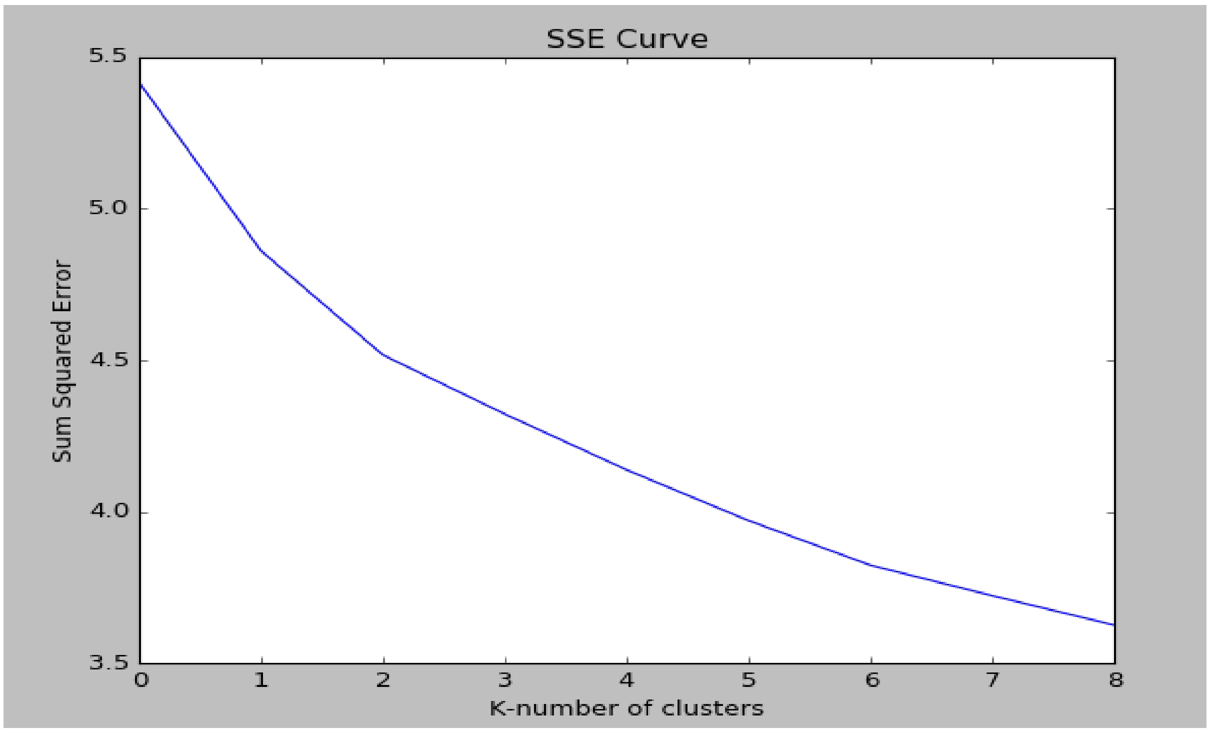


Figure 5.13: Calculation of the value of K using Elbow method for the test data

The Figure 5.13 shows the calculation of the ideal number of clusters based on Elbow method .The fundamental idea of this method is for a range of values of *k* , k-means clustering is done on the dataset and the sum of squared errors (SSE) is calculated and plotted. Then the "elbow" on the arm in the graph is chosen as the value of *k* as it has a low SSE. Then the data is clustered into K groups.

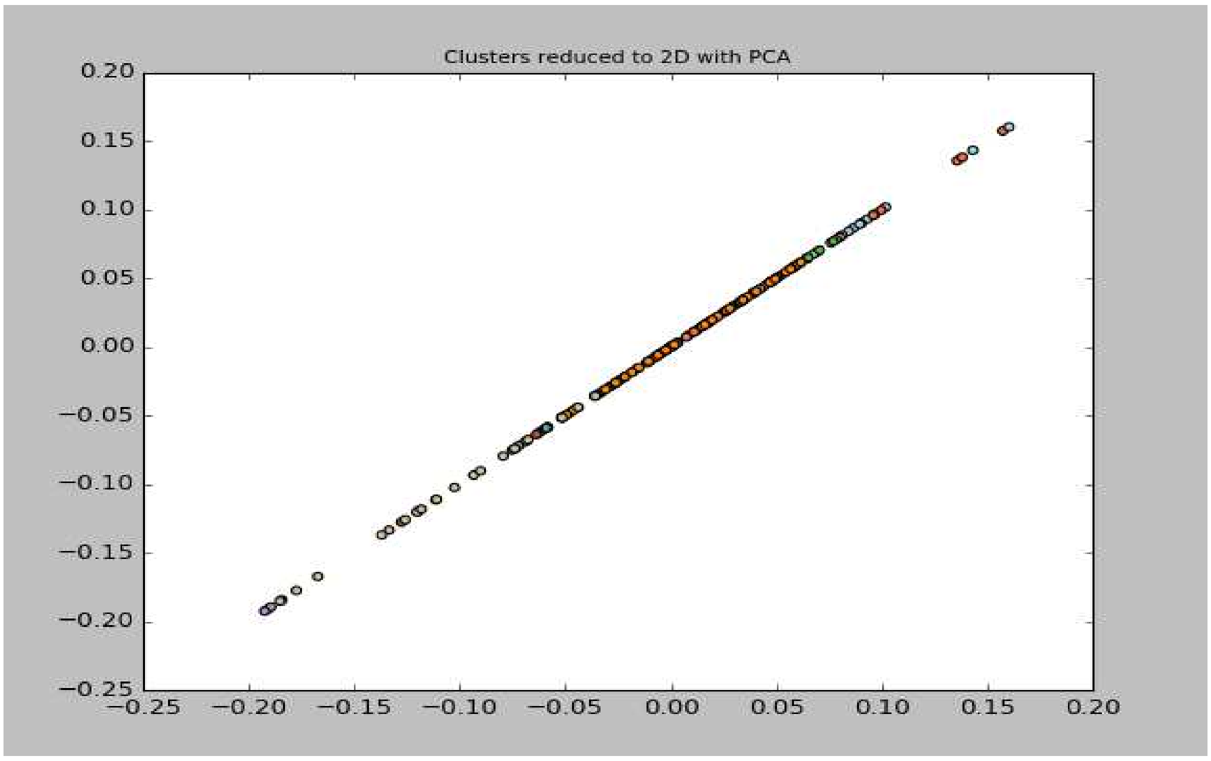


Figure 5.14: Visualisation of clusters of the test data using Principal Component Analysis

The Figure 5.14 shows the visualisation of the clusters after being reduced to two dimensions using Principal component analysis. This graph shows the 9 groups of different types of Exploit kits present in the test data.

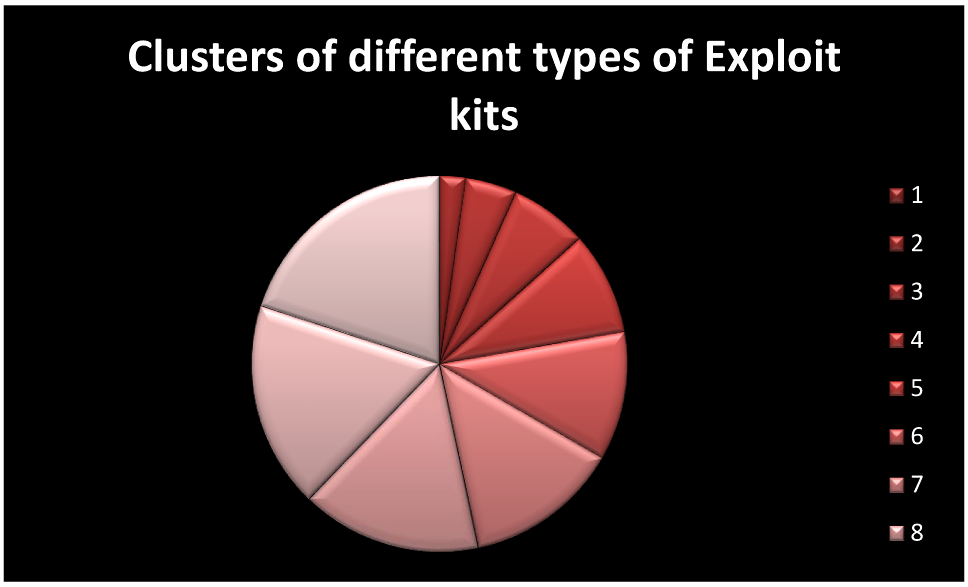


Figure 5-15: Distribution of different types of Exploit kits in the test data

The Figure 5-15 shows the distribution of different types of Exploit kits present in the test data. The accuracy of K-means clustering on the Testing Set is between 75%- 85%. This is calculated using manual analysis of the clusters formed by the algorithm . As shown in the results, the classifier is decent enough to group samples from the training set in to different groups of Exploit kits based on the domain name and URL.

In terms of verification, although the data originally came without label, it now can be treated as labelled data since this work picked the samples from the dataset and manually labelled them. In addition, the sample size is relative small, this work manually went through the detection results to verify that the detectors have done their job right.

Chapter 6 Conclusions

Exploit Kits contribute to a significant portion of web-based threats on the internet today, creating a staging ground for further malicious criminal activities. Through the extensive history of exploit kit attacks causing problems for innocent internet users, it is clear that it is important to have an effective system in place to control the damage. To minimize the threat they pose to the internet, it is important to understand how they function (entry point, distribution, exploit, infection, and execution, to launch an attack) to detect and defend users from further attacks.

A typical exploit kit attack usually begins with a victim being lured to a URL (e.g., by clicking a link in a spam email or an advertisement found on a legitimate webpage) to visit a seemingly legitimate web page. This is followed by a series of redirections, and the victim reaches an exploit kit landing page. The kit then scans for vulnerabilities to exploit by gathering information about the victim. At this stage, a kit analyses the User-Agent information of the victim (type and version number of the operating system, browser, and third-party plugins) so it can serve suitable exploits (pre-written exploit code which usually include installer scripts, number of exploits, configuration details, administration features for the kit owner, and payload) to the victim based on this information. Examples of (historically) popular Exploit Kits include RedKit, Angler, Blackhole, NuclearPack, Sakura, NeoSploit, Siberia Private, and Styx. [11]

This work has trained Naive Bayes classifier on the features derived from attack centric behaviour of the Exploit kits. The data set up to train the classifier involves data which cover all the different stages present in a typical Exploit kit attack workflow and provides a very good detection rate on the test data. The contributions of the approach include a machine learning approach that makes use of distinct features drawn from the attack-centric behaviour, an implementation and evaluation of the approach , clustering of the detected Exploit kits URL and domain name, visualisation of the clusters that contain different types of Exploit kits and an implementation of information gain algorithm for ideal feature selection from a machine learning point of view.

The features selected for effective detection of Exploit kits in this work are host (Connection-Specific), URI (Connection-Specific), referrer (Connection-Specific), user_agent (Connection-Specific), status_code (Connection-Specific) and mime_type (Content-Specific).These aggregate values of features, in fact, are very good at putting apart Exploit kit and non-Exploit kit related traffic. The connection-Specific and Content-Specific features used in the work are backed up by the statistical variations in the graph which are indicators of the discriminative power of these features in practice. The features selected guarantee a high true positive rate and very low false positives rate in the detection.

The average performance measure of the Naive Bayes Classifier on the training set in this work is 98.03%**.** Using 10-fold cross validation, the classifiers is quite precise which indicates the discriminative power of the features used based on attack-centric behaviour of the Exploit kits. The advantage of this approach is its accuracy, ability to modify the classifier (features) and very little time to classify the data.

The accuracy of classifiers on the testing test shows the accuracy of the classifiers on an independent test set. Overall, the classifier has achieved about 99% accuracy. As shown in the results, the classifier is precise enough to correctly classify samples disjoint with the training set.

The Naive Bayes classifier can help answer the question whether a given URL points to an Exploit kit. The answer to this question can have significant implications for the safety of end-users on the Internet, considering the rate of victims that have been affected by Exploit kits over the last few years. This information can be used by search engines which can use detection techniques to prevent indexing of these malicious URLs . These URLS could also be blacklisted and this information could be passed on to end-users and browser vendors. They can be protected from harmful infection. The signature dataset maintained by anti-virus companies can be updated with these URLs. The take-down operations against malicious activities on the Internet can make use of identifying these Exploit Kits. This approach helps detect malicious URLs that are used by an Exploit kit which is useful when new Exploits are added by Exploit kits by using machine learning. [11]

The accuracy of K-means clustering on the testing set is between 75%- 85%. In terms of verification, although the data originally came without label, it now can be treated as labelled data since this work has picked the samples from the dataset and manually labelled them. In addition, the sample size is relative small, this work went through the detection results manually to verify that the detectors have done their job right. As shown in the results, this approach is decent enough to group samples from the training set in to different groups of Exploit kits based on the domain name and URL.

The practical use of grouping the various types of Exploit kits could help analysts in knowing what type of Exploit kits are being used to target their network and/or customers. This is of course if that network does not already employ an IDS system, which already has signatures to detect the types of Exploit kits. This makes most sense when a customer does have HTTP logs, in this case Bro is used, but such HTTP logs could also come from a proxy or firewall log. In this work, machine learning algorithm could identify the Exploit kits that were seen, in retrospect. A lot of companies do have HTTP logs available, and not PCAP files (which is what an IDS would use). Knowing the type of Exploit kits is useful, because most Exploit kits have different types of Exploits. For example Exploit kit A could have an Exploit for all Java versions from 2015, while Exploit kit B would only have Exploits for Java versions from 2014. Another example: Exploit kit C could only have Exploits for Java, while Exploit kit D only has Exploits for Flash. Knowing the types of Exploits served by various Exploit kits can help an analyst in assessing the possible damage the Exploit kit could to the end users of a certain network. For example if Exploit D only employs Flash Exploits, while all end users in the attacked network do not have Flash installed, there is little damage to be done by this specific Exploit kit.

6.1 Future work

Despite the significant progress made in this thesis, there remain several open exciting challenges with detection of Exploit kits for large-scale machine learning. In the following list, some promising topics are mentioned that can be of worth to future research projects.

- ✓ Considering the entire PCAP for detection
- ✓ Considering other Bro logs for detection purposes
- ✓ More training data and data which includes the labels of the Exploit kit they belong to

- ✓ Analysis of the formed clusters
- ✓ Considering temporal features which can bring context to the picture

7 References

[1]Luo, T., & Jin, X. NEXT-GENERATION OF EXPLOIT KIT DETECTION BY BUILDING SIMULATED OBFUSCATORS.

[2]Grier, Chris, et al. "Manufacturing compromise: the emergence of Exploit-as-a-service." *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012.

[3]https://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-evolution-of-Exploit-kits.pdf

[4]Mitchell, Tom Michael. *The discipline of machine learning*. Vol. 9. Carnegie Mellon University, School of Computer Science, Machine Learning Department, 2006.

[5]Rätsch, Gunnar. "A brief introduction into machine learning." *21st Chaos Communication Congress*. 2004.

[6]Kotsiantis, Sotiris B., I. Zaharakis, and P. Pintelas. "Supervised machine learning: A review of classification techniques." (2007): 3-24.

[7]Kevin, Murphy. "Machine Learning: a probabilistic perspective." (2012).

[8]Yang, Y. and J. P. Pedersen, "A Comperative Study on Feature Selection in Text Categorization", The Fourteenth International Conference on Machine Learning, pp. 412–420, 1997.

[9]Mitchell, T. M., Machine Learning, McGraw Hill, 1997.

[10] McCallum, A. and K. Nigam, "A Comparison of Event Models for Naive Bayes Text Classification", AAAI-98 Workshop on Learning for Text Categorization, 1998.

[11]Eshete, Birhanu, and V. N. Venkatakrishnan. "Webwinnow: Leveraging Exploit kit workflows to detect malicious urls." *Proceedings of the 4th ACM conference on Data and application security and privacy*. ACM, 2014.

[12]http://setosa.io/ev/principal-component-analysis/

[13]https://mariuszprzydatek.com/2014/10/31/measuring-entropy-data-disorder-and-information-gain/

[14]Paxson, Vern. "Bro: a system for detecting network intruders in real-time."*Computer networks* 31.23 (1999): 2435-2463.

[15]https://www.bro.org/why_choose_bro.pdf

[16]Yang, Y. and X. Liu, "A Re-examination of Text Categorization Methods", Proceedings

of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval, Berkeley, US, 1999

[17]Joachims, T., “A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization”, ICML-97 , 1997

[18]Burges, C. J. C., “A Tutorial on Support Vector Machines for Pattern Recognition”, Data Mining and Knowledge Discovery, Vol. 2, No. 2, pp. 121–167, 1998

[19]Eberhardt, Jeremy J. "Bayesian spam detection." *Scholarly Horizons: University of Minnesota, Morris Undergraduate Journal* 2.1 (2015): 2.

[20]McCallum, Andrew, and Kamal Nigam. "A comparison of event models for naive bayes text classification." *AAAI-98 workshop on learning for text categorization*. Vol. 752. 1998.

[21]Shalev-Shwartz, Shai, and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[22]Kodratoff, Yves. *Introduction to machine learning*. Morgan Kaufmann, 2014.

[23]Khanum, Memoona, et al. "A Survey on Unsupervised Machine Learning Algorithms for Automation, Classification and Maintenance." *International Journal of Computer Applications* 119.13 (2015).

[24]Bonev, Boyan. *Feature selection based on information theory*. Universidad de Alicante, 2010.

[25]http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

[26]http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html

[27]Birhanu Eshete, Adolfo Villafiorita, Komminist Weldemariam, Mohammad Zulkernine. EINSPECT: Evolution-Guided Analysis and Detection of Malicious Web Pages. In Proceedings of the International Conference on Computer Software and Applications (COMPSAC), pages 375-380, IEEE, 2013.

[28]Birhanu Eshete, Adolfo Villafiorita. Effective Analysis, Characterization, and Detection of Malicious Web Pages. In Proceedings of the International Conference on World Wide Web (WWW) Campanion, pages 355-360, ACM, 2013.

[29]Birhanu Eshete, Adolfo Villafiorita, Komminist Weldemariam. BINSPECT: Holistic Analysis and Detection of Malicious Web Pages. In Proceedings of the International Conference on Security and Privacy in Communication Networks (SECURECOMM), pages 149-166, Springer-Verlag, 2012.

[30]Birhanu Eshete, Adolfo Villafiorita, Komminist Weldemariam. Malicious Website Detection : Effectiveness and Efficiency Issues, In Proceedings of the System Security Workshop (SysSec), pages 123- 126, IEEE, 2011.

[31]Vadim Kotov and Fabio Massacci. Anatomy of Exploit Kits - Preliminary Analysis of

Exploit Kits as Software Artefacts. In ESSoS, pages 181–196, 2013.

[32]http://www.delaat.net/rp/2015-2016/p24/report.pdf

[33]http://people.csail.mit.edu/kalyan/AI2_Paper.pdf

[34]http://stanford.edu/class/ee103/lectures/documents/documents_slides.pdf

[35]https://www.coursera.org/learn/ml-foundations/lecture/i8ilf/creating-the-word-count-vector

[36]Maas, Andrew L., et al. "Learning word vectors for sentiment analysis." *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011.

[37]Largeron, Christine, Christophe Moulin, and Mathias Géry. "Entropy based feature selection for text categorization." *Proceedings of the 2011 ACM Symposium on Applied Computing*. ACM, 2011.

[38]Justin Ma. Learning to Detect Malicious URLs. PhD thesis, University of California, San Diego, 2010.

[39]Cavnar, William B., and John M. Trenkle. "N-gram-based text categorization." *Ann Arbor MI* 48113.2 (1994): 161-175.

[40]Khanum, Memoona, et al. "A Survey on Unsupervised Machine Learning Algorithms for Automation, Classification and Maintenance." *International Journal of Computer Applications* 119.13 (2015).

[41]Smola, Alex, and S. V. N. Vishwanathan. "Introduction to machine learning." *Cambridge University, UK* 32 (2008): 34.

[42]https://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-evolution-of-Exploit-kits.pdf

[43]Stock, Ben, Benjamin Livshits, and Benjamin Zorn. "Kizzle: A Signature Compiler for Detecting Exploit Kits."

[44]De Maio, Giancarlo, et al. "Pexy: The other side of Exploit kits." *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer International Publishing, 2014.

[45]Šrndić, Nedim, and Pavel Laskov. "Hidost: a static machine-learning-based detector of malicious files." *EURASIP Journal on Information Security*2016.1 (2016): 22.

[46]Kolter, J. Zico, and Marcus A. Maloof. "Learning to detect and classify malicious executables in the wild." *Journal of Machine Learning Research*7.Dec (2006): 2721-2744.

[47]Tang, Adrian, Simha Sethumadhavan, and Salvatore J. Stolfo. "Unsupervised anomaly-based malware detection using hardware features." *International Workshop on Recent Advances in Intrusion Detection*. Springer International Publishing, 2014.

[48]Cohen, William W. "Fast effective rule induction." *Proceedings of the twelfth international conference on machine learning*. 1995.

[49]https://jair.org/media/3908/live-3908-7454-jair.pdf

[50]Bottou, Léon. "From machine learning to machine reasoning." *Machine learning* 94.2 (2014): 133-149.

[51]http://www.cs.rutgers.edu/~elgammal/classes/cs536/lectures/ANN.pdf

[52]http://kioloa08.mlss.cc/files/hutter1.pdf

[53]https://www.ics.uci.edu/~dechter/papers/paginated_binders/PART%25201.pdf

[54]http://wwwis.win.tue.nl/~tcalders/teaching/datamining09/slides/DM09-07-Clustering.pdf

[55] http://www.inf.ed.ac.uk/teaching/courses/inf2b/learnnotes/inf2b-learn-note07-2up.pdf

[56] http://machinelearningmastery.com/an-introduction-to-feature-selection/

[57]Kotov, Vadim, and Fabio Massacci. "Anatomy of Exploit kits." *International Symposium on Engineering Secure Software and Systems*. Springer Berlin Heidelberg, 2013.

[58] Allodi, Luca, Vadim Kotov, and Fabio Massacci. "Malwarelab: Experimentation with cybercrime attack tools." *Presented as part of the 6th Workshop on Cyber Security Experimentation and Test*. 2013.

[59]Grier, Chris, et al. "Manufacturing compromise: the emergence of Exploit-as-a-service." *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012.

[60]Eshete, Birhanu, et al. "EKHunter: A Counter-Offensive Toolkit for Exploit Kit Infiltration." *NDSS*. 2015.

[61]Kapravelos, Alexandros, et al. "Revolver: An automated approach to the detection of evasive web-based malware." *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*. 2013.

[62]Cova, Marco, Christopher Kruegel, and Giovanni Vigna. "Detection and analysis of drive-by-download attacks and malicious JavaScript code." *Proceedings of the 19th international conference on World wide web*. ACM, 2010.

[63]Rieck, Konrad, Tammo Krueger, and Andreas Dewald. "Cujo: Ecient Detection and Prevention of Drive-by-Download Attacks."

[63]http://www.nlpca.org/pca_principal_component_analysis.html

[64]Salton, Gerard, Anita Wong, and Chung-Shu Yang. "A vector space model for automatic indexing." *Communications of the ACM* 18.11 (1975): 613-620.

[65]https://blog.malwarebytes.com/cybercrime/Exploits/2016/09/rig-Exploit-kit-takes-on-large-malvertising-campaign/

[66]https://en.wikipedia.org/wiki/Exploit_kit

[67]http://odur.let.rug.nl/~vannoord/TextCat/textcat.pdf

[68]http://scikit-learn.org/stable/