HTTP BOTNET DETECTION USING CLASSIFICATION TECHNIQUE

LEE KHER XIN

BORANG PENGESAHAN STATUS TESIS

JUDUL: HTTP BOTNET DETECTION USING CLASSIFICATION TECHNIQUE

SESI PENGAJIAN: 2015/2016

Saya LEE KHER XIN
(HURUF BESAR)

mengaku membenarkan tesis (PSM/~~Sarjana~~/~~Doktor Falsafah~~) ini disimpan di Perpustakaan

Fakulti Teknologi Maklumat dan Komunikasi dengan syarat-syarat kegunaan seperti berikut:

1. Tesis dan projek adalah hakmilik Universiti Teknikal Malaysia Melaka.

2. Perpustakaan Fakulti Teknologi Maklumat dan Komunikasi dibenarkan membuat salinan untuk tujuan pengajian sahaja.

3. Perpustakaan Fakulti Teknologi Maklumat dan Komunikasi dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.

4. ** Sila tandakan (/)

______	SULIT	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
______	TERHAD	(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan dimana penyelidikan dijalankan)
/	TIDAK TERHAD	

[illegible]

(TANDATANGAN PENULIS)

Alamat tetap: 641,LRG ANGSANA 8/2,
TMN ANGSANA
09000,KULIM
KEDAH

Tarikh: 26 AUG 2016

[illegible]

(TANDATANGAN PENYELIA)

Nama Penyelia: DR. FAHMI ARIF

Tarikh: 26 AUG 2016

CATATAN: *Tesis dimaksudkan sebagai Laporan Projek Sarjana Muda (PSM)

** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa.

HTTP BOTNET DETECTION USING CLASSIFICATION TECHNIQUE

LEE KHER XIN

This report is submitted in partial fulfillment of the requirements for the Bachelor of Computer Science (Computer Security) with Honours

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TEKNIKAL MALAYSIA MELAKA 2016

DECLARATION

I hereby declare that this project report entitled

HTTP BOTNET DETECTION USING CLASSIFICATION TECHNIQUE

is written by me and is my own effort and that no part has been plagiarized without citations.

STUDENT : [illegible] Date : 26 AUG 2016

(LEE KHER XIN)

I hereby declare that I have read this project report and found this project report is sufficient in term of the scope and quality for the award of Bachelor of Computer Science (Computer Security) With Honours.

SUPERVISOR : [illegible] Date : 26 AUG 2016

(DR. FAHMI ARIF)

DEDICATION

This project is dedicated to Universiti Teknikal Malaysia Melaka (UTeM), which provide sufficient facility for me to conduct this project. It also dedicated to my great supervisor, who always teaches me how to solve the problem I faced during this project and supervises me to finish this project. Besides, it also dedicated to my beloved parents, who always give encouragement and support to me to complete this project. Then, this project also dedicated to my friends, who always give their cooperation to me during this study.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Fahmi Arif who giving me valuable guidance and advice that lead me to complete this project successfully. I would like to appreciate him for giving me idea and opinion about my project which helped me to understand my project better. Besides, he also taught me to think out of the box and think critically to generate a good solution to solve the problem I faced during my project. All of the guidance helped me to accomplish my project on time. I would also like to appreciate my evaluator for this project, En Mohd Zaki Bin Mas'ud for taking his time to evaluate me. This evaluation gave me a clearer understanding of my project and figure out the problem needed to solve in my project to make it better than what I already have.

I would also like to thank Universiti Teknikal Malaysia Melaka (UTeM) for providing me with a comfortable environment and sufficient facilities to finish this project. Finally, I would also like to thank my beloved families and friends for their motivations and supports for me throughout my project. With the motivation and support of everyone that was mentioned above, I am able to overcome many problems that occurred and able to complete my project successfully on time.

ABSTRACT

In this final year project, HTTP botnet was detected by using classification technique. Due to the unknown relationship between network parameter and botnet, it is difficult to detect HTTP botnet by analysis network traffic. Thus, it is requiring understanding the relationship between network parameter and botnet before classifying the malicious network traffic and non-malicious network traffic. The main objective of this project is investigating the network parameter of the network traffic. Next objective is to study the relationship between network parameter with HTTP botnet by using several classification techniques and compare which of the algorithm is more accurate for classifying the network parameter with botnet. Besides, the scopes of the investigation in this project are detecting the HTTP-based botnet only by a study on network traffic and using five type of classification technique to classify the malicious botnet traffic and non-malicious network traffic. The methodology processes of this project are literature review, data collection, pre-processing, classification and analysis and documentation. At the end of this project, it is able to identify the relationship of the network parameter with HTTP botnet and detect the HTTP botnet by classification the network parameter.

ABSTRAK

Dalam projek akhir tahun ini, HTTP botnet dikesankan dengan menggunakan teknik klasifikasi. HTTP botnet sukar dikesankan melalui analisis trafik rangkaian kerana hubungan yang tidak diketahui antara parameter rangkaian dengan botnet. Oleh itu, hubungan antara parameter rangkaian dengan botnet perlu difahami sebelum mula proses mengklasifikasikan trafik rangkaian yang berniat jahat dan trafik rangkaian yang tidak berniat jahat. Objektif utama projek ini adalah menyiasat parameter trafik rangkaian. Objektif kedua adalah mengkaji hubungan antara parameter rangkaian dengan HTTP botnet melalui penggunaan beberapa teknik klasifikasi. Objektif ketiga adalah pembandingkan algoritma yang lebih tepat untuk mengklasifikasikan parameter rangkaian dengan botnet. Selain itu, skop untuk projek ini adalah mengesan HTTP botnet melalui kajian trafik rangkaian dan menggunakan lima jenis teknik klasifikasi untuk mengklasifikasikan trafik rangkaian botnet dan trafik rangkaian biasa. Proses metodologi projek ini ialah kajian literatur, pengumpulan data, pra-pemprosesan, klasifikasi dan analisis dan dokumentasi. Pada peringkat akhir projek ini, hubungan antara parameter rangkaian dengan HTTP botnet dapat dikenalpastikan dan HTTP botnet dapat dikesankan mengikut klasifikasi parameter rangkaian.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ATTACHMENTS

CHAPTER I

INTRODUCTION

1.1 Project Background

Malware is the malicious software that used to interrupt computer operations, gather personal information, gain access to private computer systems or send unwanted advertising. Malware may be stealthy and intended to steal information or monitor the user activities for an extended period without the knowledge of the user. There is difference type of malware which possesses varying behavior such as virus, worm, rootkit, botnet, Trojan horse, spyware, adware and ransomware.

Botnet is one type of the malware that uses a collection of compromised computers to generate spam message, relay viruses or flood a network with excessive requests to cause the network fail, such as distributed denial of service (DDoS) attack. Those attacks are coming from a number of computers that are remotely controlled by a bot master that can be located anywhere across the globe. While HTTP botnet is one type of botnet which periodically visit certain web server to get updates or latest commands. Bot masters use HTTP protocol to hide their activities among the normal

HTTP flows and easily avoid detection methods like firewall. Nowadays, the HTTP services are being widely used by the Internet applications, it is not easy to block this service. Thus detection of the HTTP botnet has become a challenge.

The technique used in this study to detect the HTTP botnet is classification. Classification is a technique of data mining utilized to forecast group membership for data samples. Classification is the issue of recognizing which of a set of classes a new observation is classified, on the foundation of a training set of data containing observations which the class membership is known. The individual observations are analyzed into a set of quantifiable properties. Classification is utilized to reveal the uncommon behavior in network traffic. Classification method use supervised algorithms to appropriately label the command and control (C&C) channels in HTTP/HTTPS botnets by means of an in-depth analysis of the network traffic.

1.2 Problem Statement (PS)

In this information age society, most of the people prefer to complete their work through the internet. Thus, this makes an opportunity for the hacker to steal sensitive information with malware such as HTTP botnet. HTTP botnet is difficult to detect because it uses HTTP protocol to hide their communication flow. Since HTTP services are being used by many Internet application, it cannot simply close the service. Besides, it is difficult to detect HTTP botnet due to the unknown relationship between network parameter and botnet. Table 1.1 shows the summary of the problem statement.

Table 1.1: Summary of problem statement

PS	Problem Statement
PS1	Difficult to detect HTTP botnet using a network traffic analysis due to unknown relationship between network parameter and botnet.

1.3 Project Question (PQ)

Table 1.2 shows the three project questions of this project.

Table 1.2: Summary of project question

PS	PQ	Project Question
PS1	PQ1	What are the network parameters existing in the network traffic?
	PQ2	How is the relationship between each network parameter with botnet?
	PQ3	Which algorithm is the best in finding out relationship between network parameter and HTTP botnet?

1.4 Project Objective (PO)

Table 1.3 shows the three objectives of the project.

Table 1.3: Summary of project objective

PS	PQ	PO	Project Objective
PS1	PQ1	PO1	To investigate the network parameter of the network traffic.
	PQ2	PO2	To study the relationship between network parameter with HTTP botnet using several classification techniques.
		PO3	To compare which of the algorithm is more accurate for classify the network parameter with botnet.

1.5 Project Scope (PS)

The field of the investigation in this project is detecting the HTTP botnet by study on the network traffic. The technique used to detect malicious botnet traffic from the network traffic is classification. Classification using supervised algorithms to identify the relationship of the network parameter with the HTTP botnet. The type of botnet detect in this project is HTTP botnet. Only six types of HTTP botnets are release in a control network such as Dorkbot, Zeus, Citadel, Spyeye, Cutwail and Waledac. The network parameter was examined and categorized to separate HTTP botnet network traffic and non-malicious network traffic.

1.6 Project Contribute (PC)

This project helps to enhance Intrusion Detection System (IDS) or Firewall to detect HTTP botnet. The output of this project is to detect the HTTP botnet by classification the network parameter to identify the relationship of the network parameter with HTTP botnet. At the end of this project, this project can differentiate the non-botnet network traffic and botnet network traffic.

1.7 Thesis Organization

Chapter 1: Introduction

This chapter discusses the introduction of the project, problem statement, project question, project objective, project scope and project contribution.

Chapter 2: Literature Review

In this chapter, among 20 journals needed to study and produce the literature review. From the journal, the related work of this project, the critical review of the current problem and the proposed solution of the project were list out. The literature review can help us more understand about the title of this project.

Chapter 3: Project Methodology

This chapter discusses the methodology of the project which will be carried out and the project milestones to ensure all the activity is complete on time. In this chapter, the software and hardware requirement involve in the project was list out.

Chapter 4: Design

This chapter discusses the analysis of the initial design and the result of the detail design. The design of the network architecture is discussed in this chapter. The network design will increase the understanding of the arrangement of the network device and the IP addressing of the workstation. Throughout this chapter, this project can have a clear idea on the possible scenario.

Chapter 5: Implementation

This chapter describes the activity involved in the implementation phase and the expected output after complete this chapter. During this section, data-preprocessing was been executed to fill in missing value or delete noisy data of the dataset before implement the dataset inside classification process.

Chapter 6: Testing and Analysis

This chapter discusses the activity involved in the testing phase and what is the strategy of testing used in this project. In this section, the design of the classification process and the result of the classification was studied and analyzed.

Chapter 7: Project Conclusion

This chapter discusses the summarization of the project such as the strength and weakness of the project, contribution of the project, limitation of the project and the future works of the project.

1.8 Summary

At the end of this chapter, the problem statement, project question, project objective, project scope, and project contribution was stated. Next chapter will discuss the literature review of the project and the related work of this project.

CHAPTER II

LITERATURE REVIEW

2.1 Introduction

This chapter discusses the literature review and project methodologies of HTTP botnet detection by using classification technique. This project title is separate into 3 main parts which are botnet definition, malware detection and machine learning. The literature review discusses based on the 3 main part of the project title to ensure the literature is related to the topic and achieve the project objective. This chapter states the definition of the botnet which includes IRC botnet, Peer-to-Peer (P2P) botnet and HTTP botnet. The scope of the botnet of this project is HTTP botnet. The feature of HTTP botnet and the latest issues related to the HTTP botnet were also discussed in this chapter. For the part of malware detection, it consists of several types of malware detection technique which are anomaly-based detection and signature-based detection. This project is using anomaly-based detection technique to detect the malware based network traffic. Besides, the machine learning part was introduced the supervised learning and unsupervised learning. The classification method used in this project is under supervised learning. Inside classification, the algorithms such as Naïve Bayes, Support Vector Machines, Artificial Neural Network and Decision Trees and the evaluation criteria of the classification algorithms were discussed in this project.

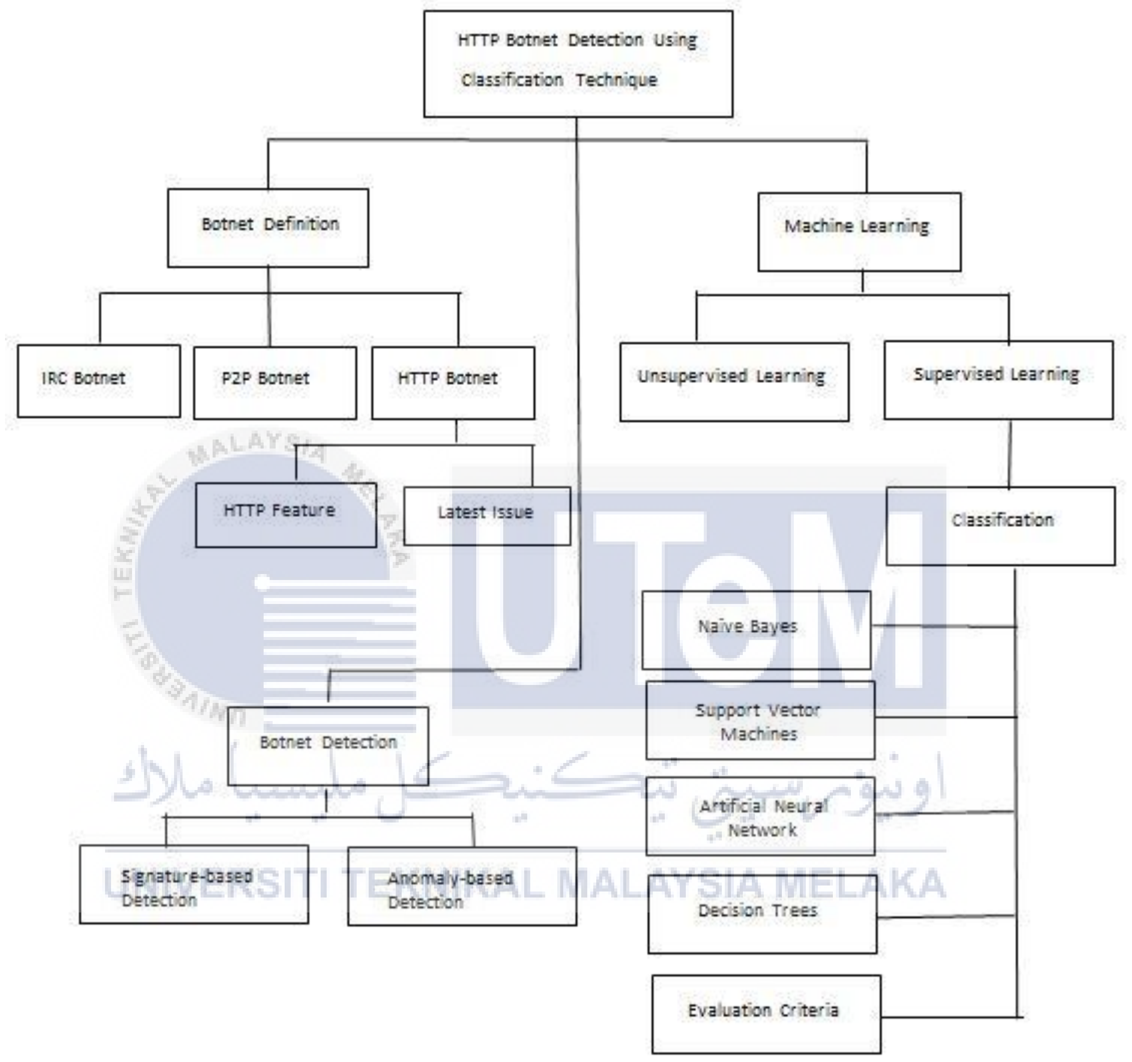


Figure 2.1: Framework of literature review

2.2 Botnet Definition

In this information age society, botnets are the new emerge dangerous threats on the Internet to carry out the sophisticated cybercrimes. A botnet is a collection of compromised computer that the botmaster can remotely control it to carry out malicious activities (Saad et al., 2011). A botmaster is a person who gives the command to C&C server. The bots will receive and responds to that command from the C&C server and execute the malicious activities which order by the botmaster. Botnets are used to carry out a distributed denial of service (DDOS), execute a click-fraud trick, spread spam, or steal individual sensitive information such as email address and credit card numbers. The lifecycle of a botnet can be divided into four phases which are formation, C&C, attack and post-attack phases (Saad et al., 2011). During the phase of formation, botmaster will continuously increase the bot number by infecting other victim machines on the network. Some botnet has the propagation tactics as worms which are duplicate and propagate themselves automatically. Then the infected victim machine will act as a bot that will receive the instructions from the botmaster regularly during the C&C phase. During the attack phase, the bots will execute the instruction from the botmaster to carry out malicious activities. After the attack phase, the botmaster will try to probe the botnet to gain information about active bots and plan for the formation of new bots. This is because some bots may be detected and removed during the phase of an attack. Botnets can be classified as IRC-based, P2P-based and HTTP-based botnets. Moreover, the botnet investigated in this project is HTTP-based botnet.

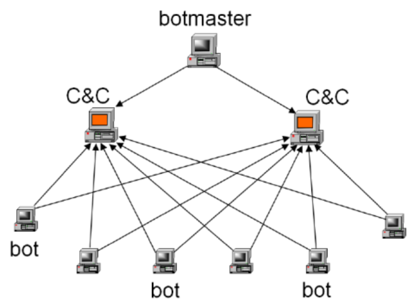


Figure 2.2: Command and control architecture of a botnet

Source: Wang et al. (2010)

2.2.1 IRC Botnet

IRC botnet is the first generation of botnet which using Internet Relay Chat (IRC) protocol and IRC servers always build up a central C&C server to issue the botmasters' command (Eslahi et al., 2013). The botmaster will communicate and control the bots by using established IRC command and C&C channels. The botmaster will keep control the bot as long as possible. The bots will only respond when the botmaster pushes new commands to the botnet. The IRC botnets are easy to control, manage and execute. The weakness of IRC botnets is it will face the problem of central point of failure. To solve this problem, P2P botnet is designed with no central point to shut down the botnet.

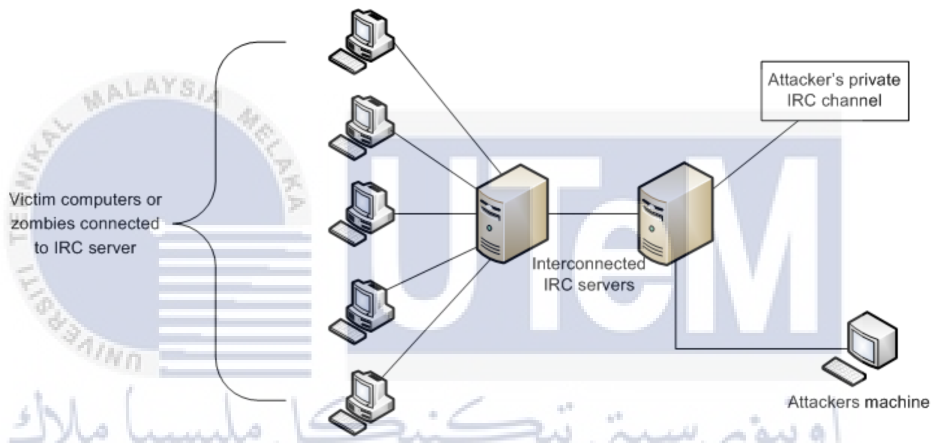

Victim computers or zombies connected to IRC server
Interconnected IRC servers
Attacker's private IRC channel
Attackers machine

Figure 2.3: A centralized IRC botnet

Source: Feily, Shahrestani, & Ramadass (2009)

2.2.2 P2P Botnet

P2P botnet uses peer-to-peer communication which is P2P based C&C channel to proxy command from the botmaster (Fedynyshyn et al., 2011). P2P botnet produces a network structure without a central C&C server which is difference from the network structure of IRC botnet and the individual bot plays a role as both server and client (Zhao et al., 2013). In P2P networks, communication between botmaster and bots forms unforeseeable paths. The botmaster sends an instruction to any one or two bots and the bot will distribute the instruction to their neighbors. The advantage of P2P botnet is it does not have a central point of failure. Besides, the absence of centralized C&C makes it hard to hijack the botnet or locate the botmaster. Unfortunately, with this decentralized architecture of the P2P botnet, the botmaster is very hard to measure the size of a network structure of P2P botnet. In this decentralized architecture, shutdown of a proxy server may cause part of the botnet unable to function normally. Thus, in the P2P topology, shutdown a single bot will not affect the entire P2P botnet as alternate paths are available. Nugache is one type of encrypted P2P botnet which able to evade the botnet detection techniques.

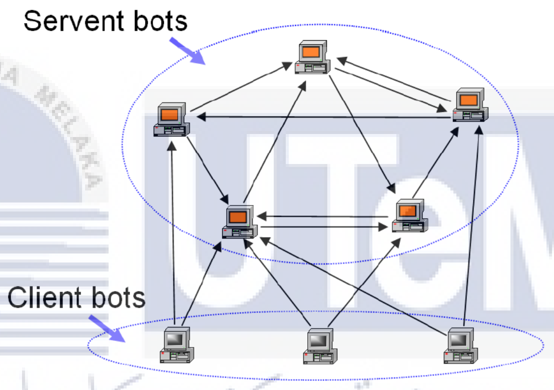


Figure 2.4: The architecture of P2P botnet

Source: Wang et al. (2010)

2.2.3 HTTP Botnet

HTTP botnet also is a central C&C model which same with the IRC botnet. HTTP botnet use HTTP protocol to distribute the malicious commands on web servers (Eslahi et al., 2013). HTTP bot periodically to gain updates or new instructions by visit certain web servers instead of always remaining in the mode of connection. This model is defined as PULL style. HTTP botnet uses a pull-based model to distribute their malicious command. The pull-based model means the bots will continue at a regular interval to request latest commands from the C&C server. To bypass traditional firewall based security, HTTP botnet can be stealthy itself by hijacking a authorize communication channel and encrypt the network packets to evade detection based on deep packet analysis (Zhao et al., 2013). The main drawback of HTTP botnet is the entire botnet can be easily disrupted by shutting the HTTP server because HTTP botnet is based on centralized architectures. It is difficult to obstruct HTTP botnet because the wide range of the HTTP service is used. Then, the detection of HTTP botnet is more difficult, because there are many applications and services on the Internet nowadays frequently use HTTP protocol to communicate between each other.

2.2.4 Latest Issues Related to HTTP Botnet

Estonia had received a massive Distributed Denial of Service (DDoS) attacks in 2007 reflected a paradigm change in relation to the protection of critical facilities. The capabilities of botnets have been proved to paralyze service online, including government servers, payment platforms or financial entities (Puerta et al., 2013). In February 2006, the network system of the Northwest Hospital at Seattle acts strangely. After the investigation, the investigator realizes that the network system of the Hospital was attacked by the botnet (Strayer et al., 2006). In August 2005, Britain's NISCC issued a warning, because they found the Trojan activity is increase which aims the UK government network. The attacker wants to collect the high sensitive valuable information of UK government. Content delivery network company CloudFlare in UK also surfaced DDoS attack which has involved mobile advertisements capable of generating around 275000 HTTP request per second. This situation is known as a Layer 7 HTTP flood attack. The HTTP botnet attack is hard to detect because the requests send by the attacker is look like legitimate traffic. These requests can overload the server or make the site down.

2.2.5 HTTP Feature

Hypertext Transfer Protocol (HTTP) is a protocol at the application level that used for distributed, collaborative and hypermedia information systems. HTTP is a generic protocol that used by the World Wide Web for data communication since 1990. HTTP has a feature which involves the representation of data and allows the system to transfer data independently. HTTP/1.0 is a common protocol in the Multipurpose Internet Mail Extensions (MIME) format which contains metainformation during the process of transferring data on the status of request or respond between client and server (Fielding et al., 1999). HTTP is a request-response protocol and the HTTP header carries much information about the browser of client, server, requested page and other (Mah 1997).

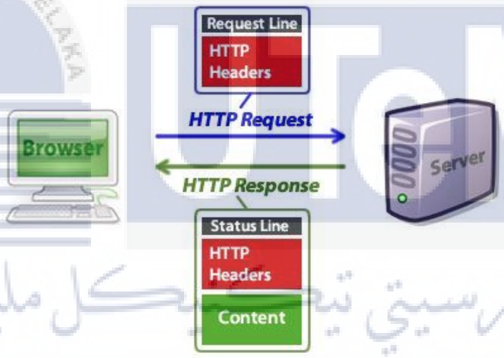


Figure 2.5: HTTP request and responds between client and server

Source: Burak Guzel (2009)

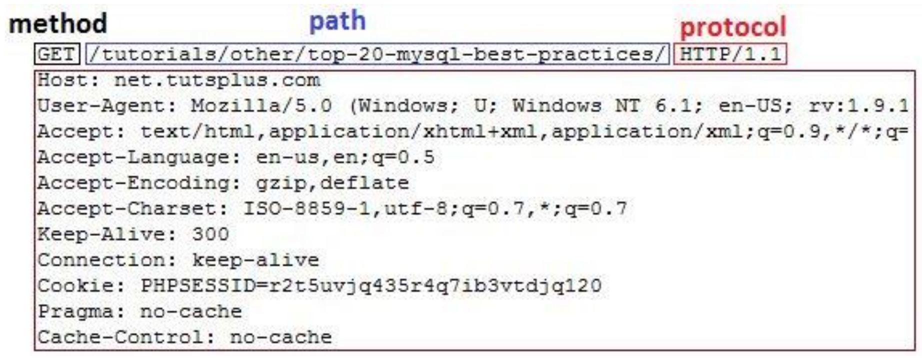
2.2.5.1 HTTP Request

HTTP header for request message is a message send from client to server. The first line of the request message can divide into three parts which are method, path and protocol. Method is the action to be performed on the request. The examples of method are OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, and CONNECT. The three common methods are GET, POST and HEAD. Path is the Request-URI which is

Uniform Resource Identifier and identifies which resource applies to the request (Fielding et al., 1999). Protocol part contains the version of the HTTP protocol. Table 2.1 shows the definition of the methods in the HTTP header.

Table 2.1: Description of methods

Methods	Description
OPTIONS	A request of message about the options of communication available on the chain of request or response.
GET	A request of retrieve information or data.
HEAD	A request same with GET but server does not return body of message in the process of response. It used to test the validity of the hypertext links.
POST	A request that used to request the server receive the substance include in the request as a new subordinate of the resource.
PUT	A request that used to store the enclosed entity under the supplied URI.
DELETE	A request that used to remove the define resource.
TRACE	A request that used to echoes a request message.
CONNECT	A request that used to dynamically switch a proxy to become a tunnel.



```
GET /tutorials/other/top-20-mysql-best-practices/ HTTP/1.1
Host: net.tutsplus.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120
Pragma: no-cache
Cache-Control: no-cache
```

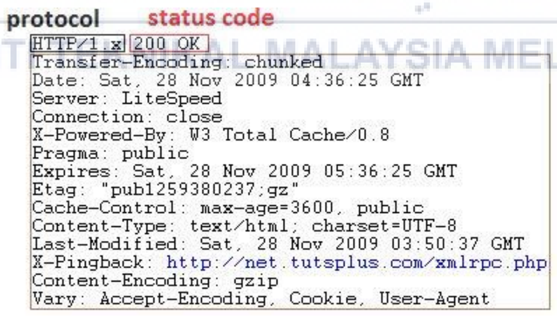
Figure 2.6: HTTP request structure

Source: Burak Guzel (2009)

User agent consists the information about the user who begins the request (Fielding et al., 1999). The user agent contains the information of the name and the version of the browser and operating system and the default language. Accept language is the set of language that favors as a reply to the request. Accept encoding is the type of encoding which is acceptable.

2.2.5.2 HTTP Response

The server will give back a responds with a HTTP response to the client after the client sends a HTTP request. Status-Line is the first line of a HTTP Response message which consists the version of the protocol, numeric code of status and its short message (Fielding et al., 1999). The version of the protocol usually is HTTP/1.1. The numeric code of status is a 3 digit code of the trying to meet the request and the short message is the description of the status code. HTTP status codes have 200's, 300's, 400's and 500's. 200's are response to a successful request. 300's are response to redirection of the request. 400's are response to an error with the request at the client side. 500's are response to an error with the server which cannot process the request.



```
HTTP/1.x 200 OK
Transfer-Encoding: chunked
Date: Sat, 28 Nov 2009 04:36:25 GMT
Server: LiteSpeed
Connection: close
X-Powered-By: W3 Total Cache/0.8
Pragma: public
Expires: Sat, 28 Nov 2009 05:36:25 GMT
Etag: "pub1259380237;gz"
Cache-Control: max-age=3600, public
Content-Type: text/html; charset=UTF-8
Last-Modified: Sat, 28 Nov 2009 03:50:37 GMT
X-Pingback: http://net.tutsplus.com/xmlrpc.php
Content-Encoding: gzip
Vary: Accept-Encoding, Cookie, User-Agent
```

Figure 2.7: HTTP response structure

Source: Burak Guzel (2009)

The header of the HTTP response consists of date, entity tag (ETag), cache-control, content-type, last modified and so on. Date is used to indicate the time and date of the information was started. ETag in the field of HTTP response header is used for the purposes of caching and give the present value of the entity tag for the difference requested. Cache-control is used to define an instruction that must follow by all caching mechanisms in the process of request and response. “max-age” declares that the cache is only valid for how many seconds. Content-Type is the type of media of the entity-body transmits to receiver. Last-Modified represent the date and time for the last modified of the file.

2.3 Botnet Detection

Botnet detection is a technique to detect the malicious traffic which originated by botnets. The botnet detection techniques are anomaly-based detection and signature-based detection. Anomaly-based detection is the technique that used in this project to detect botnet traffic.

2.3.1 Signature-based Detection

Signature-based detection technique detects the botnet by utilizing the signatures of current botnet (Zeidanloo et al., 2010). For example, Snort can find the signature of existing bots by monitoring the network traffic. Signature-based detection only can be used to discover the botnet which is already known by the public. Thus, the signature-based detection cannot detect the zero-day bot attacks. This detection approach will also cannot detect the very similar bot with slight different signature. Rishi is a famous signature-based botnet detection that matches known nick-name patterns of IRC bot (Goebel & Holz, 2007). Rishi is used to monitor the IRC servers, uncommon server ports and suspicious IRC nicknames. It uses the scoring system and n-gram analysis to detect the bots that use uncommon communication channels.

2.3.2 Anomaly-based Detection

Anomaly detection techniques are based on the definition of standard behavior and performance of certain parameter to detect the malicious traffic (Puerta et al., 2013). The parameters are the characteristics of network connection, CPU usage or any modification of the file system. Anomaly detection is good at discovering a new botnet infection which causes changes in the monitored activity (Fedynyshyn et al., 2011). Karasaridis et al. have the study on performing an in-depth passive analysis of the traffic in transport layer (Karasaridis et al., 2007). The aim is able to detect behavior patterns independently of the nature of the change of information at the application layer. Giroire et al. state that bot must contact their bot master via C&C server to receive command (Giroire et al., 2009). Giroire et al. builds a legitimate whitelist destination and observe the host. If the host connects a new destination, an alarm is raised. The connection of the communication will be blocked if the connection is pretended to be malicious. The disadvantage of this method is the whitelist need to update frequently. BotSniffer exploits the temporal similarity and special of botnet activity contrasting them with the legitimate network traffic (Gu et al., 2008). The idea behind BotSniffer is each of the bots in the same network will accept the same command from botmaster and carry out the similar activity responses at the same time. Thus, if a group of host performs the same activity to respond the same command from same server so as to mark such traffic as malicious (Fedynyshyn et al., 2011).

2.4 Machine Learning

Machine learning is an automatic learning technique which will base on the past observation to make an accurate prediction. Machine learning can avoid people to making mistake during analyses and trying to establish relationship between multiple features (Kotsiantis, 2007). It uses algorithms that can iteratively learn from data without being definitely programmed. The repeat feature of machine learning can let it able to independently analyze data when it exposed to new data. The activities involved machine learning algorithms are fraud detection, web search result, email spam filtering, network intrusion detection and so on. The two methods of machine learning are unsupervised learning and supervised learning. The method used for this project is supervised based machine learning.

2.4.1 Unsupervised learning

Unsupervised learning has been recommended to detect spam email or spam in network social by straightly leveraging the pattern of spamming that do not have the training cost (Tan et al., 2013). Training cost is human labor cost to marking training set. Thus, the data used for unsupervised learning has no historical marks or labels. (Gao et al., 2010) recognized spam mail by clustering posts based on the similarities of text and URL. Besides, unsupervised learning also can use to detect anomalies in the flow of packets on a TCP/IP network (Zanero & Savaresi, 2004). The techniques of unsupervised learning are nearest-neighbor mapping, k-means clustering, self-organizing maps and so on.

2.4.2 Supervised learning

Supervised learning is research for algorithms that study from a lot of instances to generate universal hypotheses and then the algorithms can make predictions about the future instances (Kotsiantis, 2007). Supervised learning is the process of studying a set of standards from a training set and producing a classifier that used to generalize from new instances. The learning algorithm will be given a set of instance with the correlated estimate outputs (Settles, 2010). The algorithm will learn to find error by contrasting the estimate output with the actual output. Then, the algorithms modify the model to minimize the error between estimated output and actual output (Settles, 2010). The methods of supervised learning are classification, regression, prediction and so on. Classification is the method used for this project to detect HTTP botnet.

2.5 Classification

In machine learning, classification is a process to train a classifier. The classifier was trained to recognize different patterns from given samples of training dataset accurately and classify sample of testing dataset with trained classifier (Cho & Won, 2003). In addition, classification also can define as a process to finds common attributes between a set of objects in a database (Ming-Syan Chen, Jiawei Han, 1996). Then, it will classify them into their corresponding classes according to the classification model. Firstly, a sample of dataset X is act as a training dataset which each row consist the same set of multiple attributes as the rows in a large dataset Z and each row has a known class label. The key point of classification is to analyze the training dataset first and construct an accurate classification model for each class by using the attribute in the dataset. After that, the classification model is used to classify testing dataset Z. Nowadays, there have many classification applications such as medical diagnosis, performance prediction, selective marketing and so on.

Furthermore, classification is a technique use to classify and analyze effectively behavior characteristics of the network traffic in order to classify botnet behavior network traffic and normal behavior network traffic (Saad et al., 2011). The behavior characteristics of the network traffic are the flow duration, the packet size, the number of ACK and SYN packets per flow, and so on. Classification also used to identify four major classes of network services such as bulk data transfer, streaming, interactive and transaction (Roughan & Sen, 2004). They generate very accurate network traffic flow classification by using flow duration characteristic and packet size, and observe the simple classification schemes. The algorithms of machine learning classification are Decision Trees, Naïve Bayes, Artificial Neural Network, Support Vector Machine classifier and so on.

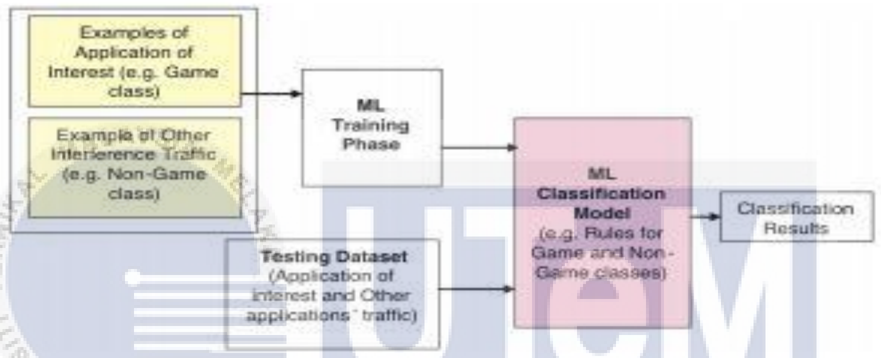

Examples of Application of Interest (e.g. Game class)
Example of Other Interference Traffic (e.g. Non-Game class)
ML Training Phase
ML Classification Model (e.g. Rules for Game and Non-Game classes)
Classification Results
Testing Dataset (Application of interest and Other applications' traffic)

Figure 2.8: The training and testing for supervised classifier

Source: Nguyen et al. (2008)

2.5.1 Naïve Bayes

According Maron and Kuhns (1960), Naïve Bayes is one of the probability methods that used to classify text and retrieve information. Naïve Bayes use the conditional probability of each attribute value given the class and the concept description of prior probability of each class (Maron & Kuhns, 1960). Prior probabilities are based on the previous experience and used it to predict the outcome. It calculates the frequency of the classes occurrence and the attribute values for each class in the training data to estimate the value of prior probability (Kolter & Maloof, 2006). In addition, Naïve Bayes has given an unknown instance to compute posterior probability of every class based on the Bayes' rule and returning as its prediction the class with the highest probability value. Naïve Bayes has been used to detect flow of the P2P botnet C&C traffic. According to the earlier research, Naïve Bayes has above 88% of the real detection percentage of the P2P botnet C&C and the classifier training time was lower compare to the classifier training time SVM and ANN (Saad et al., 2011).

2.5.2 Support Vector Machines

Support Vector Machines is good on the classification task on traditional text (Dumais, 1996). The method generates a linear classifier which is a vector of an intercept and weights (Kolter & Maloof, 2006). SVM will map the training data into higher-dimensioned space by using the kernel function, so the problem is linearly divided (Kolter & Maloof, 2006). When applied to the problem of binary classification, a SVM chooses among all possible hyperplanes dividing the two classes in kernel space a hyperplane that maximizes its margins to each class as the decision boundary for classification (Problem & Li, 2013). This means that SVM will choose the best hyperplane that leaves the maximum margin from both classes. SVM improve the standard methods of finding optimal dividing hyperplanes (Erbacher et al., 2008). SVM has been used to detect P2P botnet C&C traffic flows. According to the earlier research, SVM has above 90% of the true detection rate of the P2P botnet C&C but the classifiers training time and classification time was higher than Naïve Bayes and ANN (Saad et al. 2011).

2.5.3 Artificial Neural Network

Artificial Neural Network (ANN) is a computational modeling tool that have widely acknowledgment in many fields for modeling complex real-world problem (Basheer & Hajmeer, 2000). ANN are also capable of carrying out large-scale parallel calculations for knowledge representation and data processing (Basheer & Hajmeer, 2000). Jain et al. (1996) state the remarkable information processing features of ANN are nonlinearity, robustness, high parallelism, learning, fault and failure tolerance, fuzzy information and ability to handle imprecise and their capability to generalize. One of the advantages of ANN is it has ability of self-training. ANN also capable of finding hidden interdependencies in raw input data (Erbacher et al., 2008). ANN has been used to detect P2P botnet C&C traffic flows. According to the research of other people, ANN also has above 90% of the true detection rate of P2P botnet but the total classification error rate was higher than SVM (Saad et al., 2011).

2.5.4 Decision Trees

Decision tree is a rooted tree with internal nodes and leaf nodes (Kolter & Maloof, 2006). The internal nodes represent the attributes and the leaf nodes represent the class labels. The branches of the attributes lead to the children which represent the value of the attribute. The main function of decision tree is uses the attributes and the values of an example to build the tree start from the root to leaf. The decision tree is built by choosing the attribute that classifies the data of training set into suitable classes. Decision tree produces the node, branches and children based on the attribute and its value, eliminate the attribute after consideration and allocate the instance to the suitable child node. This process will execute iteratively until a node of a class contains an example. Diminish the induced decision tree by deleting the subtrees that are probably perform poorly on test data is important to avoid overtraining.

2.6 Evaluation Criteria

The performance of the selected classifier was evaluated using True Positive Rate (TPR), False Positive Rate (FPR) and accuracy. Table 2.2 shows the definition of TPR, TNR, FPR and FNR.

Table 2.2: Definition of performance metrics

Performance Metrics	Description
True Positive Rate (TPR)	Ratio of correctly detecting malware samples as malware.
True Negative Rate (TNR)	Ratio of correctly detecting benign samples as benign.
False Positive Rate (FPR)	Ratio of incorrectly detecting benign samples as malware.
False Negative Rate (FNR)	Ratio of incorrectly detecting malware samples as benign.

The formula of TPR, FPR and accuracy:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

$$\text{Acuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Where

TP- Number of malware samples which correctly classify.

FN- Number of malware samples which incorrectly classify as benign.

FP- Number of benign samples which incorrectly classify as malware.

TN- Number of benign samples which correctly classify.

Good performance classifiers are indicated by high value of TPR and low value of FPR (Zaki et al., 2014). The performance of each classifier was evaluated by the accuracy value which represents the level of accuracy of the classifiers to classify the samples into the correct class.

2.7 Critical Review

There have been several types of research done by other people to detect the malicious network traffic. Strayer et al. recommend a method to detect the botnets by investigating flow characteristics in order to obtain the credential of the botnet command and control activities (Strayer et al., 2006). The example of flow characteristics is duration, bandwidth and the timing of the packet. They proposed to eliminate the non-malicious network traffic first and classify the other network traffic into the group of malicious network traffic. Then, they correlate that traffic by finding the similar communication patterns that define as the botnet activity. Livades et.al suggest to recognizing the command and control (C&C) traffic of IRC botnet by using machine learning technique (Lu et al., 2009). They propose to identify the difference between IRC and non-IRC traffic and the difference between IRC botnet traffic and real IRC botnet traffic. According to Gu et al., they study the correlation and similarity of the spatial-temporal in network traffic (Gu et al., 2008). They use BotSniffer which is a prototype system to detect HTTP and IRC botnet. The botnet detection technique suggests above have the limitation to IRC-based C&C protocols and centralized botnet structure only.

In addition, Jae-Seo et al. and Tung-Ming el al. proposed one parameter according to one of the HTTP botnet characteristics which is Degree of Periodic Repeatability (DPR) (Lee et al., 2008). DPR is the regular connections pattern of HTTP botnet to botmaster. According to Jae-Seo et al. and Tung-Ming el al., when the DPR is low, this proves that the activity is a bot. This is because a bot usually uses a fixed connection interval. This detection method will produce a false negative result, if the botmaster alters the connection intervals technique. Besides, this detection method also will detect the updaters as a bot and produce a false positive result.

Then, BotMiner was introduced by Gu et al. to investigate the similarities in the malicious activities which produce by the similar botnet. Although BotMiner can detect the random interval bot which frequently change the connection intervals, it also faces the high risk of producing false positive result when it detects the service which needs to update periodically such as Gmail session that needs to check email regularly. This detection method shows less efficiency when used in a single bots or small size botnet.

Furthermore, Lu et al. introduce to use payload-signature to classify the services and application flows (Lu et al., 2009). They use the payload signature to check the bit string inside packet payload. The bit string in packet payload can use to separate the unknown traffic and known traffic from the set of network traffic to decrease the false alarm rate. This method is less effective as it cannot recognize a new pattern and maybe will increase the false negative rate.

Thus, each of the studies on botnet detection produces the different result in the field of false alarm rates and efficiency. Therefore, this project used five different technique of classification to detect the HTTP botnet from network traffic.

2.8 Summary

Finally, the chapter 2 was discussed about the literature review that related to this project. Throughout the chapter 2, the methodology or technique used in this project was clearly defined. This review will be used as a reference for the implementation and analysis phase.

CHAPTER III

METHODOLOGY

3.1 Introduction

In this chapter, the project methodology and the project milestone was discussed clearly. Project methodology and the project milestone become the guideline until the end of the project. Both of it ensure that the project meets the objective of the project.

3.2 Project Methodology

This section described the step of the methodology about the project. The step listed in the methodology should be followed to make sure the project is in a correct sequence and do not run out of the scope of the project. Figure 3.1 below show the five phases of the project methodology.

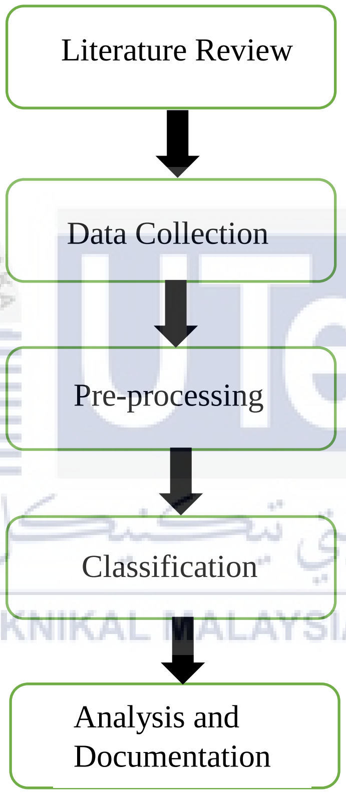


Figure 3.1: Project Methodology

3.2.1 Phase 1: Literature review

Phase 1 is the phase to study all related previous research paper to understand the scope and the requirement of the project. The related journal used as a reference to make the project become more reliable. Besides, this phase also defines the theory about the HTTP botnet, anomaly-based detection and the method of classification used to detect the botnet.

3.2.2 Phase 2: Data Collection

Phase 2 is the phase to release and collect the data of HTTP botnet. During this phase, the HTTP botnet was released in an isolate network and the data of the network traffic was collected after one week the HTTP botnet been released. In addition, this phase was defining the type of HTTP botnet. On the same time, the non-malicious network traffic also generated by browsing HTTP website for 1 day duration.

3.2.3 Phase 3: Pre-processing

Phase 3 is the phase to convert the type of the file from tcpdump file to CSV file by using tcptrace tool. After that, the data clean process is needed to carry out by delete all the unreadable character of the Rapid Miner in the CSV file. The rapid miner unable to process the data which involve alphabet character and IP address. Thus, the data should be clean before classifying the data using the Rapid Miner.

3.2.4 Phase 4: Classification

In this phase, the five types of the algorithms of machine learning classification use in this project are Decision Tree, K-Nearest Neighbors (K-NN), Naïve Bayes, Random Forest and Random Tree. After the five types of machine learning classification algorithms been applied to classify the botnet behavior network traffic and normal behavior network traffic, the result between the five types of the algorithm need to be compared to verify which algorithm is the most accurate algorithm to classify the network traffic.

Decision Tree is a predictive classification model that predicts the label of a new sample based on different parameter values of the dataset. K-nearest neighbor (K-NN) is a type of simple classification technique among memory based induction. K-NN makes decision to predict the label based on the k closest neighbors with similarity measures. Naïve Bayes is a classifier based on Bayes Theorem of conditional probability. Random Forest is a classifier that aggregates the result and it is constructed by a brunch of decision trees. The good attribute at each node of the decision tree is constructing from a randomly selected features number. Random Tree classifier functions like the decision tree classifier, but Random Tree classifier will split a random subset of the attributes.

The reason of implement these five types of classification technique is these five types classifier is a common and popular classifier used by another researcher in malware or botnet detection. According to the earlier research, Random Forest classifier able to classify the sample into correct classes and achieves 82.53% of detection accuracy (Puerta et al., 2013). Besides, Naïve Bayes has above 88% of the detection accuracy and the training time of Naïve Bayes classifier was lower compare to SVM and ANN classifiers (Saad et al., 2011). In criteria of Area under the Curve (AUC), Random Forest and KNN have achieve the highest value which around 0.9 and 0.93 (Puerta et al., 2013). The AUC is percentage of correct test results in while classifying testing data and AUC value of 1 represents a perfect test (Kapratwar, 2016). Moreover, Decision Tree classifier also achieves 80% of the correctly classified instances (Puerta et al., 2013).

3.2.5 Phase 5: Analysis and Documentation

In this phase, the result of the classification is needed to analysis by producing a graphical result. After the analysis, all the result of the classification of HTTP botnet is need to documentation and summarize. The project contribution and the project limitation is need to list out at the end of the project. Then, it will follow by evaluation and submission of the completed final year project.

3.3 Project Requirement

This section was discussed about the software and hardware requirement of the project.

3.3.1 Software Requirement

Table 3.1 shows the minimum software requirements of the project.

Table 3.1: Software requirements

Requirements	Description
Windows 7 Operating System(OS)	Machine Operating System.
Ubuntu 14.04 LTS	Machine operating system that used for malware repository.
Wireshark	Network packet analyzer tool.
Microsoft Office 2013: ▪ Microsoft Words ▪ Microsoft PowerPoint ▪ Microsoft Excel	The software of Microsoft Office 2013 is used to prepare the final project report, Gantt Chart, milestone, slide presentation and data cleaning.
Rapid Miner Studio	Intuitive graphical user interface for the

	design of analytic processes.
VMware Workstation	A virtual machine which creates an isolate environment used to download and analysis the HTTP botnet.
Tcpdump	A common packet analyzer that runs under the command line.
Process Monitor	A monitoring tool to capture real time file system activity, registry activity, process or thread activity and network activity.
Process Explorer	A system resources monitoring tool to monitoring the currently active processes and tracking down DLL-version problems or handle leaks.

3.3.2 Hardware Requirement

Table 3.2 shows the information of the network devices involve in this project which is router, switch and sniffer. In addition, Table 3.3 figures out the hardware information for window 8.1, window 7 ultimate and Ubuntu 14.04 LTS.

Table 3.2: Information of network devices

Network Devices	Detail
Router	Cisco 2800 Series
Switch	Catalyst 2950 Series
Sniffer	HP ProLiant DL160 G5

Table 3.3: Hardware requirement for Windows 8.1, Windows 7 Ultimate and Ubuntu 14.04 LTS

Requirement	Server 1	Server 2	Server 3
Operating System	Windows 8.1	Windows 7 Ultimate	Ubuntu 14.04 LTS
Processor	Intel (R) Core (TM) i5-4200U	Intel (R) Pentium (R) D	Intel (R) Core (TM) i5-3470
Memory (RAM)	4.00 GB	8.00 GB	4.00 GB
System Type	64-bit Operating System	32-bit Operating System	64-bit Operating System

3.4 Project Schedule and Milestones

This section was discussed the project schedule and milestones. Project schedule and milestone are used to ensure the progress of the project is running on time.

3.4.1 Milestones

Table 3.4 shows the milestone of the project.

Table 3.4: Milestone

Project Activity	Milestones	Week
Deciding title of Proposal PSM	Prepare and discuss proposal PSM	04-08 Jan
Proposal PSM Submission & Presentation	Deliverable proposal PSM	11-15 Jan
Chapter 1: Introduction	Preparation of chapter 1	01-04 March
Submission of Chapter 1	Deliverable chapter 1 and discuss chapter 2	07-11 March

Chapter 2: Literature Review	Preparation of chapter 2	14-25 March
Submission of Chapter 2	Deliverable chapter 2 and discuss chapter 3	28 Mar -1 April
Chapter 3: Methodology	Preparation of chapter 3	04-08 April
MID SEMESTER BREAK		
Submission of Chapter 3	Deliverable chapter 3 and discuss chapter 4	18-22 April
Chapter 4: Design	Preparation of chapter 4	25 April- 13 May
Submission of chapter 4	Deliverable of chapter 4	16 - 20 May
Prepare PSM 1 Report	Improvement of PSM 1 report	23 - 27 May
FINAL PRESENTATION (PA)	Submission and presentation PSM 1 Report	30 May - 03 June
REVISION WEEK Correction draft report based on supervisor's and evaluator's comments during the final presentation session. Submission overall marks to PSM/PD committee.	Action – Student, Supervisor, Evaluator. PSM/PD committee.	06 - 10 June
FINAL EXAMINATION SEMESTER		
Chapter 5: Implementation	Preparation of chapter 5	04-08 July
Submission of chapter 5	Deliverable of chapter 5 and discuss chapter 6	11-15 July
Chapter 6: Testing	Preparation of chapter 6	18-22 July
Submission of chapter 6	Deliverable of chapter 6 and discuss chapter 7	25-29 July

Chapter 7: Conclusion	Preparation of chapter 7	01-05 Aug
Submission of chapter 7	Deliverable of chapter 7	08-12 Aug
Complete Final Report PSM 2(draft) Progress Evaluation	Preparation and Improvement Final Report PSM 2	15-19 Aug
Presentation Schedule	Deliverable – Complete Report(draft)	22-26 Aug
-PSM 2 presentation and evaluation - Correction report based on supervisor's and evaluator's comments during the final presentation session.	Improvement of Final Report PSM 2	29-31 Aug
-Submit PSM complete report for supervisor's signature and binding	Deliverable – PSM report(3 copies) & CD (1 copy)	29-31 Aug

3.5 Summary

Finally, the chapter 3 discussed the five phases include in the project which are literature review, data collection, pre-processing, classification and analysis and documentation. This chapter also described the software and hardware requirement of the project. The project schedule and milestone are listed out to ensure the activity of the project been completed on time.

CHAPTER IV

DESIGN

4.1 Introduction

This chapter is about the preliminary design and the result of the detailed design of the project. The design involves in the project is logical network design. Network design can help to realize the structure of the implementation of the network. Having a visual view of a logical network design can help to identify or troubleshooting the problem quickly. Besides, this chapter also discusses the design of the data collection of botnet and the scenarios to stimulate the project.

4.2 Network System Architecture

In this project, the control network was set up to collects and captures the HTTP botnet network traffic. The network device involves in this control network are one router, one switch, one sniffer and three workstations. The network topology of this malicious network is start topology. In a start topology, all the workstations are connected to one switch which acts as the center of the network architecture. Data need to pass through the switch before transfer to other destination. The switch will manage all the functions of the network and acts as a repeater for the data flow. Start topology is easy to install and troubleshoot to detect the fault device. As the node are not connected to each other, thus if one node has problem, the rest of the node also can function well. The main drawback of start topology is the network will corrupt if the switch fails to function well.

Besides, network architecture was set up to collects and captures the real network traffic. The network device involves in this network are one router, one switch, one sniffer and three workstations. The network topology implement in this network architecture is start topology.

4.3 Logical Network Design

Logical network design illustrated the arrangement of the network device and the network connection between each device inside a network. The logical network design used was help to understand the IP address and subnet mask of the workstations.

4.3.1 Logical Network Design for Malicious Network

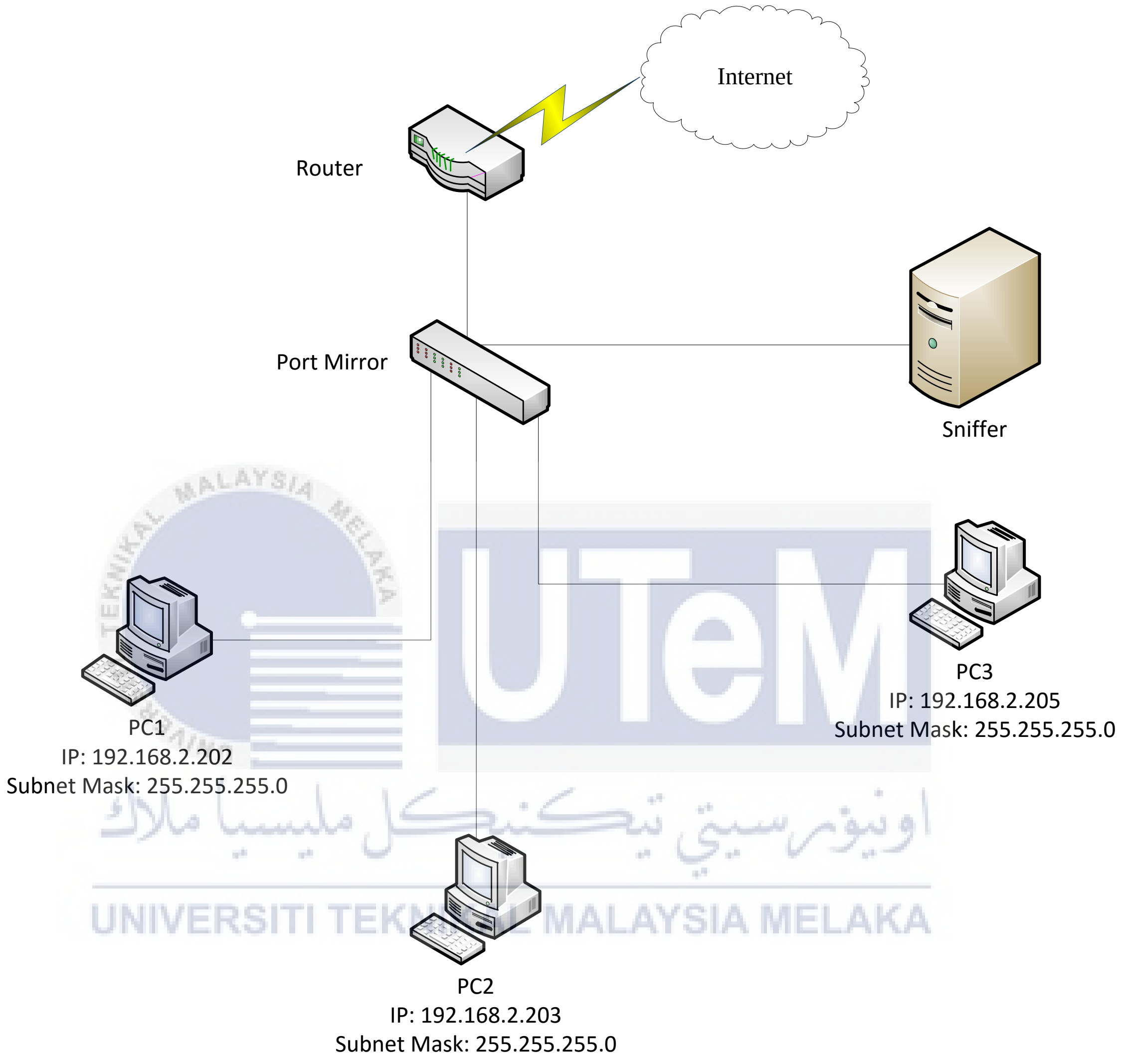


Figure 4.1: Logical design of the malicious network

Based on Figure 4.1, there have one router, one switch, one sniffer and three workstations to setup a network environment for generating malicious network traffic. The sniffer is a network analyzing tool that will capture and view the flow of the packet receive and transmit on the network. It is used to recording the malicious network traffic when a botnet is release among the workstation and the bot is communicating with the C&C server to receive the command from botmaster.

4.3.2 Logical Network Design for Real Network

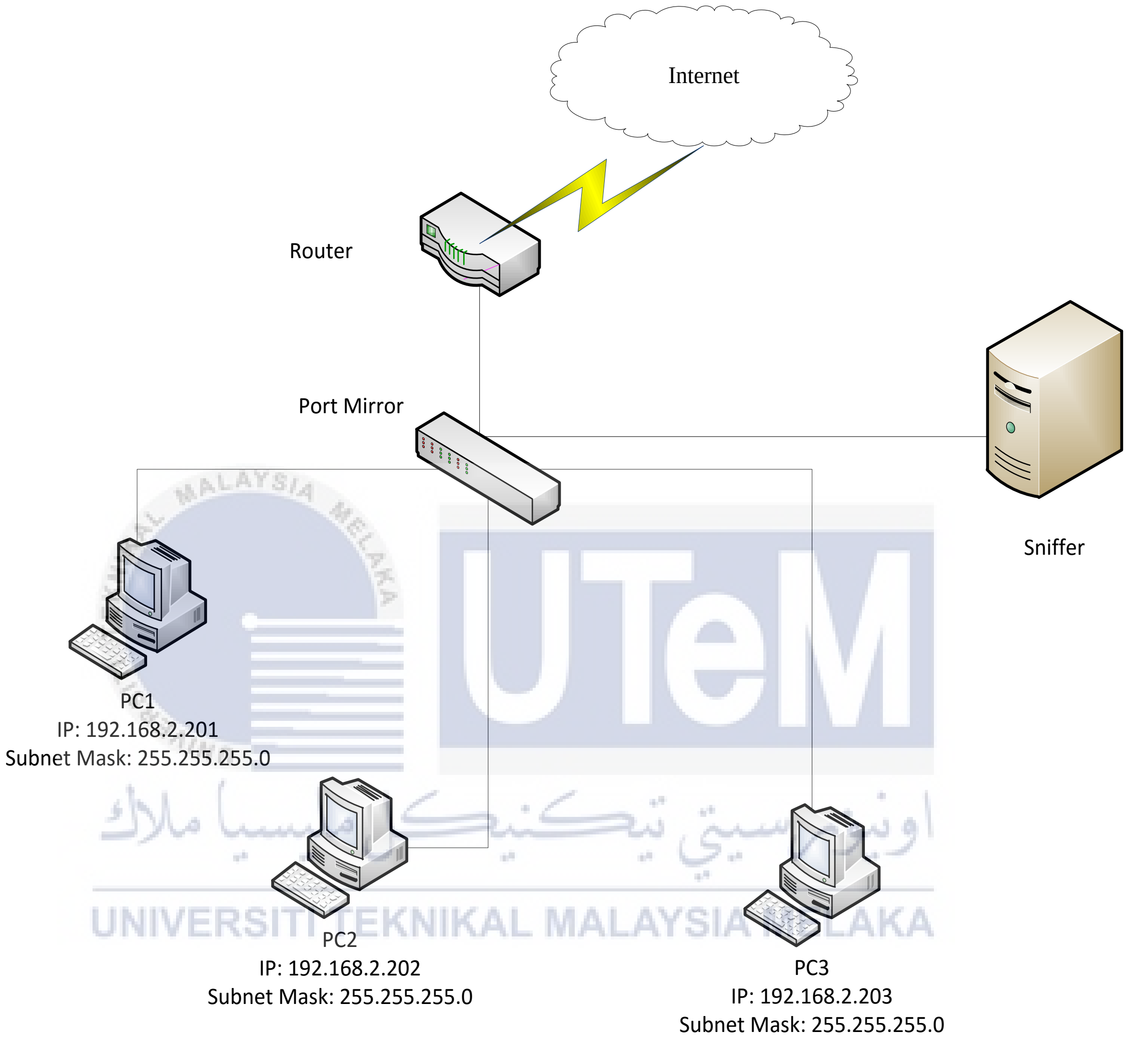


Figure 4.2: Logical design for real network

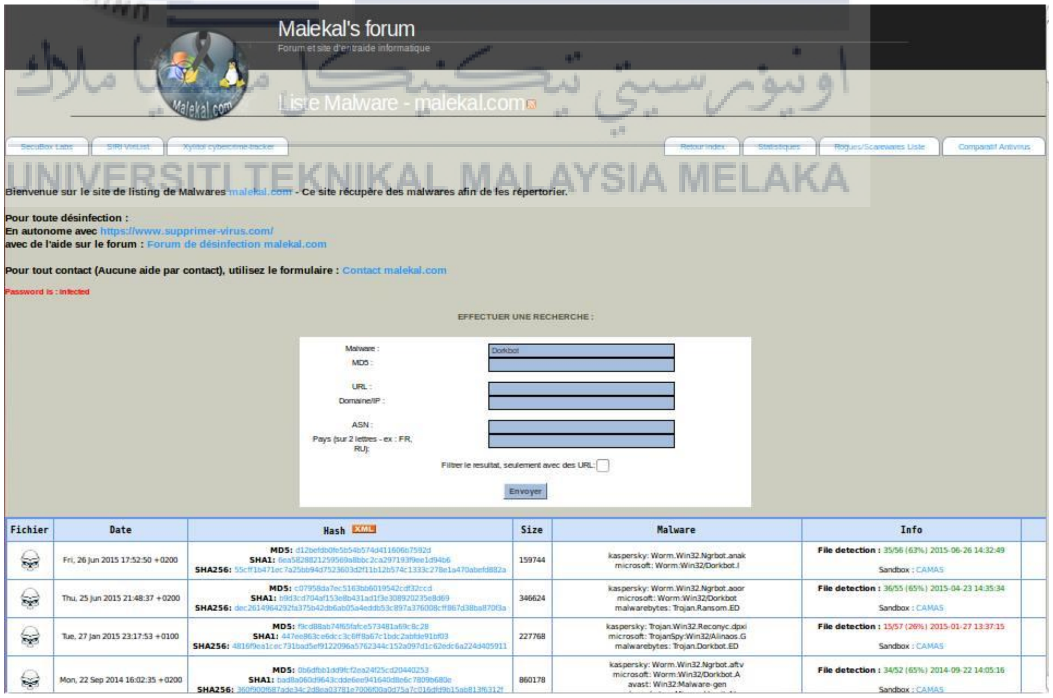
Based on Figure 4.2, there have one router, one switch, one sniffer and three workstations to setup a network environment for generating real network traffic. The network sniffer will capture the real network traffic without the existent of botnet.

4.4 Possible Scenarios

The possible scenarios of this experiment have divided into two parts which are data collecting and data analysis.

4.4.1 Data Collecting

Firstly, the malware binary or execute files was needed to collect from the Internet. The websites use to collect the malware binary files are Contagio BlogSpot (http://contagiodump.blogspot.my/), Malware Analysis by Cuckoo Sandbox (https://malwr.com/) and Malekal's Forum (http://malwaredb.malekal.com/). Figure 4.3 is an example of the step to collect the HTTP dorkbot binary files from Malekal's Forum.



Bienvenue sur le site de listing de Malwares malekal.com - Ce site récupère des malwares afin de les répertorier.

Pour toute désinfection :
En autonome avec https://www.supprimer-virus.com/
avec de l'aide sur le forum : Forum de désinfection malekal.com

Pour tout contact (Aucune aide par contact), utilisez le formulaire : Contact malekal.com

Password is : infected

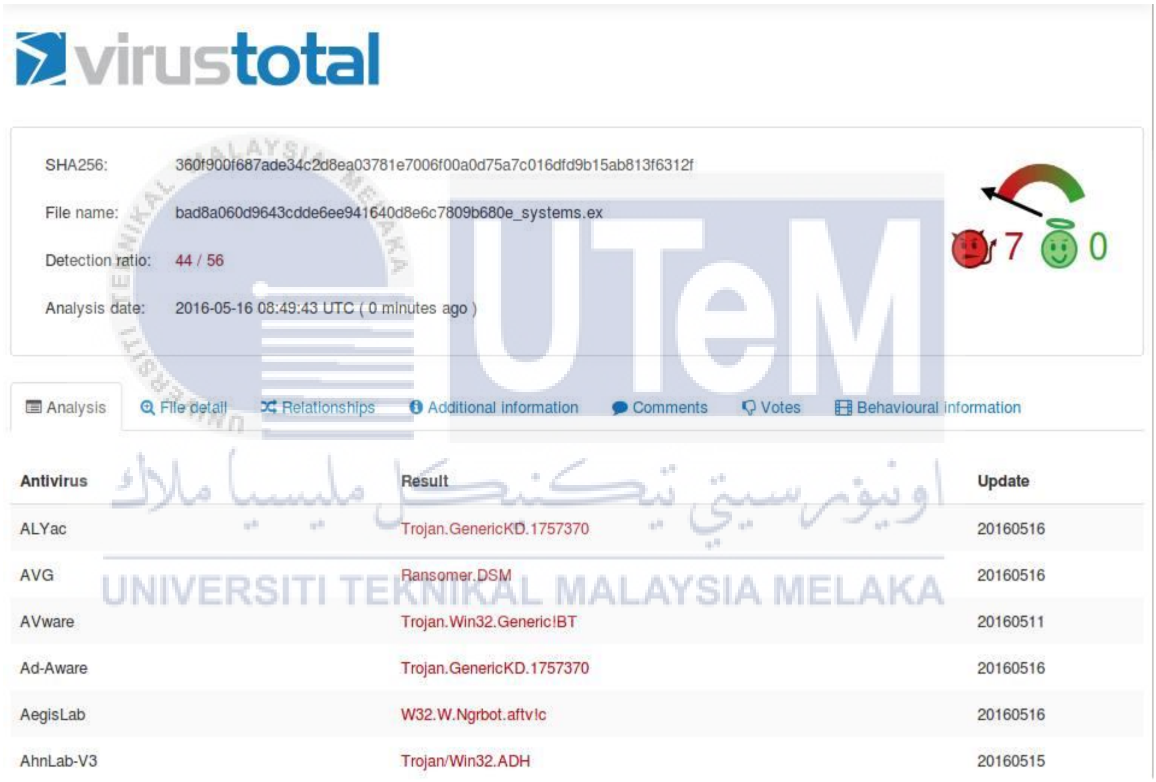
EFFECTUER UNE RECHERCHE :

Malware :	Dorkbot
MD5 :	
URL :	
Domaine/IP :	
ASN :	
Pays (sur 2 lettres - ex : FR, RU):	
Filtrer le resultat, seulement avec des URL:	☐

Fichier	Date	Hash XML	Size	Malware	Info
	Fri, 26 Jun 2015 17:52:50 +0200	**MD5:** d12befdb0fe5b54b574d411606b7592d **SHA1:** 6ea5828821259569a8bbc2ca297193f9ee1d94b6 **SHA256:** 55cff1b471ec7a25bb94d7523603d2f11b12b574c1333c278e1a470abefd882a	159744	kaspersky: Worm.Win32.Ngrbot.anak microsoft: Worm:Win32/Dorkbot.I	**File detection :** 35/56 (63%) 2015-06-26 14:32:49 Sandbox : CAMAS
	Thu, 25 Jun 2015 21:48:37 +0200	**MD5:** c07958da7ec5163bb6019542cdf32ccd **SHA1:** b9d3cd704af153e8b431ad1f3e308920235e8d69 **SHA256:** dec2614964292fa375b42db6ab05a4eddb53c897a376008cff867d38ba870f3a	346624	kaspersky: Worm.Win32.Ngrbot.aoor microsoft: Worm:Win32/Dorkbot malwarebytes: Trojan.Ransom.ED	**File detection :** 36/55 (65%) 2015-04-23 14:35:34 Sandbox : CAMAS
	Tue, 27 Jan 2015 23:17:53 +0100	**MD5:** f9cd88ab74f65fafce573481a69c8c28 **SHA1:** 447ee863ce6dcc3c6ff8a67c1bdc2abfde91bf03 **SHA256:** 4816f9ea1cec731bad5ef9122096a5762344c152a097d1c62edc6a224d405911	227768	kaspersky: Trojan.Win32.Reconyc.dpxi microsoft: TrojanSpy:Win32/Alinaos.G malwarebytes: Trojan.Dorkbot.ED	**File detection :** 15/57 (26%) 2015-01-27 13:37:15 Sandbox : CAMAS
	Mon, 22 Sep 2014 16:02:35 +0200	**MD5:** 0b6dfbb1dd9fcf2ea24f25cd20440253 **SHA1:** bad8a060d9643cdde6ee941640d8e6c7809b680e **SHA256:** 360f900f687ade34c2d8ea03781e7006f00a0d75a7c016dfd9b15ab813f6312f	860178	kaspersky: Worm.Win32.Ngrbot.aftv microsoft: Worm:Win32/Dorkbot.A avast: Win32:Malware-gen	**File detection :** 34/52 (65%) 2014-09-22 14:05:16 Sandbox : CAMAS

Figure 4.3: Dorkbot binary files on Malekal's Forum

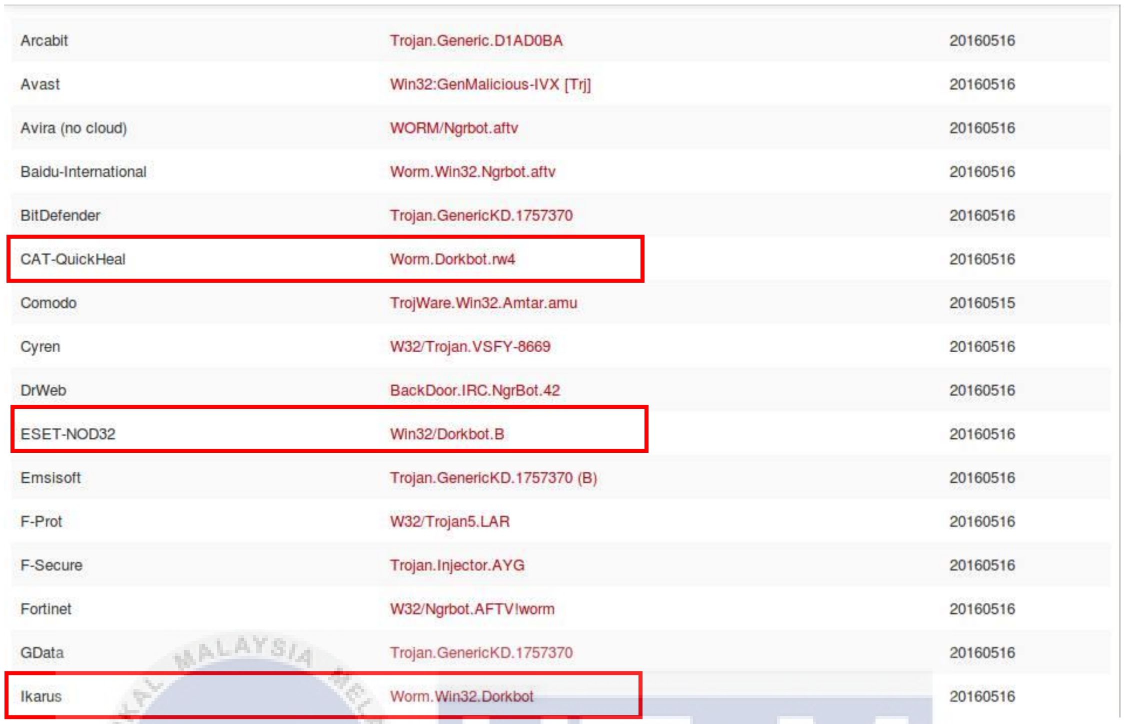
Secondly, the malware binary file was needed to upload in the website VirusTotal (https://www.virustotal.com/) to analyze and verify the malware binary file. The VirusTotal is an online antivirus engine and scanner that analyzes files and URLs to identification the malicious content such as virus, worms, Trojan, botnet, backdoor and other malware. The result of VirusTotal scanner can prove that the malware binary file is a HTTP botnet. Figure 4.4 and Figure 4.5 show the result of the scanner VirusTotal after uploading a Dorkbot binary file. Figure 4.5 show that has three antiviruses detect the binary file uploaded contains a Dorkbot.



SHA256: 360f900f687ade34c2d8ea03781e7006f00a0d75a7c016dfd9b15ab813f6312f
File name: bad8a060d9643cdde6ee941640d8e6c7809b680e_systems.ex
Detection ratio: 44 / 56
Analysis date: 2016-05-16 08:49:43 UTC (0 minutes ago)

Antivirus	Result	Update
ALYac	Trojan.GenericKD.1757370	20160516
AVG	Ransomer.DSM	20160516
AVware	Trojan.Win32.Generic!BT	20160511
Ad-Aware	Trojan.GenericKD.1757370	20160516
AegisLab	W32.W.Ngrbot.aftv!c	20160516
AhnLab-V3	Trojan/Win32.ADH	20160515

Figure 4.4: Result of VirusTotal part 1



Arcabit	Trojan.Generic.D1AD0BA	20160516
Avast	Win32:GenMalicious-IVX [Trj]	20160516
Avira (no cloud)	WORM/Ngrbot.aftv	20160516
Baidu-International	Worm.Win32.Ngrbot.aftv	20160516
BitDefender	Trojan.GenericKD.1757370	20160516
CAT-QuickHeal	Worm.Dorkbot.rw4	20160516
Comodo	TrojWare.Win32.Amtar.amu	20160515
Cyren	W32/Trojan.VSFY-8669	20160516
DrWeb	BackDoor.IRC.NgrBot.42	20160516
ESET-NOD32	Win32/Dorkbot.B	20160516
Emsisoft	Trojan.GenericKD.1757370 (B)	20160516
F-Prot	W32/Trojan5.LAR	20160516
F-Secure	Trojan.Injector.AYG	20160516
Fortinet	W32/Ngrbot.AFTV!worm	20160516
GData	Trojan.GenericKD.1757370	20160516
Ikarus	Worm.Win32.Dorkbot	20160516

Figure 4.5: Result of VirusTotal part 2

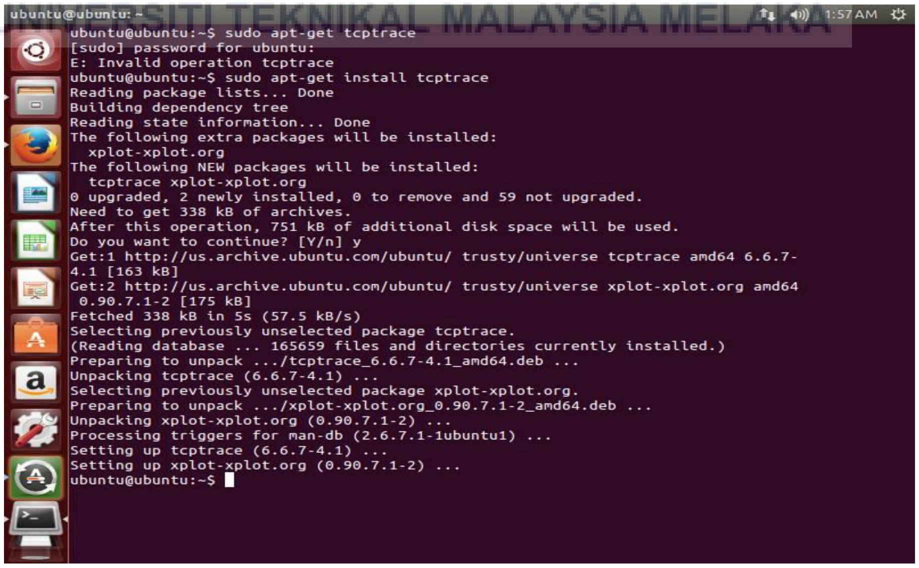
Thirdly, process monitor and process explorer were used to make sure the HTTP botnet is not in a dormant status. By using the process monitor and process explorer, all the processes or activity execute by the malware binary file on the user's system can monitor by the user. Process monitor is a real time monitoring tools that can use to capture all the single activity execute by the botnet such as registry activity, process or thread activity, network activity and the file system activity. In the registry activity, it list out the activity carry out by the botnet such as creating, reading, deleting or querying the registry keys. In process or thread activity, it list out all event for starts or exits of the process and thread. Besides, the information about the botnet connecting, sending, receiving, or reconnecting to the IP address of botmaster was recorded in the network activity. In the file system activity, it listed out all the activity execute by the botnet such as opening, closing, creating or reading the files. Process explorer act as a task manager or system monitor application that list out the active processes and Dynamic Link Library (DLL) in the computer system. The activities monitors by process explorer are the processes loading a DLL file, running an open window or opening and closing the file.

After verifying the malware binary file is a HTTP botnet and it does not in a dormant status, the malware binary file was executed in a control network for the duration of 7 days. Table 4.1 lists out the milestone of release the HTTP botnet binary files in a control network.

Table 4.1: Milestone of release HTTP botnet binary files

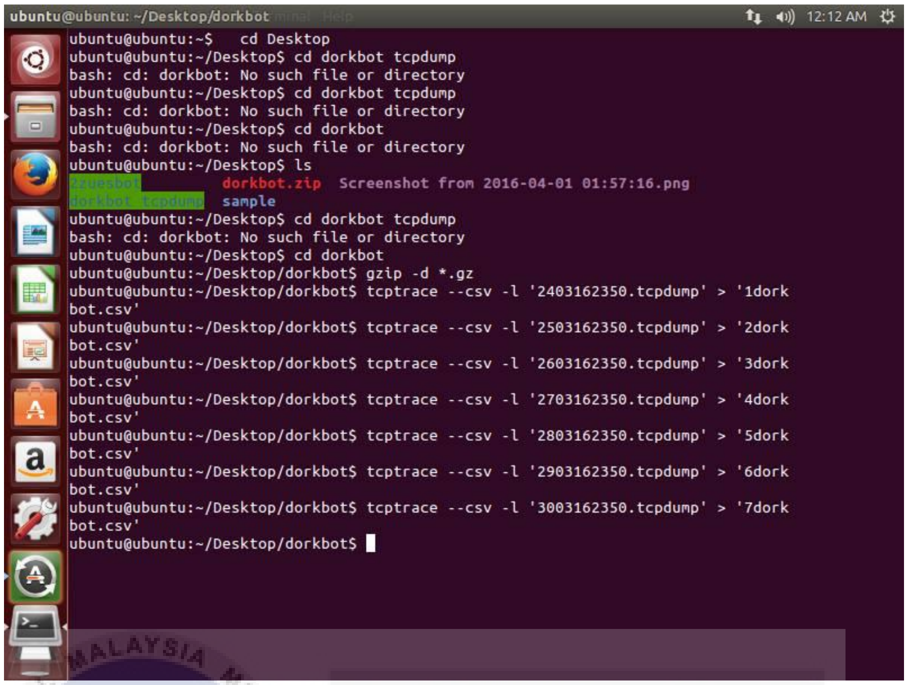
Type of HTTP botnet	Start date	End date	Duration
Dorkbot	24/03/2016	30/03/2016	Seven days
Zeus	31/03/2016	06/04/2016	Seven days
Citadel	07/04/2016	13/04/2016	Seven days
Spyeye	14/04/2016	20/04/2016	Seven days
Cutwail	21/04/2016	27/04/2016	Seven days
Waledac	28/04/2016	04/05/2016	Seven days

After 7 days, the malicious tcpdump file was collected from the network sniffer. Before analyze the malicious tcpdump file in the Rapid Miner, the tcpdump file was needed extract into CSV file by using tcptrace. Figure 4.6 shows the script of install tcptrace and Figure 4.7 shows the script of extract tcpdump file into CSV file.



```
ubuntu@ubuntu:~$ sudo apt-get tcptrace
[sudo] password for ubuntu:
E: Invalid operation tcptrace
ubuntu@ubuntu:~$ sudo apt-get install tcptrace
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  xplot-xplot.org
The following NEW packages will be installed:
  tcptrace xplot-xplot.org
0 upgraded, 2 newly installed, 0 to remove and 59 not upgraded.
Need to get 338 kB of archives.
After this operation, 751 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu/ trusty/universe tcptrace amd64 6.6.7-
4.1 [163 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu/ trusty/universe xplot-xplot.org amd64
 0.90.7.1-2 [175 kB]
Fetched 338 kB in 5s (57.5 kB/s)
Selecting previously unselected package tcptrace.
(Reading database ... 165659 files and directories currently installed.)
Preparing to unpack .../tcptrace_6.6.7-4.1_amd64.deb ...
Unpacking tcptrace (6.6.7-4.1) ...
Selecting previously unselected package xplot-xplot.org.
Preparing to unpack .../xplot-xplot.org_0.90.7.1-2_amd64.deb ...
Unpacking xplot-xplot.org (0.90.7.1-2) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Setting up tcptrace (6.6.7-4.1) ...
Setting up xplot-xplot.org (0.90.7.1-2) ...
ubuntu@ubuntu:~$ 
```

Figure 4.6: Script of install tcptrace



```
ubuntu@ubuntu: ~/Desktop/dorkbot                                12:12 AM
ubuntu@ubuntu:~$   cd Desktop
ubuntu@ubuntu:~/Desktop$ cd dorkbot tcpdump
bash: cd: dorkbot: No such file or directory
ubuntu@ubuntu:~/Desktop$ cd dorkbot tcpdump
bash: cd: dorkbot: No such file or directory
ubuntu@ubuntu:~/Desktop$ cd dorkbot
bash: cd: dorkbot: No such file or directory
ubuntu@ubuntu:~/Desktop$ ls
2zuesbot          dorkbot.zip  Screenshot from 2016-04-01 01:57:16.png
dorkbot tcpdump   sample
ubuntu@ubuntu:~/Desktop$ cd dorkbot tcpdump
bash: cd: dorkbot: No such file or directory
ubuntu@ubuntu:~/Desktop$ cd dorkbot
ubuntu@ubuntu:~/Desktop/dorkbot$ gzip -d *.gz
ubuntu@ubuntu:~/Desktop/dorkbot$ tcptrace --csv -l '2403162350.tcpdump' > '1dork
bot.csv'
ubuntu@ubuntu:~/Desktop/dorkbot$ tcptrace --csv -l '2503162350.tcpdump' > '2dork
bot.csv'
ubuntu@ubuntu:~/Desktop/dorkbot$ tcptrace --csv -l '2603162350.tcpdump' > '3dork
bot.csv'
ubuntu@ubuntu:~/Desktop/dorkbot$ tcptrace --csv -l '2703162350.tcpdump' > '4dork
bot.csv'
ubuntu@ubuntu:~/Desktop/dorkbot$ tcptrace --csv -l '2803162350.tcpdump' > '5dork
bot.csv'
ubuntu@ubuntu:~/Desktop/dorkbot$ tcptrace --csv -l '2903162350.tcpdump' > '6dork
bot.csv'
ubuntu@ubuntu:~/Desktop/dorkbot$ tcptrace --csv -l '3003162350.tcpdump' > '7dork
bot.csv'
ubuntu@ubuntu:~/Desktop/dorkbot$ 
```

Figure 4.7: Script of extract tcpdump file into CSV file

In addition, the non-malicious network traffic data was also needed to collect at the same time. Firstly, network architecture was needed to set up such as one router, one switch, one sniffer and three workstations. The sniffer was used to intercept and record the network traffic pass over the network. Secondly, a list of HTTP website was prepared. Thirdly, HTTP website was browsed for the duration of 1 day to generate the non-malicious network traffic. Finally, the non-malicious tcpdump file was collected for the data analysis. After that, the non-malicious tcpdump file was extract into CSV file by using tcptrace.

Finally, the malicious network traffic data was combined with the non-malicious network traffic data to become a big dataset.

4.4.2 Data Analysis

The data collected was analyzed in this phase by using Rapid Miner. Rapid Miner is an analytic tool use in the field of machine learning, data mining, statistical analytics and predictive analytics. In this project, the method of classification was used to classify the malicious network traffic and non-malicious network traffic. Before classifying the dataset by using Rapid Miner, the data preprocessing was needed to implement first.

Data preprocessing is the preliminary step of data mining to prepare the raw data by removes noise from data before continues to another processing procedure. The steps involve in data preprocessing are data cleaning, data integration, data transformation and data reduction. Data cleaning is clean the data by correcting the inconsistence data and filling the missing values. Data integration is integrating the data from multiple sources. Data transformation is normalizing and generalizing the data. Data reduction is reducing or eliminating the redundant parameter of the dataset. In data preprocessing, all the parameters of the dataset were needed to study. A dataset has 93 columns of parameter and a numbers of rows. Table 4.2 shows the list of the 93 parameters of the dataset. The description of the parameters of dataset showed in Appendix B.

Table 4.2: List of network parameter

Source: Lang (2010)

No	Parameter	No	Parameter	No	Parameter
1.	conn	32.	zwnd_probe_pkts_a2b	63.	min_segm_size_b2a
2.	host_a	33.	zwnd_probe_pkts_b2a	64.	avg_segm_size_a2b
3.	host_b	34.	zwnd_probe_bytes_a2b	65.	avg_segm_size_b2a
4.	port_a	35.	zwnd_probe_bytes_b2a	66.	max_win_adv_a2b
5.	port_b	36.	outoforder_pkts_a2b	67.	max_win_adv_b2a
6.	first_packet	37.	outoforder_pkts_b2a	68.	min_win_adv_a2b
7.	last_packet	38.	pushed_data_pkts_a2b	69.	min_win_adv_b2a
8.	total_packets_a2b	39.	pushed_data_pkts_b2a	70.	zero_win_adv_a2b
9.	total_packets_b2a	40.	SYN_pkts_sent_a2b	71.	zero_win_adv_b2a
10	resets_sent_a2b	41.	FIN_pkts_sent_a2b	72.	avg_win_adv_a2b

11.	resets_sent_b2a	42.	SYN_pkts_sent_b2a	73.	avg_win_adv_b2a
12.	ack_pkts_sent_a2b	43.	FIN_pkts_sent_b2a	74.	initial_window_bytes_a2b
13.	ack_pkts_sent_b2a	44.	req_1323_ws_a2b	75.	initial_window_bytes_b2a
14.	pure_acks_sent_a2b	45.	req_1323_ts_a2b	76.	initial_window_pkts_a2b
15.	pure_acks_sent_b2a	46.	req_1323_ws_b2a	77.	initial_window_pkts_b2a
16.	sack_pkts_sent_a2b	47.	req_1323_ts_b2a	78.	ttl_stream_length_a2b
17.	sack_pkts_sent_b2a	48.	adv_wind_scale_a2b	79.	ttl_stream_length_b2a
18.	dsack_pkts_sent_a2b	49.	adv_wind_scale_b2a	80.	missed_data_a2b
19.	dsack_pkts_sent_b2a	50.	req_sack_a2b	81.	missed_data_b2a
20.	max_sack_blks_ack_a2b	51.	req_sack_b2a	82.	truncated_data_a2b
21.	max_sack_blks_ack_b2a	52.	sacks_sent_a2b	83.	truncated_data_b2a
22.	unique_bytes_sent_a2b	53.	sacks_sent_b2a	84.	truncated_packets_a2b
23.	unique_bytes_sent_b2a	54.	urgent_data_pkts_a2b	85.	truncated_packets_b2a
24.	actual_data_pkts_a2b	55.	urgent_data_pkts_b2a	86.	data_xmit_time_a2b
25.	actual_data_pkts_b2a	56.	urgent_data_bytes_a2b	87.	data_xmit_time_b2a
26.	actual_data_bytes_a2b	57.	urgent_data_bytes_b2a	88.	idletime_max_a2b
27.	actual_data_bytes_b2a	58.	mss_requested_a2b	89.	idletime_max_b2a
28.	rexmt_data_pkts_a2b	59.	mss_requested_b2a	90.	hardware_dups_a2b
29.	rexmt_data_pkts_b2a	60	max_segm_size_a2b	91.	hardware_dups_b2a
30.	rexmt_data_bytes_a2b	61.	max_segm_size_b2a	92.	throughput_a2b
31.	rexmt_data_bytes_b2a	62.	min_segm_size_a2b	93.	throughput_b2a

After understand all the meaning of the parameter of the dataset, the unrelated and duplicated data was needed to delete, the missing value of the data was needed to fill in and some related data was needed to integrate. The execution of the data preprocessing can increase the effectiveness and accuracy of the result of data mining. Next, the dataset was uploaded into Rapid Miner for the classification of the data. In the stage of classification, 5 types of classification algorithm were used to classify the data. Then, the result of the performance measures between the 5 types of classification algorithm was evaluated for classify the malicious and non-malicious network traffic.

4.5 Summary

The network design is important for develop a network architecture. The logical network design can give a clear understanding about the arrangement of the network device and the IP addressing. The possible scenario states in this chapter can ensure that this project is not run out of the objective and the scope of the topic. Besides, it also gives a clear review of the process of experiment needed to carry out. This chapter was prepared a good basic and preparation for the implementation of the project.

CHAPTER V

IMPLEMENTATION

5.1 Introduction

This chapter discusses the activity execution in the phase of implementation. In this phase, the malicious and non-malicious tcpdump files are extracting into CSV file by using tcptrace. After both of the malicious and non-malicious files been extracted, both data are combine become a big dataset. Besides, data preprocessing is needed to carry out to make sure the dataset contains significant parameters without missing value.

5.2 Data preprocessing

During this phase, data preprocessing is carried out and divided into four steps which are data cleaning, data integration, data transformation and data reduction. Data preprocessing is an essential phase that should be executed carefully before classification process was started. Data preprocessing can brings benefit of gaining a high accuracy of the classification result.

5.2.1 Data Cleaning

Data cleaning is an importance part of the project before start the classification process. Data cleaning is a process used to eliminate missing value such as symbol NA which represented as not available and noisy value such as IP address in the dataset that is unreadable by rapid miner. Data cleaning also can help to improve the accuracy of classification. In this project, the column of host_a and host_b parameters will be delete because both of the parameters represented as IP addresses of host a and host b which cannot interpret by rapid miner. Besides, the NA symbol found in the column of idletime_max_b2a parameter will be replaced as zero value. The row of the throughput_a2b and throughput_b2a parameters which consists NA symbol will be deleted. The column of ttl_stream_length_a2b, ttl_stream_length_b2a, missed_data_a2b, missed_data_b2a parameters will be deleted because it contains more than 60% of NA.

5.2.2 Data Integration

Data integration is an association process of combining data from multiple sources become a big dataset. In this project, the malicious and non-malicious dataset was needed to combine become a big dataset. Besides, the first_packet and last_packet parameters which act as time of the first packet and last packet sent in the connection will be integrate become inter_arrival_time parameter which is the interval time between the time of the first packet and last packet sent in the connection.

5.2.3 Data Transformation

Data transformation is a process of transform the parameter value into an appropriate format which is suitable to undergo the classification process in rapid miner. The example of data transformation is changing the alphabet Y which act as yes into value 1 and alphabet N which act as no into value 0. Thus all alphabets Y and N of the parameters req_1323_ws_a2b, req_1323_ws_b2a, req_1323_ts_a2b, req_1323_ts_b2a, req_sack_a2b, req_sack_b2a are change into value 1 and 0. This is because rapid miner cannot interpret the alphabet so all alphabets such as Y and N will change into value 1 and 0 corresponding. To label the malicious and non-malicious dataset, class parameter is added. The malicious dataset was represented as value 1 and non-malicious dataset was represented as value 0.

5.2.4 Data Reduction

Data reduction is one of the steps of data preprocessing which needed to delete unrelated and insignificant parameters in the dataset. Although data reduction will decrease the size of the dataset but it still maintains to generate the similar classification result. In this project, the parameter which consist more than 90% value 0 will be deleted. This is because those parameter contains a lot of value 0 will not bring any effect on the classification process. Besides, the conn parameter which acts as the numbering of the dataset also will be deleted, because this parameter is not useful in the classification process. Thus the data reduction can help to improve the accuracy of the classification result by eliminating the irrelevant parameters.

5.2.5 Parameter

Table 5.1 shows the list of the network parameters selected after data preprocessing. The description of the network parameters showed in Appendix B. That network parameters was used in the classification process.

Table 5.1: List of selected network parameter

Source: Lang (2010)

No	Parameter	No	Parameter	No	Parameter
1.	port_a	21.	pushed_data_pkts_a2b	41.	avg_segm_size_a2b
2.	port_b	22.	pushed_data_pkts_b2a	42.	avg_segm_size_b2a
3.	inter_arrival_time	23.	SYN_pkts_sent_a2b	43.	max_win_adv_a2b
4.	total_packets_a2b	24.	FIN_pkts_sent_a2b	44.	max_win_adv_b2a
5.	total_packets_b2a	25.	SYN_pkts_sent_b2a	45.	min_win_adv_a2b
6.	resets_sent_b2a	26.	FIN_pkts_sent_b2a	46.	min_win_adv_b2a
7.	ack_pkts_sent_a2b	27.	req_1323_ws_a2b	47.	avg_win_adv_a2b
8.	ack_pkts_sent_b2a	28.	req_1323_ts_a2b	48.	avg_win_adv_b2a
9.	pure_acks_sent_a2b	29.	req_1323_ws_b2a	49.	initial_window_bytes_a2b
10.	pure_acks_sent_b2a	30.	req_1323_ts_b2a	50.	initial_window_bytes_b2a
11.	unique_bytes_sent_a2b	31.	adv_wind_scale_a2b	51.	initial_window_pkts_a2b
12.	unique_bytes_sent_b2a	32.	adv_wind_scale_b2a	52.	initial_window_pkts_b2a
13.	actual_data_pkts_a2b	33.	req_sack_a2b	53.	data_xmit_time_a2b
14.	actual_data_pkts_b2a	34.	req_sack_b2a	54.	data_xmit_time_b2a
15.	actual_data_bytes_a2b	35.	mss_requested_a2b	55.	idletime_max_a2b
16.	actual_data_bytes_b2a	36.	mss_requested_b2a	56.	idletime_max_b2a
17.	rexmt_data_pkts_a2b	37.	max_segm_size_a2b	57.	throughput_a2b
18.	rexmt_data_pkts_b2a	38.	max_segm_size_b2a	58.	throughput_b2a
19.	rexmt_data_bytes_a2b	39.	min_segm_size_a2b	59.	class
20.	rexmt_data_bytes_b2a	40.	min_segm_size_b2a		

5.3 Summary

Finally, this chapter discusses the detail process of the data preprocessing before we started the classification process. The data preprocessing can help to generate a dataset without unrelated parameter and noisy data which will affect the accuracy of the classification result.

CHAPTER VI

TESTING AND ANALYSIS

6.1 Introduction

This chapter discusses the classification process and result implementation through the Rapid Miner analytic tool. During this chapter, the design of classification process was discussed and the result of the classification process was analyzed. Through this chapter, the result of difference classification techniques such as Decision Tree, K-NN, Naïve Bayes, Random Forest and Random Tree was evaluated and compared. At the end of chapter six, the classification technique has the most accurate performance in classifying malicious and non-malicious network traffic was concluded.

6.2 Result and Analysis

This section studies the classification process execute in Rapid Miner. Besides, each performance measures of the difference classification technique were analyzed and all techniques were compared to choose the technique which has the most accurate performance in the classification process.

6.2.1 Input Dataset in Rapid Miner

After dataset is clean by removing noisy and irrelevant data through data preprocessing, then the dataset is prepared to proceed with classification process. Rapid Miner is an analytic tool which used in this project to execute the classification process. To input dataset inside Rapid Miner, first click on the button "Add Data" in the repository.

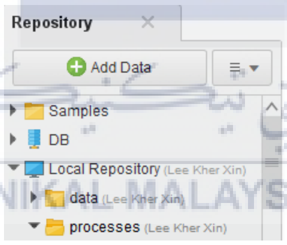


Figure 6.1: Interface of repository

Next, select the place store dataset such as My Computer.

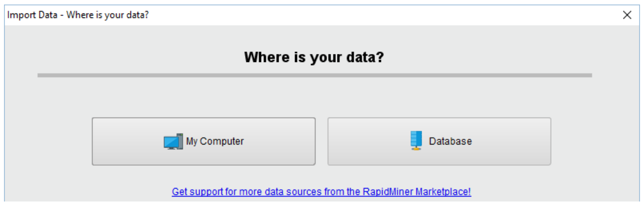
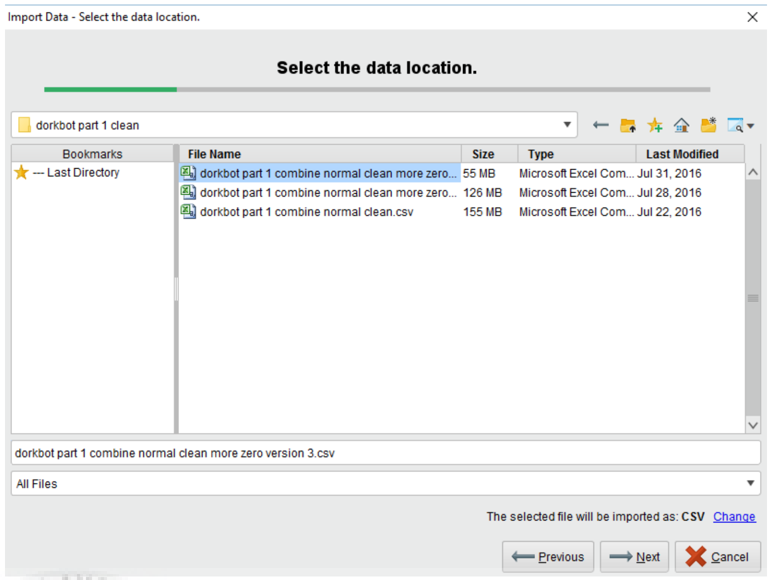

Import Data - Where is your data?
Where is your data?
My Computer
Database
Get support for more data sources from the RapidMiner Marketplace!

Figure 6.2: Interface of import data

Then, choose the file and click “Next” button.


Import Data - Select the data location.

Select the data location.

dorkbot part 1 clean

File Name	Size	Type	Last Modified
dorkbot part 1 combine normal clean more zero...	55 MB	Microsoft Excel Com...	Jul 31, 2016
dorkbot part 1 combine normal clean more zero...	126 MB	Microsoft Excel Com...	Jul 28, 2016
dorkbot part 1 combine normal clean.csv	155 MB	Microsoft Excel Com...	Jul 22, 2016

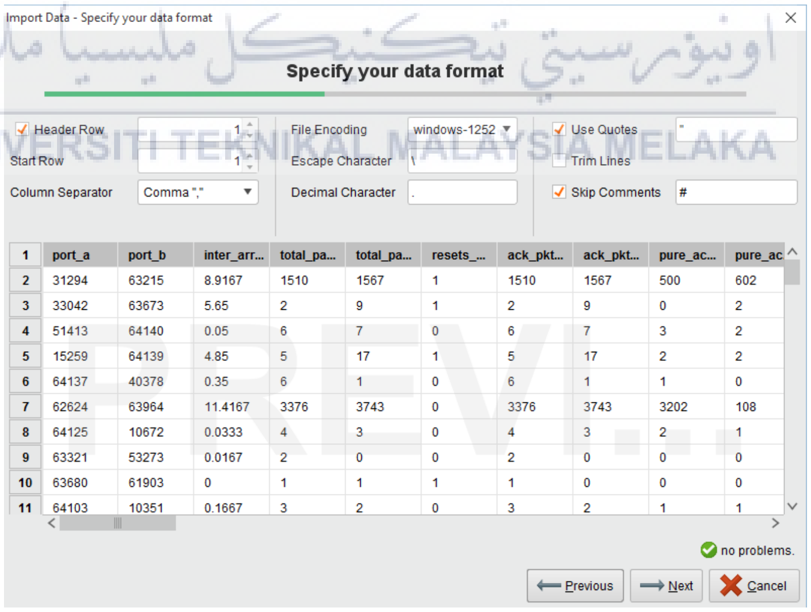
dorkbot part 1 combine normal clean more zero version 3.csv

All Files

The selected file will be imported as: **CSV** Change

Figure 6.3: Interface of select the data location

After that, specify the data format such as column separator and click “Next” button.


Import Data - Specify your data format

Specify your data format

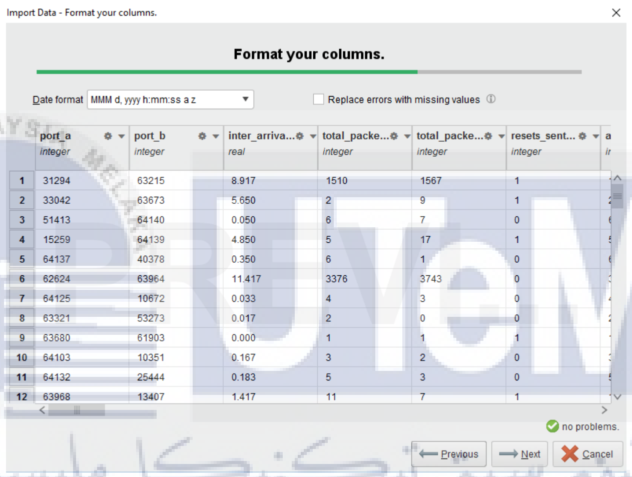
Header Row ✓ 1
Start Row 1
Column Separator Comma ","
File Encoding windows-1252
Escape Character \
Decimal Character .
Use Quotes ✓ "
Trim Lines
Skip Comments ✓ #

1	port_a	port_b	inter_arr...	total_pa...	total_pa...	resets_...	ack_pkt...	ack_pkt...	pure_ac...	pure_ac
2	31294	63215	8.9167	1510	1567	1	1510	1567	500	602
3	33042	63673	5.65	2	9	1	2	9	0	2
4	51413	64140	0.05	6	7	0	6	7	3	2
5	15259	64139	4.85	5	17	1	5	17	2	2
6	64137	40378	0.35	6	1	0	6	1	1	0
7	62624	63964	11.4167	3376	3743	0	3376	3743	3202	108
8	64125	10672	0.0333	4	3	0	4	3	2	1
9	63321	53273	0.0167	2	0	0	2	0	0	0
10	63680	61903	0	1	1	1	1	0	0	0
11	64103	10351	0.1667	3	2	0	3	2	1	1

no problems.

Figure 6.4: Interface of specify data format

Then, the appropriate data type for each parameter was needed to set. The example of data type is polynominal, binominal, real, integer, date_time, date and time. Polynominal represents the different string values such as circle, square, rectangle, and triangle. Binominal is used for the parameters which have two values only such as yes and no. Real can used for fractional number while integer used for whole number. Date_time is a data type which used for both date and time, while date is used for represents date only and time is used for represents time only. Besides, the role of class parameter was needed to set as a label which used to differentiate the malicious and non-malicious dataset.



Date format: MMM d, yyyy h:mm:ss a z

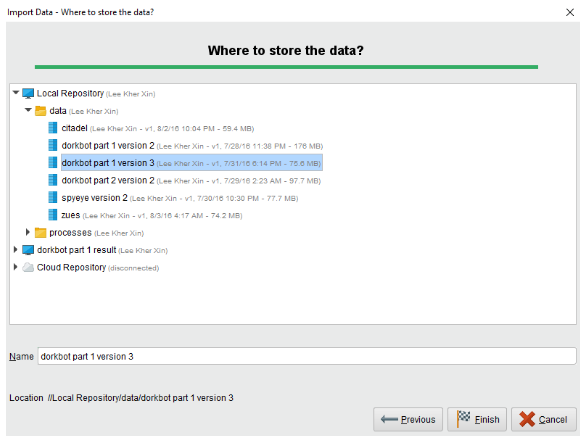
Replace errors with missing values

	port_a integer	port_b integer	inter_arriva... real	total_packe... integer	total_packe... integer	resets_sent... integer
1	31294	63215	8.917	1510	1567	1
2	33042	63673	5.650	2	9	1
3	51413	64140	0.050	6	7	0
4	15259	64139	4.850	5	17	1
5	64137	40378	0.350	6	1	0
6	62624	63964	11.417	3376	3743	0
7	64125	10672	0.033	4	3	0
8	63321	53273	0.017	2	0	0
9	63680	61903	0.000	1	1	1
10	64103	10351	0.167	3	2	0
11	64132	25444	0.183	5	3	0
12	63968	13407	1.417	11	7	1

no problems.

Figure 6.5: Interface of format columns

Next, the dataset was stored into the local repository of Rapid Miner.



- Local Repository (Lee Kher Xin)
  - data (Lee Kher Xin)
    - citadel (Lee Kher Xin - v1, 8/2/16 10:04 PM - 59.4 MB)
    - dorkbot part 1 version 2 (Lee Kher Xin - v1, 7/28/16 11:38 PM - 176 MB)
    - dorkbot part 1 version 3 (Lee Kher Xin - v1, 7/31/16 6:14 PM - 75.6 MB)
    - dorkbot part 2 version 2 (Lee Kher Xin - v1, 7/29/16 2:23 AM - 97.7 MB)
    - spyeye version 2 (Lee Kher Xin - v1, 7/30/16 10:30 PM - 77.7 MB)
    - zues (Lee Kher Xin - v1, 8/3/16 4:17 AM - 74.2 MB)
  - processes (Lee Kher Xin)
- dorkbot part 1 result (Lee Kher Xin)
- Cloud Repository (disconnected)

Name: dorkbot part 1 version 3

Location //Local Repository/data/dorkbot part 1 version 3

Figure 6.6: Interface of storing dataset in local repository

6.2.2 Design of classification process

This section discusses the design of five classification techniques such as Decision Tree, K-Nearest Neighbors (K-NN), Naïve Bayes, Random Forest and Random Tree.

6.2.2.1 Decision Tree

The Figure 6.7, Figure 6.8 and Figure 6.9 below show the design of decision tree classification. Figure 6.7 show that there have 4 operators which are retrieve spyeye, shuffle, cross-validation and tree to rules. Retrieve spyeye is the operator to retrieve the dataset from the local repository. Shuffle is an operator which used to randomize all the arrangement of the dataset. Cross-validation operator is used to predict the performance of the decision tree classifier. The number of validation is set to 10. This also means that the input dataset will be divided into 10 parts equally. Inside the 10 parts of the dataset, a single part is used as the testing dataset and the other 9 parts of the dataset will be used as the training dataset. The process of cross-validation was repeated 10 times with each of the 10 parts used exactly once time as the testing dataset. Moreover, tree to rule is an operator which used to generate a set of rule from the model of the decision tree.

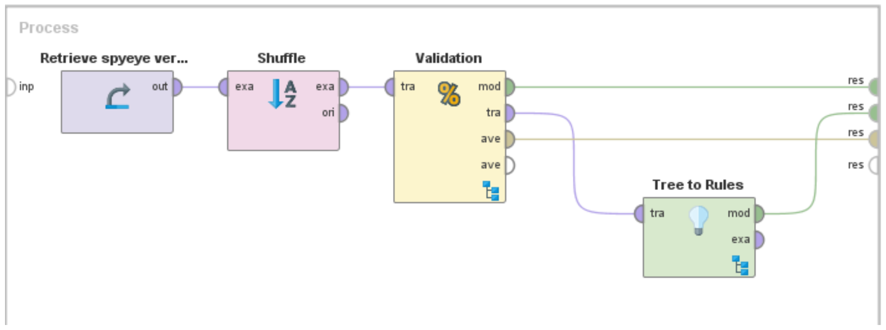


Figure 6.7: Interface 1 of decision tree classification

Figure 6.8 below shows the nested process of the cross-validation operator. Inside cross-validation operator, it divided into two parts which are training process and testing process. The training process is used for training the dataset and constructs a classification model. Then, the classification model was applied in the testing process and the performance of the classification model also will be measure in the testing process. Figure 6.9 shows the nested process of the tree to rules operator.

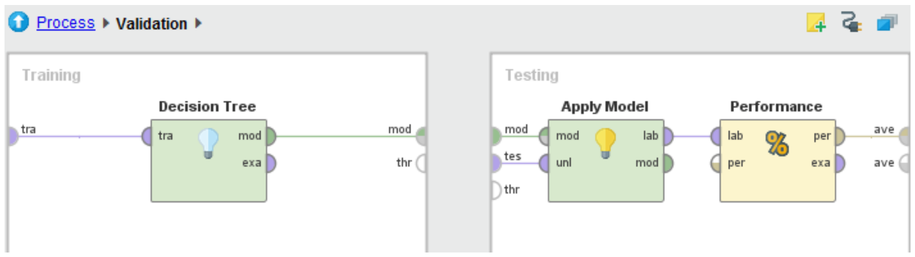


Figure 6.8: Interface 2 of decision tree classification

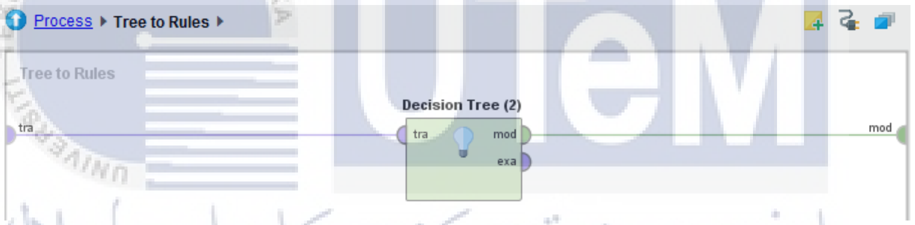


Figure 6.9: Interface 3 of decision tree classification

6.2.2.2 K-Nearest Neighbors

Figure 6.10 show that there have 3 operators which are retriever, shuffle and cross-validation. Figure 6.11 show that the nested process of cross-validation. Inside the training process, there has k-NN classifier while in the testing process there has apply model and performance operator.

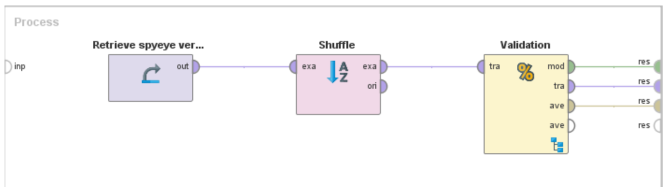


Figure 6.10: Interface 1 of K-NN classification

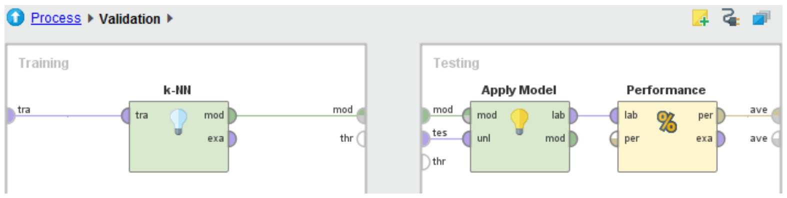


Figure 6.11: Interface 2 of K-NN classification

6.2.2.3 Naïve Bayes

Figure 6.12 and Figure 6.13 show the design of the Naïve Bayes classification. Figure 6.12 show that there have 3 operators which are retriever, shuffle and cross-validation. Besides, Figure 6.13 show that there have a Naïve Bayes classifier at training process, while in the testing process there have apply model and performance operator.

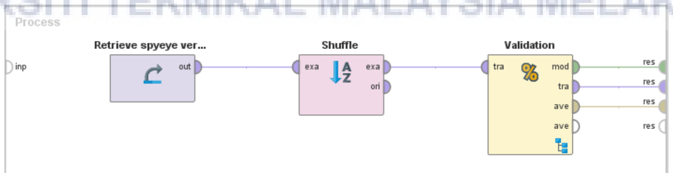


Figure 6.12: Interface 1 of Naïve Bayes classification

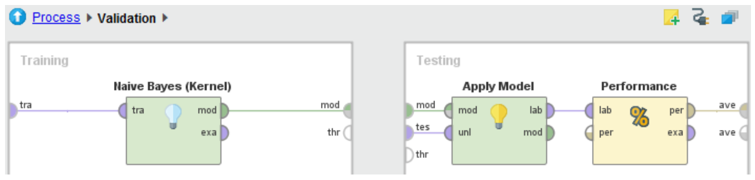


Figure 6.13: Interface 2 of Naïve Bayes classification

6.2.2.4 Random Forest

Figure 6.14 and Figure 6.15 show the design of the Random Forest classification. Figure 6.14 is the overall process of Random Forest classification. Figure 6.15 is the nested process of the cross-validation operator. Inside the training process, there have a Random Forest classifier while inside the testing process, there have a apply model and a performance operator.

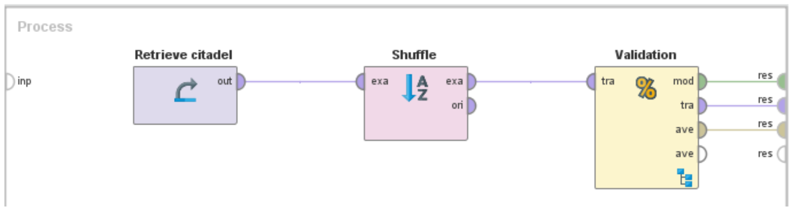


Figure 6.14: Interface 1 of Random Forest classification

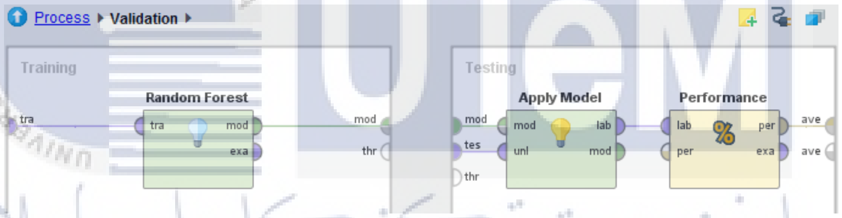


Figure 6.15: Interface 2 of Random Forest Classification

6.2.2.5 Random Tree

Figure 6.16 and Figure 6.17 show the design of the Random Tree classification. Figure 6.16 is the main process of the Random Tree classification. Figure 6.17 is the nested process of the cross-validation operator. In the training part, there have a Random Tree operator, while in the testing part, there have a apply model and a performance operator.

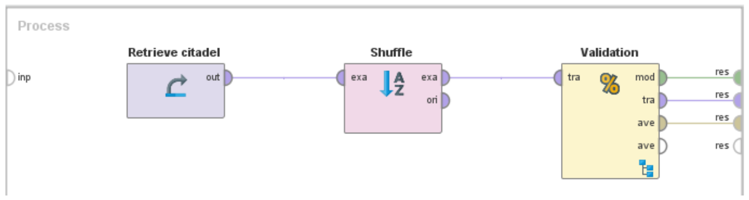


Figure 6.16: Interface 1 of Random Tree classification

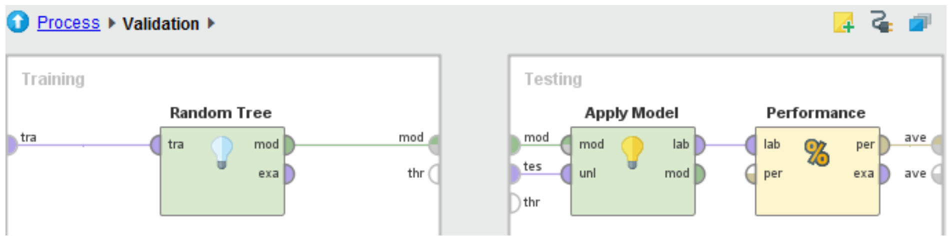


Figure 6.17: Interface 2 of Random Tree classification

6.2.3 Result of Classification

The performance operator used in the classification has calculated the performance of the classifier in six types of criteria which are accuracy, precision, recall, AUC (optimistic), AUC and AUC (pessimistic). Accuracy is the percentage of the predictive value which similar with the actual value. Precision is the percentage of the effectiveness measurement which is how correctly the classifier classifies the malicious network traffic as malware network traffic among the malicious and non-malicious network traffic i.e. precision= True Positive/ (True Positive + False Positive). Recall is the percentage of sensitivity measurement which is how correctly the classifier classifies the malicious network traffic as malware network traffic among the malware network traffic i.e. recall= True Positive/ (True Positive + False Negative). AUC represents as the area under the curve of the receiver operating characteristics (ROC) which used to measure the performance of the classifiers. The AUC with values of less than 0.5 indicates the classifier as a random predictor. The average of the AUC (pessimistic) and AUC (optimistic) is the value of AUC.

6.2.3.1 Dorkbot

Table 6.1: Result of dorkbot

Classifier / Criteria	Decision Tree	K-NN	Naïve Bayes	Random Forest	Random Tree
Accuracy (%)	87.75	90.07	70.10	81.47	77.14
Precision (%)	86.86	93.69	91.54	81.37	77.37
Recall/ TPR (%)	99.99	94.07	69.46	99.99	98.53
FPR (%)	66.14	26.88	27.22	97.12	88.68
AUC (optimistic)	0.995	0.984	0.800	0.972	0.990
AUC	0.683	0.836	0.799	0.699	0.550
AUC (pessimistic)	0.371	0.688	0.799	0.426	0.109

Table 6.1 shows the classification result of the dorkbot dataset in seven criteria of performance measure for five type of classifier. In criteria accuracy, K-NN classifier has the highest accuracy value which is 90.07% while Naïve Bayes classifier has the lowest accuracy value which is 70.10%. Decision Tree is the classifier that has the second highest percentage of accuracy which is 87.75%. In criteria precision, K-NN has the highest percentage of precision which is 93.69% while the Random Tree has the lowest percentage of precision which is 77.37%. Besides, most of the classifiers achieve higher recall value that above 94%. Only Naïve Bayes classifier has the lowest recall value which is 69.46%. The low recall value of Naïve Bayes classifier makes the classifier not have high confidence level in classifying botnet network traffic as botnet network traffic. In criteria False Positive Rate (FPR), K-NN classifier gains the lowest percentage which is 26.88% while Random Forest gains the highest FPR which is 97.12%. In criteria AUC, K-NN classifier obtains the highest AUC value which is 0.836, while Random Tree classifier obtains the lowest AUC value which is 0.550. Therefore, the classifier that has a good performance is K-NN classifier. K-NN has highest detection accuracy which is 90.07%, high TPR value and low FPR value.

6.2.3.2 Zeus

Table 6.2: Result of zeus

Classifier / Criteria	Decision Tree	K-NN	Naïve Bayes	Random Forest	Random Tree
Accuracy (%)	83.61	86.96	51.84	78.08	76.94
Precision (%)	82.16	91.21	84.85	77.42	76.70
Recall/ TPR (%)	99.82	91.42	43.58	99.93	99.49
FPR (%)	65.07	26.44	23.35	87.51	90.75
AUC (optimistic)	0.998	0.977	0.733	0.959	0.989
AUC	0.678	0.825	0.733	0.720	0.552
AUC (pessimistic)	0.359	0.673	0.733	0.481	0.115

Table 6.2 shows the classification result of the zeus dataset in seven criteria of performance measure for five type of classifier. In criteria of accuracy, K-NN and Decision Tree obtains the high detection accuracy which is 86.96% and 83.61% respectively. On the other hands, Naïve Bayes classifier achieves the lowest accuracy value which is 51.84%. In criteria of precision, K-NN, Naïve Bayes and Decision Tree classifiers obtain the high value of precision which are 91.21%, 84.85% and 82.16% respectively. Although Naïve Bayes has a higher precision value, but Naïve Bayes classifier has a lower recall value which is 43.58%. This will cause Naïve Bayes classifier become a not reliable classifier in classifying malicious and non-malicious network traffic. In criteria false positive rate, Naïve Bayes classifier has the lowest percentage which is 23.35% while Random Tree has the highest false positive rate which is 90.75%. Moreover, K-NN classifier has higher AUC values which are 0.825. K-NN classifier indicates as a good predictor since it has AUC values that nearly to value 1. Therefore, the classifier that has a good performance is K-NN classifier. K-NN has highest detection accuracy which is 86.96%, high TPR value and low FPR value.

6.2.3.3 Citadel

Table 6.3: Result of citadel

Classifier / Criteria	Decision Tree	K-NN	Naïve Bayes	Random Forest	Random Tree
Accuracy (%)	91.70	94.46	73.23	71.88	69.54
Precision (%)	90.39	95.92	85.51	71.02	69.31
Recall/ TPR (%)	98.41	96.04	73.55	99.89	100
FPR (%)	23.10	9.01	27.47	89.85	97.62
AUC (optimistic)	0.985	0.996	0.824	0.948	0.993
AUC	0.952	0.935	0.823	0.822	0.525
AUC (pessimistic)	0.918	0.874	0.823	0.697	0.056

Table 6.3 shows the classification result of the citadel dataset in seven criteria of performance measure for five type of classifier. In criteria accuracy, both of the K-NN and Decision Tree classifiers have the high detection value which above 90%. In the other hands, Random Tree classifier has the lowest detection value which is 69.54%. Besides, K-NN and Decision Tree classifier achieve the high precision value which is 95.92% and 90.39% respectively. In criteria of recall, Naïve Bayes classifier has the lowest recall value which is 73.55%. Random Tree classifier obtains 100% recall value but has the lowest precision value. This is because Random Tree classifier predicts the non-malicious network traffic as malicious network traffic. In criteria false positive rate, the classifier that has the lowest false positive rate is K-NN classifier which is 9.01%. In term of AUC, Decision Tree obtains the highest value which is 0.952. This indicates that Decision Tree classifier as a good predictor among other classifier. Random Tree has lowest AUC value which is 0.525. This states that Random Tree classifier as a random predictor classifier. Therefore, the classifier that has a good performance is K-NN classifier. K-NN has highest detection accuracy which is 94.46%, high TPR value and low FPR value.

6.2.3.4 Spyeye

Table 6.4: Result of spyeye

Classifier / Criteria	Decision Tree	K-NN	Naïve Bayes	Random Forest	Random Tree
Accuracy (%)	90.41	95.26	65.51	76.84	78.15
Precision (%)	96.33	96.84	98.31	76.68	77.73
Recall/ TPR (%)	90.86	96.94	55.66	99.98	99.92
FPR (%)	11.03	10.09	3.06	96.95	91.28
AUC (optimistic)	0.982	0.997	0.905	0.965	0.993
AUC	0.927	0.934	0.905	0.771	0.563
AUC (pessimistic)	0.872	0.872	0.905	0.577	0.133

Table 6.4 shows the classification result of the spyeye dataset in seven criteria of performance measure for five type of classifier. From the table above, K-NN has highest detection accuracy which is 95.26% which Naïve Bayes has lowest detection accuracy which is 65.51%. In criteria precision, Decision Tree, K-NN and Naïve Bayes obtain a higher precision values which greater than 95%. In criteria recall, most of the classifiers achieve higher recall value that more than 90%, only Naïve Bayes obtains 55.66% of the recall value. Naïve Bayes classifier has lower reliability in classify malicious sample as malicious sample. In criteria FPR, Decision Tree, K-NN and Naïve Bayes classifiers have the low false positive rate which is 11.03%, 10.09% and 3.06% respectively. In criteria AUC, K-NN has the highest value which is 0.934 while Random Tree has the lowest value which is 0.563. This indicates that K-NN as a good predictor and Random Tree as a random predictor. Therefore, K-NN classifier is a good classifier which has highest detection accuracy, 95.26%, high TPR value, 96.94% and low FPR value, 10.09%.

6.2.3.5 Cutwail

Table 6.5: Result of cutwail

Classifier / Criteria	Decision Tree	K-NN	Naïve Bayes	Random Forest	Random Tree
Accuracy (%)	93.73	97.88	87.76	86.38	87.14
Precision (%)	94.91	98.66	95.42	86.33	87.56
Recall/ TPR (%)	97.94	98.87	90.04	99.93	99.07
FPR (%)	31.52	8.06	25.95	94.93	84.49
AUC (optimistic)	0.988	0.999	0.939	0.953	0.959
AUC	0.837	0.954	0.939	0.909	0.734
AUC (pessimistic)	0.686	0.909	0.939	0.864	0.510

Table 6.5 shows the classification result of the cutwail dataset in seven criteria of performance measure for five type of classifier. From the table above, six classifiers achieve a good detection accuracy which all of them have greater than 85% accuracy value. K-NN obtains the highest detection accuracy which is 97.88%. In criteria precision, Decision Tree, K-NN and Naïve Bayes classifiers obtain the high precision value which is 94.91%, 98.66% and 95.42% respectively. In the criteria of recall, five classifiers achieve a high percentage which all of them greater than 90%. In criteria FPR, K-NN classifier has the lowest percentage which is 8.06%. Random Tree classifier has the lowest value of AUC which is 0.734 while K-NN has a highest value of AUC which is 0.954. Therefore, the classifier that has a good performance among the five classifiers is K-NN classifier. K-NN has highest detection accuracy which is 97.88%, high TPR value and low FPR value.

6.2.3.6 Waledac

Table 6.6: Result of waledac

Classifier / Criteria	Decision Tree	K-NN	Naïve Bayes	Random Forest	Random Tree
Accuracy (%)	90.07	89.08	44.25	85.56	85.72
Precision (%)	89.58	93.40	95.34	85.31	85.55
Recall/ TPR (%)	99.74	93.57	35.10	99.96	99.78
FPR (%)	59.48	33.92	8.80	88.25	86.41
AUC (optimistic)	0.994	0.978	0.828	0.982	0.998
AUC	0.704	0.798	0.827	0.685	0.567
AUC (pessimistic)	0.414	0.618	0.827	0.389	0.136

Table 6.6 shows the classification result of the waledac dataset in seven criteria of performance measure for five type of classifier. In criteria accuracy, Decision Tree classifier has the highest detection accuracy which is 90.07% while Naïve Bayes classifier has the lowest detection accuracy which is 44.25%. In criteria precision, Naïve Bayes classifier has highest precision value which is 95.34% but in criteria recall, Naïve Bayes has lowest recall value which is 35.10%. The lowest recall value decreases the confidence level of Naïve Bayes classifier to classify positive instance. In the criteria of false positive rate, Naïve Bayes classifier has the lowest percentage which is 8.8%. In criteria AUC, Decision Tree, K-NN and Naïve Bayes have high AUC values which are 0.704, 0.798 and 0.827 respectively. Therefore, Decision Tree is a classifier that has a good performance among the five classifiers. Decision Tree has highest detection accuracy which is 90.07%, high TPR value and low FPR value.

6.2.3.7 Comparison of the Performance Measures

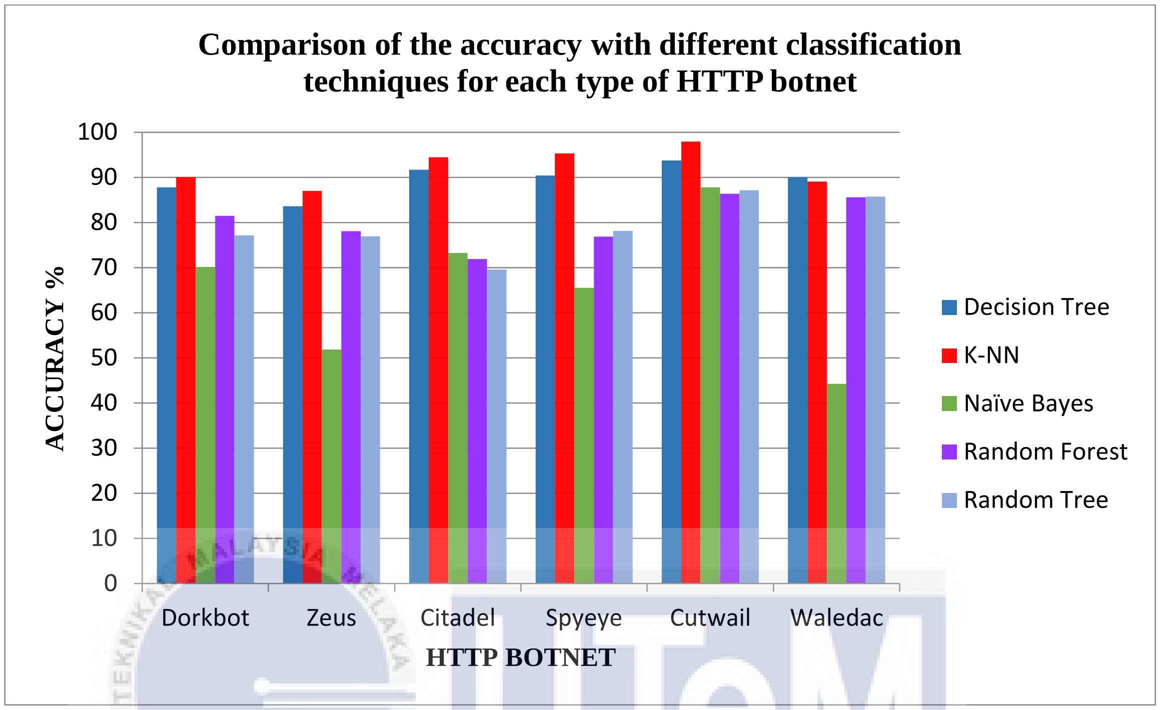


Figure 6.18: Comparison of the accuracy with different classification techniques for each type of HTTP botnet

According to Figure 6.18, the classification technique that achieves the good performance in criteria accuracy for each type of HTTP botnet datasets is K-NN classifier. From the figure above, K-NN classifier has reached almost 90% detection accuracy in classifying malicious and non-malicious network traffic. Then the classification technique which has the second highest detection accuracy is Decision Tree classifier. Figure 6.18 show that Decision Tree classifier has the good result of accuracy which inside the range of 83% until 93%. On the other hands, the classification technique that has the bad performance in criteria accuracy for each type of HTTP botnet datasets is Naïve Bayes classifier. Besides, Random Forest and Random Tree classifiers have achieves almost the same percentage of accuracy for each type of HTTP botnet datasets.

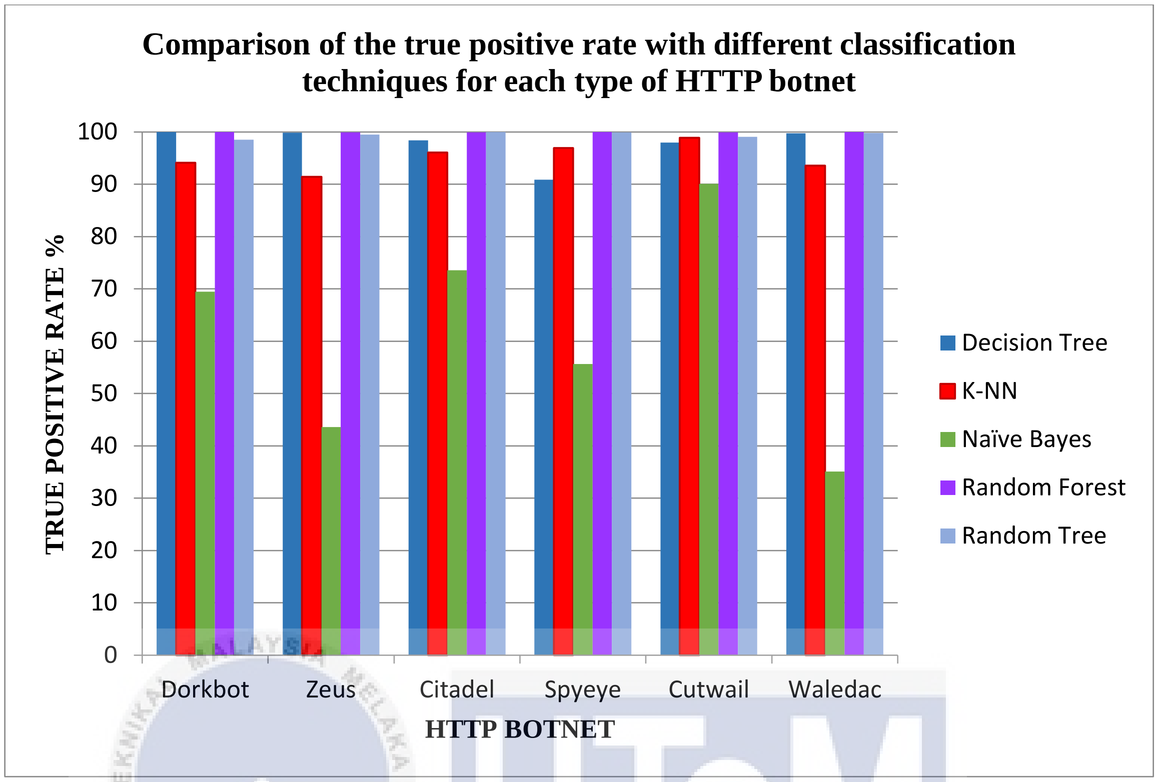


Figure 6.19: Comparison of the true positive rate with different classification techniques for each type of HTTP botnet

According to Figure 6.19, the classification techniques that have high percentage of true positive rate for each type of HTTP botnet dataset are Random Forest and Random Tree classifiers. Both of these two classifiers have more than 95% of true positive rate for each type of HTTP botnet datasets. Besides, Decision Tree and K-NN classifiers have more than 90% of true positive rate in correctly classifying malicious network traffic. From Figure 6.19, Naïve Bayes classifier has the lowest percentage of true positive rate for each type of HTTP botnet datasets. For waledac dataset, Naïve Bayes classifier has only almost 35% of true positive rate in correctly classifying the malicious network traffic.

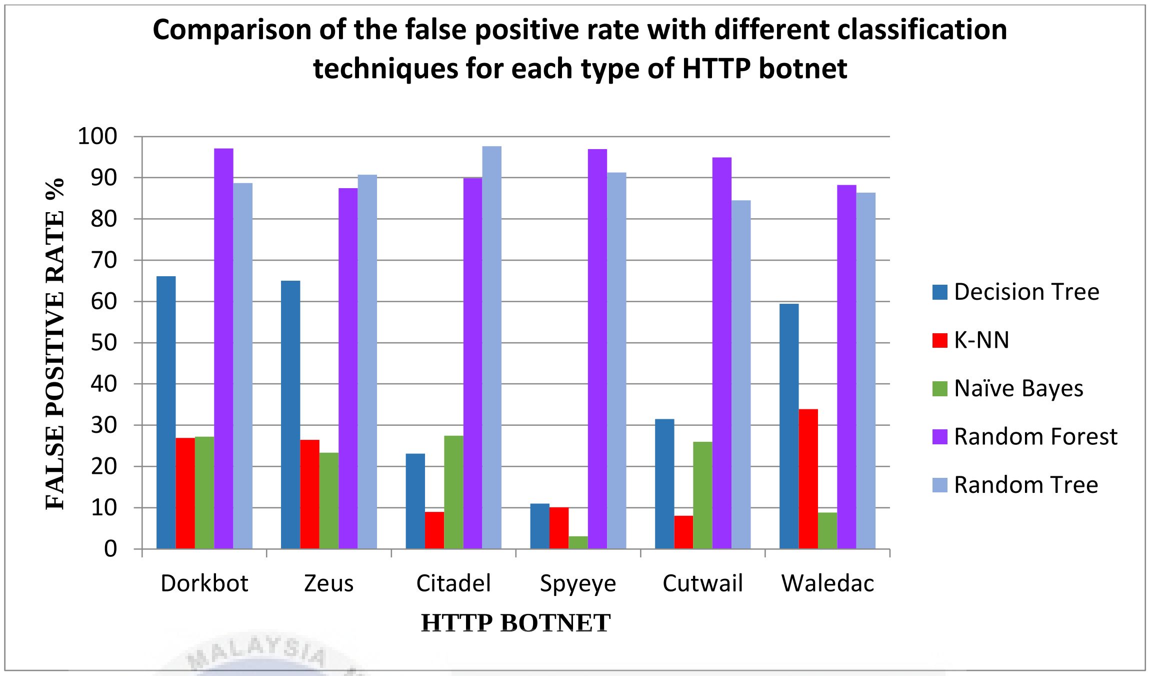


Figure 6.20: Comparison of the false positive rate with different classification techniques for each type of HTTP botnet

According to Figure 6.20, the classification techniques have good performance with the lower false positive rate is K-NN and Naïve Bayes classifiers. From Figure 6.20, both K-NN and Naïve Bayes classifiers have less than 35% of false positive rate for each type of HTTP botnet datasets. The lower false positive rate can decrease the risk of misclassifying benign sample as malware sample. On the other hands, Random Forest and Random Tree have the higher false positive rate which is both of them have more than 84% of false positive rate for each type of HTTP botnet datasets. Then, Decision Tree classifier has achieves the lower false positive rate which is almost 10% for spyeye dataset.

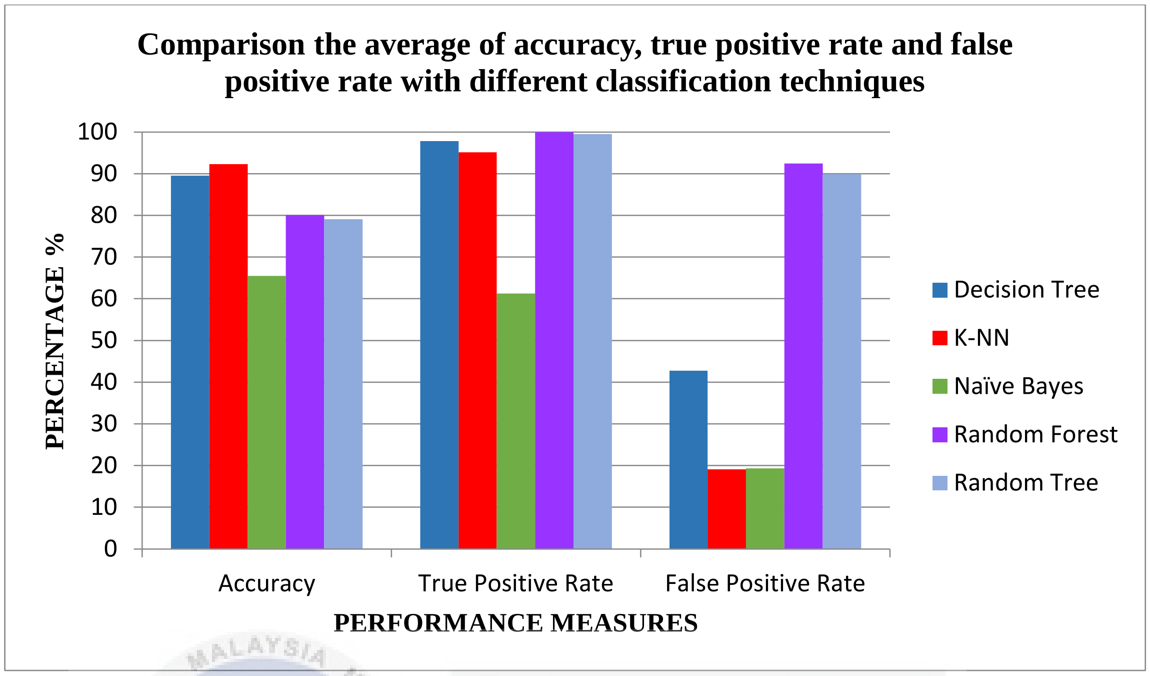


Figure 6.21: Comparison the average of accuracy, true positive rate and false positive rate with different classification techniques

Good performance classifiers are indicated by high value of accuracy, high value of TPR and low value of FPR. According to Figure 6.21, the classifier which has the highest percentage of accuracy is K-NN classifier, while Naïve Bayes classifier has the lowest percentage of accuracy. Besides, in the criteria of true positive rate, most of the classifier achieves the higher percentage of true positive rate such as Decision Tree, K-NN, Random Forest and Random Tree classifiers. Only Naïve Bayes classifier has almost 60% of true positive rate. In the criteria of false positive rate, K-NN and Naïve Bayes classifiers have less than 20% of the false positive rate, while Random Forest and Random Tree classifiers have almost 90% of the false positive rate. However, K-NN is the good performance classifier which has the highest accuracy rate, higher true positive rate and lower false positive rate.

6.3 Summary

At the end of this chapter, the design of classification process was discussed and the result of classification was evaluated. The classification technique which has high accuracy, high true positive rate and low false positive rate is K-NN classifier. Thus, the best classifier among the five classification technique is the K-NN classifier.

CHAPTER VII

PROJECT CONCLUSION

7.1 Introduction

This chapter summarizes the project by describe the overall process of the project and integrate the project objective with the information in the phase of implementation and testing. This chapter also states the weakness and strength of the project. Besides, this chapter also states out the contribution, limitation and the future work of the project.

7.2 Project Summarization

In conclusion, this project has been separate into two parts which are data collection and data analysis. Data collection was completed in PSM 1 while data analysis was completed in PSM 2. During PSM 1, six types of HTTP botnet binary files were downloaded from several websites. The six types of HTTP botnet are Dorkbot, Zeus, Citadel, Spyeye, Cutwail and Waledac. After that, the HTTP botnet binary files were released in a control network for the duration of 7 days. Then, the malicious network traffic was collected. On the same time, the real network traffic is needs to generate by browsing the HTTP website for the duration of 1 day. Then, both malicious and non-malicious network traffic is needed to combine together.

During PSM 2, the tcpdump file is needed to extract into CSV file by using tcptrace. After that, before classify the combined network traffic, data preprocessing is needed to execute such as filling the missing value and delete redundant parameter of the dataset. Then, dataset can upload into Rapid Miner and continuous the process of classification. The five classification techniques will implement in this project which are Decision Tree, K-NN, Naïve Bayes, Random Forest and Random Tree. Finally, the performance measure between the five types of classification technique was needed to evaluate for classify the malicious and non-malicious network traffic. After analyses the result of classification, K-NN is the best classifier among the five type of classifier which has the highest detection accuracy, high true positive rate and low false positive rate.

Finally, the weakness of this project is time-consuming. At the stage of collecting malicious and non-malicious network traffic, a lot of time was needed to spend to collect the network traffic and extract the tcpdump file into CSV file. Besides, at the phase of classification, the process of classification running in Rapid Miner was needed to wait for maximum until 20 hours. On the other hands, the strength of this project is compare five type of the classification technique to select the best performance classifier in classifying malicious and non-malicious network traffic.

7.3 Project Contribution

The project contribution is helping to enhance Intrusion Detection System (IDS) or Firewall to detect HTTP botnet. The output of this project is to detect the HTTP botnet by classification the network parameter to identify the relationship of the network parameter with HTTP botnet. At the end of this project, the non-botnet network traffic and botnet network traffic can be differentiated.

7.4 Project Limitation

The project limitation is this project only used six type of HTTP botnet such as Dorkbot, Zeus, Citadel, Spyeye, Cutwail and Waledac. Besides the other limitation of this project is this project only implement and analyze five type of classification technique in classifies malicious and non-malicious network traffic. The five types of classification techniques is Decision Tree, K-NN, Naïve Bayes, Random Forest and Random Tree.

7.5 Future Work

The future work of the project is added the feature selection technique to increase the accuracy of the result of classification. Feature selection can select the significant attributes and improve the performance of classification process for classifying the malware sample and benign sample.

7.6 Summary

Finally, this project aims to investigate the network parameter of the network and study the relationship between network parameter with HTTP botnet using five classification techniques. After the evaluation of the performance measure for the five classification techniques, the best classifier is the classifier with the highest accuracy, high true positive rate and low false positive rate which is K-NN classifier.

REFERENCES

Basheer, I.A. & Hajmeer, M., 2000. Artificial neural networks : fundamentals , computing , design , and application. , 43, pp.3–31.

Cho, S. & Won, H., 2003. Machine Learning in DNA Microarray Analysis for Cancer Classification.

Dumais, S., 1996. A Bayesian Approach to Filtering Junk E-Mail. , (Cohen).

Erbacher, R.F. et al., 2008. A Multi-Layered Approach to Botnet Detection. Journal of Security and Management, pp.301–308.

Eslahi, M., Hashim, H. & Tahir, N.M., 2013. An efficient false alarm reduction approach in HTTP-based botnet detection. IEEE Symposium on Computers and Informatics, ISCI 2013, pp.201–205.

Fedynyshyn, G., Chuah, M.C. & Tan, G., 2011. Detection and classification of different botnet C&C channels. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6906 LNCS, pp.228–242.

Fielding, R., Irvine, U.C. & Gettys, J., 1999. Hypertext Transfer Protocol -- HTTP / 1 . 1 Status of this Memo. , pp.1–114.

Gao, H. et al., 2010. Detecting and characterizing social spam campaigns. Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, pp.35–47. Available at: http://delivery.acm.org/10.1145/1880000/1879147/p35-gao.pdf?ip=137.166.81.123&id=1879147&acc=ACTIVE SERVICE&key=65D80644F295BC0D.C3714298A2589389.4D4702B0C3E38B35.4D4702B0C3E38B35&CFID=484687663&CFTOKEN=14116697&__acm__=1425715693_438e8d4865ac243f46.

Giroire, F. et al., 2009. Exploiting temporal persistence to detect covert botnet channels. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5758 LNCS, pp.326–345.

Goebel, J. & Holz, T., 2007. Rishi: identify bot contaminated hosts by IRC nickname evaluation. HotBots'07 Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, p.8. Available at: http://dl.acm.org/citation.cfm?id=1323128.1323136.

Gu, G., Zhang, J. & Lee, W., 2008. BotSniffer : Detecting Botnet Command and Control Channels in Network Traffic. Proceedings of the 15th Annual Network and Distributed System Security Symposium., 53(1), pp.1–13. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.8092&amp;rep=rep1&amp;type=pdf.

Kapratwar, A., 2016. Static and Dynamic Analysis for Android Malware Detection.

Karasaridis, A., Rexroad, B. & Hoeflin, D., 2007. Wide-scale botnet detection and characterization. HotBots'07 Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, p.7. Available at: http://dl.acm.org/citation.cfm?id=1323128.1323135.

Kolter, J.Z. & Maloof, M. a, 2006. Learning to Detect and Classify Malicious Executables in the Wild. Journal of Machine Learning Research, 7, pp.2721–2744. Available at: http://portal.acm.org/citation.cfm?id=1248646.

Kotsiantis, S.B., 2007. Supervised machine learning: A review of classification techniques. … Applications in Computer Engineering: Real Word …, 31, pp.249–268. Available at: http://books.google.com/books?hl=en&lr=&id=vLiTXDHr_sYC&oi=fnd&pg=PA3&dq=survey+machine+learning&ots=CVsyuwYHjo&sig=A6wYWvywU8XTc7Dzp8ZdKJaW7rc\npapers://5e3e5e59-48a2-47c1-b6b1-a778137d3ec1/Paper/p800.

Lee, J. et al., 2008. The Activity Analysis of Malicious HTTP-based Botnets using Degree of Periodic Repeatability *. , pp.83–86.

Lu, W., Tavallaee, M. & Ghorbani, A.A., 2009. Automatic Discovery of Botnet Communities on Large-Scale Communication Networks.

Mah, B.A., 1997. An Empirical Model of HTTP Network Traffic. , pp.592–600.

Maron, M.E. & Kuhns, J.L., 1960. On Relevance, Probabilistic Indexing and Information Retrieval. Journal of the ACM, 7(3), pp.216–244.

Ming-Syan Chen, Jiawei Han, P.S.Y., 1996. Data Mining: An Overview from a Database Perspective.pdf. , p.18.

Problem, P.I. & Li, Y., 2013. Application of Machine Learning Techniques to. , pp.1–21.

Puerta, D. et al., 2013. A Supervised Classification Approach for Detecting Packets Originated in a HTTP-based Botnet. , 16(03), pp.1–13.

Roughan, M. & Sen, S., 2004. Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification. Proceedings of the 4th …, pp.135–148. Available at: http://dl.acm.org/citation.cfm?id=1028805.

Saad, S. et al., 2011. Detecting P2P botnets through network behavior analysis and machine learning. 2011 9th Annual International Conference on Privacy, Security and Trust, PST 2011, (February 2016), pp.174–180.

Settles, B., 2010. Active Learning Literature Survey. Machine Learning, 15(2), pp.201–221.

Strayer, W.T. et al., 2006. Detecting botnets with tight command and control. Proceedings - Conference on Local Computer Networks, LCN, (February 2016), pp.195–202.

Tan, E. et al., 2013. UNIK: unsupervised social network spam detection. Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, pp.479–488. Available at: http://dl.acm.org/citation.cfm?id=2505581.

Zaki, M. et al., 2014. ANALYSIS OF FEATURES SELECTION AND MACHINE LEARNING CLASSIFIER IN ANDROID MALWARE DETECTION.

Zanero, S. & Savaresi, S.M., 2004. Unsupervised learning techniques for an intrusion detection system. Proceedings of the 2004 ACM symposium on Applied computing - SAC ’04, (DECEMBER 2003), p.412. Available at:

http://portal.acm.org/citation.cfm?doid=967900.967988.

Zeidanloo, H.R., Zadeh, M.J. & Zamani, M.S., 2010. A Taxonomy of Botnet Detection Techniques. Journal of Computer Science, 7(3), pp.158–162.

Zhao, D. et al., 2013. Botnet detection based on traffic behavior analysis and flow intervals. Computers & Security, 39(March 2016), pp.2–16. Available at: http://www.sciencedirect.com/science/article/pii/S0167404813000837.

APPENDIX

A. Gantt Chart

Task Name	Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Deciding title of Proposal PSM	■													
Proposal PSM Submission and Presentation		■												
Chapter 1: Introduction		■	■	■										
Submission of Chapter 1			■	■										
Chapter 2: Literature Review			■	■	■	■	■							
Submission of Chapter 2						■	■							
Chapter 3: Methodology						■	■	■						
Submission of Chapter 3								■	■	■				
Chapter 4: Design									■	■	■	■		
Submission of chapter 4												■	■	
Prepare PSM 1 Report												■	■	■
Final Presentation (Pa)														■
Submission overall marks to PSM/PD committee.														■

B. **Network Parameter**

Parameter	**Description**
conn	Unique identifier for each connection in the relation
host_a	The IP address of the ‘a’ machine involved in the connection. This is usually the machine that initiated the connection.
host_b	The IP address of the 'b' machine in the connection. This is usually the machine that was contacted by the host a machine.
port_a	The port number used by the application on the ‘a’ host machine for this connection. Port numbers can be associated with services and applications by relating them to the Services table in the database. If this is a low number (e.g. less than 1024), it corresponds to a well-known service. This is more likely to be the case for the port_b field below as clients connect to servers on their well-known ports.
port_b	The port number for the ‘b’ host machine in the connection. If this is a server machine, the port will indicate what the application is that created the connection. This is found by relating the port number to the Services table.
first_packet	The time of the first packet sent in the connection, i.e. the SYN packet establishing the connection. This time is UNIX time, i.e. the number seconds since midnight on January 1st, 1970.
last_packet	The time of the last packet in the connection, again in UNIX time.
total_packets_a2b	The total number of packets sent from host a to host b in this connection.

total_packets_b2a	The total number of packets sent from host b to host a in this connection.
resets_sent_a2b	The number of communications from host a to host b within the connection that requested that the connection be reset. These are packets with the RST flag set.
resets_sent_b2a	The number of packets with the RST flag set sent from host b to host a.
ack_pkts_sent_a2b	The number of packets that had the Acknowledgement flag set, i.e. providing feedback to host b from host a that it had seen certain packets.
ack_pkts_sent_b2a	The number of Acknowledgement packets sent from host b to host a.
pure_acks_sent_a2b	The number of packets from host a to host b that only had the Acknowledgement flag set and not any of the DATA, SYN, FIN or RST flags set.
pure_acks_sent_b2a	The number of packets from host b to host a that only had the Acknowledgement flag set and not any of the DATA, SYN, FIN or RST flags set.
sack_pkts_sent_a2b	The number of packets from host a to host b containing an Acknowledgement and a Selective Acknowledgement flag.
sack_pkts_sent_b2a	The number of packets from host b to host a containing an Acknowledgement and a Selective Acknowledgement flag.
dsack_pkts_sent_a2b	The number of packets containing a Delayed Acknowledgement flag sent from host a to host b.
dsack_pkts_sent_b2a	The number of packets containing a Delayed Acknowledgement flag sent from host b to host a.
max_sack_blks_ack_a2b	Most SACKnowledgement blocks in a single Acknowledgement from host a to host b.

max_sack_blks_ack_b2a	Most SACKnowledgement blocks in a single Acknowledgement from host b to host a.
unique_bytes_sent_a2b	The total number of non-retransmitted bytes sent from host a to host b. This can be used along with the total number of transmitted bytes to determine how many bytes were re-transmitted and hence "lost".
unique_bytes_sent_b2a	The total number of non-retransmitted bytes sent from host b to host a. This can be used along with the total number of transmitted bytes to determine how many bytes were re-transmitted and hence "lost".
actual_data_pkts_a2b	The number of segments of data sent from host a to host b.
actual_data_pkts_b2a	The number of segments of data sent from host b to host a.
actual_data_bytes_a2b	The number of bytes of data sent from host a to host b.
actual_data_bytes_b2a	The number of bytes of data sent from host b to host a.
rexmt_data_pkts_a2b	The number of retransmitted packets from host a to host b.
rexmt_data_pkts_b2a	The number of retransmitted packets from host b to host a.
rexmt_data_bytes_a2b	The number of retransmitted bytes from host a to host b.
rexmt_data_bytes_b2a	The number of retransmitted bytes from host b to host a.
zwnd_probe_pkts_a2b	The number of zero window probe packets sent from host a to host b. These are packets inquiring about the buffer size on the other machine
zwnd_probe_pkts_b2a	The number of zero window probe packets sent from host b to host a. These are packets inquiring about the buffer size on the other machine
zwnd_probe_bytes_a2b	The number of zero window probe bytes sent from host

	a to host b. These are packets inquiring about the buffer size on the other machine.
zwnd_probe_bytes_b2a	The number of zero window probe bytes sent from host b to host a. These are packets inquiring about the buffer size on the other machine.
outoforder_pkts_a2b	The number of out of order packets that were sent from host a to host b, i.e. received out of order by host b.
outoforder_pkts_b2a	The number of out of order packets that were sent from host b to host a, i.e. received out of order by host a.
pushed_data_pkts_a2b	The number of packets sent from host a to host b with the PUSH bit set, i.e. basic transfer of data rather than transmission control.
pushed_data_pkts_b2a	The number of packets sent from host b to host a with the PUSH bit set, i.e. basic transfer of data rather than transmission control.
SYN_pkts_sent_a2b	The number of Synchronization packets sent from host a to host b.
FIN_pkts_sent_a2b	The number of Finish packets sent from host a to host b.
SYN_pkts_sent_b2a	The number of Synchronization packets sent from host b to host a.
FIN_pkts_sent_b2a	The number of Finish packets sent from host b to host a.
req_1323_ws_a2b	Logical value indicating whether the '1323' window scaling was requested by host a.
req_1323_ts_a2b	Logical value indicating whether the '1323' time stamp was requested by host a.
req_1323_ws_b2a	Logical value indicating whether the '1323' window scaling was requested by host b.
req_1323_ts_b2a	Logical value indicating whether the '1323' time stamp was requested by host b.

adv_wind_scale_a2b	Window scale factor for host a to host b.
adv_wind_scale_b2a	Window scale factor for host b to host a.
req_sack_a2b	Logical value indicating whether host a requested Synchronized Acknowledgements
req_sack_b2a	Logical value indicating whether host b requested Synchronized Acknowledgements
sacks_sent_a2b	The number of SACKs sent from host a to host b.
sacks_sent_b2a	The number of SACKs sent from host b to host a.
urgent_data_pkts_a2b	Number of packets sent from a to host b with the URGENT bit set.
urgent_data_pkts_b2a	Number of packets sent from b to host a with the URGENT bit set.
urgent_data_bytes_a2b	Number of bytes sent from a to host b in packets with the URGENT bit set.
urgent_data_bytes_b2a	Number of bytes sent from b to host a in packets with the URGENT bit set.
mss_requested_a2b	Maximum segment size in communication from host a to host b.
mss_requested_b2a	Maximum segment size in communication from host b to host a.
max_segm_size_a2b	Largest amount of data in a segment sent from host a to host b.
max_segm_size_b2a	Largest amount of data in a segment sent from host b to host a.
min_segm_size_a2b	Smallest amount of data in a segment sent from host a to host b.
min_segm_size_b2a	Smallest amount of data in a segment sent from host b to host a.
avg_segm_size_a2b	Average segment size sent from host a to host b, given by the number of data bytes sent divided by the number of packets.

avg_segm_size_b2a	Average segment size sent from host b to host a, given by the number of data bytes sent divided by the number of packets.
max_win_adv_a2b	The maximum window size advertised by host a to host b in this connection.
max_win_adv_b2a	The maximum window size advertised by host b to host a in this connection.
min_win_adv_a2b	The minimum window size advertised by host a to host b in this connection.
min_win_adv_b2a	The minimum window size advertised by host b to host a in this connection.
zero_win_adv_a2b	The number of ZERO windows advertised by host a to host b.
zero_win_adv_b2a	The number of ZERO windows advertised by host b to host a.
avg_win_adv_a2b	Average window advertisement by host a to host b.
avg_win_adv_b2a	Average window advertisement by host b to host a.
initial_window_bytes_a2b	The number of bytes in the initial window from host a to host b.
initial_window_bytes_b2a	The number of bytes in the initial window from host b to host a.
initial_window_pkts_a2b	The number of segments in initial window from host a to host b.
initial_window_pkts_b2a	The number of segments in initial window from host b to host a.
ttl_stream_length_a2b	Not used.
ttl_stream_length_b2a	Not used.
missed_data_a2b	The number of bytes that were dropped by host b. This is the total stream length from host a to host b - number of unique bytes sent.
missed_data_b2a	The number of bytes that were dropped by host a. This

	is the total stream length from b to a - number of unique bytes sent.
truncated_data_a2b	Not used.
truncated_data_b2a	Not used.
truncated_packets_a2b	Not used.
truncated_packets_b2a	Not used.
data_xmit_time_a2b	Number of seconds from first to last transmission from host a to host b.
data_xmit_time_b2a	Number of seconds from first to last transmission from host b to host a.
idletime_max_a2b	Maximum idle time for host a in communicating with host b on this connection.
idletime_max_b2a	Maximum idle time for host b in communicating with host a on this connection.
hardware_dups_a2b	Not used.
hardware_dups_b2a	Not used.
throughput_a2b	The throughput for this side of the connection. This is the number of bytes divided by connection duration from host a to host b.
throughput_b2a	The throughput for this side of the connection. This is the number of bytes divided by connection duration from host b to host a.
class	To differentiate or label both malicious and non-malicious dataset while 1 represented as malicious dataset and 0 represented as non-malicious dataset.