NETWORK BEHAVIORAL ANALYSIS FOR DETECTION OF REMOTE ACCESS TROJANS

KHIN SWE YIN

UNIVERSITY OF COMPUTER STUDIES, YANGON

SEPTEMBER, 2019

Network Behavioral Analysis for Detection of Remote Access Trojans

Khin Swe Yin

University of Computer Studies, Yangon

A thesis submitted to the University of Computer Studies, Yangon in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

September, 2019

Statement of Originality

I hereby certify that the work embodied in this dissertation is the result of original research and has not been submitted for a higher degree to any other University or Institution.

…..…… ……..……

Date Khin Swe Yin

ACKNOWLEDGEMENTS

First of all, I would like to thank His Excellency, the Minister, the Ministry of Education, for full facilities support during the Ph.D Course at the University of Computer Studies, Yangon.

Secondly, I would like to express very special thanks to Dr. Mie Mie Thet Thwin, the Rector, the University of Computer Studies, Yangon, for allowing me to develop this dissertation and giving me general guidance during the period of my study.

Thirdly, I would like to express special thanks to my external examiner Dr. Thandar Phyu, Director of Technology, ATG Company Ltd., for useful comments and suggestions.

I would like to express my deepest gratitude to my supervisor, Professor Dr. May Aye Khine, for her tremendous support and guidance during the course of my doctoral study. Her insights into research has inspired me significantly in my pursuit of research.

I would also like to extend my special appreciation and thanks to Dr. Khine Moe Nwe, Professor and Dean of the PhD course, for the useful comments, advices and insight which are invaluable to me.

I would also like to thank my dissertation committee. Without their insightful comments and feedback, this dissertation would not be possible.

I would like to express my respectful gratitude to Daw Aye Aye Khine, Associate Professor, Head of English Department, for her valuable supports from the language point of view and pointed out the correct usage in my dissertation.

I would also like to thank all my teachers for mentoring, encouraging, and recommending the dissertation. I am also grateful to all talented and friendly colleagues from PhD 9th batch for their friendship and accompany during my graduate student life. Special thanks go to all the past and current members at Numerical Analysis Laboratory and Cyber Security Laboratory.

Finally, I want to express sincerest and deepest gratitude to my father, mother, sister and brother for their unconditional love, support and encouragement. I would not have been where I am today without their endless love and tremendous support. My dissertation is dedicated to them.

ABSTRACT

Today's world is connected through the Internet, everyone can connect each other and people do business on the Internet like online shopping and online banking. Social networking sites are widely used and people save their sensitive data on connected computer or laptop or mobile phone. Therefore, information security is increasingly important to be protected properly. Remote Access Trojan (RAT), a kind of malware, is one of malwares that disclose confidential information to the wrong party. It passes through network to give command to the victim and control it remotely. Researchers have proposed many approaches to detect such kind of malware. However, threat actors use canning ways to create new malwares, and so new Remote Access Trojans and variants of existing RATs are emerging every day. The popular Advanced Persistent Threat (APT) and targeted attacks also use the command and control communication like RAT to intrude and control a victim remotely.

There are three main challenges facing Remote Access Trojan detection. First, signature-based detection is not enough to catch up RATs that camouflage themselves by using encryption and polymorphism. Second, there is lack of effective features for machine learning methods to identify the behavior of Remote Access Trojans although behavior-based detection is useful for detecting unknown malware. Third, there is much overhead and time takes long in extracting features from a session that starts SYN packet of TCP three-way handshake to the end of the traffic in network traffic classification.

Both network-based and behavior-based approaches are applied in developing software for malware detection. Network behavioral analysis has been done many years for two objectives: to detect the command and control traffic of malwares like Remote Access Trojans so that confidential data cannot be disclosed to the wrong person, and to classify network traffic so that network administrator can manage his or her related network easily.

Network behavioral analysis is done and a new approach of feature extraction for detecting Remote Access Trojans in the early stage is proposed and implemented in this thesis. The malicious behavior of Remote Access Trojans is differentiated from normal network traffic in the first twenty packets of network traces. Four machine learning algorithms are applied for classification and their parameters are tuned perfectly in order to obtain the best performance. This thesis makes three primary

contributions. First, the detection solutions proposed fundamental behavior of Remote Access Trojans and are immune to malware obfuscation and traffic encryption. Second, the solutions are general enough to identify different types of Remote Access Trojans and they can also be extended to counter next-generation Remote Access Trojans. Third, the detection solutions function to meet the needs of network traffic classification in order to differentiate normal traffic and malicious one.

By accomplishing this approach, Remote Access Trojans are detected in the early stage and it is not necessary to wait until the end of the network traffic in capturing network traffic for network traffic classification. It approaches to achieve the objectives of information security: Confidentiality, Integrity and Availability (CIA). Limitations and future work are also explained clearly.

Table of Contents

List of Figures

List of Tables

List of Equations

CHAPTER 1
INTRODUCTION

As the internet and new technologies are widely used around the world, the world is increasingly connected and new forms of social interactions and market places for the sale of products and services has evolved. People use online banking, online shopping and credit cards and debit cards to buy something on the Internet. Just like in the physical world, there are people on the Internet who breach network security for financial gain, reputation, business disruption and intellectual property. They use malwares to accomplish their goals. Moreover, the hackers and crackers are looking for any security vulnerability to exploit the victim's computer or network using malware. New viruses, malwares, and other threats are still emerging and propagating on the Internet, and information security risks have been increasingly important.

Remote Access Trojans are malware camouflaged as legitimate software and, they trick users to be installed in a user's computer without user knowing. Malware authors create Remote Access Trojan for spying on, hijacking, or destroying computers. It includes a backdoor to control a remote computer. The attackers may distribute Remote Access Trojans to other vulnerable computers to establish botnet or get sensitive data from competitors through targeted attack.

A targeted attack is a type of threat in which threat actors actively pursue and compromise a target entity's infrastructure while maintaining anonymity [73]. These attackers have some expertise and enough resources to implement their plan. Targeted attacks are one of the biggest cyber-threats to an organization in Internet-connected world. Targeted attacks use Remote Access Trojans as their weapon. The targeted company loses reputation, or costs millions in damages because of targeted attacks.

Generally, the RATs do not affect the performance of computer and computers are infected by a RAT for years without user noticing anything wrong. Malwares like spyware or keylogger are created for single purpose. Spyware is a software that gathers login information, browsing history and internet usage habits of a user without user's knowledge, and sends such information to another entity through cookies. Keylogger records everything the user types and it is used for obtaining email address, password and credit card number. Unlike these malwares, RATs are special ones that give hackers complete control over infected computers.

There are three main challenges in detecting RATs. First, like other types of malware, current Remote Access Trojans commonly employ obfuscation techniques such as encryption and polymorphism. Using these techniques, the RAT code can mutate without changing the functions or the semantics of its payload. Since signature-based detection schemes look for specific data patterns in binaries or payload, it is difficult for them to identify all obfuscated RATs. Second, behavioral-based detection can uncover the variants of RATs and unknown RATs. However, there is lack of effective features that increase the detection rate and to reduce False Positive Rate and False Negative Rate. Lastly, features are extracted only when the current interaction between two hosts ends. It is much overhead and takes time.

Network behavioral analysis is used to classify application layer traffic and to detect malware [52] [85]. In our work, network behavioral features are defined and calculated based on network behavioral analysis, and supervised machine learning methods are applied for classifying normal and malicious Remote Access Trojans traffic. Time does not take long, and malicious traffic can be differentiated from normal ones. Effective features are also proposed and the approach gets comparable classification accuracy, False Positive Rate and False Negative Rate. Moreover, it is easy to manage and detect the malicious ones as it is a network-based system. In addition, it can also be applied in detecting unknown Remote Access Trojans (RAT) and variants of known RATs that use TCP connection.

1.1 Trojan Horse

A Trojan horse or Trojan is a type of malicious code or software that is often disguised as legitimate software. Generally, there are twelve types of Trojans according to their actions. [80][77]

(1) Remote Access Trojan
(2) Backdoor Trojan
(3) Distributed Denial of Service (DDoS) attack Trojan
(4) Downloader Trojan
(5) Fake AV Trojan
(6) Ransom Trojan
(7) Rootkit Trojan

(8) Trojan banker

(9) Trojan IM

(10) Game-thief Trojan

(11) SMS Trojan

(12) Mailfinder Trojan

Remote Access Trojan can give an attacker full control over victim's computer via a remote network connection. It is extremely dangerous because it gives full control of victim's PC to the attacker. Backdoor Trojan can create a "backdoor" on victim's computer that lets an attacker access and control the victim. Data can be downloaded by a third party or more malware can be uploaded to victim's computer. Distributed Denial of Service (DDoS) attack Trojan performs DDoS attacks. Downloader Trojan downloads and installs new versions of malicious programs. Fake AV Trojan behaves like antivirus software, but it demands money from user to scan computer and remove threats, although the threats that they report are actually non-existent. Ransom Trojan blocks victim's data or impairs the computer's performance and demand the ransom money to restore the computer's performance or unblock data. Rootkit Trojan conceals certain objects or activities in computer system in order to protect malicious code being detected and to stay as long as possible on an infected computer. Trojan banker steals victim's confidential data through online banking, e-payment systems and credit or debit card. Trojan IM steals users' logins and passwords on IM platforms. Game-thief Trojan steals user account information from online gamers. SMS Trojan sends text messages from a victim's mobile device to premium rate phone numbers and drive up phone costs. Mailfinder Trojan steals the email addresses accumulated on victim's computer.

1.1.1 Remote Access Trojans

Remote Access Trojan conceal themselves as legitimate software, used to trick users into unknowingly installing malware. An attacker can fully control a victim's PC through Trojan. A RAT generally consists of two sides: A client and a server. The server-side is executed on the victim, and the client-side is executed on the attacker to control the victim. The attacker controls the victim's PCs secretly from remote in order to steal confidential data, erase or overwrite data, listen the key logger or capture the system screen. Remote Access Trojans can camouflage themselves under legitimate

program and hence never appear in task manager. RATs use reverse connections to connect remote computer and to remain undetected. The activities that the attackers can do over the victim's PC are

(1) Steal confidential data, such as passwords, security codes, credit card information using keyloggers
(2) Erase or overwrite data
(3) Listen the key logger or capture the system screen
(4) Create a botnet through a victim to perform DDoS attacks
(5) Delete or replace OS critical files
(6) Create fake traffic to perform DoS
(7) Download and upload spyware, adware and malware
(8) Record screenshots, audio and video of victim´s PC
(9) Disable firewall and anti-virus software
(10) Set up victim´s PC as a proxy server to relay attacks

How Remote Access Trojan works is shown in Figure 1.1. Scenario of RAT is shown in Figure 1.2:

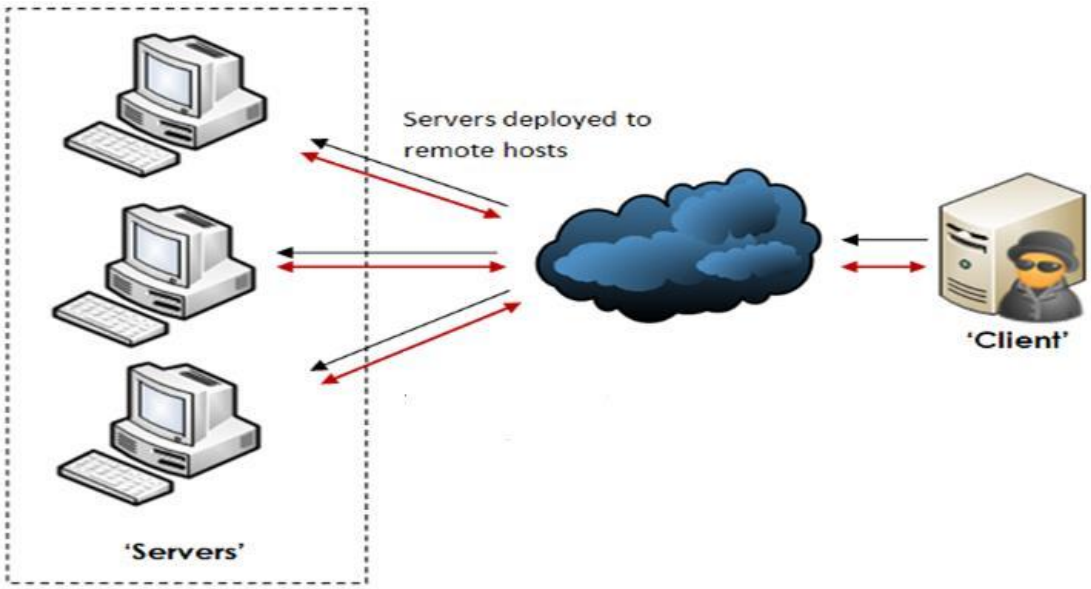

Figure 1.1: How Trojan works through Internet [44]

Scenario of RAT is shown below:

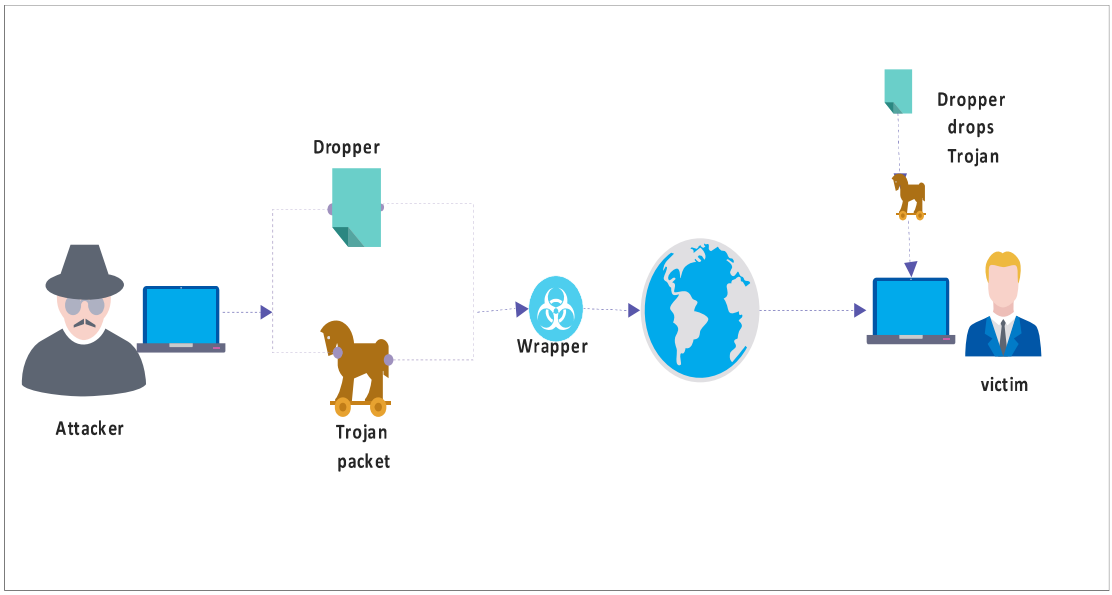

Figure 1.2: Scenario of a Remote Access Trojan

Trojans usually camouflage behind a genuine code or file to avoid detection by antivirus scanner. Behind the scene, it sets up a backdoor connection with the remote attacker. It has 3 stages: Dropper, Malicious code and Wrapper.

(1) Dropper: It installs malicious code into the target.

(2) Malicious code: It exploits the system and gives the attacker control over the target.

(3) Wrapper: It wraps dropper, malicious code, genuine code into one exe package.

When a victim downloads an infected file, dropper installs the malicious code first and then the genuine program.

1.1.2 Why RAT is Important

Center for Internet Security (CIS) mentioned TOP ten malware for January 2018 [75]. According to CIS, Trojans are the most infected malware in January 2018. Among them, there are two Remote Access Trojans – Gh0st and NanoCore.

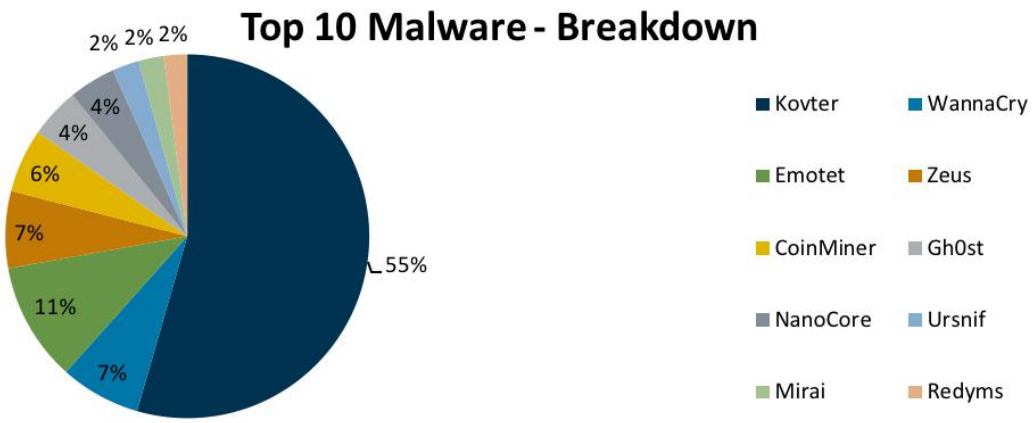

Figure 1.3: Top 10 malware January 2018 [75]

(1) Kovter is a Trojan and it is spread via malspam email attachments that contain malicious office macros. After a victim is infected by Kovter, it serves as a backdoor to attacker using command and control communication.

(2) Emotet is a Trojan and it drops banking Trojans. It infects victim's computer via emails that contain malicious download links.

(3) WannaCry is a ransomware worm that is spread via malspam.

(4) ZeuS/Zbot is a banking Trojan and it uses keystroke logging to compromise victim when a customer uses a banking website.

(5) CoinMiner is a cryptocurrency miner. It is spread via malvertising. After infection, CoinMiner uses Windows Management Instrument (WMI) and EternalBlue to exploit SMB and spread across a network. It uses the WMI Standard Event Consumer scripting to execute scripts persistently.

(6) Gh0st is a Remote Access Trojan used to control infected endpoints. It allows an attacker to control the infected machine remotely.

(7) NanoCore is a Remote Access Trojan (RAT) and it spreads via malspam as an Excel file. It downloads and executes files, visits websites, and adds registry keys to control persistently.

(8) Ursnif and Dreambot are banking trojans and they are known for weaponizing documents. It collects victim information from login pages and web forms.

(9) Mirai is a botnet and it compromises Internet of Things (IoT) devices and it conducts large-scale distributed denial of service (DDoS) attacks. It is dropped

after a successful exploit and then the attacker gains access to an infected machine.

(10) Redyms is a trojan and it is downloaded via exploit kit. It is primarily distributed in the United States.

According to Proofpoint Threat Report, Remote Access Trojans are still one of the most infected malwares in the third quarter of 2018 [56].

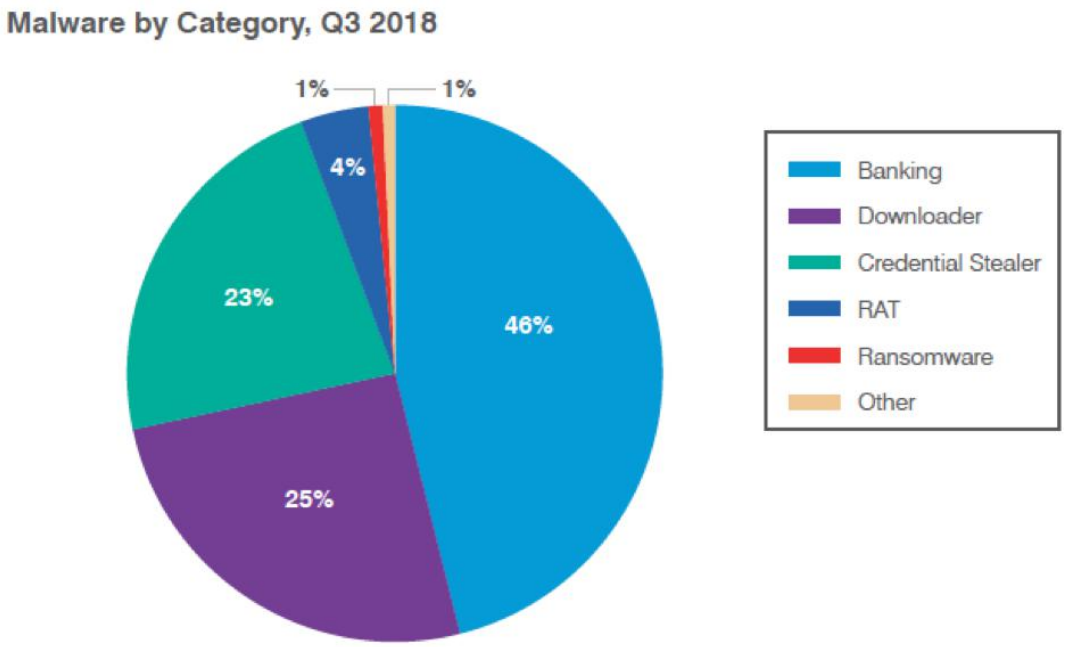

Figure 1.4: Relative distribution of malware in third quarter of 2018 [56]

1.1.3 Targeted Attacks

A targeted attack is used by threat actors to trace and compromise a target entity's infrastructure secretly. Phases of a Targeted Attack are explained below [73].

(1) Intelligence gathering
(2) Point of entry
(3) Command-and-control (C&C) communication
(4) Lateral movement
(5) Asset/Data Discovery
(6) Data Exfiltration

(1) Intelligence gathering: Threat actors collect information like the intended target's IT environment and its organizational structure to customize their attacks. They use social engineering techniques to get critical information like recent events, work-related issues or concerns, and other areas of interest.

(2) Point of entry: Spear phishing email, zero-day or software exploits, watering hole techniques, instant-messaging and social networking platforms are used to tempt targets to click a link or download malware. Then they establish a connection with the target.

(3) Command-and-control (C&C) communication: After breaching security, threat actors constantly communicate to the malware to execute malicious code or download information from the victim. They do this communication secretly.

(4) Lateral movement: Inside the network, threat actors move throughout the network to search critical information or infect further valuable systems.

(5) Asset/Data Discovery: Notable assets or data are classified and isolated for future data exfiltration. These data are transferred through tools like Remote Access Trojans. Moreover, file lists in different directories may be sent back to the attacker so that valuable data can be identified.

(6) Data Exfiltration: The objective of an attack is to gather critical information. Data can be transferred gradually. Targeted attacks can hide to avoid detection and access to the victim's confidential data. These data include intellectual property, trade secrets, and customer information. Moreover, attackers may also seek other sensitive data such as top-secret documents from government or military institutions.

1.1.4 Advanced Persistent Threats (APT)

APTs are sophisticated and well-structured cyber-attacks targeting specific information in high profile companies and governments, usually in a long-term campaign involving different steps. [14]. APTs can do large-scale security breaches and various damages before an organization detects them. APT has 6 stages and they are**:**

(1) Reconnaissance and Weaponization

(2) Delivery

(3) Initial Intrusion

(4) Command and Control

(5) Lateral Movement

(6) Data Exfiltration

In reconnaissance and weaponization, malware is prepared and the attackers utilize social engineering, tools like OSINT. Activities like spear phishing and watering hole attack are applied for delivery. Zero-day exploits or remote code execution is done as initial intrusion. Exploiting legitimate services, RAT, Encryption methods are used for command and control communication. As lateral movement, the attackers try to escalate privilege and collect data. Then, they do compression, encryption and intermediary staging for data exfiltration.

1.1.5 Comparison of RAT, Targeted Attack and APT

Table 1.1 Comparison of RAT, targeted attack and APT

	RAT	Targeted attacks	APT
Attacker	Mostly, a person	A group	Highly organized group
Target	Unspecified	Specific	Specific, governmental organization, commercial enterprise
Purpose	Financial benefits, demonstrating abilities	To infiltrate the target's network and steal information from their servers	Competitive advantages, strategic benefits
Approach	Single-run, smash and grab, short period	Persistent attack, expending considerable effort	Repeated attempts, state-of-the-art way, stays long term, adapts to resist

However, the common point of RAT, Targeted attack and APT is using command and control communication between the attack side and the victim side.

1.2 Problem Statement of the Research

Remote Access Trojans are different in name and appearance like njRAT and Xtreme, but they have similar functions such as controlling a remote computer and downloading files from it. Nowadays RATs use legitimate network ports on the infected machines, they imitate legitimate commercial remote administration tools, and they perform very surgical operations that common malware techniques do not perform. Attackers use encryptors like exeStealth to obfuscate their RAT. In the case of njRAT, it is detected by 43 of 56 AV manufactures as malicious code. When njRAT is created

with encryptor tool called exeStealth, and the encrypted file is run on Virus Total, just 22 out of 57 AV vendors have detected the file as malicious.

Signature-based malware detections like SNORT and antivirus scanners are not enough to detect Remote Access Trojans when malware authors use encryption and polymorphism to develop them. Signatures have to be created whenever a new RAT appears. Then, signature database is extremely large, and it cannot work properly in high speed network. Signature-based anti-virus scanners work by looking at the files on the computer as sequences of bytes. Malware is detected by searching for specific patterns that are occurred in a given piece of malware. If this type of sequence is detected, the file can be flagged as malicious [4]. This technique has some advantages like fast matching and low false positive rate. However, if the attack pattern has small modifications or a novel attack appears, it cannot be successfully detected. In addition, maintenance and updating of the signature database is a tedious and time-consuming task, and it can also affect system performance.

There is lack of effective features for machine learning methods in behavior-based approaches to detect different types of Remote Access Trojans. In behavioral-based detection, features and machine learning methods are used for detecting RATs. In previous researches, unimportant features are collected and they cannot differentiate malicious traffic from normal traffic. There is False Negative Rat and False Positive Rate that should be taken care of. Moreover, popular machine learning algorithms are applied in this area and just a simple ratio of dataset obtained from their work is used for building a model. As typical network contains approximately 99.99% of normal instances and small number of malicious instances, just a simple ratio of normal and malicious traffic instances is not enough to approach a best detection model with effective features and no overhead.

Generally, features are extracted from a session that starts from SYN packet in TCP three-way handshake to FIN/ACK packet and some are obtained from a session that starts a connection to the end of the traffic. So, time takes long to obtain features and confidential information will be leaked before detection. There is a problem of error recovery features when features are defined based on TCP three-way handshake connection. Some define features depending on packet time interval and they do not consider error-recovery features like TCP retransmission. So correct information for features cannot be obtained in this state, and malicious traffic will be missed.

1.3 Motivation of the Research

In the hyper-connected world, data is always stored in a computer and, individual or organization comes across data leakage or data disclosure through Internet. They will lose their finance, reputation, business and intellectual property because of these problems. Threat actors use different techniques such as Remote Access Trojans, targeted attack and advanced persistent threats to intrude the targeted network and collect valuable data from the victim's computer. Remote Access Trojan is one of the most infected malwares for data breach or data disclosure. It is important to detect command and control communication that RATs use, and such kind of detection helps to catch command and control communication of targeted attacks and advanced persistent threats (APT).

1.4 Objectives of the Research

There have been many detection approaches for detecting Remote Access Trojans such as signature-based detection. However, network behavioral based features are used in this thesis. The objectives of this research are:

(1) To mitigate the infection of Remote Access Trojans as early as possible
(2) To provide effective features for detection model of Remote Access Trojans
(3) To provide the assistance in detecting variants of Remote Access Trojans by network behavioral based approach
(4) To achieve the comparable classification accuracy for malware detection in production environments with effective feature set and no overhead

1.5 Contribution of the Research

It is important to know when the traffic will be cut or stopped in capturing network traffic in order to extract effective features for detecting Remote Access Trojans. In this research, the traffic is cut based on the number of packets. Effective features are extracted from the first twenty packets which is the optimal number of packets for detecting Remote Access Trojans in the early stage of network traffic. It does not depend on packet time interval to define features and so error recovery feature is not a problem.

The malicious network traffic of Remote Access Trojans is differentiated from normal network traffic of normal applications. Classification accuracy is higher than previous methods, and False Negative Rate and False Positive Rate are the lowest.

Ensemble learning methods are also applied for malicious traffic classification and AdaBoost is contributed for detecting RATs and for different ratios of normal and malicious RATs traffic.

1.6 Organization of the Research

This dissertation is organized with eight chapters. Chapter 1 describes introduction, including Remote Access Trojans, problem statement of the research, motivation, and objectives of the research. Contribution is also expressed in this chapter. Background theory of malware analysis, malware detection, host-based and network-based detection systems and network traffic classification are explained in Chapter 2. Literature review for dynamic analysis, packet analysis, TCP data format and relative sequence number, how to define features based on TCP connection and machine learning methods are described in Chapter 3. Feature selection for different ratios of dataset is discussed in Chapter 4. Chapter 5 explains how to extract features in the experiment, tuning parameters and performance evaluation. Chapter 6 presents experimental analysis and evaluation, and summary of different results based on different ratios of dataset. The comparison on the performance of two detection approaches are stated in Chapter 7. Conclusions and future work are explained in Chapter 8.

CHAPTER 2
BACKGROUND THEORY

Nowadays, malwares such as Trojan, worm and bot are spreading speedily through the network. Analyzing network traffic is a must to find such kinds of malware. There are network intrusion detection system and antivirus scanners to catch these malwares. Multi-faceted and diverse malware is capable of subverting even the most well defended computer networks. New invention of malware is increasing every day and the intensive use of networks and the Internet makes malicious software disseminate increasingly and rapidly. Therefore, manufacturers and researchers do research to produce effective anti-malware systems.

Remote Access Trojan is one of the most dangerous malwares and it can cause huge amount of damage depending on how the attacker applies it. As it is a kind of malware, it is necessary to do the analysis like other malwares and, the detection techniques are proposed based on this analysis.

2.1 How Remote Access Trojans Work

RATs usually do not show up in the list of running programs or tasks. Their actions are similar to those of legitimate programs. Moreover, an attacker will manage the level of resource use so that the performance of the system cannot be dropped and a user does not recognize something unusual. How to use a trojan is explained below.

(1) Attacker creates an executable file. This is server part of trojan and mostly called as server.exe.

(2) Attacker binds this server.exe with any genuine file like a song or image. Attacker gives this file to a victim and the victim is supposed to open this file or click on it. Then the server.exe is executed on the victim.

(3) As the server part is run in the victim, a port on the victim's computer gets opened and try to connect to the attacker. In this way, the attacker can control the victim’s computer remotely in any part of the world through the control panel.

There are two different types of remote connection to connect remote system: (1) Direct connection, (2) Reverse connection [76]. In direct connection, after the server part has been installed on a victim's machine, the attacker enters the public IP address of the victim's computer for making a connection to it. However, limitations of direct

connection are that public IP address is most probably dynamic and gets changed every time that disconnects and reconnects. An attacker, therefore, needs to find out IP address of victim each time. Moreover, firewall restricts the incoming connection. The main drawback of direct connection is that we cannot access the victim who is behind a router or a network because victim's machine is not assigned public IP address. It is only assigned private IP address which cannot be used or accessed outside that network. The WAN IP address is assigned to the interface of the router.

In reverse connection, an attacker enters his own IP address in server part while configuring it. When the server part is installed on a victim's machine, it automatically sends connection back to the client part that is attacker. Moreover, firewall in the victim's machine would not restrict to outgoing connections. However, there is a problem in this case. It is attacker's IP that is also dynamic and this can be overcome easily by entering a domain name in server part which always points to his dynamic IP.

The basic functionality of RAT is shown in Figure 3.2, in which, a server part of RAT has already installed on victims. An attacker which is a client part of RAT gives commands and control to victims. An attacker can control many victims at the same time.

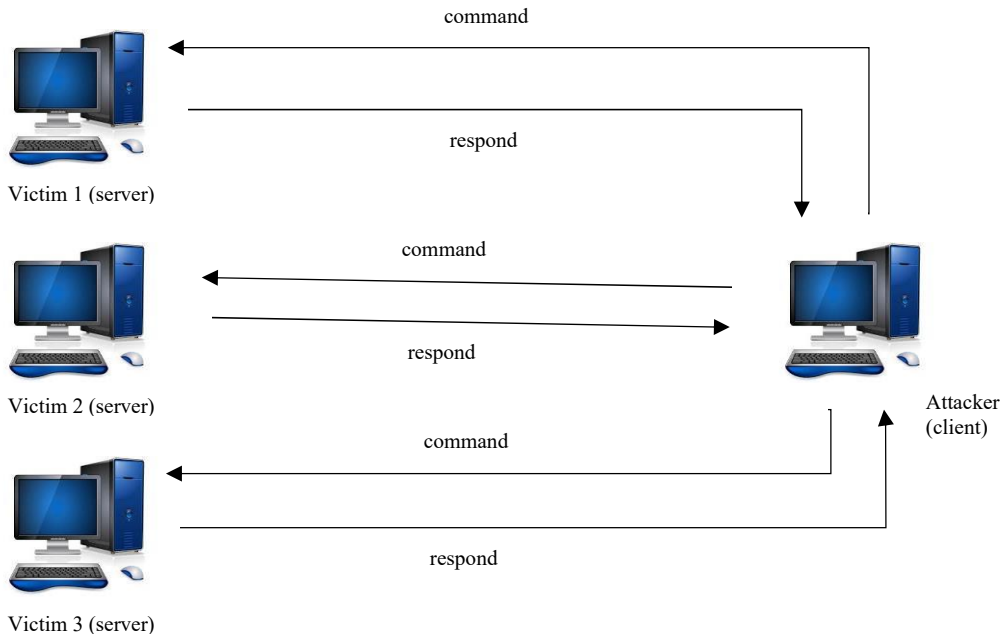

Figure 2.1: Basic functionality of a Remote Access Trojan

2.2 Malware Analysis

Malware analysis is a process that determines the intent and functionality of a malware sample such as a virus, worm, or Trojan horse. It is a necessary step to develop effective detection techniques for malicious code. Moreover, it is an important prior condition for developing removal software that can delete malware from an infected machine. Some of the common questions answered by malware analysis are:

What is the purpose of malware?

What is its target?

What did it steal?

What did it do?

How did it get there?

How long has it been there?

2.2.1 Static Malware Analysis

There are two types of malware analysis: Static analysis and Dynamic analysis. Static analysis is defined as "the discipline of automatically inferring information about computer programs without running them. [17]. Different tools and techniques are applied to determine whether a file is malicious or not. Static analysis provides information about its functionality and collects technical indicators to produce simple signatures. Technical indicators can include file name, MD5 checksums or hashes, file type and file size. Various techniques are used for static malware analysis. Some are explained below [21].

File fingerprinting: Static analysis examines external features of the binary. It computes a cryptographic hash of the binary to prove that it has not been changed.

Extraction of hard coded strings: Software typically prints status- or error-messages as readable text. The internals of the inspected binary can be understood through this text.

File format: The metadata of a file format can give useful information. For example, a lot of information can be extracted from a Windows binary, which is typically in portable executable (PE) format, such as compilation time as well as strings, menus and icons.

Packer detection: Malware is distributed in an encrypted or compressed form using a packer in order to hide from detector.

Disassembly: The binary of an executable file is disassembled in order that an analyst can inspect the program logic and then examine its intention. It is the major part of static analysis. Tools like IDA Pro is utilized to reverse the machine code to assembly language.

Understanding a given binary is the main advantage of static malware analysis. It analyzes all possible execution paths of a malicious sample. Static analysis is generally safer than dynamic analysis because the source code is not actually executed. However, it is extremely time-consuming and tedious.

There are limitations of Static malware analysis. Analyzing binaries cannot get useful results if the binary uses self-modifying code techniques. Malware depending on values such as current system date, indirect jump instructions that cannot be statically determined exacerbates static analysis techniques. Moreover, malware authors may develop malware instances that can avoid or overcome the limitations of static analysis methods.

2.2.2 Dynamic Malware Analysis

Dynamic analysis is performed by actually running the code and monitoring its behavior and its functionality [17]. It helps to detect code obfuscations, polymorphic malwares and unpacking malwares that are some limitations of static analysis, as it examines the program while in execution. The actual behavior of a program can be seen in this way. However, there are some drawbacks. The main one is dormant code: Dynamic analysis usually monitors only one execution path and it cannot cover complete code. If the analysis environment is not properly isolated or restricted, the third-party system will be harmful. Moreover, if malware realizes that they are being executed within a controlled analysis environment, they may alter their behavior or stop executing. There are two basic approaches for dynamic analysis [21]:

Analyzing the difference between defined points: A malware sample is executed for a certain period of time. Then the modifications made to the system are analyzed by comparison to the initial system state.

Observing runtime-behavior: In this approach, malicious activities launched by the malicious code are monitored during runtime using a specialized tool.

2.3 Malware Detection

Based on the use of detection technique, malware detection methods are fundamentally classified in three categories: Signature-based methods, behavior-based methods, and heuristic methods [5]. Concealment Strategies used by malware author are also explained.

2.3.1 Signature-based Detection

Signatures are unique patterns that are associated with each malware characteristics. These methods are usually built based on static analysis. Signatures are extracted from known attacks of various malware and a database is created with these signatures. Since signatures are created from known attacks of malware, it is very efficient and accurate to detect known malware with low false positive rate. An example of signature-based detection is antivirus scanner.

However, these methods cannot detect unknown malware and variants of known malware. A certain amount of time and money are required to extract signatures from known attacks. Users need to update signature database daily. These are the main drawbacks of signature-based methods. Moreover, they are unable to confront against polymorphic and metamorphic malwares that change their codes in each infection. To deal with these problems, researchers have proposed new malware detection techniques.

2.3.2 Behavior-based Detection

Behavior-based malware detection observes the behavior of a program to determine whether it is malicious or not [33]. Since behavior-based techniques observe what a program does, they are not vulnerable to the drawbacks of signature-based methods. In these methods, the behavior of various malwares is collected, and some behavior of different malwares are similar. So, a single behavior can uncover various malwares. This type of detection technique helps to detect new variants of malware because they do similar manner in the system by using same resources. A behavior-based detection consists of three parts [28]:

(1) Data Collection: It collects information from executable file.

(2) Interpretation: It converts raw data obtained from data collection into intermediate representations.

(3) Matching: The representation is compared with the behavior signatures.

Symantec patented the histogram based malicious code detection technology which is a behavior-based detection approach [33]. The advantage of these detection techniques is that it is able to detect unknown and polymorphic malwares that signature-based approaches cannot detect. However, high False Positive Rate and also long scanning time are drawbacks of behavior-based methods [2].

2.3.3 Concealment Strategies

Since malware detection software is emerging and malware developers understand that their malware will be detected, they try to avoid anti malware software by using concealment techniques. Some of the most well-known concealment strategies are introduced below [5]:

(1) Obfuscation: It includes adding garbage commands, unnecessary jumps etc.

(2) Code encryption: Malwares camouflage their malicious activity by encrypting themselves. Encrypted malware consists of a decryption algorithm, encryption algorithm, encryption keys and encrypted malicious code. When the malware runs, its malicious part is decrypted by the key and decryption algorithm. The malware copies itself and generates a new key. A new encrypted version will be produced using new generated key and encryption algorithm.

(3) Oligomorphic strategy: Such kind of malwares use encryption to protect themselves and their encryption algorithm is changed within limited time.

(4) Polymorphic strategy: Malware encrypts itself by an encryption algorithm. A different decryption key is used in each infection. Moreover, polymorphic malware uses an unlimited number of encryption algorithms to avoid detection. A part of decryption code will alter in each execution.

(5) Metamorphic strategy: It is the most complex type of malware strategy. This malware changes themselves so that the new instance is not resemble to the original one.

As malware authors are trying to introduce new strategies, traditional signature based and behavioral methods cannot uncover malware in some cases. So, a new effective method to detect malware is absolutely required.

2.3.4 Heuristic Methods

Heuristic malware detection methods overcome the disadvantages of signature-based and behavior-based detection methods. Heuristic malware detection methods use data mining and machine learning methods to learn the behavior of an executable file. There are some features investigated for malware detection which will be described in the followings [5].

2.3.4.1 Application Programming Interface (API)/System calls

Programs use API calls to send their requests to the Operating System [53] and API call sequences reflect the behavior of a piece of code like malware. Hofmeyr et al. applied API call sequences as a feature of a malware [27]. Afterward, a research was made for using API calls [6] [65] [71].

Associative classification which is post processing techniques is used in [82]. They applied Chi-squared testing in [68]. System call sequences is used for both malicious and benign executables [30] and they are used to create a topological graph called code graph. However, this graph is too large. To overcome this drawback, API calls are classified to 128 groups [83], so the code graph reduced. A classifier based on the analysis of API calls by a PE file is proposed for detecting malware from large and imbalanced gray list [83].

2.3.4.2 Operational Code (OpCode)

An OpCode which is short for Operational Code is the subdivision of a machine language instruction and it identifies the operation to be executed. A program is defined as a series of ordered assembly instructions. An instruction is a pair composed of an operational code and an operand or a list of operands.

OpCodes is used as feature for detecting malware by Bilar [7]. OpCode sequence features are presented by Santos et al. The frequency of OpCode sequences focuses on detecting the variants of obfuscated malware [63]. The relevance of each OpCode is computed using Mutual Information [55]. At last, Weighted Term Frequency (WTF) [36] is used to extract features from executables.

OpCode sequence features are trained with machine learning classifiers [64]. Large number of samples are required for each class in machine learning based classifiers.

2.3.4.3 N-Grams

N-Grams are all substrings of a larger string with a length of N [1]. For example, the string “VIRUS”, can be divided into 3-grams: “VIR”, “IRU”, “RUS”.

N-Grams is used as a feature for malware detection by Tesauro et al. [23]. It is used to detect Boot Sector Viruses using Artificial Neural Networks (ANN). A Boot Sector Virus is a variant of malware that infects DOS Boot Sector or Master Boot Record (MBR). When a system has been infected, the MBR is destroyed and the computer boot order is changed.

Byte N-Gram representation is applied to detect unknown malware Kotler and Maloof [87]. In their previous study [86], malware is classified into separate families based on the functions of respective payload.

2.3.4.4 Control Flow Graph

Control Flow Graph (CFG) is a graph that represents the control flow of programs and it has been researched for many years [29], [45], [72]. CFG is a directed graph, where each node represents a statement of the program and each edge represents control flow between the statements. Statements may be assignments, copy statements, branches, etc.

Normalization operations are performed after disassembling an executable program in order to reduce effects of mutation techniques and unveil the flow connections between benign and malicious code [12].

2.3.4.5 Hybrid Features

Features and algorithms influence the performance of machine learning classifiers. Researchers use effective features to improve the precision of machine learning classifiers. CFG based detection methods should be improved for detecting polymorphic or metamorphic malware.

So, CFG and API calls are used to detect metamorphic malware by Eskandari et al. [19]. New features and a novel classifier are proposed by Lu et al. [42]. Imported DLLs and API function calls of a PE file was used as the content-based features. The concept of information gain is used for reducing these features because of the huge number of extracted features. For behavioral-based features, each file is executed in a virtual machine and its behavior is expressed as a feature.

2.3.4.6 Some Newly Introduced Features

File content and file relations are used as features [84]. File relations are presence or absence of some files in the computer of all cloud users. File relations among samples are useful feature to detect malware. A semi parametric classification model is proposed to combine file content and file relation. For file content, API calls are extracted from the content of malicious and benign PE files. File content and files relation information are integrated by their semi parametric model and the classification problems are formulated using the graph regularization framework. A new approach is proposed to detect polymorphic script viruses [36]. The VBScript malware is represented using a dependency graph.

2.4 Host-based vs Network-based Detection System

Based on how to collect data, malware intrusion detection systems can be distinguished into two major classes: the host-based system and the network-based system.

2.4.1 Host-based Intrusion Detection System

A Host Based Intrusion Detection System (HIDS) collects information about the activities of a particular single system, or host. The term "host" refers to an individual computer or server. The host-based agents sometimes referred to as sensors, would be installed on a suspected machine that possible attacks can be occurred. HIDS works constantly by monitoring the access to the system and its application and sending alerts for any unusual activities. It monitors system logs, event logs, application logs, user policy enforcement, rootkit detection, file integrity, and other intrusions to the system. A baseline is created based on these logs. If any new activity appears, HIDS checks this activity against the baseline and if any log is found outside of this baseline, HIDS gives an alert. If any unauthorized activity is caught, HIDS will alert the user or block the activity or perform other decision based on the policy configured by system administrator [26].

Limitations of HIDS:

(1) The host-based detection system should be installed in each host and it occupies more resources than network-based system in which a host is enough to be installed in a computer network.

(2) It also relies on the operating system platform, and it requires prior knowledge of the platform.

2.4.2 Network-based Intrusion Detection System

A Network-based Intrusion Detection System (NIDS) monitors both ongoing and outgoing network traffic that is connected to an organization's network, looking for malicious activities. NIDS is installed at a particular place in the network where the ongoing and outgoing traffic are passing through the network segment. It examines every packet passing through network and matches the data against its signature database If there are any new entries against its database, it sends alarms to the management console. An example of NIDS is SNORT.

Limitations of NIDS:

(1) The detection mechanism could only detect the known Trojan and it is unable to uncover novel samples.

(2) It cannot catch up even simple encryption techniques and misses malicious traffic.

(3) Since the signatures are increasing day by day, the approach becomes computationally more expensive, and it effects the performance of NIDS.

2.5 Network Traffic Classification

Nowadays classifying network traffic is an important topic in the field of network security. It is very essential for network administrators and Internet Service Providers (ISPs) to manage the performance of a network and to enhance network security such as traffic shaping/policing, diagnostic monitoring and capacity planning, and route provisioning. It is also advantageous to find threats that pass network. There are three network traffic classification techniques: Port-based Technique, Payload-based Technique and Machine learning techniques as shown in Figure 2.1.

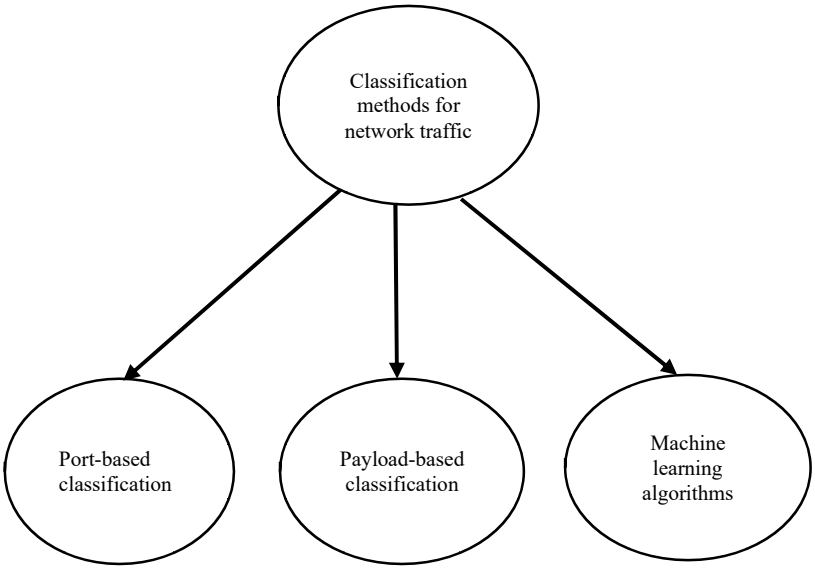

Figure 2.2: Three types of network traffic classification

2.5.1 Port-based Network Traffic Classification

In TCP/IP and UDP networks, a port is an endpoint to a logical connection and the way a client program specifies a specific server program on a computer in a network. Some port numbers are pre-assigned by the IANA, and these are called the "well-known ports" which are specified in RFC 1700 [41] [79].

Port numbers range from 0 to 65536, but only ports numbers 0 to 1024 are reserved for privileged services and designated as well-known ports. The range of port numbers from 1024 to 49151 are the registered ports. They are assigned by IANA for specific service upon application by a requesting entity.

The port numbers from 49152 to 65535 are dynamic or private port numbers that that are available for use by any application to use in communicating with any other application, using the Internet's Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP) [66].

Port-based traffic classification is simple and fast, and it is based on a well-known port number with a given traffic type such as web traffic that uses TCP port 80. Port-based technique fails when dynamic port number is assigned for applications. Dynamic port number is unregistered number with Internet Assign Number Authority (IANA). Peer to Peer applications (P2P) that use dynamic port numbers are rapidly increased. IANA assigned ports are shown in Table 2.1 and registered ports are shown in Table 2.2 [41].

Table 2.1: IANA assigned ports

Well-known ports (TCP)	Application (Description)
20	File Transfer Protocol (FTP) for data transfer
21	File Transfer Protocol (FTP) for control
22	Secure Shell (SSH), secure logins, file transfers (scp, sftp) and port forwarding
23	Telnet protocol—unencrypted communications
25	Simple Mail Transfer Protocol (SMTP) for email routing between mail servers
53	Domain Name System (DNS)
80	Hypertext Transfer Protocol (HTTP)
110	Post Office Protocol, version 3 (POP3)
123	Network Time Protocol (NTP) for time synchronization
161	Simple Network Management Protocol (SNMP)
443	Hypertext Transfer Protocol over TLS/SSL (HTTPS)

Table: 2.2 Registered ports

Registered ports (TCP)	Application (Description)
1701	Layer 2 Forwarding Protocol (L2F) Layer 2 Tunneling Protocol (L2TP)
1900	Simple Service Discovery Protocol (SSDP), discovery of UPnP devices
1985	Cisco Hot Standby Router Protocol (HSRP)
3306	MySQL database system
4500	IPSec NAT Traversal
5353	Multicast DNS (mDNS)
5432	PostgreSQL database system
6660-6664	Internet Relay Chat (IRC)
9389	adws, Microsoft AD DS Web Services, Powershell uses this port

2.5.2 Payload-based Network Traffic Classification

Payload-based technique is also called deep packet inspection (DPI). Generally, DPI is applied in network-based approach and it is a kind of signature-based detection. Each application is studied and its payload is analyzed for its characteristics and signatures are created. It performs signature analysis to understand and verify different applications. The classification engine compares the network traffic against the signatures to identify the exact applications. There are different signature analysis methods. Pattern analysis, behavioral analysis, heuristic analysis and protocol analysis are popular methods. The signatures need to be updated periodically to keep up with new applications and new versions developments in existing ones. Signatures cannot be defined properly from encrypted network traffic. Network traffic that uses of encryption has been rapidly increasing, and malware authors have taken advantage of this trend to evade signature-based detection.

An example of deep packet inspection is SNORT. A part of shell code of Poison Ivy Remote Access Tool that is included as the payload in an exploit is as follows [3]:

```
# Generated by Poison Ivy 2.3.2
# http://www.poisonivy-rat.com
# Length: 0x000012D3 (bytes)
PIshellcode = \
'\x55\x8B\xEC\x81\xC4\x30\xF0\xFF\xFF\x60\x33\xC0\x8D\xBD\x84\xF0' + \
'\xFF\xFF\xB9\x74\x0F\x00\x00\xF3\xAA\x33\xC0\x8D\xBD\x40\xF0\xFF' + \
'\xFF\xB9\x44\x00\x00\x00\xF3\xAA\xC7\x85\xAD\xF1\xFF\xFF\xE7\x00' + \
'\x00\x00\xE9\x6E\x0D\x00\x00\x55\x8B\xEC\x81\xC4\x30\xFA\xFF\xFF' + \
```

In this shell code example, there are 4,817 bytes of shell code and only 6 byes of the shell code are unique to the IP address and port for the callback.

Snort rule to detect this malicious command and control traffic is written as follows:

alert tcp any any -> $HOME_NET any (msg: "Poison IVY shellcode"; content: "|55 8B EC 8F C4 30 F0 FF FF 60 33 C0 8D BD 84 F0|"; sid: 10000001; rev: 1;)

2.5.3 Machine Learning Algorithms

Machine learning is an intelligence technique which analyzes from variety of data and provide useful information. There are three types of machine learning: (1) Supervised Learning, (2) Unsupervised Learning, and (3) Semi-supervised Learning [11]. Supervised Learning is the process of an algorithm learning from the training dataset that consists of training data and their true labels.

Supervised learning can be further grouped into two problems: regression and classification**.** Regression is the prediction of a numeric value such as weight or dollar. Classification: Classification is to what category or class an instance of data fall into such as positive or negative and red or blue. The more the data, the better the model will perform on unseen examples. Some examples of supervised machine learning algorithms are: Linear Regression for regression problems; Random Forests for classification and regression problems; and Support Vector Machines for classification problems.

Unsupervised learning is the opposite of supervised learning. In unsupervised learning, there is no label or target value given for the data. Unsupervised learning can be further grouped into clustering, density estimation and dimensionality reduction. **Clustering**: A task of grouping similar items together is known as clustering. **Density Estimation**: It is to find statistical values that describe the data. **Dimensionality Reduction:** It is to reduce the data from many features to a small number so that it can be properly visualized in two or three dimensions. Some examples of unsupervised learning algorithms are K-Means for clustering, Kernel Density Estimation, Normal or mixture models for Density estimation. Principal component analysis for dimensionality reduction.

In semi-supervised learning, there are a large amount of input data and, only some of the data is labeled and the majority are not labelled. It is applying for large amount of data because it can be expensive or time-consuming to label data. As an example, learning medical images like CT scans or MRIs. A radiologist can label a small subset of scans for tumors or diseases. It is too time-intensive and costly to manually label all the scans. The semi-supervised learning can use the small proportion of labeled data and improve its accuracy compared to an unsupervised model.

2.5.4 Feature Selection Techniques

Feature selection is the process of selecting a subset of relevant features for use in model construction [10]. The selected features are used by classification algorithms. The feature selection result is advantageous to speed up the detection time and to improve the detection accuracy. The maximum overall performance can be achieved.

Feature selection methods fall into three groups: Filter methods, Wrapper methods and Embedded methods [10]. Filter methods assign a scoring to each feature by applying a statistical measure. The features are ranked by the score and either selected to be kept or removed from the dataset. Examples of filter methods are Chi squared test, information gain and correlation coefficient scores. Wrapper methods select a set of features in which different combinations are prepared, evaluated and compared to other combinations. Example of Wrapper method is the recursive feature elimination algorithm. Embedded methods learn which features best contribute to the accuracy of the model while the model is being created. An example of embedded feature selection method is regularization method. Examples of regularization algorithms are the LASSO, Elastic Net and Ridge Regression.

The advantages of Filter methods are (1) scale easily to very high-dimensional datasets, (2) computationally simple and fast, (3) independent of the classification algorithm, (4) Feature selection needs to be performed only once, and then different classifiers can be evaluated. The disadvantage of Filter methods is that they ignore the interaction with the classifier.

The advantages of Wrapper methods are interaction between feature subset search and model selection, and the ability to take into account feature dependencies. The disadvantages of Wrapper methods are higher risk of overfitting than filter techniques and they are very computationally intensive, especially if building the classifier has a high computational cost.

The advantage of Embedded methods is less computationally intensive than wrapper methods. The disadvantage of Embedded methods is specific to a learning machine.

2.6 Summary

In this chapter, background theory of malware analysis, malware detection techniques and network traffic classification are described in detail. Researchers do malware analysis to introduce malware detection mechanism and to keep detection

method effective as always since new variants of malware are emerging every day. Both static and dynamic ways or one of the two ways are applied in the research. Signature-based, behavior-based and heuristic detection methods have their pros and cons. Researchers enhance these detection methods to catch up with new malware and the variants of existing malware. Malware authors are always trying to evade those detection methods, too.

CHAPTER 3
LITERATURE REVIEW

There are two techniques for Trojan detection: host-based detection and network-based detection. Host-based detection has focused on Trojan's activities inside an end host. It monitors legitimate application API calls [37]. However, the order of calls can be changed or irrelevant calls can be added. Host-based system occupy more resources as it needs to be installed in each host. It also depends on the operating system platform, and it requires prior knowledge of the platform [15], [78]. Thus, the usefulness of host-based detection is limited by its complexity and much overhead.

There are two different approaches in network-based malware detection system: (1) signature-based detection that uses deep packet inspection (DPI), (2) machine learning methods that use network behavior features. Two approaches are shown in Figure 3.1. The signature-based detection system utilizes the Deep Packet Inspection (DPI) techniques and it extracts the application-level signature to detect the Trojan. As it is a network-based system, one is advantageous to many machines and the end hosts do not have much overhead. This kind of detection could gain a very high accuracy. However, the detection system only detects known Trojans as the signatures are obtained from known malware. Moreover, it cannot uncover malware that utilizes encryption techniques. The signature database needs to be updated daily. As the database is big in size with the increase of signatures, the performance may be reduced in high speed network and this approach is computationally expensive.

Another way of network-based detection is machine learning methods. Network behavioral analysis is done to define features for trojan detection. It extracts network behavioral features from network traces which are obtained from dynamic analysis. Trojans are run in a controlled area and captured traces. Network behavioral features outperform trojan detection methods. Machine learning algorithms have been used for network traffic classification for several years. Supervised, unsupervised and hybrid methods are applied for classifying network traffic and the popular methods are Decision Trees (DT), Random Forests (RF), Naïve Bayes (NB) and Support Vector Machine (SVM). Features are important to be effective for a detection model built by machine learning algorithms.

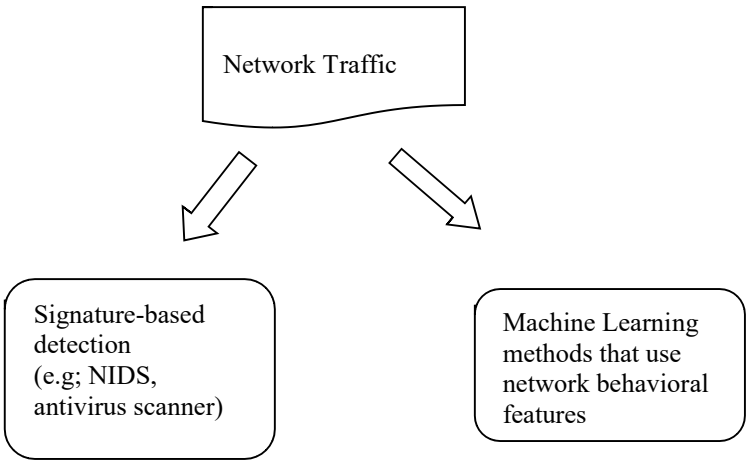

Figure 3.1: Two approaches of network-based detection

3.1 Dynamic Analysis for Behavior of a Trojan

Both host-based and network-based techniques are applied for detecting Remote Access Trojans. A dynamic analysis is performed by checking API calls to figure out whether a certain file is malicious or not. Some use sandbox to analyze the behavior of Trojans. Dynamic analysis is a must to do for analyzing network behavior of Remote Access Trojans. Based on the analysis, signatures are created for antivirus scanners or intrusion detection tools.

Real-world Trojan samples are collected from the Internet and execute them in a controlled environment to observe their network activities. As Remote Access Trojan has two sides, the server-side is executed on the victim, and the client-side is executed on the attacker to control the victim. There is command and control communication between the attacker and the victim. The attacker can download, upload files to the server. Network monitoring tools are used to capture network traffic and, the captured data is analyzed for differing malicious data from normal one.

3.2 Packet Analysis and Packet Sniffers

Packet analysis is referred to as packet sniffing or protocol analysis. It describes the process of capturing network traffic data as it flows across a network in order to better understand what is happening on that network. Packet analysis is typically performed by a packet sniffer which is a tool used to capture raw network data going across the wire. Packet analysis can help with the following [62]:

(1) Understanding network characteristics

(2) Learning who is on a network
(3) Determining who or what is utilizing available bandwidth
(4) Identifying peak network usage times
(5) Identifying possible attacks or malicious activity
(6) Finding unsecured and bloated applications

3.2.1 How Packet Sniffers Work

The packet-sniffing process involves a cooperative effort between software and hardware. This process can be divided into three steps [62]:

(1) **Collection**: In the first step, the packet sniffer collects raw binary data from the wire. To do so, the selected network interface is switched into promiscuous mode. In this mode, the network card can listen to all traffic passing on a network segment, not only the traffic that is addressed to it.
(2) **Conversion**: In this step, the captured binary data is converted into a readable form.
(3) **Analysis:** The actual analysis of the captured and converted data is done in this step. It begins to analyze the captured network data, verifies its protocol based on the information extracted, and begins its analysis of that protocol's specific features.

3.2.2 Network Capturing Tool

Tcpdump or Wireshark is utilized to capture network traffic. Tcpdump is a packet analyzer that runs under the command line. It extracts packets and displays them in real-time or logs for later analysis. Wireshark is the world's foremost and widely-used network protocol analyzer. It is used for network troubleshooting. Wireshark has graphical user interfaces (GUIs).

3.3 How Computers Communicate

Modern networks are made up of a variety of systems running on many different platforms. A set of common languages called protocols are used to aid this communication. Common protocols include Transmission Control Protocol (TCP), Internet Protocol (IP), Address Resolution Protocol (ARP), and Dynamic Host Configuration Protocol (DHCP) [61]. A protocol stack is a logical grouping of protocols that work together.

Protocols are separated according to their functions based on the industry-standard OSI reference model. The OSI model divides the network communications process into seven distinct layers. Table 1-1 lists some of the more common protocols used at each individual layer of the OSI model.

Table 3.1: Typical protocols used in each layer of the OSI model

Layer	**Protocol**
Application	HTTP, SMTP, FTP, Telnet
Presentation	ASCII, MPEG, JPEG, MIDI
Session	NetBIOS, SAP, SDP, NWLink
Transport	TCP, UDP, SPX
Network	IP, IPX
Data link	Ethernet, Token Ring, FDDI, AppleTalk

3.3.1 Connection-oriented and Connectionless Protocol

There are two alternative protocol archetypes to provide the appropriate level of quality assurance for any given situation [13]:

A connection-oriented protocol establishes and maintains a connection between communicating computers and monitors the state of that connection over the course of the transmission. In other words, each package of data sent across the network receives an acknowledgment, and the sending machine records status information to ensure that each package is received without errors, retransmitting the data if necessary. At the end of the transmission, the sending and receiving computers gracefully close the connection.

A connectionless protocol sends a one-way datagram to the destination and doesn't worry about officially notifying the destination machine that data is on the way. The destination machine receives the data and does not worry about returning status information to the source computer.

Transmission Control Protocol (TCP): A reliable, connection-oriented protocol of the Transport layer. Connection-oriented protocols provide more sophisticated flow control and error control than connectionless protocols. TCP goes to great effort to guarantee the delivery of the data. But TCP is slower than UDP. User Datagram

Protocol (UDP): An unreliable, connectionless protocol of the Transport layer. UDP offloads more of the error control responsibilities to the application.

3.3.2 TCP Data Format

The TCP data format and its functionality are shown in Figure 3.2. The complexity of this structure reveals the complexity of TCP.

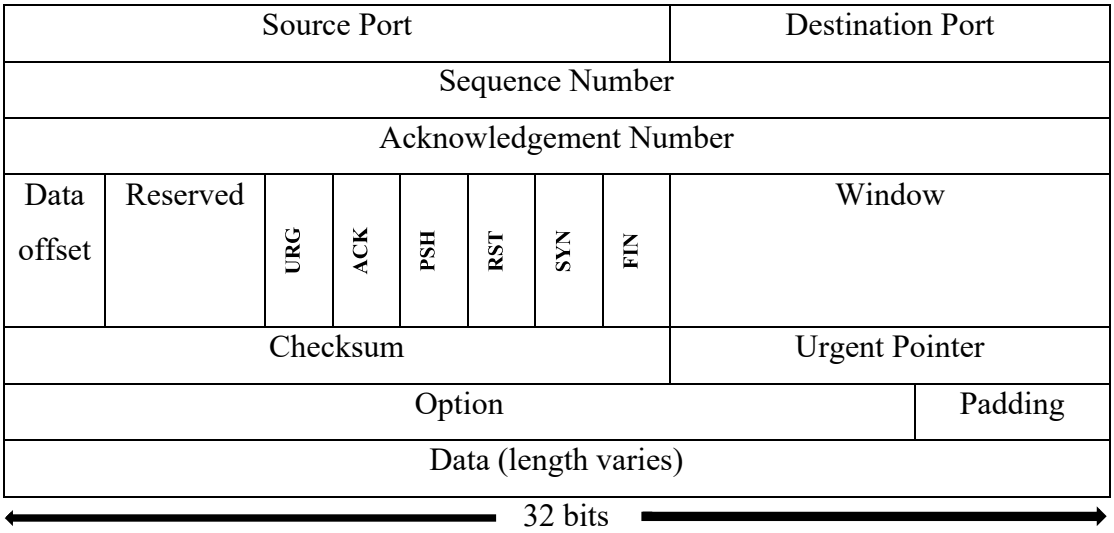

Figure 3.2: TCP data format

The fields are explained below. TCP manages all these data fields to work successfully, acknowledge, and verify network transmissions [13].

- **Source Port (16-bit):** The source port number is the port assigned to the application on the source machine.
- **Destination Port (16-bit):** The destination port number is the port assigned to the application on the destination machine.
- **Sequence Number (32-bit):** The sequence number of the first byte in this particular segment, unless the SYN flag is set to 1. If the SYN flag is set to 1, the Sequence Number field provides the initial sequence number (ISN), which is used to synchronize sequence numbers. If the SYN flag is set to 1, the sequence number of the first octet is one greater than the number that appears in this field (in other words, ISN + 1).
- **Acknowledgment Number (32-bit):** The acknowledgment number acknowledges a received segment. The value is the next sequence number the

receiving computer is expecting to receive, in other words, the sequence number of the last byte received + 1.

- **Data offset (4 bits):** A field that tells the receiving TCP software how long the header is and, therefore, where the data begins. The data offset is expressed as an integer number of 32-bit words.
- **Reserved (6 bits):** Reserved for future use. The Reserved field provides room to accommodate future developments of TCP and must be all 0s.
- **Control flags (1 bit each):** The control flags communicate special information about the segment.
- **URG:** A value of 1 announces that the segment is urgent and the Urgent Pointer field is significant.
- **ACK:** An ACK value of 1 announces that the Acknowledgment Number field is significant.
- **PSH:** A value of 1 tells the TCP software to push all the data sent so far through the pipeline to the receiving application.
- **RST:** A value of 1 resets the connection.
- **SYN:** A SYN value of 1 announces that sequence numbers will be synchronized, marking the beginning of a connection.
- **FIN:** A value of 1 signifies that the sending computer has no more data to transmit. This flag is used to close a connection.
- **Window (16-bit):** A parameter used for flow control. The window defines the range of sequence numbers beyond the last acknowledged sequence number that the sending machine is free to transmit without further acknowledgment.
- **Checksum (16-bit):** A field used to check the integrity of the segment. A receiving computer performs a checksum calculation based on the segment and compares the value to the value stored in this field. TCP and UDP include a pseudo-header with IP addressing information in the checksum calculation. See the discussion of the UDP pseudo-header later in this hour.
- **Urgent Pointer (16-bit):** An offset pointer pointing to the sequence number that marks the beginning of any urgent information.
- **Options:** Specifies one of a small set of optional settings.
- **Padding:** Extra 0 or more bits (as needed) to ensure that the data begins on a 32-bit boundary.
- **Data:** The data being transmitted with the segment.

3.3.3 Relative Sequence Number

By default, Wireshark keeps track of all TCP sessions and convert all Sequence Numbers (SEQ numbers) and Acknowledge Numbers (ACK Numbers) into relative numbers [45]. This means that instead of displaying the real/absolute SEQ and ACK numbers in the display, Wireshark displays a SEQ and ACK number relative to the first seen segment for that conversation. All SEQ and ACK numbers always start at 0 for the first packet seen in each conversation.

This makes the numbers much smaller and easier to read and compare than the real numbers which normally are initialized to randomly selected numbers in the range 0 - (2^32)-1 during the SYN phase. An example of TCP flow captured by Wireshark and how TCP three-way handshake works to transfer data are shown in Figure 3.3. According to this figure, assumes that an attacker uses IP address – 192.168.0.178, and a victim uses IP address – 192.168.0.179. The inbound data byte of victim is 96 and the outbound data byte is 252.

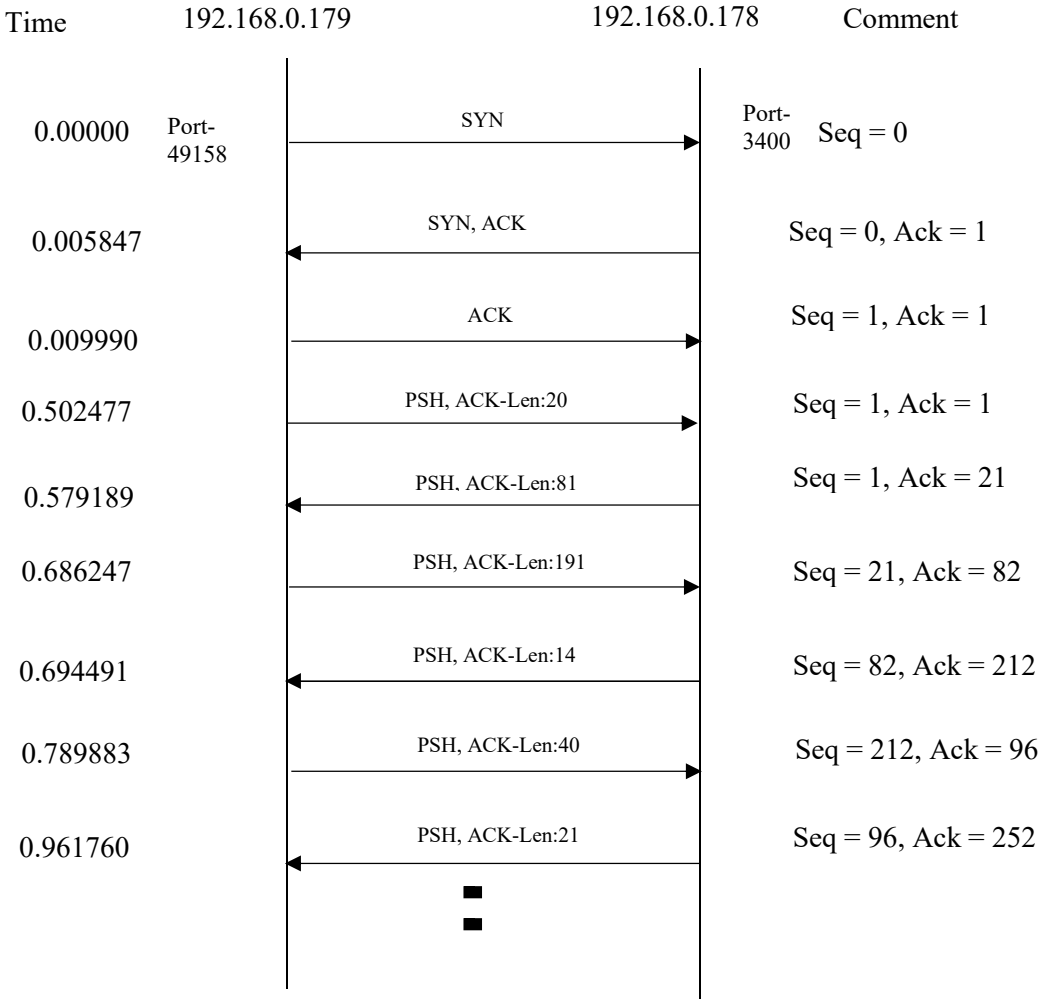

Figure 3.3: TCP flow captured by Wireshark

3.4 Defining Features Based on TCP Connection

Real-world Trojan and normal network traces are collected from the controlled environment. As almost all Trojans use TCP to communicate with victim, features are selected based on TCP based communication session. Effective features which are crucial to the detection are defined, selected and calculated. Although there are many communication sessions features to be selected, not all features provide good discrimination between the Trojan and a legitimate application. Using such features may decrease the efficiency and accuracy of the detection.

Generally, features are extracted from the start of the connection to the end of it which means that it is from a SYN packet to a FIN/RST packet in TCP based connection. Most of previous works use it to extract various features. Moreover, there is a recent approach in which features are extracted in the early stage that depends on packet interval time [31].

How to define early stage and how to define long interaction from SYN to FIN/RST for extracting features are shown in Figure 3.4. The difference of the two approaches can be seen clearly in this Figure. TCP three-way handshake is the very first step of TCP based connection. Firstly, client sends synchronization to an attacker. The attacker sends back both synchronization and acknowledgement to the client. Then, the victim sends acknowledgement to the attacker. These three steps are called TCP three-way handshake. After this handshaking, data transfer starts between two hosts. In this Figure, the short arrow shows early stage detection that depends on packet interval time, and the long arrow shows connection that starts from SYN packet to FIN/RST packet.

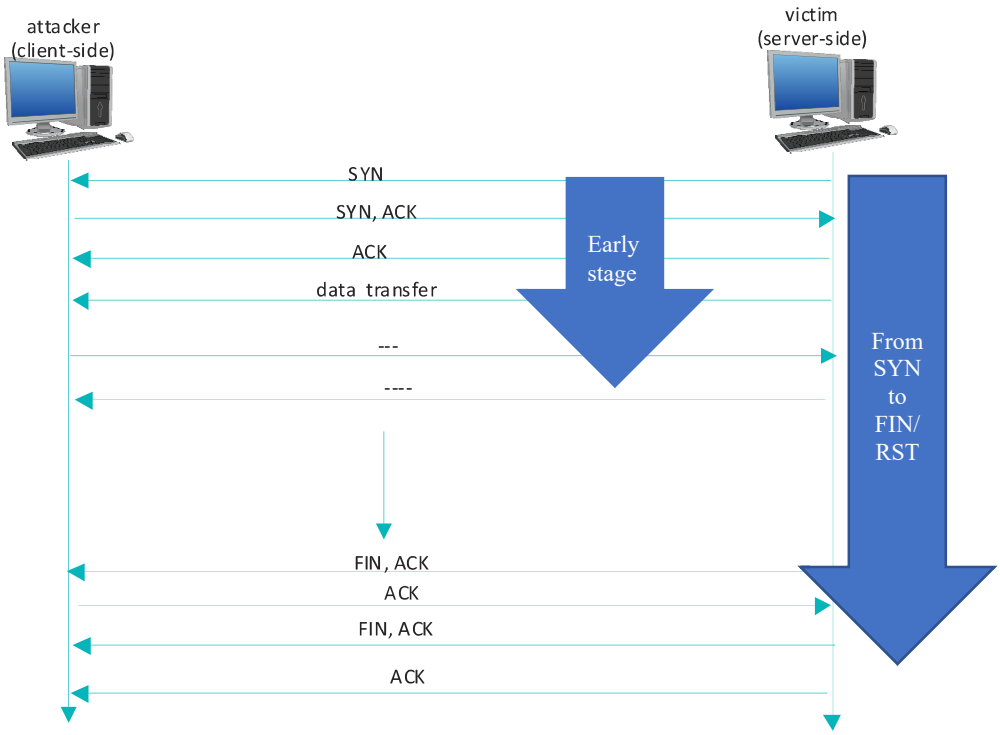

Figure 3.4: An overview of TCP based connection

Effective approaches proposed by researchers are explained below.

3.4.1 Feature Extraction from a SYN Packet to a FIN/RST Packet

The features are extracted manually in this research [39]. 46 real-world Trojan samples are analyzed in a controlled environment to select the features. The selected features for this type of feature extraction are:

(1) Duration: time from the very first SYN packet to a FIN/RST packet

(2) Packet time interval: time between each packet

(3) Number of inbound/outbound packets: the number of packets for inbound/outbound connection

(4) Number of inbound/outbound bytes: the data bytes for inbound/outbound connection

(5) Duration of the communication session and the number of transport layer connections: time from the start of connection to the end of connection and the number of TCP connection

(6) Out-in-pkts: If the total number of packets from Server-side to Client-side is larger than the opposite direction, the value equals 1, otherwise the value is 0.

(7) Out-in-bytes: If the total number of bytes from Server-side to Client-side is larger than the opposite direction, the value equals 1, otherwise the value is 0.

(8) Duration-versus: The duration of the IP-pair communication session versus the duration of Main-connection.

(9) Mean interval: the average of the packet interval time of Main-connection.

Another work uses the software network communication characteristics as features. Features are shown below [40]:

(1) ratio of sent and received traffic size

(2) number of connections

(3) proportion of upload connection

(4) proportion of concurrent connection

(5) number of distinct IP

3.4.2 Hierarchical Detection Model

The network behavior features of Remote Access Trojans are mainly distributed in the network layer, transport layer and application layer depending on the features of each stage of the Trojan communication. Features are extracted from network traffic at three layers which were the features of network layer based on IP protocol, transport layer based on TCP protocol and application layer based on HTTP protocol [32].

Network layer features e.g., many sub-connections during primary connection, standard deviation of packet interval.

Transport Layer features, e.g., communications time, "heartbeat" packet to keep-alive, upload and download traffic.

Application Layer features e.g., packet entropy, specific port in communication.

3.4.3 Closed-loop Feedback Model

Generally, there are some False Positive results for the detection of any machine learning models. A closed-loop feedback model that has a data mining module to remove some false alarms from the detected results is designed. It is shown in Figure 3.5. Finally, the False Positive results are fed back as the input for the training sets of machine learning model. Then, it can modify training model and reduce False Positive rate [32].

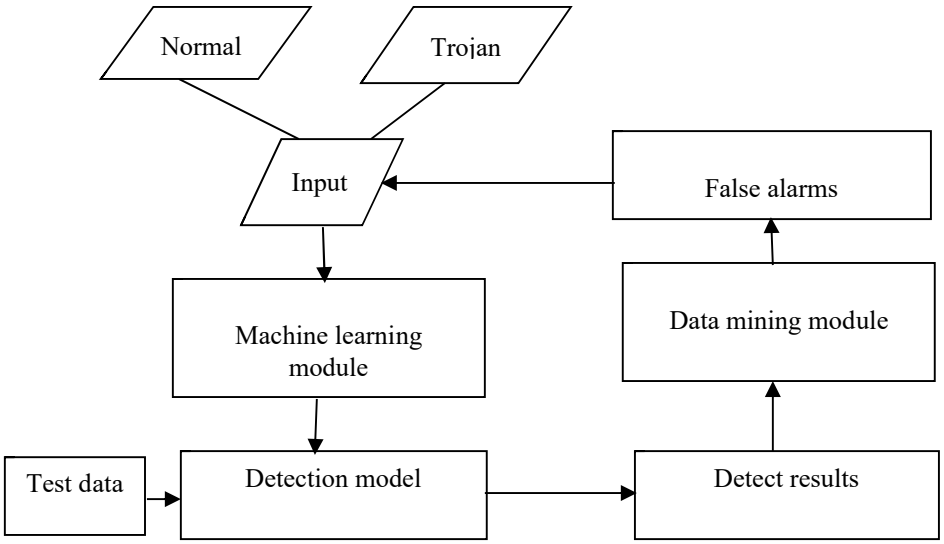

Figure 3.5: Closed-loop feedback model [32]

3.4.4 Feature Extraction in the Early Stage that Depends on Packet Interval Time

In this type of feature extraction, an early stage of the communication is defined and network features are collected from the packets of this period to detect RAT sessions by machine learning techniques. The Early Stage of a session is a packet list which satisfies conditions as follows [31]:

(1) Begins from the SYN packet of TCP 3-way handshake.

(2) Each packet interval time is less than the threshold t second(s).

Two goals are accomplished by collecting network behavior features in the early stage: detecting RATs as early as possible and dividing RAT sessions and legitimate sessions more clearly. It emphasizes on the duration of RAT's early stages that is usually shorter than that of legitimate sessions. RATs usually confine their command and control communications in order to hide themselves inconspicuously. Normal applications do not need to hide their network behaviors and they have far more traffic than RAT sessions. Early stage of RAT and normal applications traffic is shown in Figure 3.6.

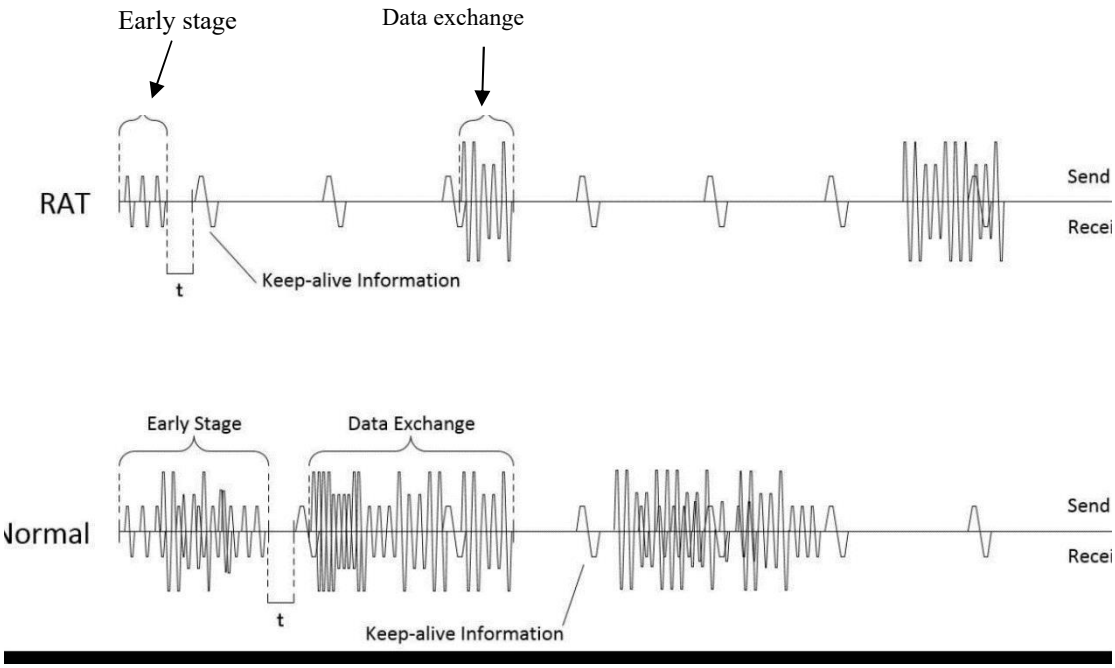

Figure 3.6: Image of the data size in the network traffic of the early stage [31]

Features used in this type of extraction are explained below.

(1) PacNum: number of packets in the early stage

(2) OutByte: Outbound data size

(3) InByte: Inbound data size

(4) OutPac: Outbound number of packets

(5) InPac: Inbound number of packets

(6) O/Ipac: rate of OutPac/InPac

(7) OB/OP: rate of OutByte/OutPac

The accuracy, FNR and FPR of this approach is better than the previous works of feature extraction that mentioned above. The most important one is that it can detect RATs in the early stage.

3.5 Dataset Used in Related Work

Not only the way of capturing data is different in previous works but also the amount of data used for classification is different. The accuracy, False Negative Rate and False Positive Rate of the detection systems are not the same. 65 types of Trojans traffic which contain 224 data flows and MSN, QQ and other normal traffic from 10

normal applications which contain 276 data flows are collected., i.e., a total of 500 data flows are obtained for the experiment [32].

RAT samples are executed in a virtual environment. On the other hand, traces of the normal application samples are collected on the laboratory network. Wireshark, a traffic monitor, is used to capture all these samples' traces.175 sessions are extracted in total, 10 are RAT sessions, the rest are legitimate sessions [31]. Other previous works do not mention the amount of dataset clearly.

3.6 Machine Learning Methods

Machine Learning algorithms are popular for building malware detection model. In Supervised Learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new unlabeled data based on pattern and associating the pattern. Scikit-learn is a standard machine learning library by python. It includes various machine learning algorithms. We use Naïve Bayes, Decision Trees, Random Forests and AdaBoost. Among them, Naïve Bayes, Decision Trees and Random Forests are widely used for malware detection. AdaBoost is also a suitable method for detecting Remote Access Trojans.

3.6.1 Decision Trees

Decision Trees (DTs) is a supervised learning method used for classification and regression. It is used to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. In Decision Trees, the process is broken down into individual tests which begin at the root node and traverse the tree, depending on the result of the test in that particular node. The tree begins at the root node. From the root node the tree branches or forks out to internal nodes. The decision to split is made by impurity measures [47].

3.6.1.1 Advantages of Decision Trees

There are many advantages of decision trees [18].

(1) Simple to understand and to interpret

(2) Requires little data preparation. It does not require data normalization, dummy variables to be created and blank values to be removed.

(3) The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.

(4) Able to handle both numerical and categorical data.
(5) Able to handle multi-output problems.
(6) As it uses a white box model, a given situation is observable in a model and the explanation for the condition is easily explained by Boolean logic.
(7) A model can be validated using statistical tests. It is possible to account for the reliability of the model.
(8) Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

3.6.1.2 Disadvantages of Decision Trees

The disadvantages of Decision Trees are as follows [18]:

(1) Decision-tree learners can overfit and create over-complex trees that do not generalize the data well. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
(2) Decision trees can be unstable because small variations in the data might result in a completely different tree being generated.
(3) The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical Decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree.
(4) There are concepts that are hard to learn because Decision Trees do not express XOR, parity or multiplexer problems easily.
(5) Decision-tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

3.6.2 Random Forests

Random Forests is an ensemble classifier that consists of many Decision Trees and outputs the class that is the mode of the class's output by individual trees. The method combines Breiman's "bagging" idea and the random selection of features and it improves prediction accuracy and control over-fitting [9].

3.6.2.1 Advantages of Random Forests

The advantages of Random Forests are as follows [43][51]:

(1) The Random Forest algorithm is not biased; it does not suffer from the overfitting problem because it takes the average of all the predictions.

(2) This algorithm is very stable. Even if a new data point is introduced in the dataset the overall algorithm is not affected much since new data may impact one tree, not all the trees.

(3) The Random Forest algorithm works well for both categorical and numerical features.

(4) Random forests can also handle missing values by using median values to replace continuous variables, or by computing the proximity-weighted average of missing values.

3.6.2.2 Disadvantages of Random Forests

There are three disadvantages of Random Forests [43][51]. They are as follows:

(1) Random Forests is slow in generating predictions because it has multiple Decision Trees. Whenever it makes a prediction, all the trees in the forest have to make a prediction for the same given input and then perform voting on it. This whole process is time-consuming.

(2) The model is difficult to interpret compared to a Decision Tree, where you can easily make a decision by following the path in the tree.

(3) Due to their complexity, they require much more time to train than other comparable algorithms.

3.6.3 Naïve Bayes

Naive Bayes is a widely used classification method based on Bayes theory. Based on class conditional density estimation and class prior probability, the posterior class probability of a test data point can be derived and the test data will be assigned to the class with the maximum posterior class probability [48]. Calculating the conditional probability is as follows:

$$P(h/D) = \frac{P(D/h)P(h)}{P(D)} \tag{3.1}$$

3.6.3.1 Advantages of Naïve Bayes

There are many advantages of Naïve Bayes [50][59].

(1) It is a simple approach which is fast and accurate for prediction.
(2) It has very low computation cost.
(3) It can efficiently work on a large dataset.
(4) It performs well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed.
(5) When the assumption of independence holds, a Naive Bayes classifier performs better compared to other models like logistic regression.

3.6.3.2 Disadvantages of Naïve Bayes

The disadvantages of Naïve Bayes are as follows [50][59]:

(1) If there is no training tuple of a particular class, this causes zero posterior probability. In this case, the model is unable to make predictions. This problem is known as Zero Probability/Frequency Problem.
(2) Another limitation of Naive Bayes is the assumption of independent predictors.

3.6.4 AdaBoost

Adaptive Boosting, called AdaBoost, is an ensemble machine learning algorithm. AdaBoost was built based on short decision tree models, each with a single decision point called decision stumps [25].

3.6.4.1 Advantages of AdaBoost

The advantages of AdaBoost are as follows [49]:

(1) AdaBoost is easy to implement.
(2) It iteratively corrects the mistakes of the weak classifier and improves accuracy by combining weak learners.
(3) Many base classifiers can be used with AdaBoost.
(4) AdaBoost is not prone to overfitting.

3.6.4.2 Disadvantages of AdaBoost

The disadvantages of AdaBoost are as follows [49]:

(1) AdaBoost is sensitive to noise data.
(2) It is highly affected by outliers because it tries to fit each point perfectly.
(3) AdaBoost is slower compared to XGBoost.

3.7 Summary of Related Works

Table 3.2 Summary of related works

Related Works	Approach	Limitation
[22][61][58]	Reviewed malware detection Signature based and behavior-based detection Static and dynamic analysis	Signature based approach cannot detect unknown malware Much false positive rate in behavior-based detection
[39]	Network behavior-based technique. It uses flow-level and IP-level features, e.g. duration, transport layer connection	It takes 5 minutes to terminate sessions in this work RATs will be detected after they stay long in the victim machine
[40]	Application network behavior-based approach Examples of features- number of connections, proportion upload of connection,	Complex to obtain features and to manage It may detect RATs after their long stay in the victim host
[31]	It detects RATs in the early stage of network traffic Early stage is defined depending on packet interval time A simple ratio of RATs and normal applications is used (10 instances of RATs and 165 instances of normal applications)	It does not consider error recovery features- TCP retransmission- and RATs will be missed Just a simple ratio of instances is not enough to obtain best detection model
[32]	Features are distributed in three layers: network layer, transport layer and application layer e.g. many sub-connections during primary connection, communications time	Complex to extract and obtain features Overhead

	False Positive Rate is fed back as input to adjust the training model 224 instances of Trojans and 276 instances of normal applications are applied	Just a simple ratio of instances is used for detection model
[34]	A hybrid approach for zero days attacks	It may not be early stage detection
[57]	Early traffic classification	It needs effective features

3.8 Summary

This chapter covers analyzing network behavior, tool that is used to analyze network traffic, the basic knowledge of TCP connection, previous research papers and their approaches for detecting Remote Access Trojans. Moreover, how to define features based on TCP connection and the amount of dataset are also explained. Supervised machine learning methods and their advantages, disadvantages are also discussed in detail. The summary of related works is also described in this chapter.

CHAPTER 4
FEATURE SELECTION

Features are important to get high performance of machine learning methods. In this work, features are extracted as much as possible from network traces captured by Wireshark, and these network traces are cut based on packet time interval. At first, information gain is used to select features. Weka data mining tool is used for this process. Then, features are selected based on comparing the differences of features. In this state, the difference of features is calculated manually and features are tuned according to the performance of the classifiers.

4.1 Remote Access Trojans and Normal Applications

Remote Access Trojans applied in this work are shown in Table 4.1. Remote Access Trojans used in this experiment are popular Trojans and they are also called Trojan building tools. They can be downloaded from Internet's black market. An infected binary file called Trojan can be created by using these tools.

Table 4.1: RATs that used in the experiment

	RATs	Version
1	ImminentMonitor	4.1.0.0
2	KilerRat	10.0.0
3	NjRat	0.6.4
4	Cerberus	1.0.3.4 Beta
5	Xtreme	3.8
6	Pandora	2.2
7	CyberGate	1.07.5
8	SpyGate	3.2
9	Xena	2.0.0
10	Babylon	1.6.0.0

RAT is one of the most dangerous Trojans because it consists of features of all types of Trojans. It provides an attacker with nearly unlimited access to host computer along with Screen Capture, File management, shell control and device drivers control

[8][16][20][24][35][46][54][60][69][70][81]. The main features of Remote Access Trojans applied in the thesis are explained in the Table 4.2. Babylon and KilerRAT can perform DDOS attack according to the specifications of the attacker's choice [8][60].

Table 4.2 Main features of ten Remote Access Trojans

no	Main features	Babylon	Cerberus	Cyber Gate	Imminent Monitor	Kiler RAT	NjRAT	Pandora	Spy Gate	Xena	Xtreme
1	Access to remote PCs, Computer control (Record and control victim's screen remotely, Record sound with a connected microphone, Record video with a connected webcam, and so on)	√	√	√	√	√	√	√	√	√	√
2	Downloads, uploads, deletes, and rename files, format drives	√	√	√	√	√	√	√	√	√	√
3	Keylogging (Steal login, passwords, credit card numbers, banking accounts)	√	√	√	√	√	√	√	√	√	√
4	Shell control	√	√	√	√	√	√	√	√	√	√
5	Registry management	√	√	√	√	√	√	√	√	√	√
7	Encoded or encrypted communication	√	√	-	√	-	-	-	-	-	-
8	Victim's computer added to a botnet	-	-	-	√	-	-	-	-	-	-
9	DDOS (weaponization)	√	-	-	-	√	-	-	-	-	-

√ : The RAT has this feature

- : The RAT does not have this feature

Although the functions of Remote Access Trojans are almost the same, some are also mentioned as password stealing virus, banking malware and spyware. Threat types are shown in Table 4.3. The possible damage of these 10 RATs are the same and they are described in Table 4.4.

Table 4.3: Threat type

no	Threat Type	Babylon	Cerberus	Cyber Gate	Imminent Monitor	Kiler RAT	njRAT	Pandora	SpyGate	Xena	Xtreme
1	RAT	√	√	√	√	√	√	√	√	√	√
2	Password stealing virus	-	-	√	√	-	-	-	-	-	-
3	Banking malware	-	-	√	√	-	-	-	-	-	-
4	Spyware	-	-	√	√	-	-	-	-	-	-

√ : The Trojan is this kind of threat type

-: The Trojan is not this kind of threat type

RAT: Remote Access Trojan

Table 4.4: Possible damage

no	Possible Damage	Babylon	Cerberus	Cyber Gate	Imminent Monitor	Kiler RAT	njRAT	Pandora	SpyGate	Xena	Xtreme
1	Data disclosure, data breach	√	√	√	√	√	√	√	√	√	√
2	Stolen banking information, passwords, identity theft	√	√	√	√	√	√	√	√	√	√

√ : The RAT can cause this kind of damage

Normal applications used in the experiment are shown in Table 4.5. Google, Facebook, Skype, Yahoo Messenger are internet-related service, social networking site, telecommunications application and instant messaging client that most of people use today. Many people use browser applications: Firefox, Chrome to navigate World Wide Web. Dropbox and PCloud are a file hosting service and the secure cloud storage that provide cloud services that many people use these days. BitTorrent and BitComet are P2P download tools. Many people use these tools for peer-to-peer file sharing.

Table 4.5: Normal applications used in the experiment

	Normal applications	Description
1	Dropbox	Cloud service
2	Pcloud	Cloud service
3	Bittorrent	P2P download tool
4	BitComet	P2P download tool
5	Facebook	Social media and service
6	Google	Internet-related services
7	Firefox	Browser
8	Chrome	Browser
9	Skype	Instant messenger
10	YahooMessenger	Instant messenger

4.2 System Architecture Based on Information Gain

The initial subset of features is extracted from network traces and it includes 14 features in this work. 14 features are explained in Table 4.6. Firstly, all features are used for classification and its performance in shown in Table 4.7. Then, features are evaluated and ranked using information gain and, some are removed based on criterion. Next, candidate set of features are chosen and utilized for classifiers. Finally, the performance result is produced. The proposed process of feature selection using information gain is shown in Figure 4.1.

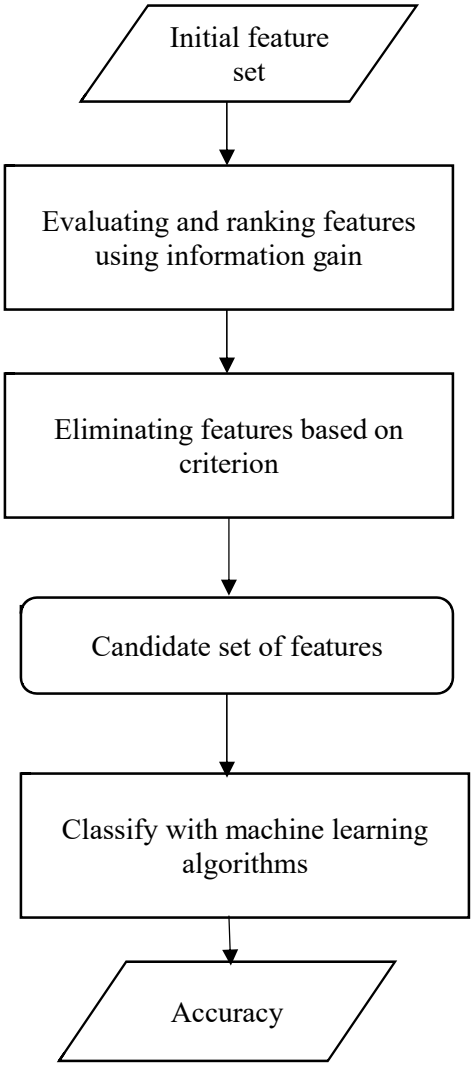

Figure 4.1: System architecture for feature Selection using information gain

4.2.1 The Proposed Features and Information Gain

Table 4.6: Proposed features

no	Feature	Description
1	PacNum	Number of packets
2	OutByte	Outbound data byte
3	OutPac	Outbound number of packets
4	InByte	Inbound data byte
5	InPac	Inbound number of packets
6	OutPacByInPac	Outbound number of packets/ inbound number of packets
7	OutByteByOut Pac	Outbound data byte/outbound number of packets

8	Duration	duration of the packets from the start of the communication to the given threshold
9	Bit/s(vict_att)	bit per seconds (from victim to attacker)
10	Bit/s(att_vict)	bit per seconds (from attacker to victim)
11	SentByReceive TrafficSize	Sent traffic size / Receive traffic size
12	OutByteByInByte	Outbound data byte / Inbound data byte
13	InByteByInPac	Inbound data byte / Inbound number of packets
14	DurByNoOfPac	Duration / Number of packets

The process of feature selection using Information gain is applied for two different ratios of normal and RAT instances: (1) dataset that includes 160 normal instances and 10 RAT instances, (2) dataset that includes 300 normal instances and 10 RAT instances.

4.2.2 The Proposed Features and their Gain Value for the First Dataset

The first dataset is one that includes 160 normal sessions and 10 Remote Access Trojans sessions. In this case, the information gain values of 7 features among 14 features are less than 0.055 and they are mentioned as 0 in Weka. These 7 features are removed and the remaining 7 features are utilized for classifiers. At last, 4 features whose information gain values are the highest among them are applied for classification. The important one is that information gain value of each feature changes if the ratio of normal and malicious sessions is changed.

Table 4.7: Features and their information gain value for the first dataset

No(N160-RAT10)	Feature	Gain Value
1	InByteByInPac	0.184
2	InByte	0.1541
3	SentByReceiveTrafficSize	0.1206
4	OutByteByInByte	0.1143
5	Bit/s(att_vict)	0.1034
6	OutByte	0.0787
7	OutPac	0.0573
8	InPac	< 0.055
9	OutPacByInPac	< 0.055
10	OutByteByOutPac	< 0.055

11	Bit/s(vict_att)	< 0.055
12	DurByNoOfPac	< 0.055
13	Duration	< 0.055
14	PacNum	< 0.055

4.2.3 Result for the First Dataset

For normal 160 sessions and 10 Remote Access Trojans sessions, parameters are tuned three times according to the classification results. Firstly, all 14 features are applied for classification. Then, 7 features are chosen after removing features whose information gain values are less than 0.055. These 7 features are shown in Table 4.8. Lastly, the best 4 features are utilized for four classifiers in order to get the best accuracy, the least False Negative Rate and False Positive Rate. These 4 features are shown in Table 4.9. The results classified by different numbers of features are shown in Table 4.10.

Table 4.8: Seven features after removing features whose information gain values are less than 0.055.

No(N160-RAT10)	Feature	Gain Value
1	InByteByInPac	0.184
2	InByte	0.1541
3	SentByReceiveTrafficSize	0.1206
4	OutByteByInByte	0.1143
5	Bit/s(att_vict)	0.1034
6	OutByte	0.0787
7	Outpac	0.0573

Table 4.9: Four features whose information gain values are greater than 0.11

No(N160-RAT10)	Feature	Gain Value
1	InByteByInPac	0.184
2	InByte	0.1541
3	SentByReceiveTrafficSize	0.1206
4	OutByteByInByte	0.1143

Table 4.10: Accuracy, FNR, FPR of NB, DT and AdaBoost for the first dataset

N160-RAT10	NB			DT			RF			ADaBoostM1		
	Acc	FNR	FPR	Acc	FNR	FPR	Acc	FNR	FPR	Acc	FNR	FPR
With 14 features	0.794	0.2	0.206	0.953	0.5	0.019	0.959	0.5	0.013	0.947	0.6	0.019
With 7 features whose IG value > 0.055	0.788	0.2	0.213	0.953	0.5	0.019	0.959	0.5	0.013	0.953	0.6	0.013
4 features whose IG value > 0.11	0.871	0.1	0.131	0.959	0.5	0.013	0.971	0.3	0.013	0.976	0.2	0.013

N160_RAT10 means normal 160 instances and RAT 10 instances

Naïve Bayes did not get high accuracy compared to other three methods but it got least False Negative Rate. It obtained 0.1 False Negative Rate by classifying with the best 4 features. However, compared to other methods its False Positive Rate is much.

The accuracy, False Negative Rate and False Positive Rate of Decision Trees do not change much in three classifications with different number of features. Its False Negative Rate is 0.5 which means that it missed 5 sessions in 10 malicious sessions. It is a bad situation for detecting Remote Access Trojans.

The performance of Random Forests does not change in two classifications with 14 features and 7 features. However, the best 4 features gave accuracy – 0.971, False Negative Rate – 0.3, and False Positive Rate - 0.013. In this state, Random Forests gave better performance than Decision Trees.

With 14 features and 7 features, AdaBoost obtains 0.947 and 0.953 for accuracy, False Negative Rate is 0.6, and False Positive Rate is 0.019 and 0.013 respectively. With 4 features, the accuracy is increased- 0.976, False Negative Rate is 0.2, and False Positive Rate is 0.013.

Among these classifiers, AdaBoost gets the highest accuracy, the least False Negative Rate and False Positive Rate. Moreover, the best 4 features gave the best result

than 14 features. The best result of all is accuracy-0.976, False Negative Rate-0.2 and False Positive Rate- 0.013.

4.2.4 The Proposed Features and their Gain Value for the Second Dataset

For the second dataset that includes 300 normal sessions and 10 Remote Access Trojan sessions, 14 features and their information gain values are shown in Table 4.11. The information gain values of six features are less than 0.03 and they are mentioned as 0 in Weka. They are removed and then, the remaining 8 features are applied for classification. Moreover, based on criterion, classification is performed again with best 5 features whose information gain value is the highest among them.

Table: 4.11: Features and their information gain value for the second dataset

No(N300-RAT10)	Feature	Gain Value
1	InByteByInPac	0.1037
2	InByte	0.0842
3	Bit/s(att_vict)	0.0825
4	OutByteByInByte	0.0675
5	SentByReceiveTrafficSize	0.0657
6	OutByte	0.0469
7	OutPac	0.0329
8	InPac	0.031
9	OutPacByInPac	< 0.03
10	OutByteByOutPac	< 0.03
11	Bit/s(vict_att)	< 0.03
12	DurByNoOfPac	< 0.03
13	Duration	< 0.03
14	PacNum	< 0.03

4.2.5 Result for the Second Dataset

For 300 normal instances and 10 RAT instances, eight features after removing six features whose information gain is less than 0.03 are shown in Table 4.12. The best five features whose information gain is greater than 0.05 are shown in Table 4.13.

Table 4.12: Eight features after removing six features whose information gain values are less than 0.03

No(N300-RAT10)	Feature	Gain Value
1	InByteByInPac	0.1037
2	InByte	0.0842
3	Bit/s(att_vict)	0.0825
4	OutByteByInByte	0.0675
5	SentByReceiveTrafficSize	0.0657
6	Outbyte	0.0469
7	Outpac	0.0329
8	Inpac	0.031

Table 4.13: Selected five features whose information gain values are greater than 0.05

No(N300-RAT10)	Feature	Gain Value
1	InByteByInPac	0.1037
2	InByte	0.0842
3	Bit/s(att_vict)	0.0825
4	OutByteByInByte	0.0675
5	SentByReceiveTrafficSize	0.0657

Table 4.14: Accuracy, FNR, FPR of NB, DT and AdaBoost for the second dataset

N300-RAT10	NB			DT			RF			ADaBoostM1		
	Acc	FNR	FPR	Acc	FNR	FPR	Acc	FNR	FPR	Acc	FNR	FPR
With 14 features	0.806	0.4	0.187	0.974	0.5	0.01	0.977	0.6	0.003	0.977	0.5	0.007
With 8 features whose IG > 0.03	0.816	0.2	0.183	0.974	0.6	0.007	0.981	0.5	0.003	0.974	0.7	0.003
5 features whose IG value > 0.05	0.765	0.1	0.24	0.977	0.6	0.003	0.977	0.6	0.003	0.977	0.6	0.003

IG: Information Gain

The comparison of classification results is shown in Table 4.14. The comparison is made by not only three different number of features based on information gain but also four different classifiers.

The accuracy of Naïve Bayes is not much like other three classifiers, but its False Negative Rate is the least and its False Positive Rate is the most among 4 classifiers. Decision Trees, Random Forests and AdaBoost cannot classify well because their False Negative Rate is 0.5 or 0.6 or 0.7 which means that they missed 5 or 6 or 7 sessions among 10 RAT sessions. Although some features are removed based on information gain value (criterion), False Negative Rate (FNR) is not reduced, and FNR of Decision Trees is increased from 0.5 to 0.6. Therefore, classification after removing features whose gain value is 0 cannot always produce good results for these features.

4.2.6 Best Features for Two Datasets

According to the results above, the best features whose information gain values are greater than 0.055 for normal 160 and RAT 10 sessions and the best features whose values are greater than 0.03 for normal 300 and RAT 10 sessions can be defined and they are shown below.

Table 4.15: Best features for two datasets

no	Best features of N160-RAT10	Best features of N300-RAT10
1	InByteByInPac	InByteByInPac
2	InByte	InByte
3	SentByReceiveTrafficSize	Bit/s(att_vict)
4	OutByteByInByte	OutByteByInByte
5	Bit/s(att_vict)	SentByReceiveTrafficSize
6	OutByte	OutByte
7	OutPac	OutPac
8		InPac

4.3 Choosing Features Based on the Differences of Normal and RAT Sessions

Another way for comparing features is to calculate differences of features manually without using information gain. Feature comparison that means how much each feature is different between normal session and RAT session is shown in Table 4.16. Then, features are arranged in descending order based on their difference and it is shown in Table 4.17. The process of feature selection based on the difference of features are also applied for two datasets: (1) dataset that includes 160 normal sessions and 10 RAT sessions, (2) dataset that includes 300 normal sessions and 10 RAT sessions.

Table 4.16: Comparing features based on the differences of normal and RAT sessions

no	Features	Type	Trend
1	PacNum	N	20% are more than 10 packets
		R	60.333% are more than 10 packets
2	OutByte	N	31.333% are more than 500 bytes
		R	19.333% are more than 500 bytes
3	OutPac	N	63.667% are more than 10 packets
		R	18.333% are more than 10 packets
4	InByte	N	84.667% are more than 200 bytes
		R	9.667% are more than 200 bytes
5	InPac	N	35.333% are more than 10 packets
		R	17.333% are more than 10 packets
6	OutPacByInPac	N	48% are more than 1
		R	87% are more than 1
7	OutByteByOutPac	N	32.333% are more than 100 byte per packet
		R	27.667% are more than 100 byte per packet
8	Duration	N	48.667% are more than 10 seconds
		R	40.667% are more than 10 seconds
9	vict_att	N	68.667% are more than 500
		R	60.333% are more than 500
10	att_vict	N	84% are more than 500
		R	33.333% are more than 500
11	RatioOfSent& ReceiveTrafficSize	N	20% are more than 1
		R	100% are more than 1
12	RatioOfOutByte& InByte	N	28.667% are more than 1
		R	97.667% are more than 1
13		N	69.667% are more than 100 byte

	InByteByInPac		
		R	0 are more than 100 byte
14	DurByNoOfPac	N	33.333% are more than 1
		R	41.667% are more than 1

Table 4.17: Features that shows the difference in descending order

no	Features	Difference
1	RatioOfSent& ReceiveTrafficSize	80
2	InByte	75
3	InByteByInPac	69.667
4	RatioOfOutByte& InByte	69
5	att_vict	50.667
6	OutPac	45.334
7	PacNum	40.333
8	OutPacByInPac	39
9	InPac	18
10	OutByte	12
11	vict_att	8.334
12	DurByNoOfPac	8.334
13	Duration	8
14	OutByteByOutPac	4.666

4.3.1 Result for the First Dataset

First dataset is one that includes 160 normal instances and 10 RAT instances. At first, eight features whose difference is greater than or equal 39 are used for classification and the result is obtained. Then, the difference of features is raised to greater than 39 and one more feature in this experiment is removed again. 7 features

whose difference is the highest among them are left for classification and they are applied for classifiers. The results are shown in Table 4.18.

Table 4.18: Accuracy, FNR and FPR of Naïve Bayes, Decision Trees, Random Forests and AdaBoost

N160-RAT10	NB			DT			RF			AdaBoostM1		
	Acc	FNR	FPR	Acc	FNR	FPR	Acc	FNR	FPR	Acc	FNR	FPR
With 14 features	0.794	0.2	0.206	0.953	0.5	0.019	0.959	0.5	0.013	0.947	0.6	0.019
With 8 features whose IG value >= 39	0.735	0.2	0.269	0.953	0.5	0.019	0.971	0.4	0.006	0.959	0.5	0.013
7 features whose IG value > 39	0.747	0.0	0.269	0.953	0.5	0.019	0.971	0.3	0.013	0.959	0.5	0.013

IG: Information Gain

For 160 normal sessions and 10 RAT sessions, the accuracy of Naïve Bayes is just 73% to 79%, False Negative Rate is the least compared to other methods-0.2 by classifying with 14 features and 8 features, but its FNR is 0 by classifying with 7 features. False Positive Rate is much more than the others – 0.269 by classifying with 14 or 8 or 7 features.

False Negative Rate of Decision Trees, Random Forests and AdaBoost is high – 0.4 or 0.5. The result of classifying by 7 features is not much different from that of 8 features.

Classification with 8 features and 7 features do not change much compared to classification with 14 features for these ratios of sessions.

4.3.2 Result for the Second Dataset

The second dataset is the dataset that includes 300 normal instances and 10 RAT instances. The classification results based on different numbers of features are shown in Table 4.19. In 300 normal sessions and 10 RAT sessions, the accuracy, False Negative Rate and False Positive Rate of Decision trees, Random forests and AdaBoost are the same.

False Negative Rate of Naïve Bayes is reduced to 0.1, but its False Positive Rate is 0.26 by classification with 8 or 7 features. False Negative Rate of Decision Trees and

Random Forests are high – 0.5 and 0.6, and its value does not change although some features are removed. AdaBoost's False Negative Rate is also high -0.6.

Table 4.19: Accuracy, FNR, FPR of NB, DT, RF and AdaBoost for the second dataset

N300-RAT10	NB			DT			RF			ADaBoostM1		
	Acc	FNR	FPR	Acc	FNR	FPR	Acc	FNR	FPR	Acc	FNR	FPR
With 14 features	0.806	0.4	0.187	0.974	0.5	0.01	0.977	0.6	0.003	0.977	0.5	0.007
With 8 features whose IG value >= 39	0.745	0.1	0.26	0.977	0.5	0.007	0.977	0.6	0.003	0.977	0.6	0.003
7 features whose IG value > 39	0.742	0.1	0.263	0.977	0.5	0.007	0.977	0.6	0.003	0.977	0.6	0.003

IG: Information Gain

4.4 Summary

In this chapter, the proposed approach for selecting features is explained. Feature ranking by information gain is changed based on different ratios of normal and malicious sessions. However, calculating the difference of features manually can show the best features easily and clearly. Generally, False Negative Rate of three classifiers: DT, RF and AdaBoost, are still high. Although False Negative Rate of Naïve Bayes is just 0.1, its accuracy is low and its False Positive Rate is much. Therefore, features extracted from network session that is cut based on packet interval time cannot give less False Negative Rate, less False Positive Rate and high accuracy for Remote Access Trojans applied in the experiment.

CHAPTER 5
SYSTEM ARCHITECTURE

Network-based malware detection is classified into two categories: signature-based detection and network behavior-based detection. Network behavior analysis has been researched to detect malware for many years. The command and control traffic of Remote Access Trojans is uncovered by network behavior-based approach.

Network behavior features outperform signatures of deep packet inspection (DPI) techniques as they do not need to be updated daily, they can detect new Remote Access Trojans and their variants. As it is a network-based approach, it does not need to be installed at every terminal and it makes less cost.

5.1 Feature Extraction

Feature extraction is important to extract effective features for machine learning methods. Features may contain irrelevant and redundant features that effect classification performance. It is important to keep effective features. However, the number of features may be as small as possible in order that the cost and the complexity of building a classifier can be reduced. Moreover, removing unimportant features facilitates data visualization, improves modeling, prediction performance, and speeds up classification process. Therefore, dimensionality reduction, feature extraction and feature selection, has been applied to machine learning methods.

The approach of extracting features that depends on packet interval time is not suitable for all Remote Access Trojans because the accuracy of classifying RATs and normal applications applied in the experiment is low and FNR is much. It has already been described in chapter 4. Moreover, it has a problem like TCP retransmission. Therefore, an effective approach for all Remote Access Trojans that behave the same way but may be different in name, appearance, port usage and some minor changes, is proposed in this thesis.

5.1.1 Research Flow

The proposed research flow is shown in Figure 5.1. Remote Access Trojans are run in a controlled area, and the network traffic is captured by Wireshark for analysis. Then features are extracted from the captured file and instances are obtained for

classification. These instances are learned by machine learning methods and the classification is evaluated by 10-fold cross validation. At last, Accuracy, False Negative Rate and False Positive Rate are resulted.

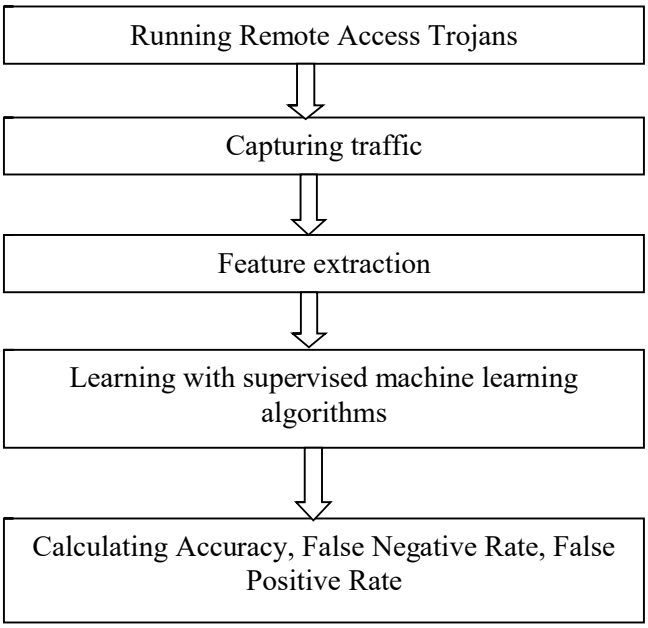

Figure 5.1: Research flow

5.1.2 How to Extract Features in the Experiment

In this approach, the number of packets n is used to cut the traffic and then features are extracted and calculated. The number of packets n used in this thesis is first twenty packets that is the optimal number of packets for ten RATs and ten normal applications in the experiment. The first twenty packets between two different hosts are selected for extracting features in order to avoid error recovery features like TCP Retransmission in TCP based connection.

If there is an interaction between two different IP addresses, the basic features are defined during 20 packets: (1) OutByte (outbound data byte), (2) InByte (inbound data byte), (3) Dur(duration from the first packet to twenty packets), (4) OutPac (outbound number of packets), (5) InPac (inbound number of packets), (6)PacNum. After that, 4 features (7) OutByteByOutpac, (8) InByteByInPac, (9) OutByteByInByte (10) OutPacByInPac are calculated based on these basic features. However, 3 features among basic features and the calculated 4 features are used for classifying the traffic. The process of feature extraction is shown in Figure 5.2.

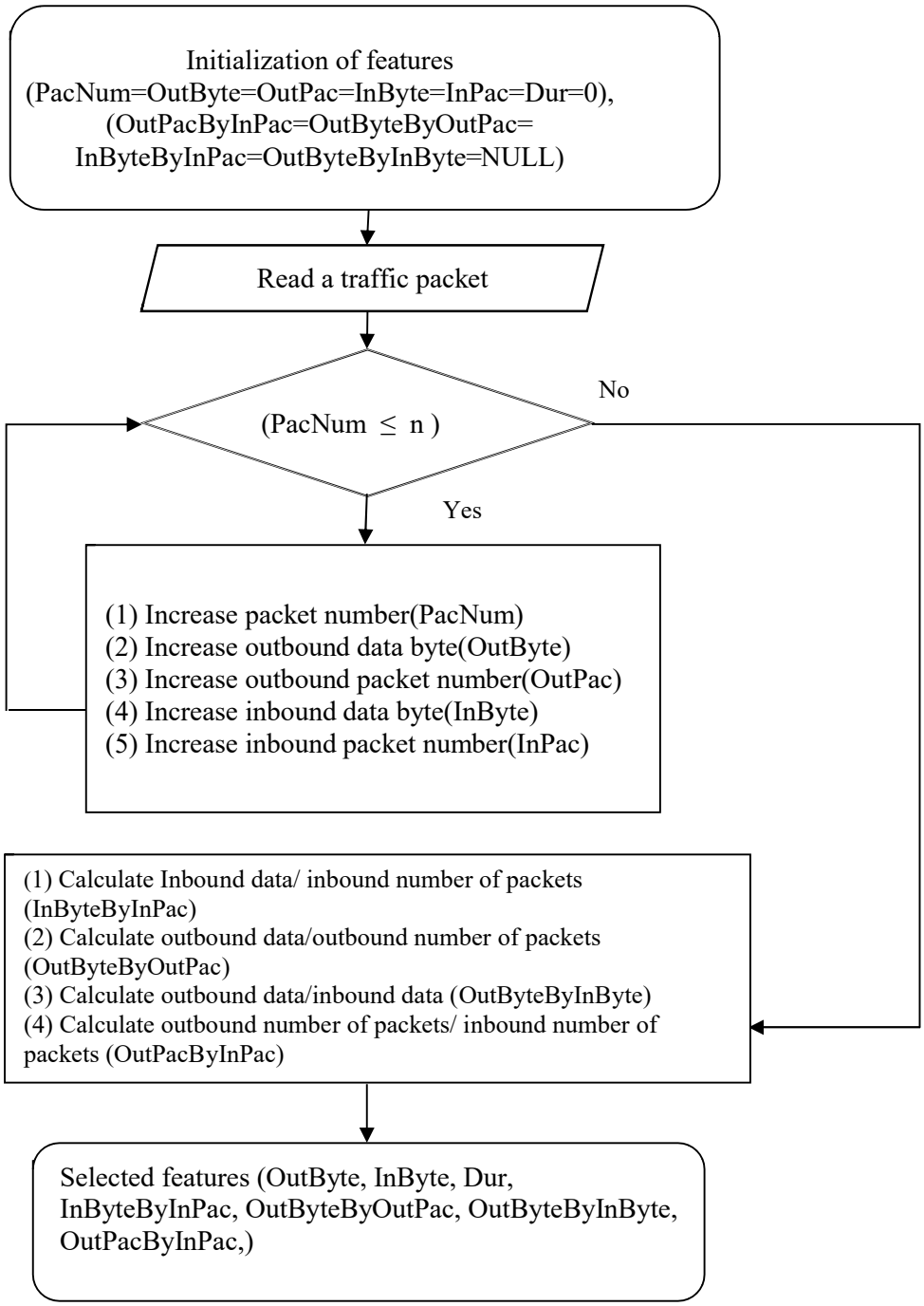

Figure 5.2: The Process of feature extraction

5.1.3 Benefit of First Twenty Packets

Features are extracted within the first twenty packets that starts SYN of TCP three-way handshake to twentieth packets without depending on how long packet interval time takes. First twenty packets are enough to define effective features and they can distinguish normal sessions and malicious sessions clearly to detect malicious

traffic of Remote Access Trojans in the early stage, and it can overcome the problem of error-recovery features too. In addition, RATs are run by many times and their different behaviors are captured. Thus, normal and malicious instances are increased, different ratios are obtained and they are applied for classification.

5.2 Parameters of Machine Learning Methods in scikit-learn

Algorithm tuning or parameter tuning is an important step for improving algorithm performance before presenting results or preparing a system for production. It is sometimes called Hyperparameter optimization. Hyper-parameters are parameters that are not directly learnt within estimators. In scikit-learn they are passed as arguments to the constructor of the estimator classes.

5.2.1 Parameters of Decision Trees in scikit-learn

(1) Criterion: string, optional (gini or entropy), default=" gini". It defines the function to measure the quality of a split. Sklearn supports "gini" criteria for Gini Index & "entropy" for Information Gain. By default, it takes "gini" value.

(2) Splitter: string, optional (best or random), default=" best". It is applied to choose the split at each node. Supports "best" value to choose the best split & "random" to choose the best random split. By default, it takes "best" value.

(3) max_features: int, float, string or None, optional (default=None). It is the number of features to consider when looking for the best split. We can input integer, float, string & None value.

If an integer is inputted then it considers that value as max features at each split.

If float value is taken then it shows the percentage of features at each split.

If "auto" or "sqrt" is taken then max_features=sqrt(n_features).

If "log2" is taken then max_features= log2(n_features).

If None, then max_features=n_features. By default, it takes "None" value.

(4) max_depth: int or None, optional (default=None). It is the max_depth parameter that denotes maximum depth of the tree. It can take any integer value or None. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples. By default, it takes "None" value.

(5) min_samples_split: int, float, optional (default=2). It is above the minimum number of samples required to split an internal node. If an integer value is taken then consider min_samples_split as the minimum no. If float, then it shows percentage. By default, it takes “2” value.

(6) min_samples_leaf: int, float, optional (default=1). It is the minimum number of samples required to be at a leaf node. If an integer value is taken then consider min_samples_leaf as the minimum no. If float, then it shows percentage. By default, it takes “1” value.

(7) max_leaf_nodes : int or None, optional (default=None). It is the maximum number of possible leaf nodes. If None then it takes an unlimited number of leaf nodes. By default, it takes “None” value.

(8) min_impurity_decrease: float, optional (default=0.)

A node will be split if this split induces a decrease of the impurity greater than or equal to this value.

(9) presort: bool, optional (default=False)

It defines whether to presort the data to speed up the finding of best splits in fitting. For the default settings of a decision tree on large datasets, setting this to true may slow down the training process. When using either a smaller dataset or a restricted depth, this may speed up the training. (default=False)

(10) random_state: int, RandomState instance or None, optional (default=None)

If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.

(11) min_weight_fraction_leaf: float, optional (default=0.)

The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.

(12) class_weight: dict, list of dicts, “balanced” or None, default=None

Weights associated with classes in the form {class_label: weight}. If not given, all classes are supposed to have weight one. For multi-output problems, a list of dicts can be provided in the same order as the columns of y.

5.2.2 Parameters of Random Forests in scikit-learn

(1) n_estimators: integer, optional (default=10). It is the number of trees in the forest

(2) max_features: int, float, string or None, optional (default =" auto"). It is the maximum number of features considered for splitting a node.

If int, then consider max_features feature at each split.

If float, then max_features is a percentage and int (max_features * n_features) features are considered at each split.

If "auto", then max_features=sqrt(n_features).

If "sqrt", then max_features=sqrt(n_features) (same as "auto").

If "log2", then max_features=log2(n_features).

If None, then max_features=n_features.

(3) max_depth: integer or None, optional (default=None). It is the maximum number of levels in each decision tree.

(4) max_leaf_nodes: int or None, optional (default=None). Grow trees with max_leaf_nodes in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.

(5) min_samples_split: int, float, optional (default=2). It is the minimum number of data points placed in a node before the node is split.

If int, then consider min_samples_split as the minimum number.

If float, then min_samples_split is a percentage and ceil (min_samples_split * n_samples) are the minimum number of samples for each split.

(6) min_samples_leaf : int, float, optional (default=1). It is the minimum number of data points allowed in a leaf node.

If int, then consider min_samples_leaf as the minimum number.

If float, then min_samples_leaf is a percentage and ceil (min_samples_leaf * n_samples) are the minimum number of samples for each node.

(7) criterion: string, optional (default =" gini"). The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

(8) min_weight_fraction_leaf: float, optional (default=0.). The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.

(9) min_impurity_decrease: float, optional (default=0.). A node will be split if this split induces a decrease of the impurity greater than or equal to this value.

(10) bootstrap: boolean, optional (default=True). It is a method for sampling data points (with or without replacement).

(11) oob_score: bool (default=False). It is used out-of-bag samples to estimate the generalization accuracy.

(12) n_jobs: integer, optional (default=1). The number of jobs to run in parallel for both fit and predict.

(13) random_state: int, RandomState instance or None, optional (default=None). If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.

(14) verbose: int, optional (default=0). Controls the verbosity of the tree building process.

(15) warm_start: bool, optional (default=False). When set to True, reuse the solution of the previous call to fit and add more estimators to the ensemble, otherwise, just fit a whole new forest.

(16) class_weight: dict, list of dicts, "balanced". "balanced_subsample" or None, optional (default=None) Weights associated with classes in the form {class_label: weight}. If not given, all classes are supposed to have weight one. For multi-output problems, a list of dicts can be provided in the same order as the columns of y.

5.2.3 Parameter of Naïve Bayes in scikit-learn

(1) random_state: int, RandomState instance or None, optional (default=None). If int, random_state is the seed used by the random number generator; If

RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.

5.2.4 Parameters of AdaBoost in scikit-learn

(1) base_estimator: object, optional (default=None). The base estimator from which the boosted ensemble is built. Support for sample weighting is required, as well as proper classes_ and n_classes_ attributes. If None, then the base estimator is DecisionTreeClassifier(max_depth=1)

(2) n_estimators : integer, optional (default=50). The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early.

(3) learning_rate: float, optional (default=1.) Learning rate shrinks the contribution of each classifier by learning_rate. There is a trade-off between learning_rate and n_estimators.

(4) algorithm: {'SAMME', 'SAMME.R'}, optional (default='SAMME.R'). If 'SAMME.R' then use the SAMME.R real boosting algorithm. base_estimator must support calculation of class probabilities. If 'SAMME' then use the SAMME discrete boosting algorithm. The SAMME.R algorithm typically converges faster than SAMME, achieving a lower test error with fewer boosting iterations.

(5) random_state: int, RandomState instance or None, optional (default=None). If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.

5.3 Tuning Parameters

The most important part of implementation is to tune the parameters of machine learning algorithms in order to obtain incredible results. Parameters in these methods are either to increase the predictive power of the model or to make it easier to train the model. Parameters are like the settings of an algorithm that can be adjusted to optimize the performance of a model.

Tuning parameter depends more on experimental results than theory. Therefore, the best method to determine the optimal settings is to try many different combinations

of parameters and evaluate the performance of each model. However, evaluating each model only on the training set can lead to overfitting problem that is one of the most fundamental problems in machine learning.

If a model is optimized for the training data, then the model will work very well on the training set and score high performance, but it will not be able to generalize to new data like a test set. When a model performs highly on the training data but poorly on the test data, this is known as overfitting. This model cannot be applied to new problems and it cannot be applied in real world. Thus, the standard procedure for tuning parameter accounts for overfitting through cross validation.

5.4 Performance Evaluation

There are different types of evaluation metrics to evaluate machine learning algorithms. They are Classification Accuracy, Logarithmic Loss, Confusion Matrix, Area Under Curve, F1 Score, Mean Absolute Error, Mean Squared Error and so on.

Classification accuracy, FNR and FPR are used for evaluating the proposed approach in this thesis.

5.4.1 Classification Accuracy

Classification Accuracy is the ratio of number of correct predictions to the total number of input samples. It works well only if there are balanced samples belonging to each class. Equation of accuracy is shown below.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions\ made} \qquad (5.1)$$

If unequal number of samples belonging to each class are trained to build a model, it gives high accuracy to a major class and it gives us the false sense of achieving high accuracy. So, accuracy is not enough to evaluate algorithms trained by unbalanced samples.

5.4.2 Confusion Matrix

Confusion Matrix is a performance measurement for machine learning classification where output can be two or more classes. It describes the complete

performance of the model. It is shown as a table with 4 different combinations of predicted and actual values in Figure 5.3.

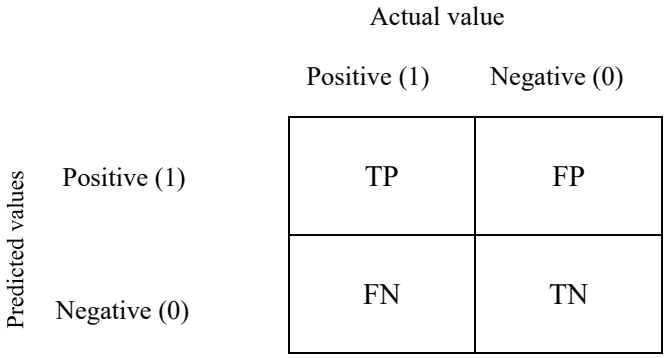

Figure 5.3: Confusion matrix

True Positives (TP): True attacks and alerts are generated.
False Positives (FP): alerts are wrongly generated when attacks are not real.
True Negatives (TN): no alerts and no attack.
False Negatives (FN): no alerts but true attack.

In this dissertation, Accuracy, False Negative Rate (FNR) and False Positive Rate (FPR) are used for evaluation. Accuracy means that the correctly classified number of both normal and malicious RAT instances are divided by the total number of both normal and RAT instances. False Positive Rate (FPR) means that the incorrectly classified number of normal instances are divided by the total number of normal instances. False Negative Rate (FNR) means that the incorrectly classified number of malicious RAT instances are divided by the total number of RAT instances. The less FNR while maintaining high accuracy, the better the detection system is for not missing malicious sessions. How to calculate Accuracy, FPR and FNR is shown below:

$$Accuracy = \frac{Correctly\ Classified\ Number\ of\ normal\ and\ RAT\ Instances}{Total\ Number\ of\ normal\ and\ RAT\ Instances} \quad (5.2)$$

$$FNR = \frac{Incorrectly\ Classified\ Number\ of\ RAT\ Instances}{Total\ Numberof\ RAT\ Instances} \quad (5.3)$$

$$FPR = \frac{Incorrectly\ Classified\ Number\ of\ Normal\ Instances}{Total\ Number\ of\ Normal\ Instances} \quad (5.4)$$

5.4.3 K-Fold Cross Validation

K-Fold Cross Validation is used to validate the result of classification in the experiment. In K-Fold Cross Validation data is split into k different subsets called fold. k-1 subsets are used to train our data and the last fold is left as test data. Then the model is averaged against each of the folds and our model is finalized. 10-fold cross validation is utilized to calculate Accuracy, False Negative Rate (FNR) and False Positive Rate (FPR) in our approach. The 10-fold cross validation is shown in Figure 5.4.

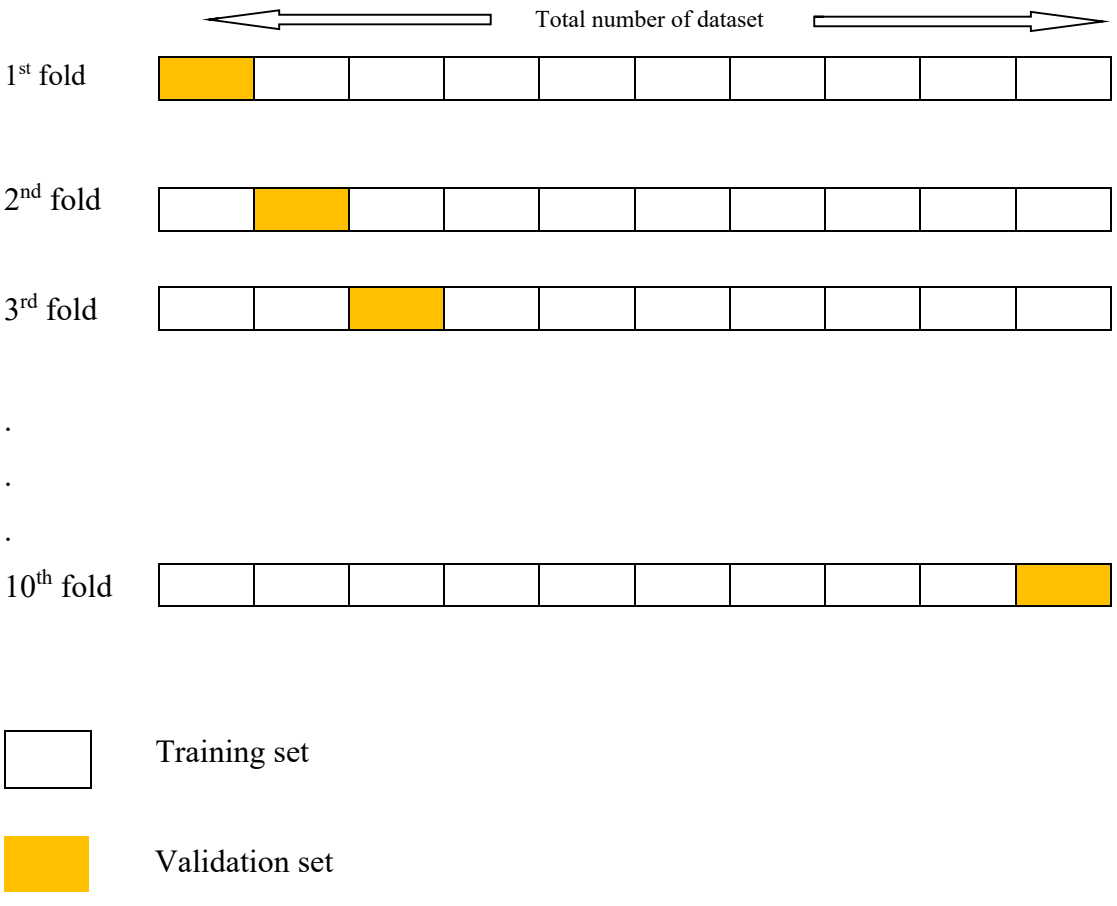

Figure 5.4: 10-fold cross validation

5.5 Summary

A new approach of extracting features based on the number of packets has been proposed, implemented and, its benefits are explained clearly in this chapter. First twenty packets are the optimal number of packets for Remote Access Trojans and normal applications in the experiment. Four machine learning methods and their parameters are also described in detail. Evaluation is performed well using confusion matrix and 10-fold cross validation.

CHAPTER 6
EXPERIMENTAL ANALYSIS AND EVALUATION

The majority of research in the area of network behavior analysis is based on real datasets. A simulated dataset that is created using simulator like ns1 and ns2 cannot represent the real network intrusion scenario and cannot express network behavior. It is important to generate real and timely datasets to ensure accurate and consistent evaluation of methods. A new dataset is proposed to implement this crucial point in our work. A testbed is set up to launch network traffic of attack as well as normal application's nature.

If a malware is installed in a system for analysis, it will destroy the functions of the system like creating large amount of shortcut files, copying or deleting files and folders and, the computers can be controlled by attacker. VMware is incredibly helpful in this process of investigating network behavior of malware without infecting the host and observing how they interact with file system, registry, attacker through network. VMware can simulate multiple computers running simultaneously on a single physical system.

6.1 Experimental Setup

A virtual environment that includes two virtual machines: an attacker and a victim, is set up. Two virtual machines have Windows operating system that Remote Access Trojans are likely to target. The attacker is a place where RAT is executed, and the victim is a place where the attacker's server.exe is executed. 10 types of Remote Access Trojans and 10 normal applications are used in the experiment. Wireshark is run on the victim to capture and collect network traffic. Testbed for detecting Remote Access Trojans is shown in Figure 6.1. The widely used RATs are applied in the experiment. Normal applications include cloud services, p2p download tools, browsers and social services that most of people use in Internet today. RATs and normal applications used in the experiment are shown in Table 6.1.

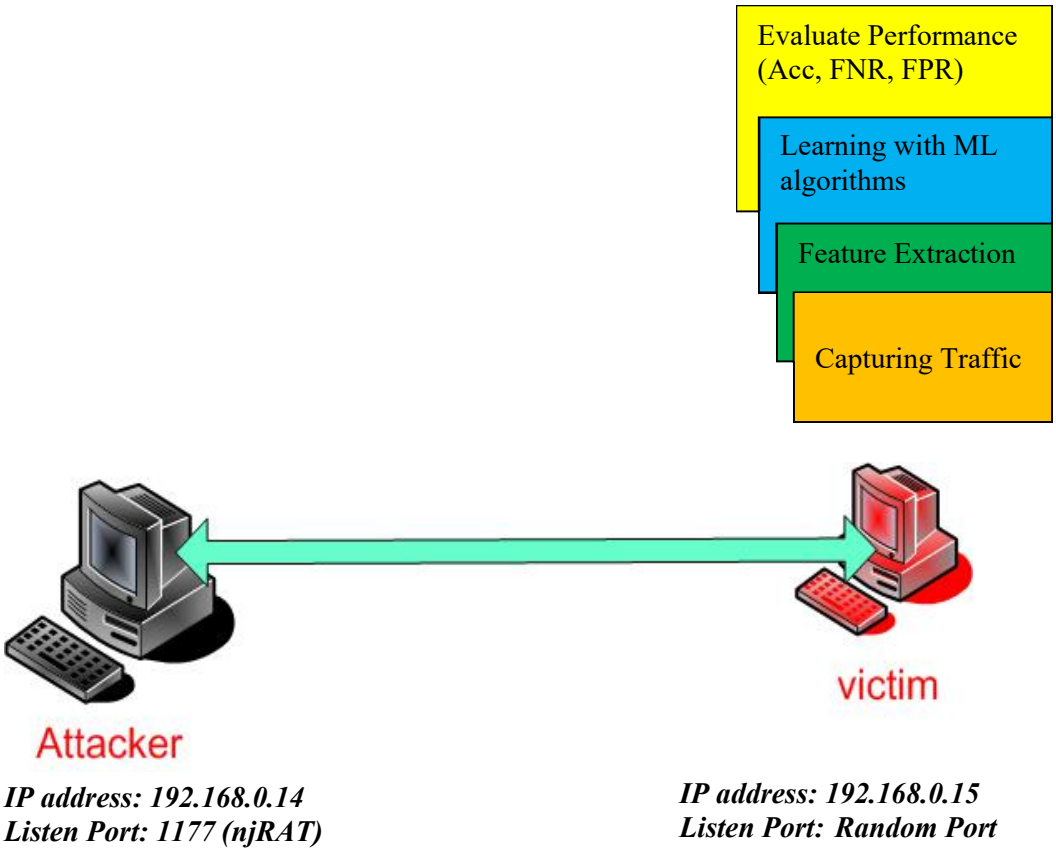

Figure 6.1: Testbed for detection of Remote Access Trojan

Table 6.1. RATs and normal applications used in the experiment

No	RATs	Normal applications
1	Babylon	BitComet
2	Cerberus	Bittorrent
3	CyberGate	Chrome
4	ImminentMonitor	Dropbox
5	KilerRat	Facebook
6	NjRat	Firefox
7	Pandora	Google
8	SpyGate	Pcloud
9	Xena	Skype
10	Xtreme	YahooMessenger

6.2 Capturing Network Traffic

How an attacker controls a victim is shown in Figure 6.2. The attacker used njRAT to intrude the victim's computer in this testbed. The attacker can do many things – accessing drives from File Manager, accessing victim's computer through Remote Desktop, running command line from Remote shell, controlling Registry, Process Manager and so on.

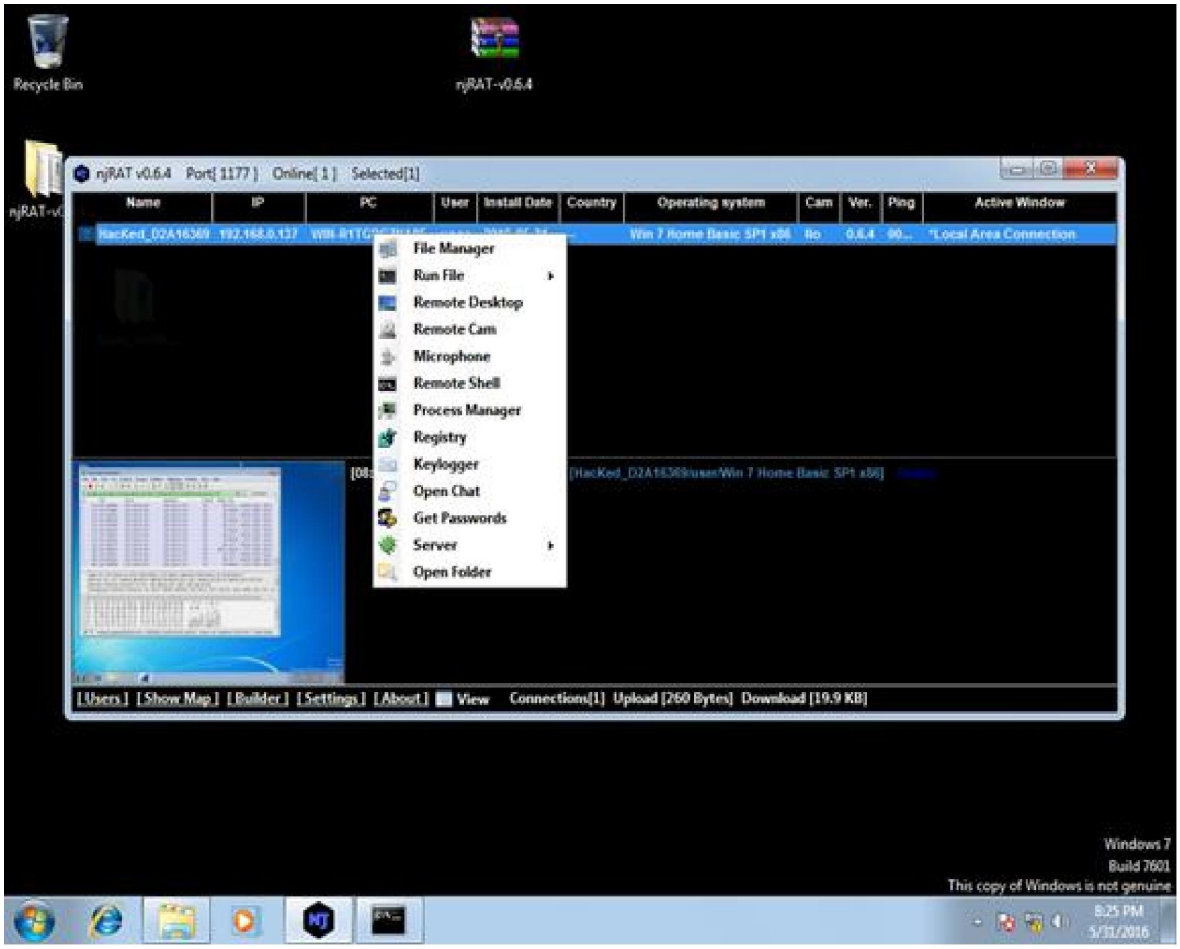

Figure 6.2: View from attacker side

njRAT traffic captured by Wireshark in the victim's machine is shown in Figure 6.3. The attacker used port 1177, and this port number is assigned by attacker. Typically, he or she doesn't use well-known ports from 1 to 1024.

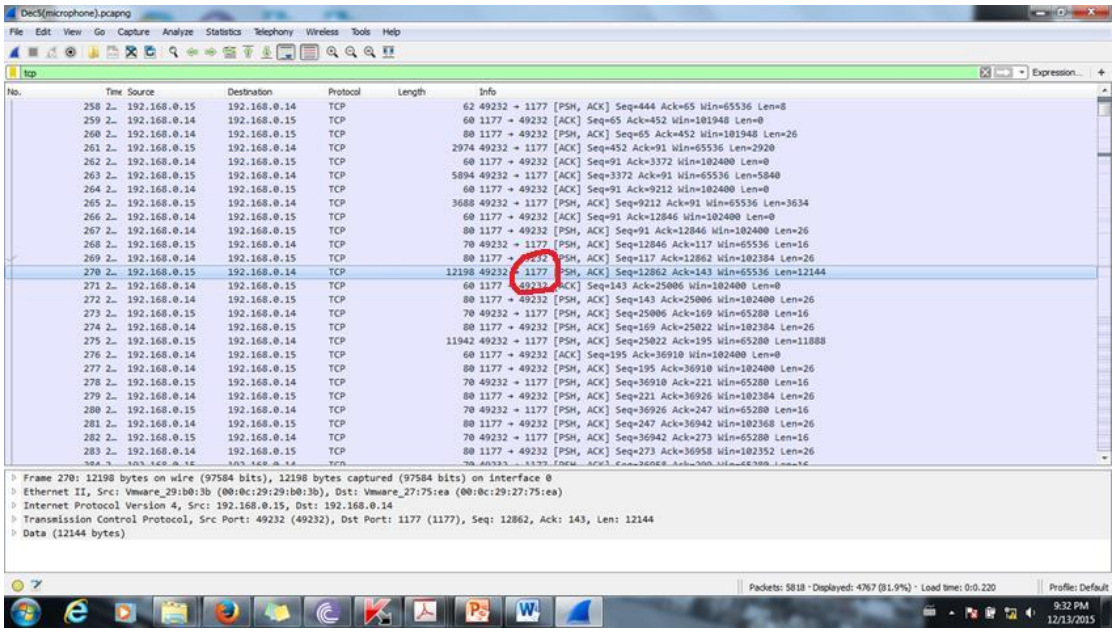

Figure 6.3: njRAT traffic in Wireshark

6.3 Problem of Previous Work

In order to extract features, the previous work cut the traffic from SYN to the packet that packet interval time is at least more than 1 second. An example of typical network traffic of Remote Access Trojans is shown in Figure 6.4 and Figure 6.5. In this example, 192.168.0.33 is attacker, 192.168.0.137 is victim. Port number of njRAT is 1177. Packet interval time that is more than 5 seconds occurs 6th packet in this example. Inbound data byte is 1, outbound data byte is 245.

An arrow in Figure 6.5 shows the way of cutting network trace that starts from SYN to the packet whose packet interval time is at least more than 1 second, in that case, packet interval time is more than 5 seconds.

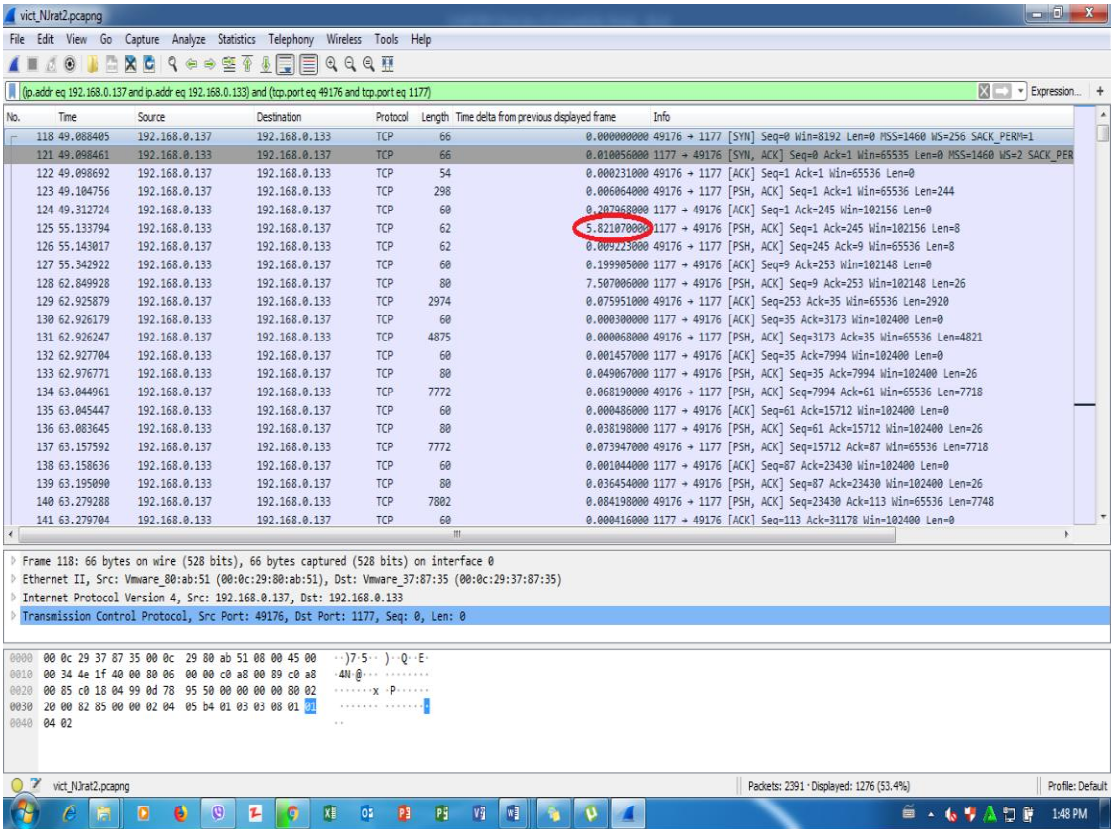

Figure 6.4: Packet interval time is more than 5 seconds

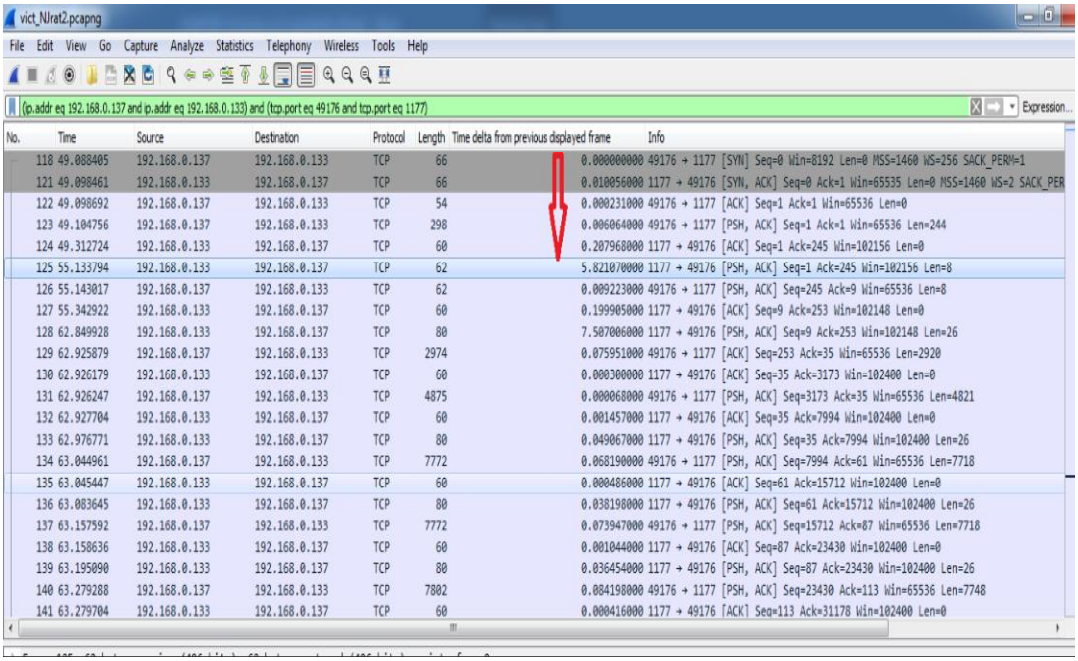

Figure 6.5: Network trace that starts from SYN to a packet whose packet interval time more than 5 seconds

In Figure 6.6, TCP Retransmission occurs and packet interval time takes more than 3 seconds before TCP three-way handshaking finishes. In this situation, data for features cannot be obtained correctly because there is no data that exchanges between two sides in this situation. It is a problem of previous work. An example is shown in Figure 6.6.

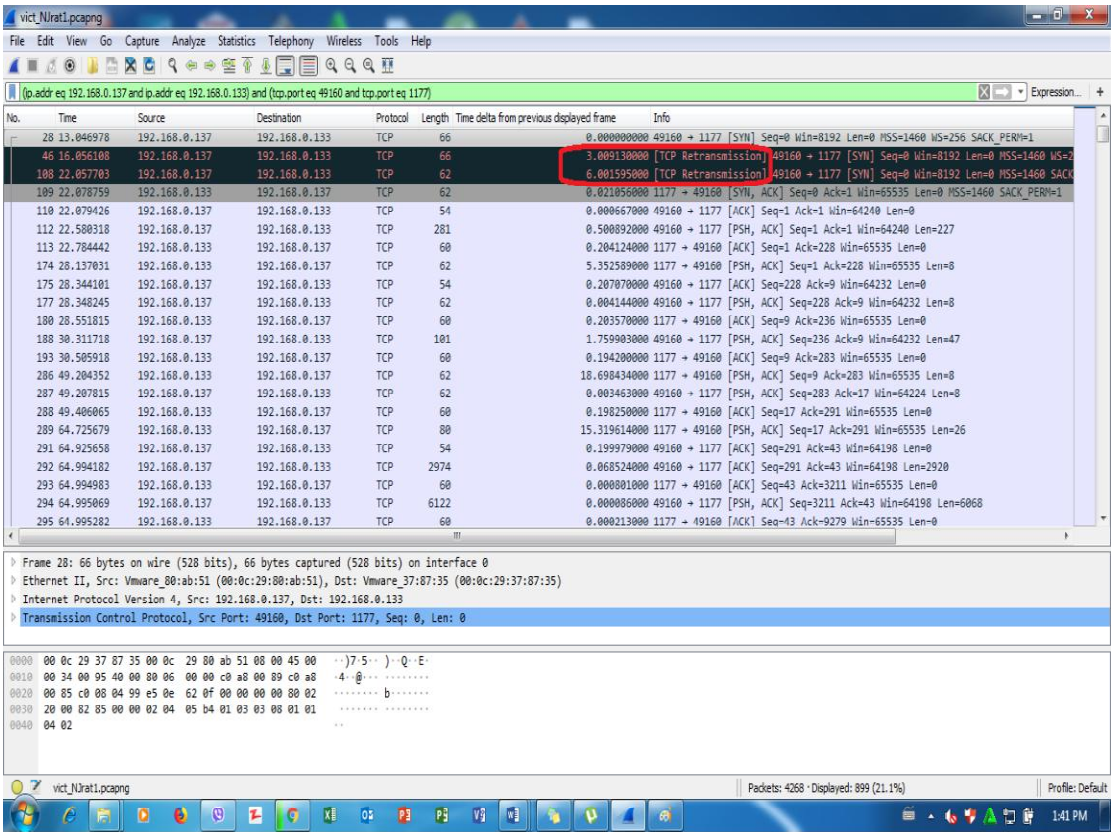

Figure 6.6: TCP Retransmission

6.4 First Twenty Packets

In order to solve this problem, the first twenty packets between two sides are cut and features are extracted. First twenty packets of network traffic are shown in Figure 6.7 and Figure 6.8. TCP Retransmission is not a problem in this way of cutting the traffic.

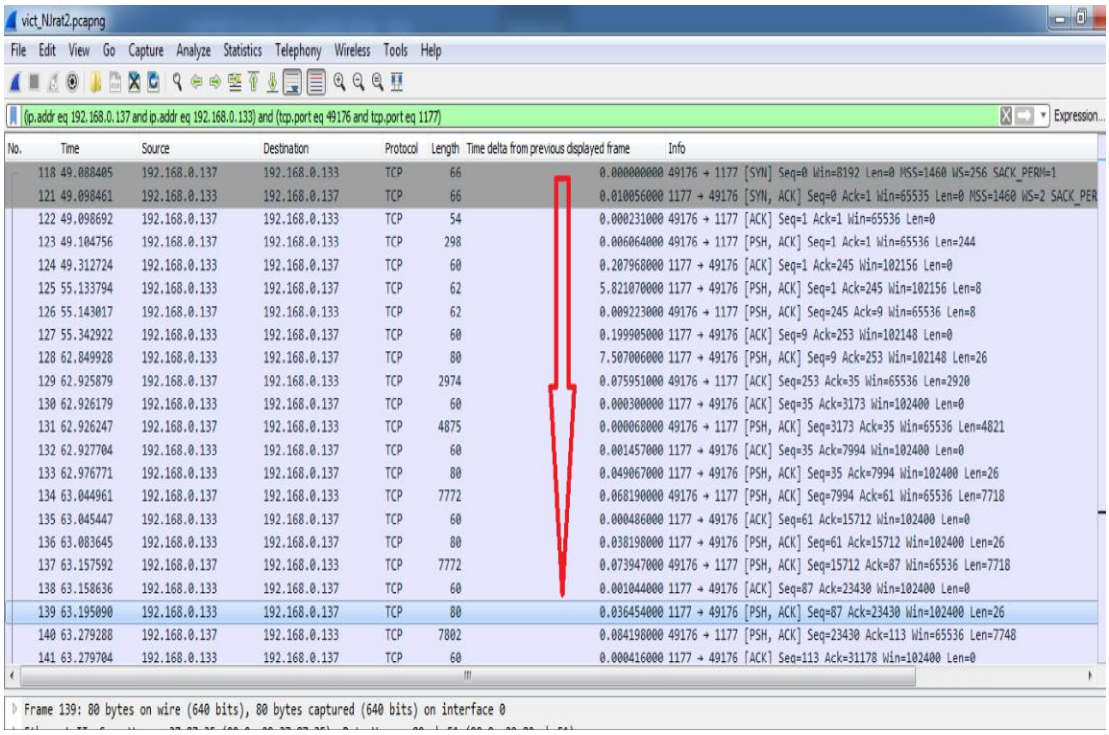

Figure 6.7: First twenty packets

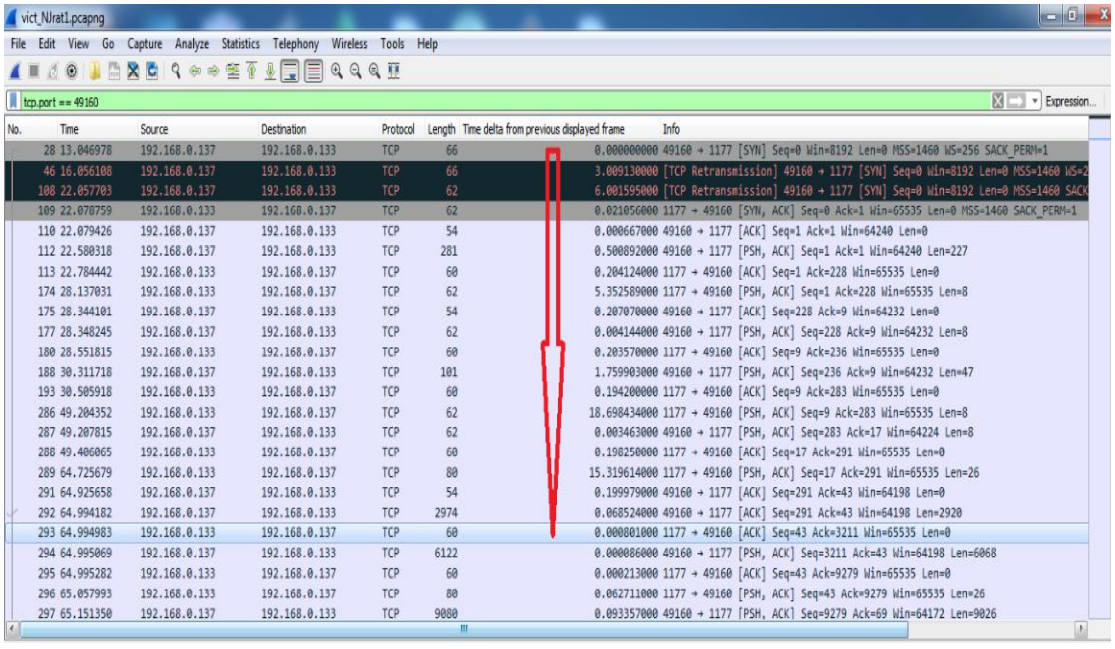

Figure 6.8: First twenty packets with TCP retransmission

Network behavior features for a session are extracted from the trace that starts from a SYN packet in the TCP three-way handshake and ends at twentieth packets. It is the very first-time traffic that collects after the victim is infected by RAT. If this RAT is not detected and it is not removed from victim's computer, it always connects back to the attacker. It is a considerable situation because the attacker may stay as long as possible to control the victim. When a RAT is run next time after system reboots, it always sends connection back to the attacker.

Seven effective features are proposed and they are applied for all classifiers. These 7 features and their difference between normal and RAT instances are shown in Table 6.2 and Table 6.3.

Table 6.2: Seven features during first twenty packets

No	Features	Description
1	Outbyte	Outbound data byte
2	InByte	Inbound data byte
3	InByteByInPac	Rate of Inbound data Byte/ Inbound number of packets
4	OutByteByOutPac	Rate of Outbound data byte/Outbound number of packets
5	Duration	Duration from the first packet to twenty packets
6	OutByteByInByte	Rate of Outbound data byte/Inbound data byte
7	OutPacByInPac	Rate of outbound number of packets/ inbound number of packets

Table 6.3 Features comparison for the first twenty packets

no	Features	Type	Trend	Difference
1	OutByte	N	88.67% are more than 500 bytes	25.67
		R	63% are more than 500 bytes	
2	InByte	N	99.33 % are more than 200 bytes	89.33
		R	10% are more than 200 bytes	
3	InByteByInPac	N	99.667 % are more than 20 byte per packet	89.667
		R	10% are more than 20 byte per packet	
4	OutByteByOutPac	N	87.333% are more than 50 byte per packet	26.333
		R	61% are more than 50 byte per packet	
5	Duration	N	19.33% take more than 20 seconds	37.67
		R	57% take more than 20 seconds	
6	OutByteByInByte	N	28% are more than 1	72
		R	100 % are more than 1	
7	OutPacByInPac	N	41% are more than 1	16
		R	25 % are more than 1	

A standard python scikit-learn library is used to implement machine learning algorithms. Gaussian Naïve Bayes, Decision Trees, Random Forests and AdaBoost classifiers are utilized in our work and 10 folds cross validation, Accuracy, False Negative Rate and False Positive Rate are used for evaluation.

6.5 The Unbalanced Dataset

There are cases that unbalanced classes exist such as diagnosis test that classifies positive or negative to many people. Just like this, there may be 99.9 percent of normal traffic and 0.01 percent of malicious traffic in a typical network in the real world. Therefore, building the detection model for normal and Remote Access Trojans traffic was tried based on the unbalanced ratio of normal and malicious sessions in this work. Two different ratios of normal and malicious sessions are utilized for classification.

Firstly, 15 sessions from each normal application and 1 session from each RAT are collected and, 150 normal instances and 10 RAT instances are used for building a model. The classification methods are Naïve Bayes, Decision Trees, Random Forests and AdaBoost. Parameters of each method have to be optimized in order to get the best performance.

6.5.1 Tuning Parameters I

Firstly, parameters of NB, DT, RF and AdaBoost are tuned for 150 normal instances and 10 RAT instances. Although all parameters are applied for classification, some parameters need to be tuned. The results of classification that uses tuned parameters are different from that of classification that uses default parameters in some cases.

6.5.1.1 Parameters of NB

Naïve Bayes has just one parameter -random_state. If random_state = 0, its accuracy is 95.62, FNR is 0.6 and FPR is 0.007.

6.5.1.2 Tuning Parameters of DT

There are 12 parameters in decision trees. (1) Criterion, (2) Splitter, (3) max_features, (4) max_depth, (5) min_samples_split, (6) min_samples_leaf, (7) max_leaf_nodes, (8) min_impurity_decrease, (9) presort, (10) random_state, (11) min_weight_fraction_leaf, (12) class_weight. random_state = 0 for all classifiers for reproducibility, and class_weight = None for different ratios of normal and malicious sessions in Decision Trees and Random Forests classifiers.

All parameters of DT are tuned to fit the dataset. By tuning min_weight_fraction_leaf and min_impurity_decrease, the accuracy reduces to 93.75%. Moreover, changing values for min_sample_leaf, presoft decreases the accuracy to 97%. Using entropy for criterion obtains the accuracy - 96%.The best result is obtained by tuning two parameters – max_depth and max_features, and by using others as default value. In 150 normal instances and 10 RAT instances, max_depth is 3 which means that the maximum depth of the tree is 3. max_features = None which means that the maximum number of features for splitting a node is the same as the number of features when classification is performed. The default value of parameters and the tuned values are shown in Table 6.4. The best result for this ratio of normal and

RAT instances is 98.75% accuracy, FNR is 0.1 and FPR is 0.007. This result is shown in Table 6.5.

Table 6.4: Default value and tuned value of Decision Trees parameters

No	Name of parameter	Default value	Tuned value
1	random_state	0	0
2	max_depth	None	3
3	max_leaf_nodes	None	None
4	min_samples_split	2	2
5	min_samples_leaf	1	1
6	presort	False	False
7	max_features	auto	None
8	min_weight_fraction_leaf	0.0	0.0
9	min_impurity_decrease	0.0	0.0
10	Criterion	gini	gini
11	Splitter	best	best
12	class_weight	None	None

Table 6.5: Experimental result of Decision Trees

DT(N150_RAT10)	Acc	FNR	FPR
With default values	97.5	0.3	0.007
With tuned values	98.75	0.1	0.007

6.5.1.3 Tuning Parameters of RF

In the case of Random Forests, there are 16 parameters but random_state is assigned zero for reproducibility and class_weight = None. All 14 parameters and their related values are tuned to obtain the best result. Tuning values of max_leaf_nodes, min_weight_fraction_leaf obtains the accuracy between 93% and 97%. The best result is obtained by tuning two parameters – max_features and bootstrap. max_features = 4 which means that the maximum number of features for splitting a node is 4. bootstrap = False which means that sampling data points is done without replacement. The other

parameters are used as default value. The default and tuned values are shown in Table 6.6. The results of default and tuned parameters is shown in Table 6.7.

Table 6.6: Default and tuned values of Random Forests parameters

No	Name of parameter	Value	Tuned value
1	random_state	0	0
2	n_estimators	10	10
3	criterion	gini	gini
4	max_features	auto	4
5	max_depth	None	None
6	min_samples_split	2	2
7	min_samples_leaf	1	1
8	min_weight_fraction_leaf	0.0	0.0
9	max_leaf_nodes	None	None
10	min_impurity_decrease	0.0	0.0
11	bootstrap	True	False
12	oob_score	False	False
13	n_jobs	1	1
14	verbose	0	0
15	warm_start	False	False
16	class_weight	None	None

Table 6.7: Experimental result of Random Forests

RF(N150_RAT10)	Acc	FNR	FPR
With default values	97.5	0.3	0.007
With tuned values	98.75	0.1	0.007

6.5.1.4 Parameters of AdaBoost

In AdaBoost, by changing values of n_estimators and learning_rate, the accuracy is 98%. However, tuning the parameter- Algorithm = SAMME from the default gives the best accuracy, FNR and FPR. The default values of parameters and the tuned values are shown in Table 6.8. The best result is shown in Table 6.9.

Table 6.8: Default and tuned value of AdaBoost parameters

No	Name of parameter	Value	Tuned value
1	random_state	0	0
2	n_estimators	50	50
3	learning_rate	1	1
4	algorithm	SAMME.R	SAMME
5	base_esimator	DecisionTreeClassifier (max_depth=1)	DecisionTreeClassifier (max_depth=1)

Table 6.9: Experimental result of AdaBoost

AdaBoost(N150_RAT10)	Acc	FNR	FPR
With default values	98.75	0.1	0.007
With tuned values	99.38	0.1	0.0

6.5.2 Tuning Parameters II

Next, 300 normal sessions and 10 RATs sessions are applied for detection model. The value of parameter -random_state, is assigned zero for reproducibility for all different ratios of normal and malicious sessions and for all classifiers, and class_weight = None for all ratios of normal and RAT instances in Decision Trees and Random Forests classifiers.

6.5.2.1 Parameters of NB

Naïve Bayes has just one parameter -random_state. If random_state = 0, the accuracy of Naïve Bayes is 97.74, FNR is 0.4 and FPR is 0.01.

6.5.2.2 Tuning Parameters of DT

All parameters and their related values are tuned to get the best result in Decision Trees. Tuning parameters:max_depth, criterion, splitter, max_leaf_nodes, min_simple_split, min_samples_leaf, and presort, obtains the accuracy that is less than or equal to 98.71%. If the value of max_features is 1 and, it is combined with default

value of other parameters, the best result is obtained. The default and tuned values of parameter are shown in Table 6.10. The best result is shown in Table 6.11.

Table 6.10: Default and tuned value of Decision Trees parameters

No	Name of parameter	Default value	Tuned value
1	random_state	0	0
2	max_depth	None	None
3	max_leaf_nodes	None	None
4	min_samples_split	2	2
5	min_samples_leaf	1	1
6	presort	False	False
7	max_features	auto	1
8	min_weight_fraction_leaf	0.0	0.0
9	min_impurity_decrease	0.0	0.0
10	Criterion	gini	gini
11	Splitter	best	best
12	class_weight	None	None

Table 6.11: Experimental result of Decision Trees

DT(N300_RAT10)	Acc	FNR	FPR
With default values	98.71	0.3	0.003
With tuned values	99.68	0.1	0.0

6.5.2.3 Tuning parameters of RF

All parameters are tuned to fit the data. The optimized values of parameters are shown in Table 6.12. If n_esimators = 100 and other parameters are default value, the accuracy is 99.03%. Tuning max_depth obtains the accuracy that is less than or equal to 99.03%. The values of two parameters: max_features and bootstrap, are changed from default value. Tuning these two parameters and using others as default value obtains the best result. For the 300 normal instances and 10 RAT instances, max_feaures = 4 which means that the maximum features for splitting a node is 4. Bootstrap = False means that sampling data points is done without replacement. Other

parameters are used as default value. The default and tuned values of parameters are shown in Table 6.12. The results are shown in Table 6.13.

Table 6.12: Default and tuned value of Random Forests parameters

No	Name of parameter	Value	Tuned value
1	random_state	0	0
2	n_estimators	10	10
3	criterion	gini	gini
4	max_features	auto	4
5	max_depth	None	None
6	min_samples_split	2	2
7	min_samples_leaf	1	1
8	min_weight_fraction_leaf	0.0	0.0
9	max_leaf_nodes	None	None
10	min_impurity_decrease	0.0	0.0
11	bootstrap	True	False
12	oob_score	False	False
13	n_jobs	1	1
14	verbose	0	0
15	warm_start	False	False
16	class_weight	None	None

Table 6.13: Experimental result of Random Forests

RF(N300-RAT10)	Acc	FNR	FPR
With default values	98.71	0.3	0.003
With tuned values	99.35	0.1	0.003

6.5.2.4 Tuning Parameters of AdaBoost

All parameters with different values are tried for classification but the result is not better than that of classification with the default value. So, parameters with default value are applied in the case of AdaBoost. The default values of parameters are shown in Table 6.14. AdaBoost with default parameters obtained the high accuracy – 99.35, FNR – 0.1 and FPR – 0.003. This result is shown in Table 6.15.

Table 6.14: Parameters of AdaBoost

No	Name of parameter	Value
1	random_state	0
2	n_estimators	50
3	learning_rate	1
4	algorithm	SAMME.R
5	base_esimator	DecisionTreeClassifier(max_depth=1)

Table 6.15: Experimental result of AdaBoost

AdaBoost (N300_RAT10)	Acc	FNR	FPR
With default values	99.35	0.1	0.003

6.6 The Balanced Dataset

If the main goal of classification is to get the highest possible accuracy, it does not need to do any balancing because most of the percentage accuracy will come from the classes with more training examples. Moreover, balancing is not necessary if the minority classes do not contribute much to achieving main goal. However, class balancing techniques are necessary when the minority classes are critical points that should not be missed.

In the experiment, each RAT is run many times in order to capture the variant behavior of RAT. In this way the number of malicious sessions is also increased without using sampling method and the balanced normal and malicious sessions are obtained for building a detection model. 300 normal sessions from 10 normal applications and 300 RAT sessions from 10 RATs are applied to the Naïve Bayes, Decision Trees, Random Forests and AdaBoost classifiers.

Parameters of Naïve Bayes, Decision Trees, Random Forests and AdaBoost are tuned according to the balanced instances: 300 normal instances and 300 RAT instances.

6.6.1 Parameters of NB

If random_state = 0, the accuracy of Naïve Bayes is 89.0, FNR is 0.2 and FPR is 0.02.

6.6.2 Tuning Parameters of DT

For 300 normal instances and 300 RAT instances, by tuning parameters: criterion, splitters, min_sample_leaf, min_samples_split, presort, the accuracy is less than or equal to 99.0%. The best result is obtained by tuning two parameters: max_depth and max_features, along with others as default value. In this case, max_depth = 3 which means that the maximum depth of the tree is 3. max_features = None which means that the maximum number of features for splitting a node is same as the number of features when classification is performed. The tuned parameters of DT for 300 normal instances and 300 RAT instances are the same as that of DR for 150 normal instances and 150 RAT instances. The default and tuned parameters are shown in Table 6.16. The results are shown is Table 6.17.

Table 6.16: Default and tuned value of Decision Trees parameters

No	Name of parameter	Default value	Tuned value
1	random_state	0	0
2	max_depth	None	3
3	max_leaf_nodes	None	None
4	min_samples_split	2	2
5	min_samples_leaf	1	1
6	presort	False	False
7	max_features	auto	None
8	min_weight_fraction_leaf	0.0	0.0
9	min_impurity_decrease	0.0	0.0
10	Criterion	gini	gini
11	Splitter	best	best
12	class_weight	None	None

Table 6.17: Experimental result of Decision Trees

DT(N300_RAT300)	Acc	FNR	FPR
With default values	98.83	0.01	0.013
With tuned values	99.67	0.0	0.007

6.6.3 Tuning Parameters of RF

In Random Forests, tuning parameters: max_leaf_nodes, n_estimators, min_samples_split, min_samples_leaf, max_depth, oob_score, warm_start and criterion, obtains the accuracy that is less than or equal to 99.5%. The best accuracy, the least FNR and FPR are obtained by tuning two parameters: max_features and bootstrap. For 300 normal instances and 300 RAT instances, max_features = 4 which means that the maximum features for splitting a node is 4. Bootstrap = False means that sampling data points is done without replacement. The default and tuned values of parameters are shown in Table 6.18. The results are shown in Table 6.19.

Table 6.18: Default and tuned value of Random Forests parameters

No	Name of parameter	Value	Tuned value
1	random_state	0	0
2	n_estimators	10	10
3	criterion	gini	gini
4	max_features	auto	4
5	max_depth	None	None
6	min_samples_split	2	2
7	min_samples_leaf	1	1
8	min_weight_fraction_leaf	0.0	0.0
9	max_leaf_nodes	None	None
10	min_impurity_decrease	0.0	0.0
11	bootstrap	True	False
12	oob_score	False	False
13	n_jobs	1	1
14	verbose	0	0
15	warm_start	False	False
16	class_weight	None	None

Table 6.19: Experimental result of Random Forests

RF(N300_RAT300)	Acc	FNR	FPR
With default values	99.5	0.003	0.007
With tuned values	99.67	0.0	0.007

6.6.4 Tuning Parameters of AdaBoost

In AdaBoost, default parameters are also the best parameters for performance. These parameters are shown in Table 6.20. The result is shown in Table 6.21.

Table 6.20: Default parameters of AdaBoost

No	Name of parameter	Value
1	random_state	0
2	n_estimators	50
3	learning_rate	1
4	algorithm	SAMME.R
5	base_esimator	DecisionTreeClassifier(max_depth=1)

Table 6.21: Experimental result of AdaBoost

AdaBoost(N300_RAT300)	Acc	FNR	FPR
With default values	99.5	0.0	0.01
With tuned values	99.5	0.0	0.01

6.7 Experimental Data

A Trojans has two parts: client and server. As the attacker's pc performs like a client and the victim's pc works like a server, he or she downloads data from the victim which performs a server part. Therefore, the outbound data is much more than inbound data in the victim side. It is the main point that can differentiate the RAT traffic from normal traffic. In normal applications, the user part does like a client and he or she downloads data from the server. Generally, the inbound data is much more than outbound data in the case of normal application in the victim side. When a RAT runs many times, there is a slight difference in the value of the same feature. However, this variation is still different from the behavior of normal applications. An example of 20 instances for normal and RAT are shown in Table 6.22. Class 0 means normal application and class 1 means Remote Access Trojan in the Table 6.22.

Table 6.22: Twenty instances for normal and RAT

	OutByte	InByte	InByteByInPac	OutPacByInPac	OutByteByOutPac	Duration	RatioOutByteInByte	Class
1	458	4259	473.2222222	1.22222222	41.63636364	0.155	0.107536981	0
2	2146	281	31.22222222	1.22222222	195.0909091	0.294	7.637010676	0
3	573	4212	468	1.22222222	52.09090909	0.735	0.136039886	0
4	2061	5674	630.4444444	1.22222222	187.3636364	0.97	0.363235812	0
5	462	4787	478.7	1	46.2	92.35	0.096511385	0
6	468	5900	590	1	46.8	4.552	0.079322034	0
7	1081	7069	589.0833333	0.666666667	135.125	0.717	0.152921205	0
8	1013	4297	477.4444444	1.22222222	92.09090909	0.339	0.235745869	0
9	615	4212	468	1.22222222	55.90909091	0.657	0.146011396	0
10	653	4250	425	1	65.3	0.268	0.153647059	0
11	421	17	1.7	1	42.1	15.002	24.76470588	1
12	287	25	2.5	1	28.7	20.012	11.48	1
13	2409	17	1.7	1	240.9	9.718	141.7058824	1
14	253	17	1.7	1	25.3	17.39	14.88235294	1
15	415	17	1.7	1	41.5	16.324	24.41176471	1
16	2517	25	2.5	1	251.7	7.702	100.68	1
17	279	13	1.3	1	27.9	20.221	21.46153846	1
18	251	17	1.7	1	25.1	17.148	14.76470588	1
19	287	17	1.7	1	28.7	17.192	16.88235294	1
20	287	17	1.7	1	28.7	16.118	16.88235294	1

6.8 Summary of Different Results Based on Different Ratios

Different ratios of normal and malicious instances, and their results are shown below:

Table 6.23: Different ratios of normal and malicious instances, and their results

Ratio of normal and malicious instances	NB			DT			RF			AdaBoost		
	Acc	FNR	FPR	Acc	FNR	FPR	Acc	FNR	FPR	Acc	FNR	FPR
N300-RAT300	89.0	0.2	0.02	99.67	0.0	0.007	99.67	0.0	0.007	99.5	0.0	0.01
N300-RAT10	97.74	0.4	0.01	99.68	0.1	0.0	99.35	0.1	0.003	99.35	0.1	0.003
N150-RAT10	95.62	0.6	0.007	98.75	0.1	0.007	98.75	0.1	0.007	99.38	0.1	0.0

Acc: Accuracy, FNR: False Negative Rate, FPR: False Positive Rate

Different ratios of normal and malicious instances and their results are shown in Table 6.23. In Naïve Bayes, Accuracy is high- 97% and 95% but False Negative Rate is 0.4 and 0.6 while using unbalanced ratios. Although its' FNR reduces to 0.2 with balanced instances, its accuracy decreases to 89%. Decision Trees has a slight difference in accuracy although different ratios of instances are used for classification. False Negative Rate of Decision Trees is 0.1 when unbalanced ratios of instances are used. But its False Negative Rate and False Positive Rate are reduced to 0 and 0.007 when balanced instances are classified. The accuracy of Random Forests does not change much with both balanced and unbalanced instances. The False Negative Rate of RF is 0.1 when unbalanced instances are classified. The accuracy is 99.67, no False Negative Rate and False Positive Rate is 0.007 when the balanced instances are used. The accuracy of AdaBoost is high – 99.5, 99.35 and 99.38 for both balanced and unbalanced instances. False Negative Rate is 0 when balanced instances are classified. It got less False Negative Rate – 0, 0.003 and 0.01.

6.9 Summary

In this chapter, how the experiment is set up, how to capture network traffic, the problem of previous work and the approach to overcome this problem are explained. Then, the results of unbalanced instances, balanced instances, and the parameters of different classifiers are described. The values of tuned parameters for 150 normal instances and 10 RAT instances are the same as that of tuned parameters for 300 normal instances and 300 RAT instances in both case of Decision Trees and Random Forest classifiers.

CHAPTER 7
THE COMPARISON ON THE PERFOMRANCE OF TWO DETECTION APPROACHES

The previous work used packet time interval in cutting the traffic in order to extract features [31]. In our work, first twenty packets are used in cutting the traffic and new features are proposed in order to detect Remote Access Trojans as early as possible. False Negative Rate that is a matter in previous work is also reduced to get an acceptable level for production environment.

7.1 Normal Applications and Remote Access Trojans that are Applied in Previous Work

Normal applications utilized in previous work is shown in Table 7.1 and Remote Access Trojans in Table 7.2.

Table 7.1: Normal applications used in previous work

no	Type	Description
1	Dropbox	cloud service
2	Skype	instant messenger
3	Chrome	web browser
4	YahooM!	instant messenger
5	Firefox	web browser
6	BitComet	P2P download tool
7	Teamviewer	P2P remote management tool
8	TorBrowser	anonymous web browser
9	BitTorrent	P2P download tool
10	PPTV	P2P video sharing tool

Table 7.2: Remote Access Trojans used in previous work

no	Remote Access Trojans
1	Bandook
2	Bozok
3	Cerberus
4	Nuclear
5	Poison Ivy
6	Turkojan
7	Gh0st
8	Netbus
9	OptixPro
10	ProRat

7.1.1 Seven Selected Features in Previous Work

In related work, early stage detection that depends on packet interval time uses 7 features for building a model. These features are PacNum, OutPac, InPac, OutByte, InByte, OutPacByInPac, and OutByteByOutPac and they are shown in Table 7.3. The difference of these 7 features between normal application and RAT are manually calculated and it is shown in Table 7.4. In Table 7.5, the amount of difference is shown in decreasing order.

Table 7.3: Seven features used in previous work

No	Features	Description
1	PacNum	packet number
2	OutPac	outbound packet number
3	InPac	inbound packet number
4	OutByte	outbound data size
5	InByte	inbound data size
6	O/Ipac	rate of OutPac/InPac
7	OB/OP	rate of OutByte/OutPac

Table 7.4: Comparing features for early stage detection that depends on packet interval time

no	Name of Features	Type	Difference between normal and RAT traffic
1	PacNum	N	78% are more than 10 packets
		R	20% are more than 10 packets
2	OutByte	N	93% are more than 500 bytes
		R	20% are more than 500 bytes
3	OutPac	N	52% are more than 10 packets
		R	10% are more than 10 packets
4	InByte	N	71% are more than 500 bytes
		R	10% are more than 500 bytes
5	InPac	N	38% are more than 10 packets
		R	10% are more than 10 packets
6	O/Ipac (OutPacByInPac)	N	99% are more than 1
		R	60% are more than 1
7	OB/OP (OutByteByOutPac)	N	68% are more than 100 byte per packet
		R	30% are more than 100 byte per packet

Table 7.5: Difference in descending order

no	Features	Difference between normal and RAT traffic
1	OutByte	73
2	InByte	61
3	PacNum	58
4	OutPac	42
5	O/Ipac(OutPacByInPac)	39
6	OB/OP(OutByteByOutPac)	38
7	InPac	28

7.2 Remote Access Trojans and Normal Applications Used in this Thesis

Remote Access Trojans and normal applications used in the work is shown in Table 7.6.

Table 7.6: Remote Access Trojans and normal applications used in our work

No	RATs	Normal applications
1	ImminentMonitor	Dropbox
2	KilerRat	Pcloud
3	NjRat	Skype
4	Cerberus	YahooMessenger
5	Xtreme	Facebook
6	Pandora	Bittorrent
7	CyberGate	BitComet
8	SpyGate	Google
9	Xena	Firefox
10	Babylon	Chrome

7.2.1 Using Seven Features of Previous Work for Our Dataset

The seven features of Omote's work [31] are applied in our dataset and, the differences of these features are calculated manually and they are shown in Table 7.7. The amount of the differences is shown in descending order in Table 7.8.

Table 7.7: Comparing seven features that depends on packet interval time for RATs and normal applications applied in our experiment

no	Name of Features	Type	Difference between normal and RAT traffic
1	PacNum	N	76% are more than 10 packets
		R	36.333% are more than 10 packets
2	OutByte	N	68.667% are more than 500 bytes
		R	19.333% are more than 500 bytes
3	OutPac	N	36% are more than 10 packets
		R	18% are more than 10 packets
4	InByte	N	84.667% are more than 200 bytes
		R	9.667% are more than 200 bytes
5	InPac	N	35.333% are more than 10 packets
		R	17.333% are more than 10 packets
6	O/Ipac (OutPacByInPac)	N	28% are more than 1
		R	29.667% are more than 1
7	OB/OP (OutByteByOutPac)	N	32.333% are more than 100 byte per packet
		R	27.667% are more than 100 byte per packet

Table 7.8: Difference of features in descending order

no	Features	Difference between normal and RAT traffic
1	InByte	75
2	OutByte	49.334
3	PacNum	39.667
4	OutPac	18
5	InPac	18
6	OB/OP (OutByteByOutPac)	4.666
7	O/Ipac (OutPacByInPac)	1.667

7.2.2 Seven Selected Features Applied in this Work

The 7 features applied in this work is shown in Table 7.9. The differences of these features are also calculated manually and they are shown in Table 7.10. The amount of difference is shown in descending order in Table 7.11.

Table 7.9: Seven features from first 20 packets used in our work

No	Features	Description
1	Outbyte	Outbound data byte
2	Inbyte	Inbound data byte
3	InByteByInPac	rate of Inbound data Byte/ Inbound number of packets
4	OutByteByOutPac	rate of Outbound data byte/Outbound number of packets
5	Duration	duration from the first packet to twenty packets
6	OutByteByInByte	rate of Outbound data byte/Inbound data byte
7	OutPacByInPac	rate of outbound number of packets/ inbound number of packets

Table 7.10: Features comparison for the first 20 packets

no	Name of Features	Type	Difference between normal and RAT traffic
1	OutByte	N	88.67% are more than 500 byte
		R	63% are more than 500 byte
2	InByte	N	99.33 % are more than 200 byte
		R	10% are more than 200 byte
3	InByteByInPac	N	99.667 % are more than 20 byte per packet
		R	10% are more than 20 byte per packet
4	OutByteByOutPac	N	87.333% are more than 50 byte per packet
		R	61% are more than 50 byte per packet
5	Duration	N	19.33% take more than 20 seconds
		R	57% take more than 20 seconds
6	OutByteByInByte	N	28% are more than 1
		R	100 % are more than 1
7	OutPacByInPac	N	41% are more than 1
		R	25 % are more than 1

Table 7.11: Difference of features in descending order

no	Features	Difference between normal and RAT traffic
1	InByteByInPac	89.667
2	InByte	89.33
3	OutByteByInByte	72
4	Duration	37.67
5	OutByteByOutPac	26.333
6	OutByte	25.67
7	OutPacByInPac	16

7.3 Comparing the Values of Features for the Approach that Depends on Packet Interval Time and the One that Depends on First Twenty Packets

Features are compared based on the difference between normal application and RAT. It is shown in Table 7.12. There are three results shown in three different columns. The first column is the difference of features from Omote's work for RATs and normal applications expressed in his paper. Their approach depends on packet interval time to cut the traffic. The second column shows the difference of features for RATs and normal applications applied in this thesis. Same features and same way of cutting the traffic like Omote's work but different RATs are applied in this work. If the same approach is applied for different RATs, there is variation in values of features. After new way of cutting the traffic is applied, 7 features including 3 new features and their differences are shown in the last column.

Table 7.12: Summary of three works

no	Features	Difference between normal and RAT traffic		
		An approach that depends on packet interval time		For first twenty packets
		For RATs and normal applications applied in Omote's work [12]	For RATs and normal applications applied in our experiment	For RATs and normal applications applied in our experiment
1	PacNum	58	39.667	-
2	OutPac	42	18	-
3	InPac	28	18	-
4	OutByte	73	49.334	25.67
5	InByte	61	75	89.33
6	O/Ipac(OutPacByInPac)	39	1.667	16
7	OB/OP(OutByteByOutPac)	38	4.666	26.333
8	Duration	-		37.67
9	OutByteByInByte	-		72
10	InByteByInPac	-		89.667

7.4 Tuned Parameters of DT, RF and AdaBoost

The parameters of DT and RF are tuned again in order to increase the detection rate, and they are shown in Table 7.13 and 7.14. The default parameters of AdaBoost that are applied in this work are shown in Table 7.15. These parameters are applied for both the previous approach which is a detection system depending on packet time interval and the approach which depends on first twenty packets.

Table 7.13: Parameters for Decision Trees

no	Name of parameter	Value (Ref paper)	Value (first twenty packets)
1	random_state	0	0
2	max_depth	None	None
3	max_leaf_nodes	None	None
4	min_samples_split	2	2
5	min_samples_leaf	1	1
6	presort	False	False
7	max_features	3	4
8	criterion	gini	entropy
9	splitter	best	best
10	Min_weight_fraction_leaf	0	0
11	Min_impurity_decrease	0	0

Table 7.14: Parameters for Random Forests

no	Name of parameter	value
1	random_state	0
2	n_estimators	100
3	criterion	Entropy
4	max_depth	None
5	min_samples_split	2
6	min_samples_leaf	1
7	min_weight_fraction_leaf	0.0
8	max_features	4
9	max_leaf_nodes	None
10	min_impurity_decrease	0.0
11	bootstrap	False
12	oob_score	False
13	n_jobs	1
14	verbose	0
15	warm_start	False

Table 7.15: Parameters for AdaBoost

No	Name of parameter	Value
1	random_state	0
2	n_estimators	50
3	learning_rate	1
4	algorithm	SAMME.R
5	base_esimator	DecisionTreeClassifier(max_depth=1)

7.5 A Performance Result of Two Approaches

A comparison on the performance of two different approaches on the same dataset is shown in Table 7.16. The first approach depends on packet interval time [31] and the second one is first twenty packets which is our proposed approach. 165 normal instances and 10 RATs instances are classified by Naïve Bayes, Decision Trees, Random Forests and AdaBoost. The detection that depends on packet interval time gets

high accuracy but its False Negative Rate is much – 0.4 by DT, 0.5 by RF and 0.6 by AdaBoost. The second approach that used the first twenty packets obtains 98% accuracy and 0.1 FNR by DT, RF and AdaBoost.

Table 7.16. A performance comparison of two detection approaches

Different approach		NB			DT			RF			AdaBoost		
		Acc	FNR	FPR	Acc	FNR	FPR	Acc	FNR	FPR	Acc	FNR	FPR
Early stage detection that depends on packet interval time	For RATs and normal app in Omote's work	0.43	0.1	0.597	0.96	0.2	0.03	0.971	0.1	0.023	-	-	-
	For RATs and normal applications in our experiment	0.85	0.2	0.151	0.97	0.4	0.006	0.96	0.5	0.006	0.95	0.6	0.012
Our approach (first twenty packets)		0.96	0.5	0.012	0.988	0.1	0.006	0.988	0.1	0.006	0.988	0.1	0.006

7.6 Summary

The comparison of the two approaches on the same dataset is explained in this chapter. From previous work, 2 features- PacNum and InPac- are removed. Two effective features -Duration and InByteByInPac are added. Then, the combination of 7 features proposed and implemented in this thesis can describe much difference between normal and RAT instances than the 7 features of previous work which is early stage detection that depends on packet interval time. As features are extracted in the first twenty packets, Remote Access Trojans can be uncovered as early as possible. Error-recovery features like TCP retransmission is not a problem for first twenty packets. Parameters of machine learning methods are also tuned in order to get the best result.

CHAPTER 8
CONCLUSION AND FUTURE WORK

Nowadays, the number of computer security related incidents and crimes has grown rapidly. The reason for this sudden increase is not only due to the increase of Internet users around the world, but also because of raising the chance of creating more security holes. Malicious users are being able to find security holes faster than system developers can handle. In addition, there has been a radical change in intruder techniques, increased amounts of damage, increased difficulty of detecting an attack, and increased difficulty of catching the attackers.

Now people cannot avoid using Internet as they use online banking, credit or debit cards and some do business on social networking sites. Any information stored or transmitted can be stolen. Data can be stolen if an intruder illegally access to a system or eavesdrop on confidential data transmissions. Depending on the type of information stolen, the results can be catastrophic. Data pertaining to customer accounts, personnel data, and data related to finances is usually extremely sensitive and, it is the most valuable asset to the organization. Therefore, it is very important to prevent unauthorized access, information leakage and to protect computer systems from "threats" by improper actions of a third party or attacker.

Remote Access Trojans are one of the most infected malwares that can cause data leakage, data disclosure to the wrong party. Although antivirus scanners and intrusion detection tools have been developed to catch them, malware authors use an advanced way like encryption, polymorphism and concealment strategy in order to avoid the detection tools. New form of detection approach is being researched in order to catch up the strategy that malware authors create.

In this thesis, a new approach that can overcome obfuscation techniques like encryption is proposed and implemented. It can uncover the variants of Remote Access Trojans and unknown Remote Access Trojans because it uses the general features extracted from packet header. Features are effective to obtain the highest accuracy, least False Negative Rate and least False Positive Rate.

The contribution of this research work is using the number of packets for extracting features, and effective features to detect Remote Access Trojans in the early stage are proposed and implemented. Machine learning methods with effective features give better result than deep packet inspection. This detection technique depends on

behavior of RATs and each behavior that is combined with 7 features uncovers the different types of Remote Access Trojans.

8.1 Advantages

Although Remote Access Trojans are different in name, appearance, and how they are developed, they have the same behavior and the same objectives. Therefore, this detection approach can detect the variant behavior of Remote Access Trojans and new Remote Access Trojans.

As it is a network-based approach, it can overcome the limitations of host-based methods. It does not need to be installed in each host; it does not depend on operating system platform. Moreover, this approach can deal with some limitations of network-based system like using encryption techniques to avoid detection, extremely big signature database, and time-consuming work to create signatures. It could uncover both known and unknown Remote Access Trojans since encryption techniques cannot camouflage the behavior of RATs. In addition, the behaviors of RATs are not as much as signatures in deep packet inspection techniques as each behavior can uncover one or more RATs. In deep packet inspection, one or more signatures have to be created for each attack.

Although different approaches are applied and features are defined differently in previous works, they cannot reduce False Negative Rate and False Positive Rate effectively and their accuracy is not significantly high. Early stage detection is also an important factor in detecting malware. Data disclosure can be controlled as early as possible if Remote Access Trojans are detected in the early stage of infection.

Supervised machine learning methods – Naïve Bayes, Decision Trees, Random Forests and AdaBoost perform the classification well. With optimized parameters, False Negative Rate can be reduced to 0.0 by Decision trees, Random Forests and AdaBoost, and False Positive Rate to 0.007 by Decision Trees and Random Forests for 300 normal instances and 300 Remote Access Trojans instances.

8.2 Limitation

In this research, the proposed approach is developed in five main parts: capturing network traffic, preprocessing and extracting features based on both packet interval time and first twenty packets, selecting features based on information gain, selecting features based on the difference of features that are calculated manually, and

classification. This approach relies on the network behavior features that uses TCP protocol. Capturing network traffic is done by Wireshark and, features are extracted manually. The command and control traffic of Remote Access Trojans can only be uncovered.

The values of features are extracted depending on the number of packets and first twenty packets are used in this thesis. It is a rare case that there are less than twenty packets in the communication between two hosts. If it occurs, it is not suitable for this approach. First twenty means that starting from SYN to twentieth packets, not from SYN/ACK or ACK or PSH, ACK.

Remote Access Trojans are run on virtual environment and there may be some variation on real environment. Three different ratios of normal and RAT instances are applied for classification: (1) 150:10, (2) 300:10, and (3) 300:300. As there are large amount of normal application traffic and small amount of malicious traffic in the real world, both normal and malicious traffic should be increased. Ten types of normal applications and ten Remote Access Trojans are applied in the experiment. There are many normal applications besides these ten applications and there are many popular Remote Access Trojans besides ten Remote Access Trojans that used in the experiment. The more normal applications and Remote Access trojans, the better the approach.

8.3 Future Work

It is expected that the number of RATs samples and normal applications will be increased in future work in order to achieve comparable classification accuracy, FPR and FNR for different variants of Remote Access Trojans and normal applications, and to develop detection system for Remote Access Trojans in production environments with effective feature set and no overhead. The amount of normal and RAT traffic will be increased to get a better approach.

The command and control communication of other malware such as other types of Trojans and botnet can be extended in future work. Real-time RAT detection system can be developed by two aspects: (1) the first twenty packets to collect, (2) seven features are simple and effective to detect Remote Access Trojans as early as possible.

Wireshark is used to capture network traffic in this thesis. There are other network capturing tools that are available to capture network traffic and these tools can also be used to analyze network traffic. Command line tool like tcpdump may also be useful for analyzing network traffic.

New features are also possible from the packet header information not only for Remote Access Trojans but also for other types of malware like Bot or DDOS. The command and control communication of Advanced Persistent Threats can also be applied in this approach and, its result will be useful for further research.

AUTHOR'S PUBLICATIONS

[p1] Khin Swe Yin, May Aye Khine, Effective Features for Detection of Remote Access Trojans, 15th International Conference on Computer Applications (ICCA), Yangon, Myanmar, 16th - 17th February, 2017. Pg 328-334.

[p2] Khin Swe Yin, May Aye Khine, Network Behavioral Features for Detecting Remote Access Trojans in the Early Stage, 6th International Conference on Network, Communication and Computing (ICNCC), Kunming, China, 8th – 10th December, 2017. Pg 92-96.

[p3] Khin Swe Yin, May Aye Khine, Ensemble Learning for Detecting Remote Access Trojans, 11th International Conference on Future Computer and Communication (ICFCC), Yangon, Myanmar, 27th February – 1st March, 2019, Pg 42-47.

[p4] Khin Swe Yin, May Aye Khine, Optimal Remote Access Trojans Detection Based on Network Behavior, International Journal of Electrical and Computer Engineering (IJECE), Indonesia, Volume 9, No.3, June, 2019, Pg 2177-2184.

BIBLIOGRAPHY

[1] T. Abou-assaleh, N. Cercone, V. Ke, and R. Sweidan, “N-gram-based Detection of New Malicious Code,” no. 1, 2004.

[2] A. Ahmed, E. Elhadi, M. A. Maarof and A. H. Osman, “Malware Detection Based on Hybrid Signature Behavior Application Programming Interface Call Graph Information Assurance and Security Research Group.” Journal, A., Sciences, A., & Publications, S., Faculty of Computer Science and Information Systems, 9(3), pp. 283-288,2012.

[3] “Assessing Outbound Traffic to Uncover Advanced Persistent Threat”, SANS Technology Institute, 2011.

[4] J. Aycock, 2006, Computer Viruses and Malware, Advances in Information Security, Springer.

[5] Z. Bazrafshan, H. Hashemi, S. Mehdi Hazrati Fard, A. Hamzeh, “A survey on heuristic malware detection techniques”, 2013 5th Conference on Information and Knowledge Technology (IKT), Department of Computer Science and Engineering, Shiraz University, Shiraz Iran.

[6] J. Bergeron, M. Debbabi, J. Desharnais, M. M. Erhioui, and N. Tawbi, “Static detection of malicious code in executable programs.” Int. J. of Req. Eng., 2001.

[7] D. Bilar, “OpCodes as predictor for malware,” International Journal of Electronic Security and Digital Forensics, vol. 1, no. 2, p. 156, 2007.

[8] D. Bisson, "KilerRat spying software takes njrat to the next level," [Online]. Available: https://www.grahamcluley.com/kilerrat-spyware-rat/, [Accessed: Sept. 22, 2019].

[9] L. E. O. Breiman, “Random forests,” Machine Learning, vol/issue: 45(1), pp. 5-32, 2001.

[10] J. Brownlee, “An Introduction to Feature Selection”, [Online]. Available: https://machinelearningmastery.com/an-introduction-to-feature-selection/, [Accessed: July.24, 2019].

[11] J. Brownlee, “Supervised and Unsupervised Machine Learning Algorithms”, [Online]. Available: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms, [Accessed: July.23, 2019].

[12] D. Bruschi, L. Martignoni and M. Monga “Detecting self-mutating malware using control-flow graph matching,” In: Büschkes, R. and Laskov, P. (eds) Detection of Intrusions and Malware & Vulnerability Assessment, volume 4064 of LNCS, pp 129–143. Springer, Berlin. 2006.

[13] J. Casad, “Sams Teach Yourself TCP/IP in 24 Hours”, 5th edition, 2012.

[14] P. Chen, L. Desmet, C. Huygens. (2014), A Study on Advanced Persistent Threats. In: De Decker B., Zúquete A. (eds) Communications and Multimedia Security. CMS 2014. Lecture Notes in Computer Science, vol 8735. Springer, Berlin, Heidelberg.

[15] B. Cogswell and M. Russinovich, “Rootkit revealer v1. 71,” Rootkit detection tool by Microsoft, 2006.

[16] “Cybergate”, [Online]. Available: https://sinister.ly/Thread-Cybergate-RAT-Tutorial-For-Beginners-best-version-ever, [Accessed: Sept. 22, 2019].

[17] B. Dang, A. Gazet, E. Bachaalany, and S. Josse, “Practical Reverse Engineering: x86, mx64, ARM, Windows Kernel, Reversing Tools, and Obfuscation”, Wiley, 111 River Street, Hoboken, NJ, USA, 1st edition, 2014.

[18] “Decision Trees”, [Online]. Available: https://scikit-learn.org/stable/modules/tree.html, [Accessed: July.23, 2019].

[19] M. Eskandari and S. Hashemi “Metamorphic malware detection using control flow graph mining”. International Journal of Compute Science Network Security,pp 1-6 ,2011.

[20] FIREEYE, [Online]. Available: https://www.fireeye.com/blog/threat-research/2014/02/xtremerat-nuisance-or-threat.html, [Accessed: Sept. 22, 2019].

[21] S. Gadhiya, K. Bhavsar, “Techniques for Malware Analysis”, Volume 3, Issue 4, April 2013, International Journal of Advanced Research in Computer Science and Software Engineering.

[22] E. Gandotra, D. Bansal, S. Sofat, “Malware Analysis and Classification: A Survey”, India Journal of Information Security, 2014; 56-64.

[23] G. B. S. Gerald, J. Tesauro, Jeffrey O. Kephart, “Neural Network for Computer Virus Recognition.” IEEE Expert, 1996.

[24] HACKTOHELL, "Setting up Cerberus RAT (Remote Administration tool)," [Online]. Available: http://www.hacktohell.org/2011/05/setting-up-cerberus-ratremote.html, [Accessed: Sept. 22, 2019].

[25] P. Harrington, Machine Learning in Action, 2012, pp-129-148.

[26] U. Hodeghatta Rao and U.Nayak, The InfoSec Handbook, An Introduction to Information Security, 2014.

[27] S. Hofmeyr, S. Forrest, and A. Somayaji, “Intrusion detection using sequences of system calls.” Journal of Computer Security, pp. 151–180, 1998.

[28] G. Jacob, H. Debar, and E. Filiol, “Behavioral detection of malware: from a survey towards an established taxonomy,” Journal in Computer Virology, pp. 251–266, 2008.

[29] P. Jalote, "An Integrated Approach to Software Engineering", Springer, New York, NY, 2005.

[30] K. Jeong and H. Lee, “Code graph for malware detection. In Information Networking.” ICOIN. International Conference on, Jan 2008.

[31] D. Jiang, and K. Omote, “A RAT Detection Method Based on Network Behavior of the Communication’s Early Stage”, The Institute of Electronic, Information and Communication Engineers (IEICE) Trans.Fundamental, 2016; vol. E99-A (1):145-153.

[32] W. Jinlong, G. Haidong, and X. Yixin, 2015. “Closed-loop Feedback Trojan Detection TechniqueBased on Hierarchical Model”, in Proceedings of Joint International Mechanical, Electronic and Information Technology Conference, Chongqing, China, 2015; 240-243.

[33] KALPA, “Introduction to Malware”, [Online]. Available: “http://securityresearch.in/index.php/projects/malware_lab/introduction-to-malware/8/”, 2011, [Accessed: June 23, 2017].

[34] R. Kaur and M. Singh, “A Hybrid real-time zero-day attack detection and analysis system”, I.J. Computer Network and Information Security, 2015; 19-31.

[35] S. Khandelwal, [Online]. Available: " Popular Remote access Trojan njRAT fuels Middle East Cyber Crime”, https://thehackernews.com/2014/03/popular-remote-access-trojan-njrat.html, [Accessed: Sept. 22, 2019].

[36] K. Kim and B. R. Moon, “Malware detection based on dependency graph using hybrid genetic algorithm.” In Proceedings of the 12th annual conference on Genetic and evolutionary computation, July 07-11, 2010.

[37] E. Kirda, C. Kruegel, G. Banks, G. Vigna, and R. Kemmerer, “Behavior-based spyware detection,” in Usenix Security Symposium, 2006.

[38] J. Lee, K. Jeong, and H. Lee, “Detecting metamorphic malwares using code graphs” In Proceedings of the ACM Symposium on Applied Computing, ser. New York, NY, USA: ACM, pp. 1970-1977, 2010.

[39] S. Li, X. Yun, Y. Zhang, J. Xiao, and Y. Wang, “A General Framework of Trojan Communication Detection Based on Network Traces”, in IEEE Seventh International Conference on Networking, Architecture, and Storage, 2012; 49-58.

[40] Y. Liang, G. Peng, H. Zhang, and Y. Wang,” An Unknown Trojan Detection Method Based on Software Network Behavior”, in Wuhan University Journal of Natural Sciences, 2013; vol.18(5):369-376.

[41] “List of TCP and UDP port numbers”, [Online]. Available: https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers, [Accessed: July.24, 2019].

[42] Y. Lu, S. Din, C. Zheng and B.Gao “Using multi-feature and classifier ensembles to improve malware detection”. Journal of CCIT 39(2), 57–72. 2010.

[43] U. Malik, “Random Forests Algorithm with Python and Scikit-learn”, [Online]. Available: https://stackabuse.com/random-forest-algorithm-with-python-and-scikit-learn/, [Accessed: July.23, 2019].

[44] “Malware Analysis - Dark Comet RAT”, [Online]. Available: https://www.contextis.com/en/blog/malware-analysis-dark-comet-rat, [Accessed: July.24, 2019].

[45] T. McCabe, "A complexity measure", IEEE Transactions on Software Engineering SE-2(4): 308–320, 1976.

[46] T. Meskauskas, [Online]. Available: https://www.pcrisk.com/removal-guides/14729-imminent-monitor-rat, [Accessed: Sept. 22, 2019].

[47] T. M. Mitchell, “Machine Learning, Decision Trees Learning,” pp. 52-76, 1997.

[48] T. M. Mitchell, “Machine Learning, Bayesian Learning,” pp. 154-178, 1997.

[49] A. Navlani , “AdaBoost Classifier in Python”, [Online]. Available: https://www.datacamp.com/community/tutorials/adaboost-classifier-python, [Accessed: July.24, 2019].

[50] A. Navlani, “Naïve Bayes Classification using Scikit-learn”, [Online]. Available: https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn, [Accessed: July.24, 2019].

[51] A. Navlani, “Understanding Random Forests Classifiers in Python”, [Online]. Available: https://www.datacamp.com/community/tutorials/random-forests-classifier-python, [Accessed: July.23, 2019].

[52] T. Nguyen, and G. Armitage, 2008, “A survey of techniques for internet traffic classification using machine learning, Communications Surveys & Tutorials”, IEEE, vol. 10, no. 4, 56–76.

[53] D. Orenstein, “Application Programming Interface (API),” Quick Study: Application Programming Interface (API), 2000.

[54] “Pandora RAT”, [Online]. Available: https://doublecodes.blogspot.com/2018/04/pandora-rat-22-beta_16.html, [Accessed: Sept. 22, 2019].

[55] C. Peng, H. Long and F. Ding, “Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy,” In Proceedings of the “IEEE Transactions on Pattern Analysis and Machine Intelligence”, 2005.

[56] “Proofpoint Threat Report: Banking Trojans and downloaders dominate malware while threat actors double down on proven techniques in the third quarter of 2018”, [Online]. Available: https://www.proofpoint.com/us/threat-insight/post/proofpoint-threat-report-banking-trojans-and-downloaders-dominate-malware-while, [Accessed: July.24, 2019].

[57] B. Qu, Z. Zhang, L. Guo, and D. Meng, “On accuracy of early traffic classification”, In Proceedings of the “IEEE Seventh International Conference on Networking, Architecture, and Storage”, 2012, 348-354.

[58] S. Ranveer, S. Hiray, “Comparative Analysis of Feature Extraction Methods of Malware Detection”, International Journal of Computer Applications (0975 8887), 2015; vol.120(5):1-7.

[59] S. RAY, “Six Easy Steps to Learn Naive Bayes Algorithm with codes in Python and R”, [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/, [Accessed: July.23, 2019].

[60] A. Riley, "Babylon RAT Raises the Bar in Malware Multi-tasking," [Online]. Available: https://cofense.com/babylon-rat-raises-bar-malware-multi-tasking/, [Accessed: Sept. 22, 2019].

[61] I.A. Saeed, A. Selamat, Ali M. A. Abuagoub, “A Survey on Malware and Malware Detection Systems”, International Journal of Computer Applications, 2013; vol.67(16):25-31.

[62] C. Sanders, “Practical Packet Analysis using Wireshark to solve real-world Network Problems”, 2011.

[63] I. Santos, F. Brezo, J. Nieves, and Y. Penya, “Idea: OpCode-sequence based malware detection,”, Engineering Secure Software and System, 2010.

[64] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, “OpCode sequences as representation of executables for data-mining-based unknown malware detection,” Information Sciences, Aug. 2011.

[65] R. Sekar, M. Bendre, P. Bollineni, and D. Dhurjati, “A Fast Automaton-Based Approach for Detecting Anomalous Program Behaviors.” In IEEE Symposium on Security and Privacy,2001.

[66] “Service Name and Transport Protocol Port Number Registry”, [Online]. Available: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml, [Accessed: Sept. 22, 2019].

[67] U. Shamir, “The 7 Most Common RATS in Use Today”, [Online]. Available: https://www.darkreading.com/perimeter/the-7-most-common-rats-in-use-today-/a/d-id/1321965? [Accessed: July.24, 2019].

[68] W. Snedecor and W. Cochran, “Statistical Methods”, 8th ed. Iowa City, IA: Iowa State Univ. Press, 1989.

[69] “Spygate”, [Online]. Available: https://hacktoolstuff.blogspot.com/2016/10/spygate-rat-32-cracked-full-and-final.html, [Accessed: Sept. 22, 2019].

[70] R. Stewart, [Online]. Available: https://cyware.com/news/xtreme-rat-a-deep-insight-into-the-remote-access-trojans-high-profile-attacks-14dea04b, [Accessed: Sept. 22, 2019].

[71] A. H. Sung, J. Xu, P. Chavez, and S. Mukkamala, “Static Analyzer of Vicious Executables”, In Proceedings of the “20th Annual Computer Security Applications Conference”, 2004, pp. 326-334.

[72] L. Tan, "The Worst Case Execution Time Tool Challenge", The External Test, Technical report, 2006.

[73] “Targeted attacks”, [Online]. Available: https://www.trendmicro.com/vinfo/us/security/definition/targeted-attacks, [Accessed: July.20, 2019].

[74] “TCP Relative Sequence Numbers”, [Online]. Available: https://wiki.wireshark.org/TCP_Relative_Sequence_Numbers, [Accessed: July.23, 2019].

[75] “Top 10 Malware January 2018”, [Online]. Available: https://www.cisecurity.org/blog/top-10-malware-january-2018/, [Accessed: July.24, 2019].

[76] “Trojan Horse (Basics)”, [Online]. Available: http://www.hackers-team.com/2012/01/trojan-horse-basics.html, [Accessed: July.24, 2019].

[77] “Trojans”, [Online]. Available: https://www.kaspersky.com/resource-center/threats/trojans, [Accessed: July.24,2019].

[78] Y. Wang, D. Beck, B. Vo, R. Roussev, and C. Verbowski, “Detecting stealth software with strider ghostbuster,” in Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on. IEEE, 2005, pp. 368–377.

[79] “What is a port number (Logical Port)?”, [Online]. Available: https://averma82.blogspot.com/2013/05/what-is-portnumber-logical-port.html, [Accessed: July.24, 2019].

[80] “What is a Trojan”, [Online]. Available: https://us.norton.com/internetsecurity-malware-what-is-a-trojan.html, [Accessed: July.24, 2019].

[81] “Xena RAT”, [Online]. Available: https://hacktoolstuff.blogspot.com/2016/10/xena-rat-20-cracked-full-version.html, [Accessed: Sept. 22, 2019]

[82] Y. Ye, T. Li, Q. Jiang, and Y. Wang, “CIMDS: adapting postprocessing techniques of associative classification for malware detection,” IEEE Trans. Syst., Man, Cybern. C, vol. 40, no. 3, pp. 298-307, 2010.

[83] Y. Ye, T. Li, K. Huang, Q. Jiang and Y. Chen, “Hierarchical associative classifier (HAC) for malware detection from the large and imbalanced gray list”. Journal of Intelligent Information Systems, 35(1), pp.1-20. 2010.

[84] Y. Ye, T. Li, S. Zhu, W. Zhuang, E.Tas, U. Gupta and M. Abdulhayoglu, “Combinig File Content and File Relations for Cloud Based Malware Detection”, Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, 2011.

[85] TF. Yen, M.K. Reiter, 2008, “Traffic Aggregation for Malware Detection. In: Zamboni D. (eds) Detection of Intrusions and Malware, and Vulnerability Assessment”, DIMVA 2008, Lecture Notes in Computer Science, vol5137. Springer, Berlin, Heidelberg.

[86] J. Zico Kolter and M. A. Maloof, “Learning to Detect and Classify Malicious Executables in the Wild,” vol. 7, pp. 2721–2744, 2006.

[87] J. Zico Kolter and M. A. Maloof, “Learning to detect malicious executables in the wild.” in roc of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006.

ACRONYMS

Acc	Accuracy
ANN	Artificial Neural Network
API	Application Programming Interface
ARP	Address Resolution Protocol
ASCII	American Standard Code for Information Interchange
CFG	Control Flow Graph
CT scans	Computed Tomography scans
DHCP	Dynamic Host Configuration Protocol
DOS	Disk Operating System
DT	Decision Trees
FDDI	Fiber Distributed Data Interface
FNR	False Negative Rate
FPR	False Positive Rate
FTP	File Transfer Protocol
GUI	Graphical User Interface
HIDS	Host-based Intrusion Detection System
HTTP	Hypertext Transfer Protocol
IANA	Internet Assign Number Authority
IM	Instant Messaging
IP	Internet Protocol
IPSec	Internet Protocol Security
IPX	Internetwork Packet Exchange
JPEG	Joint Photographic Experts Group
MBR	Master Boot Record
MD5	Message-Digest algorithm 5
MIDI	Musical Instrument Digital Interface
MPEG	Moving Picture Experts Group
MRI	Magnetic Resonance Imaging
MSN	An instant messaging software operated by Microsoft
NAT	Network Address Translation
NB	Naïve Bayes
NIDS	Network-based Intrusion Detection System

NWLink	NetWare Link
OpCode	Operational Code
OSI	Open Systems Interconnection
PC	Personal Computer
P2P	Peer to Peer
QQ	An instant messaging software operated by TENCENT
RAT	Remote Access Trojan
RF	Random Forests
SAP	Session Announcement Protocol
SDP	Session Description Protocol
SMTP	Simple Mail Transfer Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VBScript	Virtual Basic Script
WAN	Wide Area Network
WTF	Weighted Term Frequency

APPENDIX A

CONFIGURATION AND PREPROCESSING

The experiment is implemented with VMware workstation 10.0.3. Host operating system is Windows 7, 64-bit, Guest operating system is Windows 7, 32-bit. Wireshark Version 2.6.3 is used for capturing network traffic. On the attacker side, a RAT building tool is installed and then a Trojan file is built called server.exe that is combined with a legitimate file and distributed through email or a link that offers a useful and legitimate software. The server.exe file is installed on the victim's computer and then the victim's PC connects back to the attacker. As it is outbound connection, the victim's firewall does not block this kind of connection and, the command and control communication between the attacker and the victim flows easily. At that time, Wireshark is capturing network traffic on the victim's PC and the captured file is saved as pcapng format.

Preprocessing

To find the RAT traffic in pcapng file and the followings processes have to be done: as an example, in the case of njRAT, open a Wireshark file that is saved as pcapng format, click Statistics on the menu bar, choose Conversation. The Conversation window will appear. On the TCP tab, check the IP addresses and port number :1177 that is port number assigned by attacker. The interaction between two IP addresses using this port number -1177, is the command and control communication between the attacker and the victim. There will be more than one communication between the attacker and the victim. The attacker always uses port-1177 to connect with the victim although the victim uses dynamic port that is assigned on a per request basis. The dynamic port numbers are in the range from 49152 through 65535. In Figure 1, there are four communications on the port – 1177. They are described below.

No	IP address	port	IP address	port
1	192.168.0.137	49160	192.168.0.133	1177
2	192.168.0.137	49167	192.168.0.133	1177
3	192.168.0.137	49168	192.168.0.133	1177
4	192.168.0.137	49169	192.168.0.133	1177

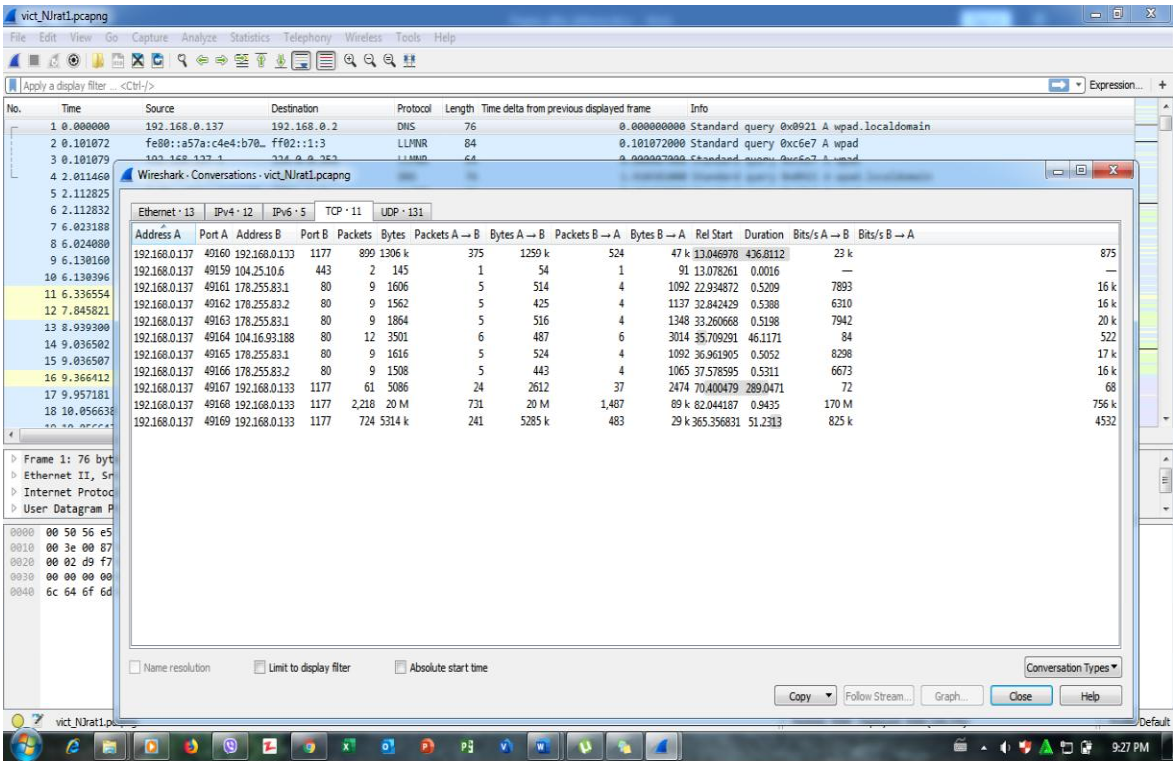

Figure 1: The Conversation window

As njRAT uses reverse connection, the victim firstly connects back to the attacker. The attacker uses port -1177 and thus, 192.168.0.133 is the attacker's IP address and 192.168.0.137 is the victim's IP address. As the smaller port number is used for the first connection and the larger one is assigned for later one, 49160 is the very first connection that exists between the victim and the attacker. Later connections occur when the attacker gives command such as downloading files or keylogging and so on. Therefore, 49160 is the port number that should be taken care of for extracting features.

In order to filter the connection between the attacker and the victim based on port 49160, tcp.port == 49160 is typed and then Enter is pressed in filter box of Wireshark. Only the interactions between them based on port 49160 are filtered and appeared in Wireshark. It is shown in Figure 2.

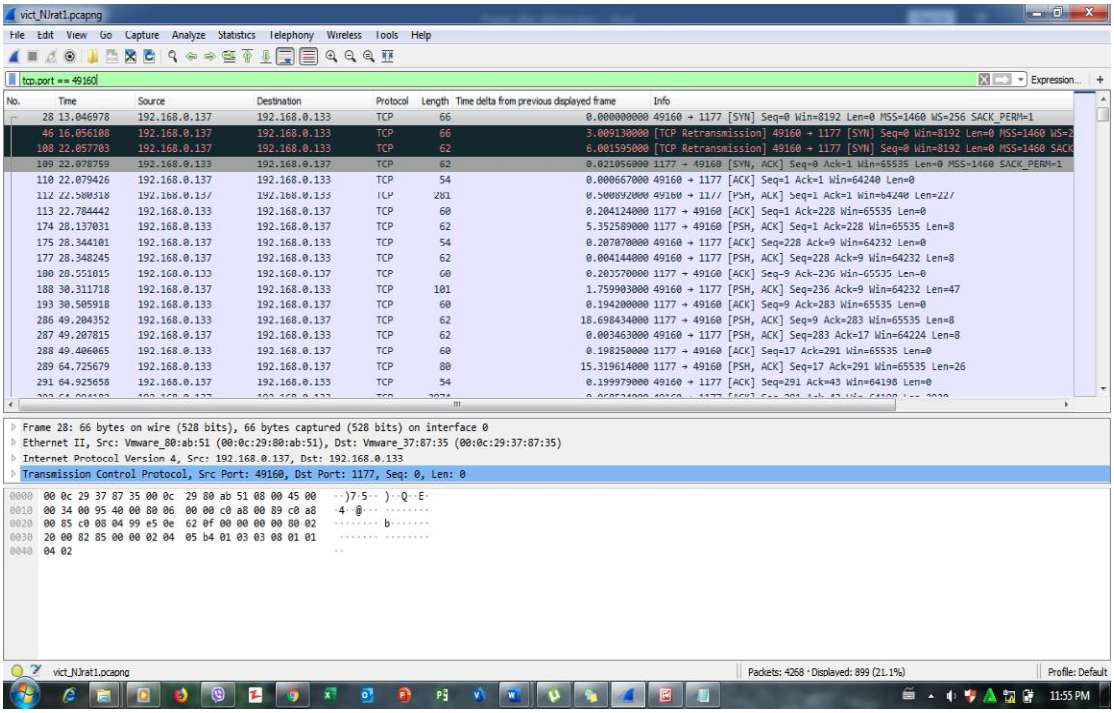

Figure 2: Filtered traffic

If features are extracted depending on packet interval time, “Time delta from previous displayed frame” column have to be checked. This column shows packet interval time between packets.

Features are extracted depending on first twenty packets in the experiment. To extract the first twenty packets,

(1) Right click on the first packet, select Mark/unmark packet

(2) Do the same until twentieth packet,

(3) Then, click File menu, choose Export Specified Packets

(4) Give filename

(5) Select Marked packets in Packet Range, 20 is shown In Marked packets

(6) Save the file

How to mark and save the file are shown in Figure 3 and Figure 4. A pcapng file that includes only first twenty packets is obtained and it is shown in Figure 5.

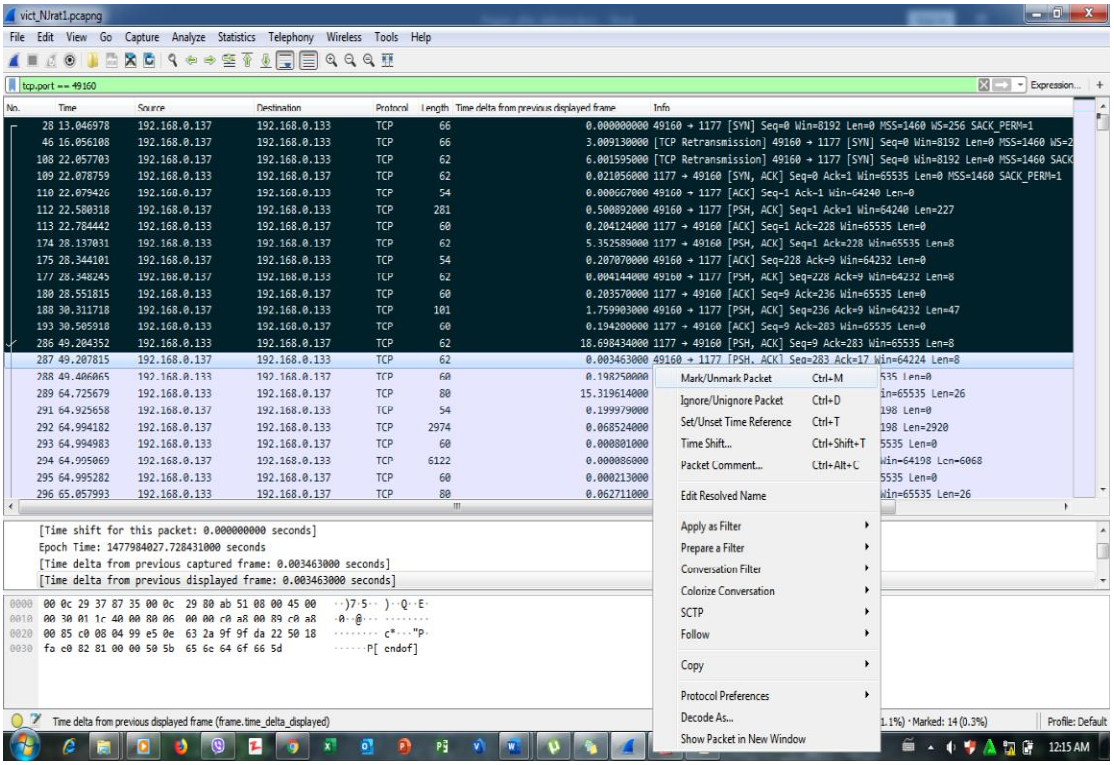

Figure 3: Marking packets

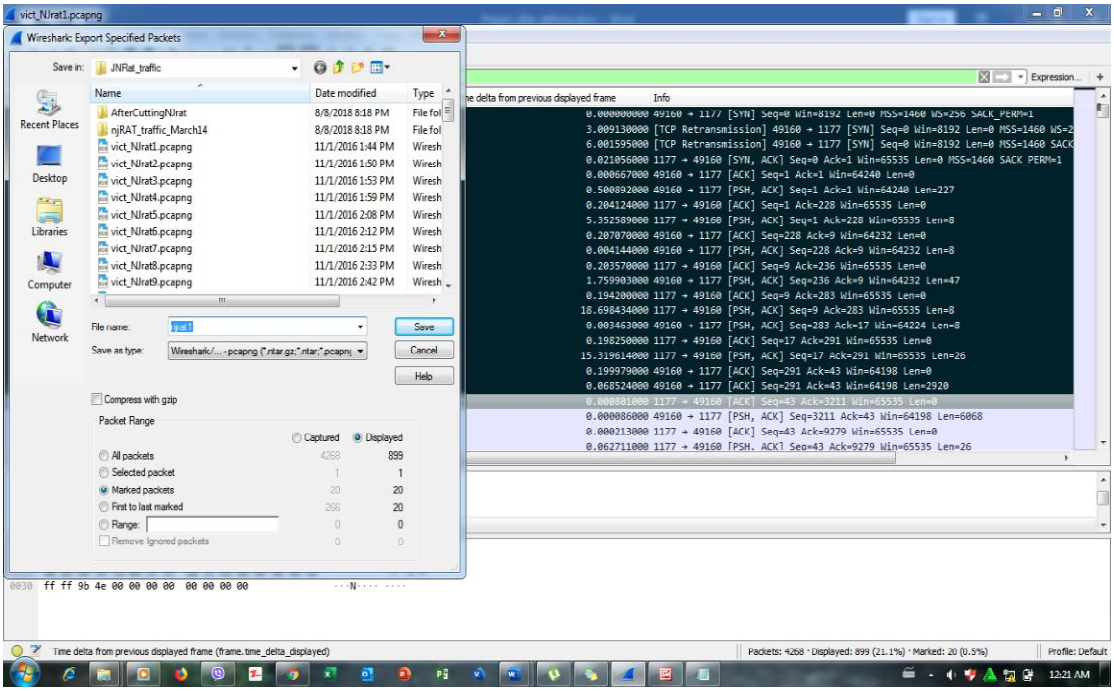

Figure 4: Saving file

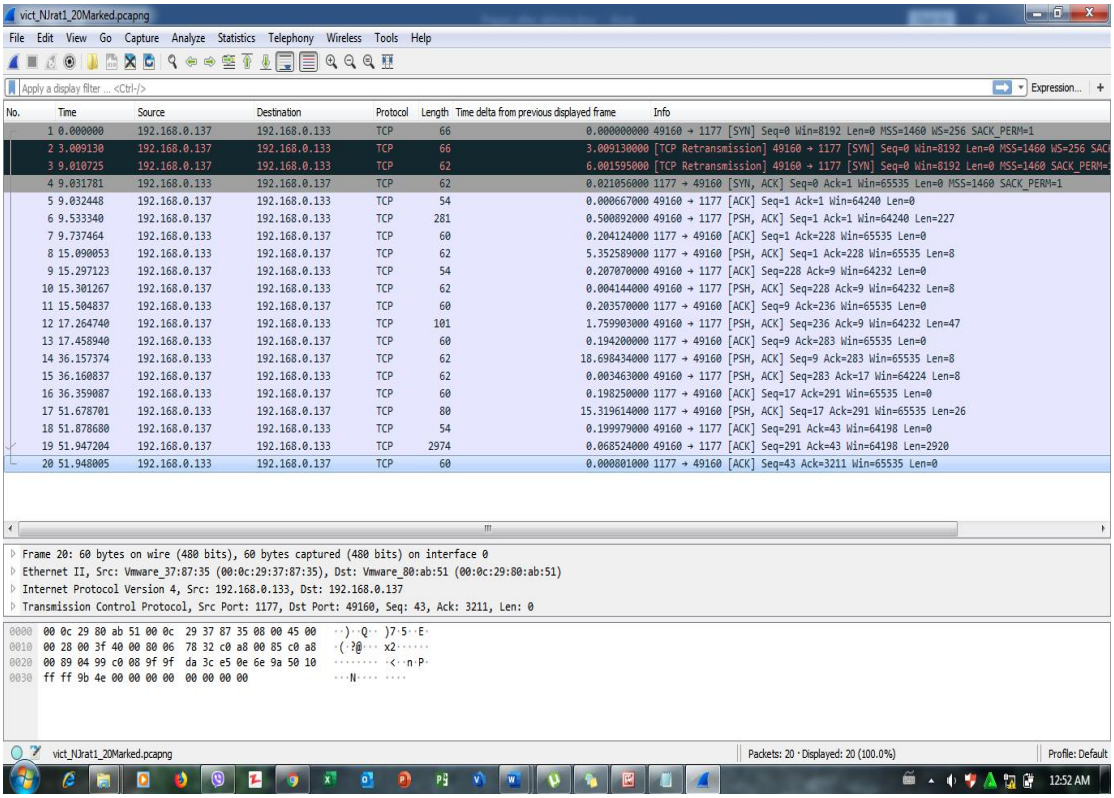

Figure 5: A file that includes only first twenty packets

The value of outbound data byte and inbound data byte is obtained by checking the twentieth packets in Wireshark. According to the Figure 5, outbound data byte is 3211, inbound data byte is 43.

Then, click Statistics on the menu bar, choose Conversations. The Conversations box appears and the value of other features are obtained from this box. It is shown in Figure 6.

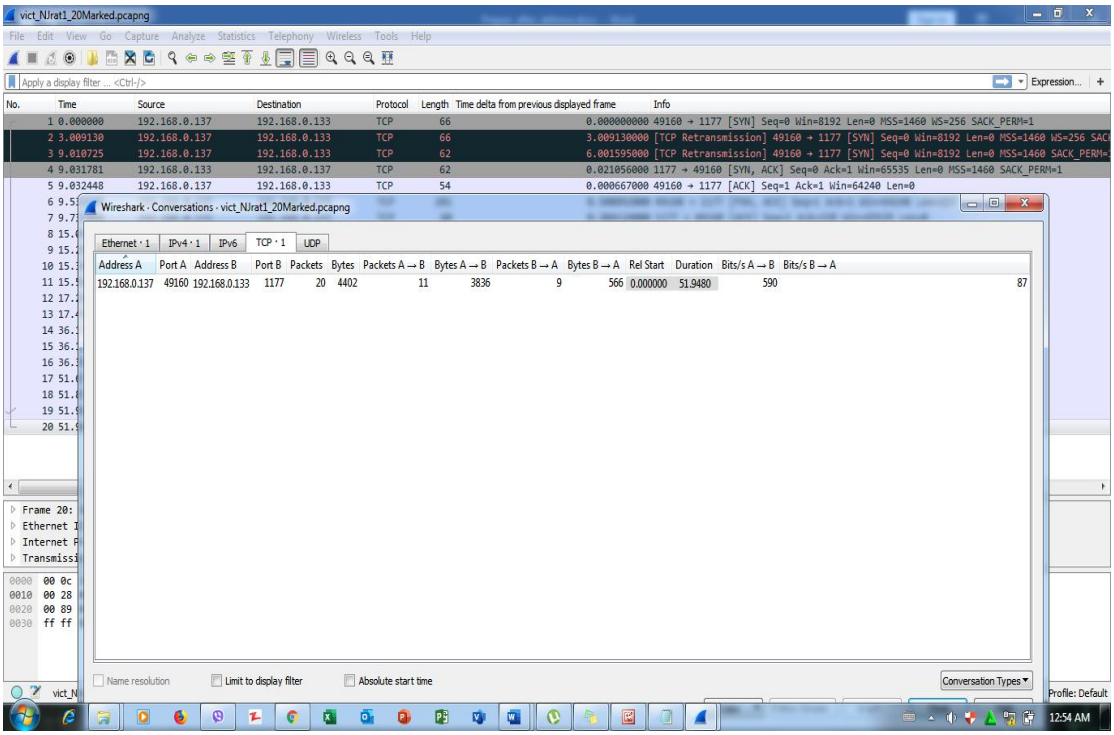

Figure 6: The Conversation window of first twenty packets

According to this Figure 6, outbound number of packets is 11, inbound number of packets is 9. Duration is 51.9480. Based on these basic features: OutByte, InByte, InPac, OutPac, other features are calculated. InByteByInPac = InByte/InPac, OutByteByOutPac = OutByte/OutPac, OutPacByInPac = OutPac/InPac, OutByteByInByte = OutByte/InByte. An instance that includes seven features are obtained in this way.

(1) OutByte = 3211

(2) InByte = 43

(3) OutByteByOutPac = 291.90909

(4) InByteByInPac = 4.77778

(5) OutPacByInPac = 1.22222

(6) Duration = 51.9480

(7) OutByteByInByte = 74.67442

APPENDIX B

ADDING MORE TROJANS AND RESULTS

Besides ten Remote Access Trojans, more Trojans are added and classification is performed again. Two Remote Access Trojans and one Trojan-downloader, one backdoor Trojan and one banking Trojan are added to the experiment. They are shown in Table1.

Table 1: New Trojans

no	RAT	Trojan-downloader	Backdoor Trojan	Banking Trojan
1	DarkComet	Kelihos	Netwire	TinyZBot
2	Novalite			

Two Remote Access Trojans (RAT) are installed in vmware environment like other RATs. Their network traffic is captured by Wireshark and, 47 instances are extracted from these two RATs.

The other three Trojans are not installed in the experiment. The pcapng files captured by Wireshark of three Trojans are downloaded from a shared web link. Three instances are obtained from three Trojans.

Fifty new instances are obtained from two RATs and three Trojans and they are replaced in the place of 50 instances in the 300 RAT instances that we already have. Then, classification is performed for 300 normal instances and 300 RAT instances in which 300 normal instances are from 10 normal applications and 300 RAT instances are from 15 Trojans.

Results and Discussions

When new Trojans are added, the classification result is a little different from the result of ten RATs and ten normal applications. The classification results are shown in Table 2.

Table 2: Experimental result after adding new Trojans

N300_RAT300	NB			DT			RF			AdaBoost		
	Acc	**FNR**	**FPR**	**Acc**	**FNR**	**FPR**	**Acc**	**FNR**	**FPR**	**Acc**	**FNR**	**FPR**
15 Trojans	53.67	0.927	0.0	99.33	0.007	0.007	99.17	0.007	0.01	99.33	0.007	0.007
10 RATs	89.0	0.2	0.02	99.67	0.0	0.007	99.67	0.0	0.007	99.5	0.0	0.01

Capturing network traffic while running many normal applications in the background is the best way in order to obtain the possible values of features. In this experiment, the malicious RAT traffic is captured by Wireshark while many normal applications like Google, Gmail, Facebook and dropbox are running in the background. An example of twenty instances including DarkComet and Novalite RATs are shown in Table 3. Novalite RAT works like other RATs and its outbound data is much more than inbound data. However, DarkComet is completely different from the others and its inbound data is much more than outbound data like most of normal applications. Therefore, the accuracy that includes new Trojans is a little reduced in Decision Trees, Random Forests and AdaBoost. As the accuracy of Naïve Bayes is reduced from 89% to 53.67%, Naïve Bayes is not suitable for this kind of classification.

Table 3: Twenty instances including DarkComet and Novalite RAT

	OutByte	InByte	InByteByInPac	OutPacByInPac	OutByteByOutPac	Duration	RatioOutByteInByte	Class
1	458	4259	473.2222222	1.22222222	41.63636364	0.155	0.107536981	0
2	2146	281	31.22222222	1.22222222	195.0909091	0.294	7.637010676	0
3	573	4212	468	1.22222222	52.09090909	0.735	0.136039886	0
4	2061	5674	630.4444444	1.22222222	187.3636364	0.97	0.363235812	0
5	462	4787	478.7	1	46.2	92.35	0.096511385	0
6	468	5900	590	1	46.8	4.552	0.079322034	0
7	1081	7069	589.0833333	0.66666666	135.125	0.717	0.152921205	0
8	1013	4297	477.4444444	1.22222222	92.09090909	0.339	0.235745869	0
9	615	4212	468	1.22222222	55.90909091	0.657	0.146011396	0
10	653	4250	425	1	65.3	0.268	0.153647059	0
11	421	17	1.7	1	42.1	15.002	24.76470588	1
12	287	25	2.5	1	28.7	20.012	11.48	1
13	2409	17	1.7	1	240.9	9.718	141.7058824	1
14	253	17	1.7	1	25.3	17.39	14.88235294	1
15	256	33	3	0.818	28.444	29.1667	7.757575758	1 (N)
16	251	33	3	0.818	27.889	27.892	7.606	1(N)
17	343	33	3.3	1	34.3	29.98	10.394	1(N)
18	647	1473087520	147308752	1	64.7	63.149	4.39214E-07	1(DC)
19	823	3933096196	393309619.6	1	82.3	60.823	2.0925E-07	1(DC)
20	733	119	11.9	1	73.3	113.198	6.159663866	1(DC)

0: Normal application, 1: RAT, N: Novalite, DC: DarkComet