Conversation-based P2P Botnet Detection with Decision Fusion

by

Shaojun Zhang

BSc, Southeast University, 2010

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Computer Science

in the Graduate Academic Unit of Computer Science

Supervisor(s):	Ali Ghorbani, Ph.D., Computer Science
Examining Board:	Rodney Cooper, Prof., Computer Science, Chair
	Huajie Zhang, Ph.D., Computer Science
	Brent Petersen, Ph.D., Electrical and Computer Engineering

This thesis is accepted by the

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

January, 2013

Abstract

Botnets have been identified as one of the most dangerous threats through the Internet. A botnet is a collection of compromised computers called zombies or bots controlled by malicious machines called botmasters through the command and control (C&C) channel. Botnets can be used for plenty of malicious behaviours, including DDOS, Spam, stealing sensitive information to name a few, all of which could be very serious threats to parts of the Internet. In this thesis, we propose a peer-to-peer (P2P) botnet detection approach based on 30-second conversation. To the best of our knowledge, this is the first time conversation-based features are used to detect P2P botnets. The features extracted from conversations can differentiate P2P botnet conversations from normal conversations by applying machine learning techniques. Also, feature selection processes are carried out in order to reduce the dimension of the feature vectors. Decision tree (DT) and support vector machine (SVM) are applied to classify the normal conversations and the P2P botnet conversations. Finally, the results from different classifiers are combined based on the probability models in order to get a better result.

Dedication

For my family, who offered me unconditional love and support throughout the course of this thesis.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Dr. Ali Ghorbani for the continuous support of my study and research. His guidance helped me in all the research and writing of this thesis.

Also I would like to thank all my labmates for their help and encouragement.

Last but not the least, I would like to thank my parents for giving birth to me and supporting me throughout my life.

Table of Contents

List of Tables

List of Figures

List of Symbols, Nomenclature or Abbreviations

IRC	Internet Relay Chat
$C\&C$	Command and Control
$P2P$	peer to peer
DT	Decision Tree
SVM	Support Vector Machine
$DDoS$	Distributed Denial of Service
RFL	Ranked Feature List
TPR	True Positive Rate
FPR	False Positive Rate
TNR	True Negative Rate
FNR	False Negative Rate
$ISOT$	Information Security and Object Technology
$LBNL$	Lawrence Berkeley National Lab

Chapter 1

Introduction

1.1 Introduction

With the rapid development of the Internet, many different kinds of cyber-attacks continue to emerge. As the technologies are evolving all the time, attack tools become more and more complicated, yet it is now easier for attackers to conduct attacks. That is why network security has become one of the most important topics.

A botnet first appeared in 1999, an IRC-based botnet called PrettyPark. After decades of development, botnets have become more and more complicated and robust. People have become aware of the threats to the Internet security caused by botnets. So botnet detection, botnet mitigation and some related researches have become hot topics in the area of Internet security.

Although botnets use Trojan or some other malware for propagation, the

command and control communication (C&C) is the main difference between botnets and other kinds of malware, namely, a one-to-many control mechanism. That is why botnet can be the most dangerous threat in the Internet world. Feily et al. pointed out that with the C&C mechanism, the bots are not physically controlled by a bot master. They can be located everywhere with different time zones, laws, languages, etc. These differences make botnet detection more difficult [1].

This thesis presents a conversation-based botnet detection approach with decision fusion. We first propose the conversation-based features for detection and then carry out a feature selection process to find suitable feature subsets for each machine learning technique. Finally, a decision fusion algorithm is used to improve the performance of our proposed peer-to-peer (P2P) botnet detection approach based on the results of each classification model.

1.2 Summary of thesis contributions

Nowadays, botnets, especially P2P botnets have become one of the most dangerous threats in the field of network security. Numerous botnet detection approaches have been put forward by researchers to detect botnets, including IRC-based botnets, HTTP-based botnets, P2P botnets, etc. Detection of centralized botnets (IRC- and HTTP-based botnets) is relatively easy because of the central point of failure. In this thesis, we propose a conversation-based P2P botnet detection with decision fusion in order to de-

tect P2P botnets effectively. This thesis makes the following contributions.

1. The proposed P2P botnet detection solution uses conversation-based features. To the best of our knowledge, this is the first time that conversation-based features are used in P2P botnet detection. As the features are derived from the headers of the network packets, they do not rely on the packets' payloads. With this characteristic, our detection approach will not be affected by traffic encryption. Moreover, the proposed approach can also be used to detect unknown P2P botnets which will be demonstrated in Chapter 4;

2. Two supervised machine learning techniques: decision tree (DT) and support vector machine (SVM) are used to detect unknown P2P botnets. Both of them have produced a high true positive rate (TPR) and a low false positive rate (FPR), which confirm that our proposed detection method works well in detecting P2P botnets.

3. A feature selection process is proposed in order to reduce the dimension of the feature vectors for DT and SVM. The original conversation-based feature vector has 16 dimensions. However, we believe that not all the features in the feature vector are needed when training the classifiers. Thus, different feature selection algorithms are used to remove the less effective features from the original feature vector.

4. Decision fusion makes the use of the probability models generated by DT and SVM. To the best of our knowledge, this is the first time

in network security that a probabilistic method is used to combine the results of multiple machine learning techniques. The M-branch smoothing method is used to built a probability model based on DT. For SVM, the distance from an instance to the hyperplane is employed to calculate the probability of the instance being classified to a specific class. Decision fusion based on probabilities allows us to improve the detection results by considering the results from DT and SVM together.

1.3 Thesis organization

The rest of the thesis is organized as follows:

Chapter 2 gives an overview of botnets. The life cycle of a botnet is described and the hazards of botnets are also listed which gives the general information of botnets. The previous work about botnet detection and related research using machine learning are reviewed as well.

In Chapter 3, our proposed approach is discussed in detail. First, the overview of the proposed P2P botnet detection method is described. Then, the feature selection processes for DT and SVM are presented separately. The details of probability models for DT and SVM are shown as well. Finally, the decision fusion algorithm is displayed in order to improve our detection approach.

Chapter 4 describes the datasets and metrics we used in our experiments. Also, the procedures of the experiments are discussed as well. Different

feature selection methods are evaluated with the experimental datasets. The best feature selection methods are then used to select features for DT and SVM. With the feature subsets, both DT and SVM have competitive results compared with the original feature vector. Finally, the results of decision fusion are compared with the results from individual DT or individual SVM in terms of the metrics.

Finally, Chapter 5 concludes the thesis and outlines some possible further directions to improve or extend our work.

Chapter 2

Background information and literature review

2.1 Life cycle of botnet

Knowing the life cycle of a botnet can help people in understanding how a botnet works. Different researchers have different opinions about the botnet life cycle. They have different definitions, but, the basic structures are the same. In this chapter, the life cycle of a botnet is described as shown in Figure 2.1.

2.1.1 Spread

This is the initial phase of a botnet's life cycle. The attacker spreads malware to find the vulnerabilities in remote hosts. If a vulnerability is found, then

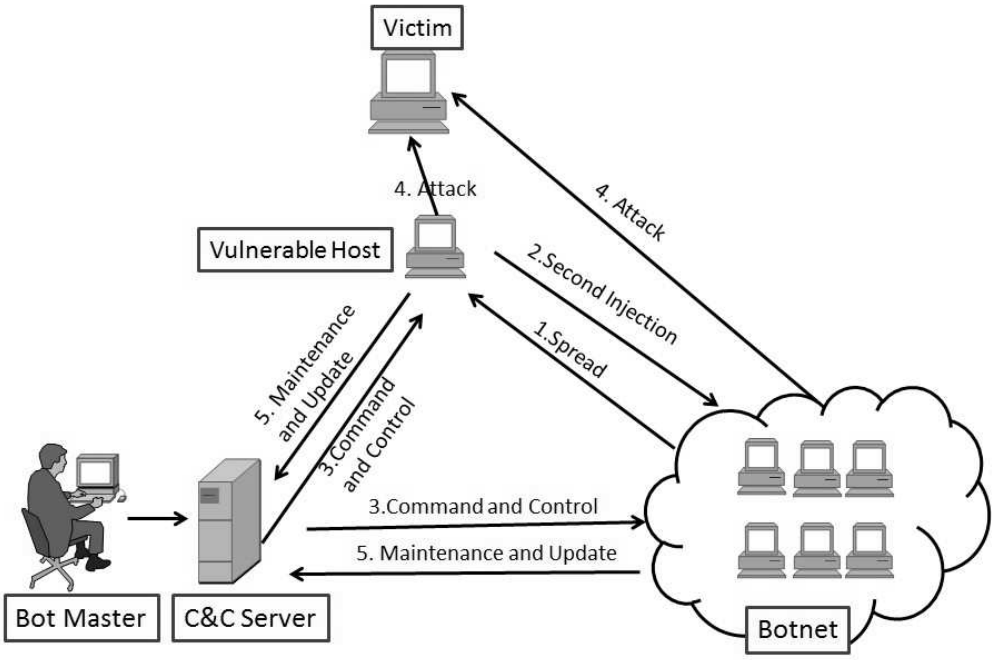


Figure 2.1: Typical life cycle of a botnet

the malware copies itself to that host and executes it for further spreading. Another way to do this is sending email with an attractive title and the body containing some words to trick the recipient to click the link or the attachment in the email. By using this social engineering technique, some recipients are tricked and the malicious binaries are installed on their machines. Holz et al. gives a detailed explanation [2].

2.1.2 Secondary injection

The hosts execute malicious scripts in bot binaries. As a result, these scripts download the latest bot binaries from the Internet automatically. It makes sure that the bot binary on a host is up to date. At this point, a host becomes a bot in the current botnet.

2.1.3 Command and Control

- For IRC-based and HTTP-based botnets, a bot establishes a connection to the C&C server through the C&C mechanism. Depending on this, a bot receives commands or orders from the attacker through this connection and responses back. There are two ways to spread a command in centralized botnets, namely push-style and pull-style. IRC-based botnets belong to the former category (See Figure 2.2), while HTTP-based botnets belong to the latter one (See Figure 2.3).

- For P2P-based botnets, C&C connections are established between dif-

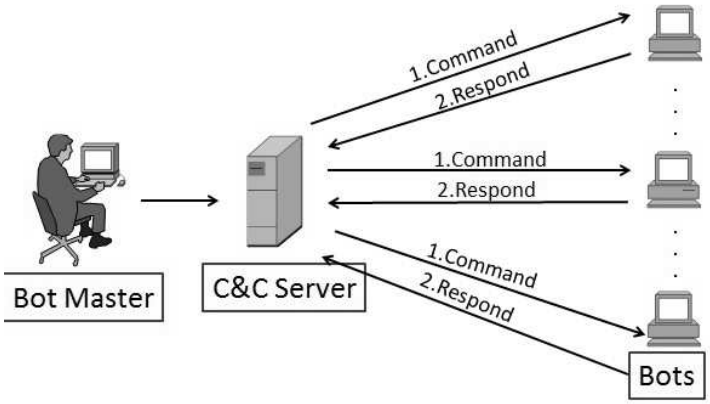


Figure 2.2: Push style

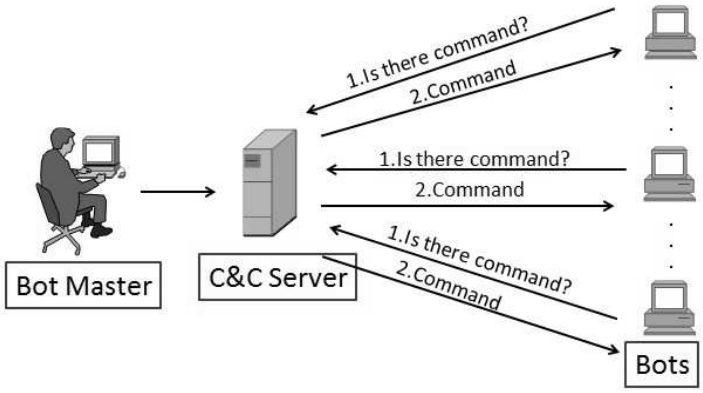


Figure 2.3: Pull style

ferent bots. The bots in a P2P-based botnet not only send commands, but also receive commands from other bots depending on the current status of the network. The attacker sends a command to some of the bots. After receiving a command from the attacker, these bots will forward the command to the remaining bots in the same botnet.

2.1.4 Attack

This phase is the most important phase in a botnet life cycle. It is also the purpose of a botnet. Botnets typically have the following malicious behaviours: 1) Spam: by using Spam, attackers can expand their botnets or spread some harmful viruses; 2) DDoS (Distributed Denial of Service): attackers can control a large number of bots to launch DDoS to a victim through C&C mechanism; 3) Stealing information: attackers can steal some personal and sensitive information from bot hosts; and 4) Make use of resources: attackers using the resources of bot hosts without permission to launch attacks or to do some unauthorized activities.

2.1.5 Update and maintenance

- Update

  As more and more researchers start to work on botnet detection, the attackers also need to upgrade their tools against the detection techniques. Bot masters upgrade the bot binaries by distributing new ver-

sions of the bot binaries in this stage. This can make the botnet always robust to new detection techniques.

- Maintenance

  After a bot host joins a botnet, it has to communicate with other bots to make others and the bot master aware that it is online. Therefore, the bot master always knows the status of the botnet.

2.2 Classification of botnets

It can be seen from the life cycle of a botnet that the C&C mechanism is the core part of a botnet. Depending on the C&C mechanism, a bot can communicate with the bot master or other bots, which makes botnets different from traditional trojans, worms and other malware. The bot master can issue commands to thousands of bots to launch an attack. Cooke et al. pointed out that from the perspective of C&C mechanism, botnets can be classified into three different categories, namely, centralized, distributed, and random. This paper also first introduced P2P botnet into academia. Researchers always come up with new ideas for C&C communication [3]. In 2010, Wang et al. combined both centralized C&C and distributed C&C and developed a new botnet, which has the advantages of both centralized and distributed botnets [4]. Dittrich and Dietrich [5] divide botnets into four categories based on the development of botnets: namely IRC, HTTP, P2P and hybrid botnets.

Based on C&C communications and the previously published papers, we classify botnets into three categories as follows.

2.2.1 Centralized botnets

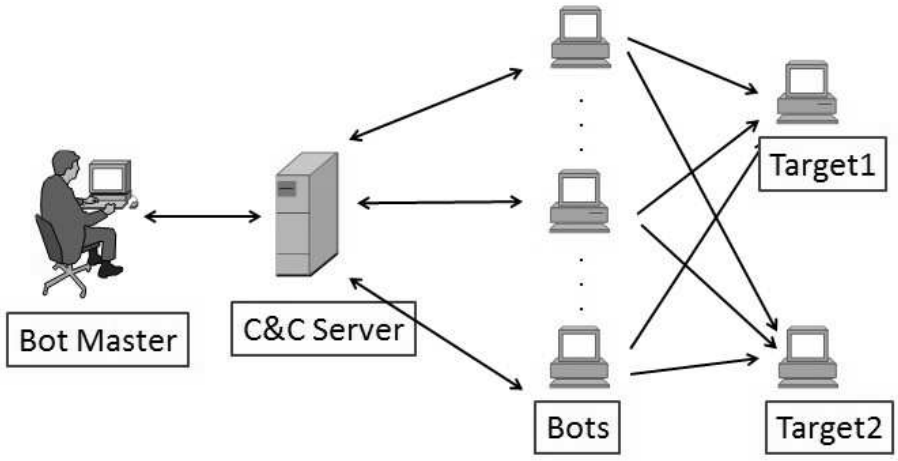


Figure 2.4: Centralized botnet

A centralized botnet usually contains a C&C server, which is used by its bot master to send commands to the bots as shown in Figure 2.4. All the commands are sent by the bot master, there is no command transmitted among the bots. Bots ask for new commands from the C&C server periodically or the C&C server sends commands periodically. As the C&C server is the core of a centralized botnet, a botnet will be disabled as soon as it is found by a detector. However, because the commands are transmitted fast and the latency is low, it is still being widely used in the wild. IRC-based botnets and HTTP-based botnets are both centralized botnets.

- IRC-based botnets

  IRC-based botnets play an important role in the development of botnets. It is the originator of all kinds of botnets. The first botnet was based on IRC. At the beginning, people developed IRC-based bots to help people when they were chatting. But, the idea was used by some malicious developers, and then the earliest IRC-based botnet emerged. It could be used by an attacker to control the hosts in the network and to carry out malicious activities to some victim targets. It is the foundation of today's botnets. Because the communication characteristics were easily detected and the data was not encrypted, it is relatively easier to detect IRC-based botnets compared to other kinds of botnets. However, it is also easier to construct IRC-based botnets, which makes them popular among attackers.

  For the detection of IRC-based botnets, researchers already did a great deal of theoretical studies of them and also a large number of experiments. So, there are some effective approaches to detect IRC-based botnets. Goebel et al. extracted an IRC bot nickname pattern and used it as a signature to detect an IRC-based botnet [6]. Lu et al. first classified the network traffic into IRC and non-IRC classes based on the traffic characteristics, and then applied K-means, merged X-means and unmerged X-means to the IRC traffic to classify the IRC traffic into two clusters, namely, normal IRC traffic and botnet IRC traffic. By comparing these three methods, they reached the conclusion that merged

X-means has an outstanding performance in clustering the traffic. This method is less effective when the data is encrypted [7].

- HTTP-based botnets

  Hackers started to seek new C&C mechanisms for botnets. As a result, HTTP-based botnets came into being. Different from IRC-based botnets, the traffic of HTTP-based botnets can be hidden in normal HTTP traffic. So, it can evade the interdiction of firewalls so as to achieve the purpose of avoiding detection. The spam model of the Rustock rootkit used encrypted HTTP in C&C communications to make it difficult to be detected [8]. BlackEnergy is another typical HTTP-based botnet [9]. Detecting HTTP-based botnets is harder than IRC-based botnets, because HTTP-based botnets can hide their traffic better.

2.2.2 P2P botnets

Detecting or mitigating a P2P botnet is very challenging. Different from centralized botnets, C&C servers are not used in P2P botnets. All the communication relies on the P2P protocols used by P2P botnets. Attackers can send a command through any bot in a botnet and this bot will forward the command until all the other bots receive the command. As shown in Figure 2.5, a bot in a P2P botnet acts as a server as well as a client, which means it can both send commands and receive commands. There is a large portion of P2P traffic in the Internet, so the traffic coming from P2P botnets

is hidden in the benign P2P traffic just as HTTP-based botnets are. This makes it hard to find P2P botnet traffic among the traffic with normal P2P applications.

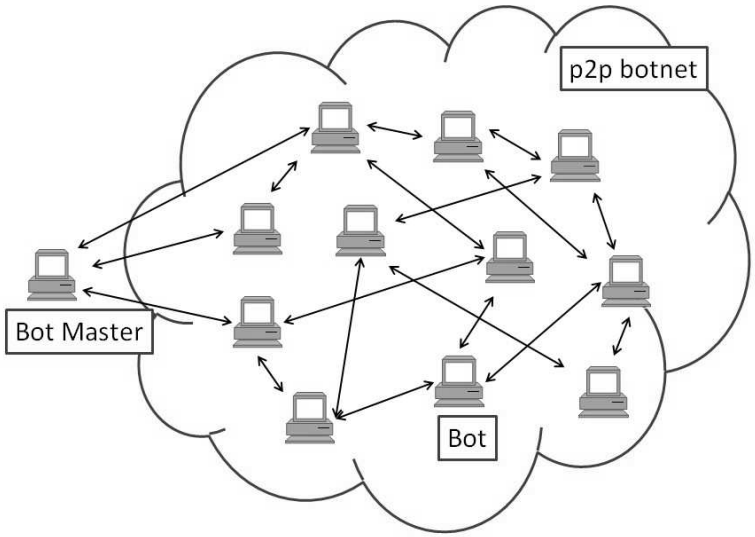


Figure 2.5: P2P botnet

Even if you find some bots in a P2P botnet, it still can be a problem to mitigate an entire botnet, because a P2P botnet can work well when some bots are removed from it. Although P2P botnets are more complicated than centralized botnets, they are more robust. Thus, they are the most popular botnets in recent years.

2.2.3 Hybrid botnets

In 2007, Wang et al. proposed an advanced P2P botnet. It combines the characteristics of centralized botnets and P2P botnets. There are two types

of bots in their advanced P2P botnet. They are servent bots and client bots. The servent bots construct a P2P network and each servent bot acts as a C&C server which can be found in centralized botnets. So it is much more robust than centralized because there is no central point of failure in their proposed P2P botnet. What is more, it can relay the commands faster than P2P botnets with the client bots [4]. In 2008, Dong et al. predicted new features of the next generation P2P botnet, and came up with a new advanced P2P botnet based on the discussion of existing botnets [10]. In their proposed botnet, the bots are split into different clusters; each cluster contains both server bots and client bots. Foreign links are used for the communication between different clusters. Server bots have public IP addresses and client bots just have private addresses. It means that all the client bots should connect to at least one server bot to receive commands, and server bots are used to maintain the overlay network. Moreover, they use a shared information box for robustness. Dong and his colleagues compared their proposed botnet with a Kademlia based botnet [11] (such as Peacomm) and a hybrid P2P botnet by using different metrics (such as bot degree, network synchronization performance and robustness).

2.3 The hazards of botnets

With botnets, it is much easier to launch attacks which are more dangerous than the traditional attacks. Lanelli et al. claimed that botnets can be used

for different kinds of cybercrimes [12]. The Honeynet project also listed several types of attacks including: DDoS, Spam, keylogging, installing malware, etc [13]. Some common attacks are listed below.

2.3.1 DDoS

DDoS is one of the most dangerous threats caused by botnets. The incredible number of bots in a botnet makes DDoS very destructive. Attackers make use of a botnet to control the bots to send requests or send data to a victim system in order to take it down. Some large botnets can even be harmful to ISPs (Internet Service Provider). Table 2.1 shows the number of DDoS targets increased from 2006 to 2009 [14]; we can speculate that it is because of the emergence of P2P botnets. From 2009, it began to decrease during the next 3 years. However, the number of attacks was still large. A reasonable explanation is that researchers have paid more attention to botnets, especially P2P botnets. They started to investigate botnets and came up with different detection or mitigation methods which caused the decline. Table 2.2 shows us that the papers published since 2009 mainly concentrate on how to detect P2P botnets. This table comes from a survey of more than 100 papers carried out at the Information Security Center of eXcellence (ISCX), University of New Brunswick, 2012.

Table 2.1: DDoS

Year	Unique C&C	Target Count	Unique Targets
2006	414	50650	25953
2007	848	35566	15755
2008	618	202678	21312
2009	590	7058221	10991
2010	430	1545208	13757
2011	322	275459	5327

Table 2.2: Paper statistics

	2011		2010		2009		2004-2008		All years	
Applied	12	55%	16	46%	26	57%	7	26%	61	47%
Theoretical	6	27%	12	34%	6	13%	11	41%	35	27%
System	3	14%	2	6%	8	17%	6	22%	19	15%
Review	1	5%	5	14%	6	13%	3	11%	15	12%
Total	22	100%	35	100%	46	100%	27	100%	130	100%

2.3.2 Spam

It is a good choice for attackers to use a botnet as a tool to send spam. The bots in a botnet send spam to target addresses after they receive the attack commands from the bot master. So it is difficult to find out the real attacker and blacklisting becomes useless. Ramachandram et al. mentioned in their paper that most spam are generated by botnets [15]. John et al. used a platform called botlab to monitor approximately 200,000 email addresses at the University of Washington. On average, these addresses can receive 2.5 million emails every day and over 90% of these were classified as spam. Almost 80% of the spam were caused by 6 different botnets [16]. The result is quite consistent with Ramachandran's work.

2.3.3 Stealing information

A bot master can make use of bot binaries to gather sensitive information from bot hosts by using techniques like screen capture, reading log files, key logging, etc. For example, SDBot is a botnet that uses a keylogging binary to steal users' personal information. That can be sold to others to perform illegal activities [17].

2.3.4 Embezzling resources

An attacker can control part of a bot host's resources to do illegal activities for some purposes. For example, the bots can be controlled to visit some websites frequently so as to increase the traffic of the website without the users' permissions. They can also be used to cast spurious votes.

2.4 Detection approaches

People have worked on botnet detection for several years and many detection approaches were proposed by researchers. We classify these approaches into the following five categories described in the next subsections.

2.4.1 Network signature-based botnet detection approaches

These approaches can be used to detect botnets by using the content of network traffic. Typically, these approaches extract some signatures from the content of some known botnets' traffic. And then these signatures are used to identify the malicious traffic coming from the same botnet. Unfortunately, these kinds of approaches cannot be used to detect an unknown botnet. For example, in the early stage, Nugache used port 8. At that time, this signature could be used to detect Nugache. Obviously, this method will not work anymore, because today's botnets are much more complicated. Lu et al. applied n-grams to the content of packets in order to detect IRC-based botnets [7]. To classify the packets into different known applications, they extract the temporal-frequent characteristics from the packets. These known applications' temporal-frequent characteristics were labelled and trained by a novel decision tree. For a specific application domain, namely IRC application in their work, different clustering algorithms were applied to detect IRC botnets depending on the standard deviation of all the 256 payload features. As this is a content-based method, so it can be only applied to unencrypted botnets. Geobel et al. used regular expressions to represent all suspicious IRC nicknames. Then n-grams were used to analyze and evaluate the nicknames to determine if a nickname belonged to a bot [6].

2.4.2 Network behaviour-based botnet detection approaches

From another perspective, there are some differences between the behaviour of botnet traffic and the behaviour of normal traffic. Compared to network signature-based botnet detection approaches, these kinds of methods are more general, and do not restrict their applications to unencrypted botnets. To some degree, these approaches can also detect unknown botnets, so they can be used widely. However, the true positive rate is relatively low compared with network signature-based detection approaches. Strayer et al. introduced a network behaviour-based botnet detection method which can detect IRC-based botnets. In their processes, the first step is to filter out non-IRC flows, because these flows cannot be a part of IRC-based botnet traffic. Then, they use machine learning techniques to cluster the remaining flows into different applications. The flows which are clustered as chat-like applications are then passed to a correlator stage. In this phase, the flows which have a similar pattern are grouped together. Then, all the flows in each group are passed to topology analysis to determine if they have a common controller. Finally, analysts determine if the flows belong to a botnet or not based on the results of topology analysis [18]. As the final determinations are made by human experts, it is a limitation of this detection approach.

2.4.3 Honeypot-based botnet detection approaches

Honeypot is a technology that simulates a host to make an attacker believe it is a bot in his botnet. Although it receives the commands published by the bot master, it will not do anything bad to any target. A honeypot acts as a bot in a botnet and it receives all the commands that the bot master sends. Therefore, it can easily obtain some important information from the bot binary by using reverse engineering or some other techniques. It can also obtain information about how the botnet works, what the commands are, etc. This information can be used to analyze some characteristics about a botnet, which can be used by different botnet detection approaches.

The honeypot approaches cost fewer resources in gathering valuable information about a botnet. It can also collect some information about new attack techniques and their signatures. Honeyd [19] and GenIII [20] are two frequently used honeypots in this field.

The honeypot approaches have some shortcomings. It takes time to collect information and analyze the obtained binaries. What is worse, if an attacker is aware of the existence of a honeypot, the attacker will not send anything to it and the honeypot will receive nothing. Some attackers may even send some fake binaries or commands to the honeypot which could mislead the detector. This could be more dangerous. Freiling et al. used the honeypot approach to collect malwares for analysis [21]. In their experiments, the honeypot found automated malwares in a short time. Gu et al. also gathered two P2P botnets, Storm and Nugache, by using honeypot techniques [22].

2.4.4 Host-based botnet detection approaches

Host-based botnet detection is a relatively straightforward detection method. It treats a bot binary as a virus, Trojan or some other malicious malware. It detects bots in the way that anti-virus software does. It monitors a host's behaviours, recording system events and some other information about the host to see if the host is infected by a bot binary. The information includes register modifications, remote control activities, file deletion, the traffic received by or sent to a host, etc. An alert is triggered when some typical botnet behaviours are detected on a host. Masud et al. used tcpdump to record the log of packets sent and received through the network interface. At the same time, exedump was used to record the time when a program started and ended. With these records, they applied machine learning techniques like SVM and decision tree to build classification models by correlating the records. These classification models were used for further botnet detection [23]. Al-Hammadi et al. also proposed a novel host-based botnet detection approach. Their experiments showed that by correlating the behaviours on a host based on a time window, their method worked well in detecting bot hosts. Three factors are considered in their approach: (1) in a time window t, the rate of change of the following fields, namely, DU (destination unreachable), FCA (failed connection attempts) and RST (reset connections). The choice of the three fields is based on the observation of P2P botnets. These fields are strong factors that can indicate if there are bad behaviours on the host; (2) in a time window t, the rate of the change

of the number of packets sent per second; (3) in a time window t, the time between two continuous system calls to send outgoing data such as [(send, send), (sendto, sendto)]. Then, a correlate algorithm was applied to these three factors to detect P2P botnets [24]. Nummipuro et al. concentrated on the behaviours of a host after receiving the commands. They hooked the SST (system service table) to see how the data coming from the Internet affected the function calls of a system and detected the botnet based on the suspicious host behaviours. Behaviour blocking was applied when suspicious behaviours were found; it stopped the suspect system calls [25]. Giroire et al. detected IRC-based botnets based on the connection frequency between a bot and a C&C server. In their opinion, there are only two situations in which a host will stably connect to an end-point for a long period. In one situation, a user visits some websites often (such as news websites, work related websites or entertainment websites), or some servers are contacted by client applications (such as RSS servers, mail servers). All the IPs belonging to this situation are added into a white list and these kinds of connections are not considered in the botnet detection process. In the other situation, the connections between a host and a C&C server are maintained for a long time. Because a bot needs to connect to C&C server to get commands or maintain the connection for a certain period. In the paper, there is a new concept called *persistence* which indicates how stable the connection is. They divide a time window into several tiny time slots. The persistence in a time slot is assigned a 1 if there is a connection between a host and a C&C server, and 0

otherwise. If the sum of all the persistences in a time window is larger than a predefined threshold, an alert arises. However, this method may be useless to detect P2P botnets, because all the connections between different bots in a P2P botnet are changing all the time depending on the current topology of the botnet. The connection between two IPs will not be stable and last for a long time [26].

2.4.5 Hybrid botnet detection approaches

Hybrid botnet detection approaches usually combine two or more previous methods to detect botnets. For instance, Yin et al. proposed a method which combines both the network-based botnet detection approach and the host-based botnet detection approach. At the host level, sensors are deployed on the hosts to obtain exceptions and suspicious processes. On the other hand, a detection system is deployed at the network level. It monitors all the traffic in the entire network to discover abnormal traffic. Finally, it correlates all the data from host and network level to detect P2P botnets. Wang et al. combined three detection approaches [27]. The honeypot-based botnet detection approach, the host-based botnet detection approach and the network-based botnet detection approach are all used in their proposed method.

2.5 Machine learning

Machine learning plays an important role in artificial intelligence. It is widely used in many fields (such as pattern recognition, date mining, medical diagnosis and so on) because of its excellent performance. Machine learning algorithms extract the internal relationship of the data which can be presented by some rules or models for prediction and classification. Following are three common machine learning categories.

2.5.1 Supervised learning

Supervised learning algorithms learn from an existing dataset whose records are labelled. Then the learning result is used to analyze unlabelled records and label the records with reasonable values. In botnet detection, machine learning techniques are used to train with both botnet datasets and normal datasets to generate classifiers (As can be seen in Figure 2.6 and Figure 2.7, different machine techniques generate different classification models. For instance, a decision tree generates a tree which can classify every record into different classes. Whereas, a hyperplane is generated after training with a support vector machine; it has the ability to classify almost all the records into the correct class). Decision tree and SVM are typical supervised learning algorithms.

Strayer et al. propose a method to detect IRC-based botnets with supervised learning algorithms. The whole process has two steps. Three machine

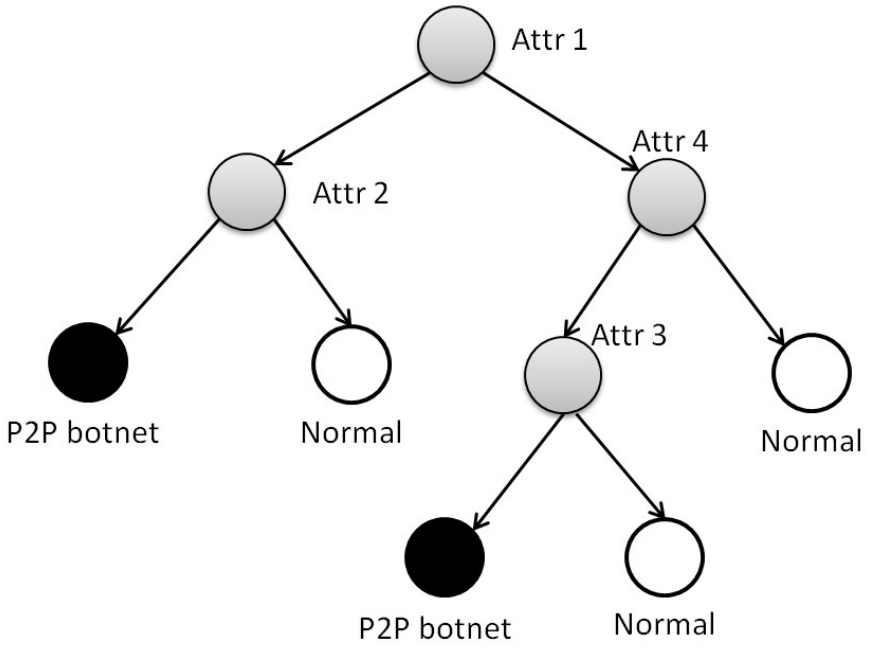


Figure 2.6: Decision tree

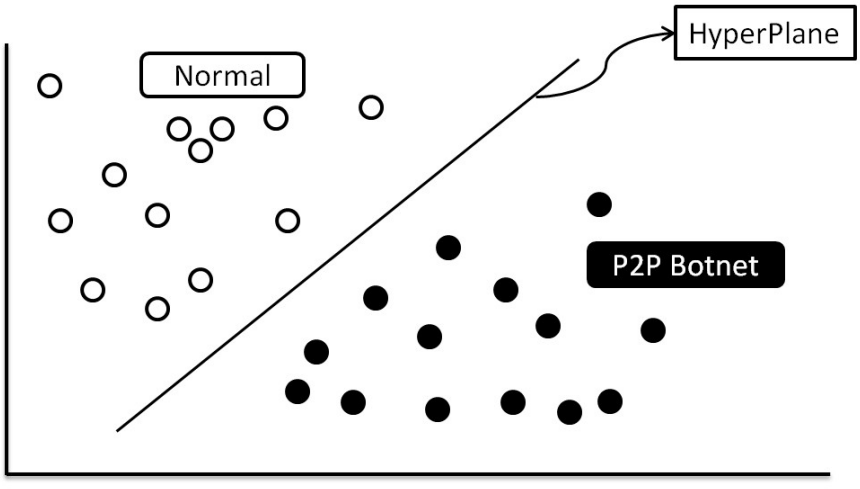


Figure 2.7: Support vector machine

learning techniques are used in each step, namely, decision tree, naïves bayes and bayesian network. In the first step, they used these three techniques to divide the traffic into IRC traffic and non-IRC traffic. The results showed naïve bayes classifier performed best in this step. After filtering out most of non-IRC traffic, the remaining traffic was passed to the following step. In this step, IRC traffic was split into malicious IRC traffic and normal IRC traffic. Different from the first step, the bayesian network had the best performance in this step [28]. Lu et al. applied the decision tree to classify the network traffic into different classes depending on the applications which generate the traffic. Then they analyzed the temporal-frequent characteristics of flows for each application's traffic to differentiate malicious traffic generated by bots and normal traffic created by normal applications. They also applied cluster techniques to IRC traffic: malicious traffic and normal traffic were clustered into two clusters with good performance [29].

2.5.2 Unsupervised learning

In comparison with supervised learning, labels are not needed in the training dataset in unsupervised learning. The purpose of unsupervised machine learning techniques is to divide unlabelled data into different clusters based on some certain metrics. The main idea is, if two data instances are close to each other based on the predefined metrics, these two data instances should be clustered into the same cluster. In the case of a botnet, the records can be classified into two clusters, a normal cluster and a malicious cluster. In

most cases, we have to label the clusters, but, unsupervised learning does not have the ability to tell which cluster should be labelled as malicious. So some characteristics of the clusters are used to label the clusters. For example, "botnet" can be labelled to a cluster if the standard deviation of all the records is smaller than the other cluster. Or we can assign "botnet" to a cluster which has fewer records than the other cluster, because botnet records are much fewer than normal records in real traffic. K-means and X-means are two commonly used unsupervised learning algorithms.

Lu et al. used K-means to split IRC traffic into two clusters [30]. As a bot is preprogrammed, all the bots in the same botnet have similar behaviours. They thought that the traffic generated by bots should be much more stable than the traffic created by human beings. So the authors believed that the standard deviation of temporal-frequent characteristics of flows can tell apart malicious traffic and normal traffic. For two clusters, the cluster with a smaller standard deviation of temporal-frequent characteristics of flows is malicious.

2.5.3 Semi-supervised learning

Semi-supervised learning is a combination of supervised learning and unsupervised learning. For many practical problems, it is very difficult to gain a great deal of data with labels. It also may cost a lot of resources and take a long time. Semi-supervised learning is used to deal with the problem that there is only a small part of the training records that have labels. Mo-

hammad et al. considered that in a real environment, where a large number of data are transmitted with high speed, collecting labelled data could be very difficult. So they used semi-supervised learning to make use of a small amount of labelled data for training with large number of unlabelled data. Their experiments showed the accuracy is good by using the classification model trained by the semi-supervised learning technique they used [31].

2.6 Concluding remarks

This chapter introduced some basic information about botnets and previous research on botnet detection. Botnets can be classified into three types including centralized botnet, P2P botnet and hybrid botnet. They can be used to launch DDoS attacks, send spams, steal information, etc. Some published botnet detection approaches were discussed as well. Most of the studies are based on flow which lead us to extend to conversation. Meanwhile, machine learning techniques are commonly used in a lot of researches. Usually, the machine learning techniques are used independently, so we came up with the idea that combining some of the machine learning techniques with probabilities. The next chapter will present the proposed detection approach based on conversation and machine learning.

Chapter 3

Proposed approach

In chapter 2, a number of botnet detection methods are given. These detection methods and their background information has led us to propose a new P2P botnet detection approach in this thesis.

The P2P botnet detection approach we propose in this thesis is based on 30-second conversations between two IPs (A conversation represents all the traffic transmitted between two IPs with a specific protocol). A conversation between two IPs is split into 30-second phases and each 30-second phase is presented by a 16-dimensional feature vector, which will be explained later. Decision Tree (DT) and Support Vector Machine (SVM) are applied to build classifiers to distinguish the P2P botnet conversations from normal conversations.

Section 3.1 presents an overview of our proposed approach. Section 3.2 discusses all the component in detail, including a feature extractor, a feature

selection component and the classification techniques.

3.1 Overview

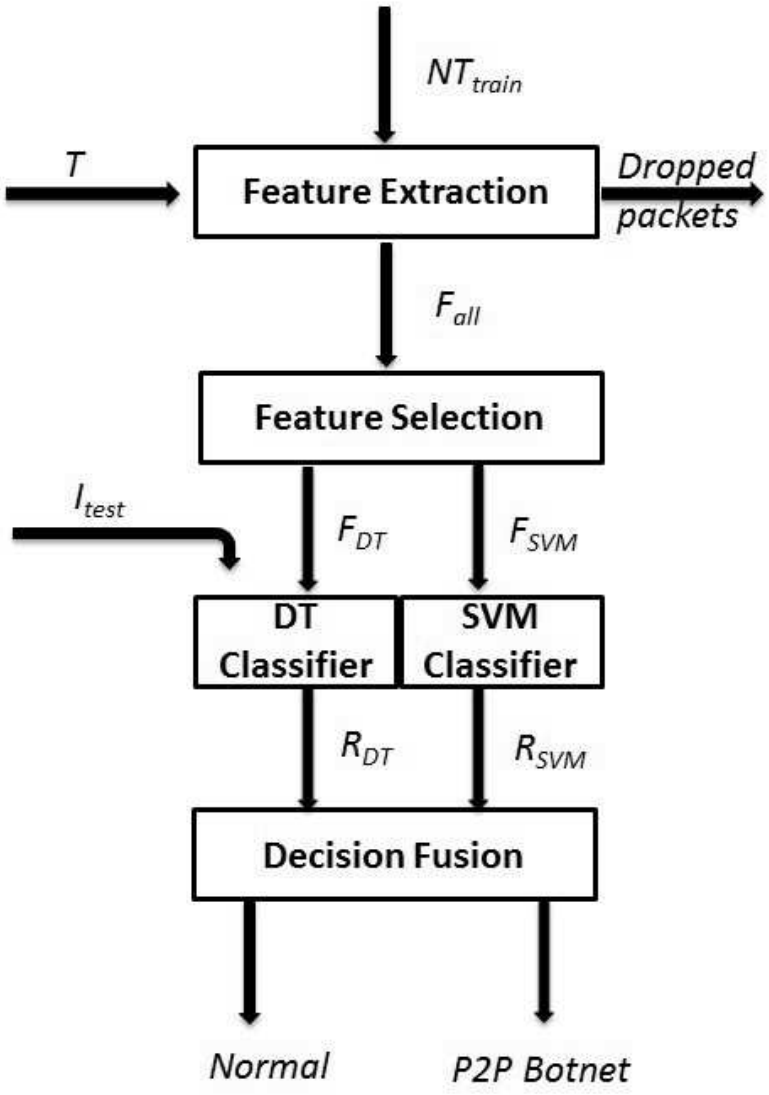


Figure 3.1: Proposed framework

Our proposed P2P botnet detection solution uses a 16-dimensional feature vector to represent a 30-second conversation between two IPs (TCP, UDP, ICMP and some other protocols are all included.) Feature selection processes are applied to reduce the dimension of the feature vectors, consequently reducing the training and test time. The reduced dimensional feature vectors are used to construct classifiers. Finally, based on the probability models in the classifiers, a decision fusion algorithm is used to combine all the results from multi-classifiers in order to make a final decision to distinguish P2P botnet traffic from normal traffic.

Figure 3.1 shows the basic framework and processes of our detection approach. Let NT_{train} represent the network traffic used for training (traffic includes all the traffic captured through the network interface). In the feature extraction component, we set the value of time window to 30 seconds. During the feature extraction phase, some P2P botnet unrelated packets, such as broadcast packets, multicast packets are discarded in order to reduce the processing time. During each time window, all the packets (except for the dropped packets) in a same time window are grouped into different conversations, and feature vectors F_{all} are built from the conversations depending on the headers of the packets. F_{all} represents the 16-dimensional feature vectors which are the input of the feature selection component. Using F_{all} the feature selection component generates optimal feature subsets for DT and SVM (F_{DT} and F_{SVM}), respectively. The goal of the feature selection component is that, by using the reduced dimension feature vectors, the classification

result is either as good as using 16-dimensional feature vector or equally has a competitive result with higher performance.

After the feature selection procedure, all the original 16-dimensional feature vectors need to be reconstructed depending on the F_{DT} and F_{SVM} obtained from the feature selection component. The new training datasets are used to build DT and SVM classifiers. Assume I_{test} is a test instance which needs to be judged as a normal or a P2P botnet instance. Before I_{test} is sent to the classifiers, it must be reconstructed, and two new test instances are generated, namely I_{testDT} and $T_{testSVM}$. These two instances represent the same conversation that are used in DT and SVM, respectively.

The last step requires making a final decision with the results from all of the classifiers. For I_{test}, when the results from all classifiers are the same, there is no doubt that I_{test} can be labelled as the result from one of the classifiers. However, the results of different classifiers may be different. In this case, we need to apply a probability model to each classifier in order to make a reasonable decision to classify I_{test} as a normal or a P2P botnet conversation.

3.2 Components

3.2.1 Feature extraction

In previous studies, a large number of the detection methods used packet signatures or flow features to detect P2P botnets. To the best of our knowledge, conversation-based P2P botnet detection has never been proposed in

the past, and this is the first time that conversation-based feature vectors are used in P2P botnet detection. We believe that our conversation-based P2P botnet detection approach has a number of advantages over signature-based and flow-based detection approaches due to the following reasons.

- Signature-based P2P botnet detection methods have a high degree of detection rate, but they heavily depend on the content of the packets, so we can infer that this kind of detection approaches can only detect unencrypted P2P botnets. It is difficult to detect unknown P2P botnets because the signature for every botnet is different. However, our conversation-based P2P botnet detection method is a totally content-free method. All the features can be obtained from the packets' headers. Therefore, it can detect not only encrypted P2P botnets, but also some related unknown P2P botnet. In other words, our proposed method is more general with a relatively high detection rate.

- Compared to flow-based features used in P2P botnet detection methods, the conversation-based features can reveal more information about the relationship between two connected IPs. As we know, a P2P botnet uses P2P protocol to establish the C&C communication. Therefore, there is a peer-to-peer relationship between two interacting hosts. Our conversation-based P2P botnet detection method considers all traffic between two zombies to generate conversation-based features. However, flow-based features generated by P2P botnet detection methods

can only investigate one-way traffic between two hosts. In other words, it takes a one-sided view of the C&C communication. That is, conversations can reflect the "peer-to-peer" relationship much better than flows can.

For instance, host A and host B send packets to each other. If we apply flow-based detection methods to these two zombies, we will extract two flows that represent the packets from A to B and from B to A, respectively. Actually, there is a chance that these two flows are similar to other "normal flows" (there are too many normal flows). In this situation, there may be a misclassification. However, if we employ conversation to show the C&C communication, the result can be more accurate, because the conversation features can reveal the "peer-to-peer" relationship better.

- Conversation-based P2P botnet detection method uses less memory than flow-based P2P botnet detection methods. As mentioned previously, we need two flows to present the communication pattern. However, a conversation feature vector can also show the pattern. When comparing these two methods, it is evident that the conversation-based P2P botnet detection method uses almost half of the memory that flow-based P2P botnet detection methods consume.

- Unlike the features in previous work [32][33], all features used in our approach are basic features or some combinations of the basic features.

As a result, the time complexity is small, and it consumes fewer system resources.

Table 3.1 shows the 16 features we create for the proposed conversation-based P2P botnet detection approach. These features are generated from a 30-second conversation and is composed into a feature vector (we call it an instance) to represent a 30-second conversation. With the label of the instance (generated from labelled datasets), the feature vector can be expressed as: $<F_1, F_2, \cdots F_{16}, LABEL>$ where F_i represents the i_{th} feature in Table 3.1.

The feature extractor is responsible for converting the network raw packets into a 16-dimensional feature vector for each 30-second conversation. Figure 3.2 shows that, the raw packets from the network are the input of the feature extractor. In this component, the traffic capturer captures packets based on a time window. Then, all packets in the same time window are grouped into different groups. All packets belonging to the same conversation are put into the same group, meaning that the number of the groups is equal to the number of the conversations in a time window. In Chapter 4, we will use this component to generate training datasets and test datasets.

In real life, a small part of the network raw packets do not need to be pushed into the time-based buffer (See Figure 3.3), for these packets (such as broadcast packets, multicast packets) cannot be part of the P2P botnet traffic. Therefore, we can discard these kinds of packets without any operation or analysis, which can reduce not only the processing time, but also the buffer

Table 3.1: Selected network traffic features

Feature	Description
F_1	The total bytes transmitted in a conversation
F_2	The total number of packets transferred in a conversation
F_3	Average length of a packet in a conversation
F_4	The number of the packets whose size are smaller than 146 in a conversation
F_5	The proportion of packets whose size are smaller than 146 in a conversation
F_6	The number of the packets whose size are larger than 146 in a conversation
F_7	The proportion of packets whose size are larger than 146 in a conversation
F_8	The size of the first packet in a conversation
F_9	$ratio1 + \frac{1}{ratio1}$; $ratio1$: The ratio between the number of packets in one direction and the number of packets in the other direction
F_{10}	$ratio2 + \frac{1}{ratio2}$; $ratio2$: The ratio between the number of bytes in one direction and the number of bytes in the other direction
F_{11}	$ratio3 + \frac{1}{ratio3}$; $ratio3$: The ratio between average bytes per packet in one direction and average bytes per packet in the other direction
F_{12}	The difference between the number of packets in one direction and the number of packets in the other direction in a conversation
F_{13}	The proportion of the difference between the number of packets in one direction and the number of packets in the other direction in a conversation $(\frac{feature12}{total\ number\ of\ packets\ in\ a\ conversation})$
F_{14}	The difference between the number of bytes in one direction and the number of bytes in the other direction in a conversation
F_{15}	The proportion of the difference between the number of bytes in one direction and the number of bytes in the other direction in a conversation $(\frac{feature14}{total\ number\ of\ bytes\ in\ a\ conversation})$
F_{16}	The difference between the number of average bytes per packet in one direction and the number of average bytes per packet in the other direction

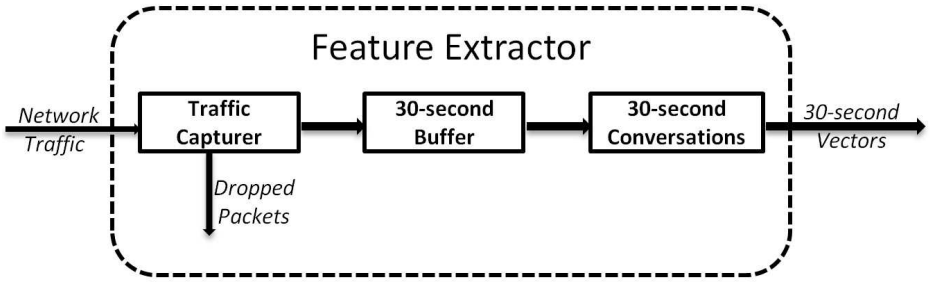


Figure 3.2: Feature extractor

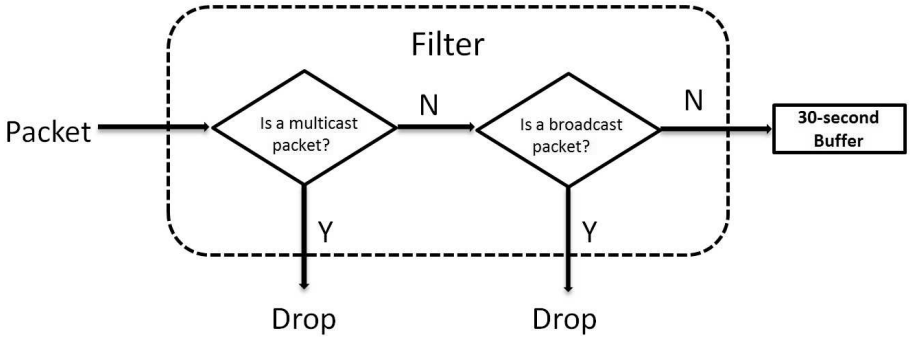


Figure 3.3: Packet filter

usage. As soon as the capturer captures a packet, it sends the raw packet to the buffer which is used to store all the packets in the same time window. Then, these packets are analyzed based on the information in their headers. They are grouped into different conversations, and then a 16-dimensional conversation feature vector with an additional class label is extracted for each conversation. A feature vector with class label is shown below:

$<F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8, F_9, F_{10}, F_{11}, F_{12}, F_{13}, F_{14}, F_{15}, F_{16}, Label>$

We create different threads to deal with the capture process and the converting operation, which ensures these two operations can be done simultaneously so as to avoid losing packets when the converting operation is processed.

3.2.2 Feature selection

We employ 16 features to represent a conversation. However, we believe that the 16 features are not all needed when training the classifiers. There may be some features that will not impact the classification result at all, or perhaps might make the result worse. In this section, we introduce some feature selection methods in order to reduce the dimension of the feature vectors. The output of an individual feature selection algorithm is an ordered feature list which ranks the features based on their own metrics. Finally, we generate a final feature list based on these ranked feature lists. The aim of feature selection is to choose a suitable feature subset for improving classification performance or decreasing the complexity of a classification model without significantly decreasing the performance.

3.2.2.1 Feature selection for decision tree

We applied two different feature selection algorithms for DT, which are based on information gain and information gain ratio separately. Since the construction of a decision tree is based on information entropy, these two algorithms are appropriate to be used in the feature selection process for DT. The following equation is the definition of information entropy in information theory:

$$H(C) = -p\left(C_1\right)\log_2\left(p\left(C_1\right)\right) - p\left(C_2\right)\log_2\left(p\left(C_2\right)\right)\cdots - p\left(C_n\right)\log_2\left(p\left(C_n\right)\right)$$

where $H(C)$ represents the information entropy of C and C_i is a value that C can be assigned to. $p(C_i)$ shows the probability that C_i can be assigned to C.

- Feature selection based on information gain:

  The feature selection method based on information gain evaluates every attribute by using information gain (See equation (3.1)), which shows how much information entropy an attribute can contribute to a system. The features with higher ranks have better ability to split a dataset into different parts, since they contain more information than other features. The features at the bottom of the ranked feature list are not good at separating a dataset, since they are less informative. That is, for a specific feature F, the larger the information gain is, the more information F has. Of course, we prefer to put the top ranked features

into our final feature list and to discard the bottom features. The ideal situation is that a feature has the largest information gain, which means that we can build a classification model just based on this feature.

How to calculate the information gain of a feature F is shown by Equation (3.1):

$$InfoGain(F) = H(C) - H(C|F) \tag{3.1}$$

where C is the target value of a classification model, it can be C_1, C_2, $\cdots$, C_m.

$H(C|F)$ in Equation (3.1) represents the classification model's entropy without feature F. It can be calculated with the following equation:

$$H(C|F) = P_1 * H\,(C|F{=}F_1) + P_2 * H\,(C|F{=}F_2) + \cdots + P_n * H\,(C|F{=}F_n) \tag{3.2}$$

where n is the number of values that can be assigned to feature F;

$$P_i = \frac{\text{number of the instances with the value of } F \text{ is } F_i}{\text{total number of the instances}} \tag{3.3}$$

and

$$H(C|F{=}F_i) = \sum_{j=1}^{m} \big(- p(C{=}C_j|F{=}F_i)\big) * \log_2 \big(p(C{=}C_j|F{=}F_i)\big)$$

$$p(C{=}C_j|F{=}F_i) = \frac{\text{number of instances with } F_i \text{ and } C_j}{\text{number of instance with } F_i}$$

- Feature selection based on information gain ratio:

  The feature selection algorithm based on information gain ratio also evaluates the features depending on the information entropy basically. But, it uses information gain ratio instead of information gain to rank the features. Information gain ratio is a combination of information gain and intrinsic information which can be seen from Equation (3.4). This algorithm ranks the features according to the information gain ratio of each feature. Also, the top features in the ranked feature list are much more important than the bottom features. Clearly, the top features should be in the final feature subset while some features at the end of the ranked list should be discarded.

$$GainRatio(F) = \frac{InfoGain(F)}{SplitInfo(F)} \tag{3.4}$$

  where $SplitInfo(F)$ represents the intrinsic information of feature F.

$$SplitInfo(F) = -\sum_{i=1}^{n} \frac{|F_i|}{|F|} * \log_2\left(\frac{|F_i|}{|F|}\right) \tag{3.5}$$

  where n is the number of values can be assigned to feature F; and $|F_i|$ is the number of the instances with the value of F is F_i.

3.2.2.2 Feature selection for SVM

As with DT, we also applied two feature selection algorithms for SVM. Each feature selection method outputs a ranked feature list as we mentioned in Section 3.2.2.1. These two algorithms are ReliefF and SVM-RFE.

- ReliefF:

  This novel method was proposed by Kononenko in 1994. It is an extension of the Relief algorithm, which was proposed by Kira and Rendell in 1992. It is widely used in feature selection for SVM. The basic idea is that, for an instance and its adjacent instances (which come from the same class), if they have a similar or same value of one feature. Then, we can say it is a good feature. Otherwise it is not. On the other hand, a good feature should make an instance and the adjacent instances (which come from the opposite class) have totally different values as well. On the contrary, if a feature does not have these kinds of characteristics, it is not good enough. ReliefF assigns a feature a large weight if it is a good feature. Finally, all the features are sorted by their weights. Algorithm 1 shows the ReliefF feature selection algorithm [34]. In our case, we only have two different class values. Thus, between line 5 and line 7, there is only one class value which is different with R_i's label.

- SVM-RFE:

  SVM-RFE is short for support vector machine-recursive feature elim-

Algorithm 1 Pseudo code of ReliefF

Input:

the dataset D contains training instances with class value;

Output:

RFL feature ranking from best to worst;

number of features n;

initiate the weight for each feature $W[F] = 0$;
for $i = 1$ to m **do**
  randomly select an instance R_i;
  find k nearest hits H_j;
  for each class $C \neq \text{class}(R_i)$ **do**
    from class C, find k nearest misses $M_j(C)$;
  end for
  for $F = 1$ to n **do**
    $W[F] = W[F] - \sum_{j=1}^{k} \text{diff}(F, R_i, H_j)/(m \cdot k) +$
    $\sum_{C \neq \text{class}(R_i)} [\frac{P(C)}{1-P(\text{class}(R_i))} \sum_{j=1}^{k} \text{diff}(F, R_i, M_j(C))]/(m \cdot k)$
  end for
end for
RFL = features are sorted based on the $W[F]$.
return RFL;

ination. As the name says, it combines SVM and RFE to assess the features. It is an iterative process to eliminate features from the original feature list. In each round, each feature is assigned a weight depending on the attribute ranking criterion, and the feature with the minimum weight is eliminated from the feature list. This process is repeated until all the features are removed from the original feature list. The order of the features that have been removed shows the importance of the features. The first feature that removed from the list has the smallest importance. While the last feature removed from the list is the most important feature in this case. So a ranked feature list is generated ac-

cording to the order of the features removed from the original feature list. The pseudo code is shown in Algorithm 2.

Algorithm 2 Pseudo code of SVM-RFE with RBF kernel

Input:

the dataset D contains training instances with class value;
number of features n;

Output:

RFL feature ranking from best to worst;

initial $D_{train} = D$;
for $i = 1$ to n **do**
  weight vector $w = libsvm(D_{train})$;
  $Q = K(D_{train})$;
  for all the features in D_{train} **do**
    rank$(i) = \frac{1}{2} w^T Q w - \frac{1}{2} w^T Q(-i) w$;
  end for
  find the feature F with the smallest selection criterion, F_{min}=min(rank);
  remove F_{min} from all the instances in D_{train};
  update D_{train} without F_{min};
  add F_{min} into RFL;
end for
return RFL;

3.2.2.3 Combination of the ranked feature lists

During the feature selection process, we employ multiple training datasets to avoid the impact of some special training datasets. Although all the data are selected randomly, there still can be a chance that a training dataset is special enough to produce incorrect results. When we apply feature selection algorithm to this kind of dataset, the ranked feature lists cannot reveal the exact relationship among the features. So we generate several datasets and

Table 3.2: Ranked feature list example

GainRatio																
$1WN$	3[1]	6	5	8	7	1	16	14	11	15	10	2	4	13	9	12
$2WN$	3	5	6	8	14	1	7	16	11	15	10	2	4	13	9	12
$3WN$	3	6	5	7	1	8	14	16	11	10	15	2	4	13	9	12
$4WN$	3	5	6	7	1	8	14	16	11	10	15	2	12	9	13	4
$5WN$	3	5	6	8	7	1	14	16	11	15	10	2	9	13	4	12
	informative												less informative			

[1] 3 : feature F_3 in Table 3.1.

consider all the ranked feature lists together to reduce the impact from a special dataset. We use n datasets $<D_1, D_2, \cdots, D_n>$ in our feature selection component. As we are using four feature selection algorithms, we will have $4n$ ranked feature lists at the end. Here is an example that applying GainRatio-based feature selection algorithm to five training datasets and the ranked feature lists are shown in Table 3.2.

Let M represent a feature selection method, after applying M to a training dataset D_i. A ranked feature list $M_i = <RF_{i1}, RF_{i2}, RF_{i3}, \cdots, RF_{im}>$ is generated, where m=16 and RF_{ij} represents the j_{th} feature in the ranked feature list which come from the training dataset D_i.

With these ranked feature lists, two different ways are introduced to produce feature subsets groups. That is, for each feature selection algorithm, the ranking results from different training datasets are considered together to produce two feature subsets groups.

- The first way to create feature subsets group is called "First$X$":

  As we discussed before, with a feature selection method M, we can

Algorithm 3 Pseudo code of First$X$

Input:

ranked feature lists $RFL[][]$;
value of X;

Output:

features in $FirstX[]$;

```
1: for i = 1 to 16 do
2:    initiate each feature's frequency fre_i=0;
3: end for
4: for j = 1 to 5 do
5:    for k = 1 to X do
6:       fre_{RFL[j][k]} += 1;
7:    end for
8: end for
9: for l = 1 to 16 do
10:   if fre_{F_l} ≥ 0.8 * 5 then
11:      add F_l to FirstX[];
12:   end if
13: end for
14: return FirstX[];
```

generate n ranked feature lists $<M_1, M_2, M_3, \cdots, M_n>$ corresponding to training datasets $<D_1, D_2, D_3, \cdots, D_n>$. Algorithm 3 describes First$X$. It picks up the first X (X belongs to 2 $\sim$ (m-1)) features from each ranked feature lists. If a feature appears more than $0.8n$ times, this feature will be added into feature subset MFirst$X$ (here M can be of the feature selection algorithm and X can be a number between 2 and (m-1)). For each feature selection algorithm M, the result of First$X$ is a feature subsets group containing (m-2) feature subsets (some feature subsets may be the same). We call this feature subsets group MFirst$X$.

- The second way to create feature subsets group is called CFB (Combination of Fuzzy and Brute-force):

  In this method, we separate all the 16 features into three parts, namely TOP, MIDDLE and BOTTOM. The features in TOP part are definitely added into feature subsets, because they are the more significant for classification. And the features in BOTTOM part are discarded and will not appear in the feature subsets. Each combination of the features in the MIDDLE part is merged with the TOP features as a feature subset. Here is how we generate the feature parts. For each M_i in $<M_1, M_2, M_3, \cdots, M_n>$, we draw one third of the features from the top of M_i. If a feature appears more than $0.8n$ times, which means, in most of the training datasets, this feature is important, then it is added into the TOP part. In a similar way, we also take one third features from the bottom of M_i. If a feature appears more than $0.8n$ times, and it is added to the BOTTOM part which means that this feature will not show up in feature subsets. The remaining features are automatically added to the MIDDLE part. As a result of CFB, for each feature selection algorithm M, there are 2^t (t is the number of features in MIDDLE part) feature subsets in the feature subsets group (every feature subset is different). We name it $MCFB$.

3.2.3 Classification techniques

Machine learning techniques are widely used in different research areas because of the outstanding performance they have. In our proposed approach, we are going to use DT and SVM to detect P2P botnet traffic. If the classification models have different opinions on an instance, we make a decision based on the possibility models corresponding to the classification model. The feature vectors used in DT and SVM are the results of the feature selection component.

- The final feature vector, which will be used for DT is as follows: $<F_1, F_3, F_5, F_6, F_7, F_8, F_{10}, F_{11}, F_{14}, F_{15}, Label>$, this feature subset is retrieved from the Waledac datasets, which will be discussed in detail in Chapter 4.

- The final feature vector, which will be used for SVM is as follows: $<F_2, F_3, F_4, F_5, F_7, F_8, F_9, F_{10}, F_{11}, F_{13}, F_{15}, F_{16}, Label>$, this feature subset is retrieved from the Waledac datasets, which will be discussed in detail in Chapter 4.

3.2.3.1 Decision tree

Figure 3.4 is a statistics survey given by Kdnuggets [35]. This survey shows decision tree is the most used machine learning technique in the field of data analysis. Moreover, decision tree and support vector machine are the two most popular supervised machine learning techniques.

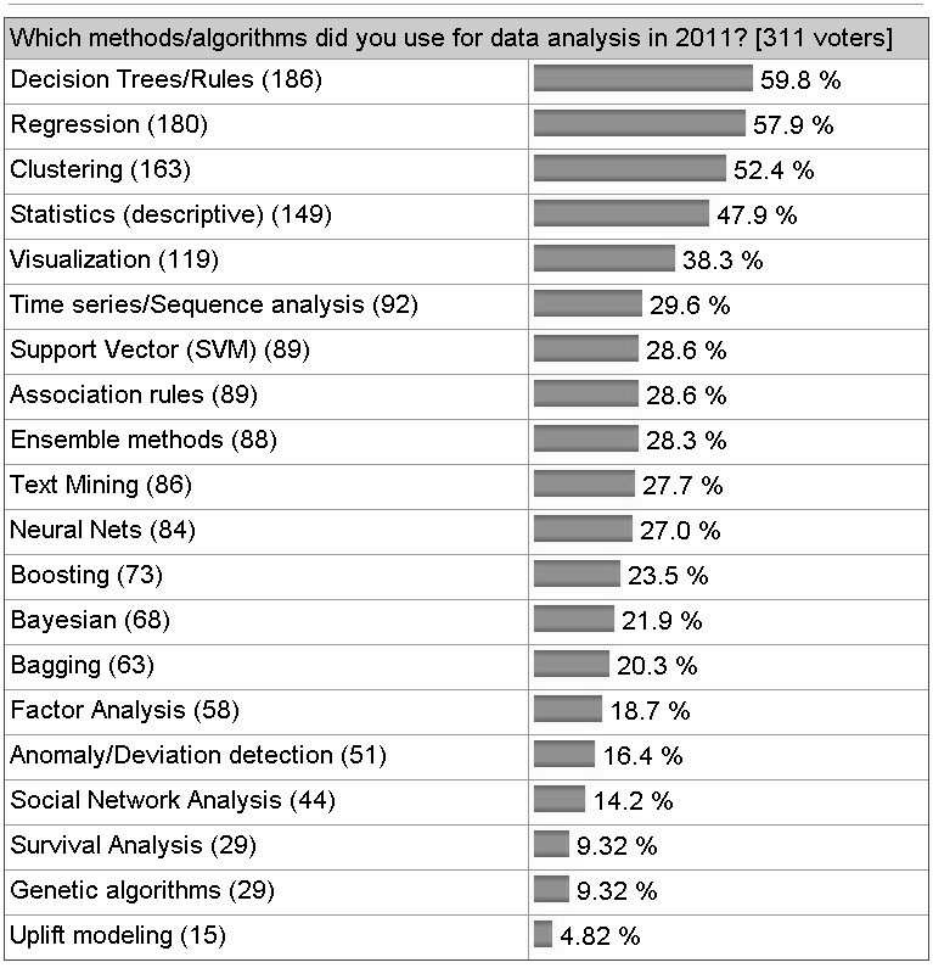


Figure 3.4: Techniques used in data analysis

Decision tree is a powerful classication algorithm, which is very useful for exploring data to gain insight into relationships among a large number of training instances. Take ID3 (an implementation of decision tree) as an example, by investigating a training dataset, the feature with the highest information gain ratio is placed at the root of the decision tree. For every possible value of this feature, a descendant node is created. The training

instances are then grouped into several parts for each descendant node corresponding to the possible values of the root node. The process is repeated to find the root node for each subtree using the grouped instances. This process is processed recursively until all the instances in a node belong to the same class. There are two types of nodes in a decision tree, namely leaf node and non-leaf node. Each non-leaf node represents a feature and each leaf node states a target value. To classify a test instance, a top-down process is executed to test the instance according the features' values. At the end, the test process reaches a leaf node which indicates the target value of the test instance.

In this case, DT is a tree-like model consisting of a set of rules for dividing the dataset into two categories, namely P2P botnet and normal. Each leaf node in the tree has a label, which indicates the class to which the instance belongs.

3.2.3.2 Support vector machine

Support Vector Machine, an extremely useful algorithm based on statistical learning theory, is a good choice for two classes discrimination. It performs very well on high dimension datasets (the classifier only depends on the support vectors, which is a small part of the whole training space). The labelled training dataset is split by an optimal hyperplane which maximizes the distance between support vectors. Here is the way that SVM can always find a linear hyperplane. For a linearly separable problem, there is no difficulty in

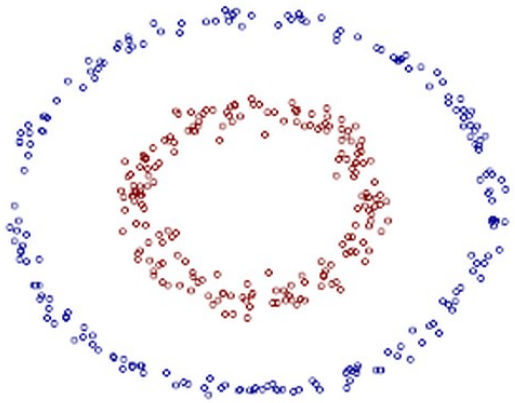
constructing a linear hyperplane. But if the problem is a non-linearly separable problem, SVM cannot build a linear hyperplane based on the original feature space directly. In a novel way, it first projects the low-dimensional feature space to a high-dimensional feature space using a kernel function until the original problem can be converted into a linearly separable problem. Then, SVM can easily find a linear hyperplane. In other words, SVM projects a problem to a linearly separable problem before constructs a hyperplane if the original problem is non-linearly separable.

Figure 3.5 shows a simple example. Figure 3.5(a) is a non-linearly separable problem in 2-dimensional space. But if we project the feature space into 3-dimensional space (Figure 3.5(b)) and do some rotation. Finally, a linearly separable problem in 3-dimensional space is shown in Figure 3.5(c).

In our problem, the labelled training instances are separated into two parts. Most instances of P2P botnet class are at one side of the hyperplane (SVM allows some considered-error data are fallen on the wrong side), and the rest of the instances remain at the other side, which stands for normal class.

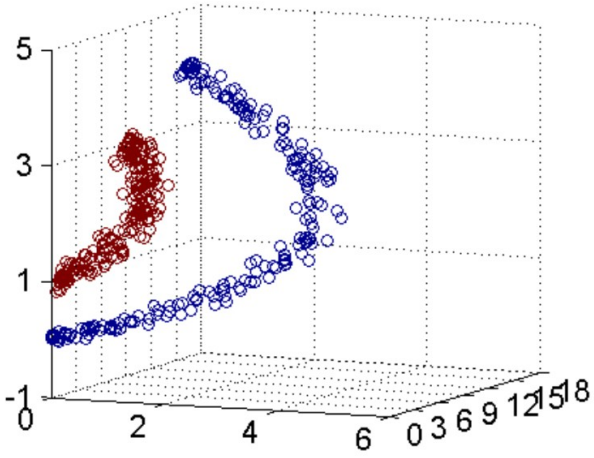
3.2.4 Probability model

In our proposed P2P botnet detection approach, multiple classifiers are used to classify the same instance, thus, different classifiers may have different results. If this happens, we make the final decision with the help of possibilities from the classifiers. Therefore, we need to build possibility models for decision tree and support vector machine.



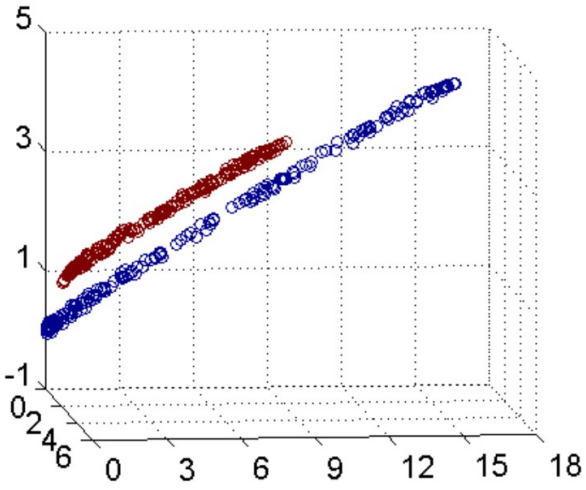
(a) Non-linearly separable problem in 2D space

Figure 3.5: Problem description



(b) Linearly separable problem in 3D space

Figure 3.5: Problem description



(c) Linearly separable problem in 3D space

Figure 3.5: Problem description

3.2.4.1 Decision tree

A smoothing method called M-Branch Smoothing is used to calculate the probabilities for decision tree. It takes all the nodes in one path into account to obtain the probability of a leaf node, because every node in the path make some contribution to the final target value of an unseen instance [36].

Assume there is a leaf node l and the path from the root to a leaf l is $< V_1, V_2, ..., V_l >$, where V_l is a leaf l and V_1 is the root. n_c^j is the number of training instances belonging to class c at node V_j. The probability of a node in the tree is represented in Equation 3.6:

$$p_c^j = \frac{n_c^j + m * p_c^{j-1}}{\sum_{c \in C} n_c^j + m} \tag{3.6}$$

From the equation we can see the probabilities of node V_j are based on V_{j-1}. p_c^0 is defined as $1/c$. By computing the probability recursively, the probabilities of leaf l are given by p_c^l.

In this equation, m depends on the total number of the training dataset N and the height of node V_j h. It can be calculated by the following equation:

$$m = M * \left(1 + \left(1 - \frac{1}{h}\right)\sqrt{N}\right) \tag{3.7}$$

where M is a constant, Ferri et al. stated that the best experimental value of M is 4 [36].

3.2.4.2 Support vector machine

For support vector machine, we make use of the distance from an instance to the hyperplane to predict the probability of each classification. We believe that the distance distribution of the training dataset can reveal the probabilities. If most of the distances fall into a range, and the distance from an unseen instance to the hyperplane also falls into this range, it is natural to think that the classification result for this unseen instance has a relatively high probability. On the other hand, if the distance of an unseen instance is too large or too small, then the probability is relatively low. We applied kernel density estimation (KDE) to the distances to estimate the distribution of the distances. As the distances derived from training dataset are not continual, KDE helps to smooth the distance distribution.

We evaluate the probabilities of results from SVM according to the distance from an instance to the hyperplane. Depending on bayes' rule, the probability of SVM's result is calculated as follows:

$$p\big(c|d(x)\big) = \frac{f_c\big(d(x)\big) * p(c)}{f\big(d(x)\big)} \tag{3.8}$$

where c is the label and *d(x)* represents the distance from point x to the hyperplane. *p(c)* is the prior probability of class c. Function f is the density estimate of the distribution of the distance to the hyperplane. We compute it as follows [37]:

$$f(d(x)) = \frac{1}{nb} \sum_{x_i \in C} K\left(\frac{d(x) - d(x_i)}{b}\right) \tag{3.9}$$

K is the kernel function used in KDE and b is the bandwidth.

3.2.5 Decision fusion

In our proposed detection approach, we use n different training datasets as mentioned. Using the average performance can reduce the influence from a specific dataset. After training the n different training datasets, we have $2n$ classifiers (n decision tree classifiers and n support vector machine classifiers). It means, we will have $2n$ classification results for each test instance. It is demonstrated in Chapter 4, both of these two kinds of machine learning techniques are good at classifying our conversation-based feature vector.

Therefore, in most cases, the results from different classifiers are the same. We can do nothing to these kinds of instances even the results are wrong. What we can do to improve the accuracy is to make a better decision for the remaining instances. If an instance is classified into different classes by different classifiers, we perform our decision fusion algorithm. A target value is assigned to the instance based on all the probabilities from the classifiers.

3.3 Concluding remarks

In this chapter we explained all the components of our proposed P2P botnet detection approach. Each component is described in detail including the processes and the algorithms. The whole approach has three components, which are, feature extraction component, feature selection component, and detection component. The feature extraction component converts the traffic into conversation-based feature vectors that represent the conversations. By introducing different feature selection algorithms in the feature selection component, some features are omitted from the feature vector. The final feature vectors for DT and SVM are used in the detection component for P2P botnet detection. The next chapter carries out experiments to evaluate the proposed approach.

Chapter 4

Experiments and Evaluation

Our proposed conversation-based P2P botnet detection approach is described in detail in Chapter 3. This chapter contains the results of the experiments we have carried out by implementing the components of the proposed approach. Section 4.1 introduces all the open source libraries and tools we used in our experiments; Section 4.2 lists the metrics we used for feature selection and evaluating the detection approach; Section 4.3 describes the datasets we used for training and test; Following that, the results of feature selection processes are shown in Section 4.4; Finally, we evaluates our detection approach with the experimental datasets in Section 4.5.

4.1 Tools

Table 4.1 shows the tools and libraries we have used in our experiments.

Table 4.1: Tools

Name	Description	Version
WireShark	A network protocol analyzer	1.6.2
Tcpreplay	Replays pcap files onto the network	3.4.4
Jpcap	A java library for capturing and sending network packets	0.7
Weka	A popular machine learning library	3.6
Libsvm	An open source library for SVM	3.12
Tcpdump	A command-line packet analyzer	4.2.1

4.2 Metrics

A good classifier should put as many instances as possible into the right class and avoid classifying instances into the opposite class. In our evaluation, we treated P2P botnet instances as positive instances and normal instances as negative instances. We use some metrics to evaluate the experiments, namely true positive rate (TPR), false positive rate (FPR), [TPR+(1-FPR)]/2, precision and accuracy. They are listed below. It is noteworthy that true negative rate (TNR) and false negative rate (FNR) are not used as metrics in our experiments. Because that a high TPR always with a low FNR and a high TNR always with a low FPR. Equations 4.1 and 4.2 shows the relationship between them.

$$\mathrm{TPR} + \mathrm{FNR} = 1 \tag{4.1}$$

$$\mathrm{FPR} + \mathrm{TNR} = 1 \tag{4.2}$$

In order to express the experiments' results clearly, we choose TPR and FPR instead of using all the four metrics.

Table 4.2: Confusion matrix

		Predicted class	
		normal	P2P botnet
Actual class	normal	TN (True Negative)	FP (False Positive)
	P2P botnet	FN (False Negative)	TP (True Positive)

A confusion matrix (See Table 4.2) can help us to understand the metrics better. There are two classes in our problem: P2P botnet and normal. True positive (TP) represents real bot instances correctly classified as bot instances. False positive (FP) indicates real normal instances misclassified as bot instances. True negative (TN) and False negative (FN) have similar meanings.

The metrics we used in our experiments are listed below:

- TPR True positive rate, also called sensitivity or recall, finds the percentage of positive instances classified as positive by using the following equation:

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

- FPR False positive rate shows the percentage of negative instances misclassified as positive by using the following equation:

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

- Precision Precision gives the percentage of instances classified as posi-

tive correctly by using the following equation:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$$

- Accuracy Accuracy indicates the percentage of correct predictions of all the instances by using the following equation:

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

- [TPR+(1-FPR)]/2

  This is a metric that combines TPR and FPR. In most cases, FPR goes up as TPR decreases, and FPR decreases as TPR increases. With this metric, we aim to find a compromise solution which has a good result of [TPR+(1-FPR)]/2. [TPR+(1-FPR)]/2 has the same effect as the Youden index, which is TPR-FPR. Actually, TPR+(1-FPR) equals TPR+TNR, which makes more sense than the Youden index [38].

4.3 Datasets

In this stage, the experimental network traffic used for experiments are shown in the following table (See Table 4.3). We will use these network traffic files to evaluate our detection approach in the following sections.

Table 4.3: Trace files

Name	Capture from	Size	Type
Storm.pcap	University of Victoia	40M	P2P Botnet
Waledac.pcap	University of Victoia	22.6M	P2P Botnet
Bittorrent.pcap	Virtual Machine	896M	P2P Normal
Normal1.pcap	Laptop	802M	Normal
Normal2.pcap	Laptop	981M	Normal
Testbed.pcap	Lab's testbed	16G	Normal
Victoria.pcap	University of Victoia	245M	Normal

The first column of the table is the name of the network traffic files. The environment where the files are retrieved from is shown in the second column. The third column describes the traffic data size for each traffic file. The last column indicates whether the traffic is normal traffic or P2P botnet traffic. The P2P botnet traffic files are derived from the University of Victoria called Information Security and Object Technology (ISOT) dataset. ISOT dataset contains both malicious traffic and non-malicious traffic. It merges several traffic data from different places. The malicious traffic comes from the French chapter of the honeynet project [39] involving two typical P2P botnets: Storm and Waledac. On the other hand, the non-malicious background traffic come from two different labs. One is the Traffic Lab at Ericsson Research in Hungary [40], the other one is the Lawrence Berkeley National Lab (LBNL) [41]. *Tshark* is used to separate the whole ISOT traffic into network

traffic files by using different filters. The file called Storm only contains the traffic from Storm, and the other file called Waledac only includes the traffic from Waledac. Furthermore, from the ISOT traffic, we obtained a part of the non-malicious traffic to generate another file called Victoria, containing only pure non-malicious network traffic. To sum up, from the ISOT dataset, we extract three different traffic files: "Storm.pcap", "Waledac.pcap" and "Victoria.pcap". In the remaining part of this chapter, the file extension will be omitted when discussing the traffic files.

Moreover, besides the normal traffic from the ISOT dataset. We also collected some different normal traffic trace files from different applications and environments. The P2P normal traffic called Bittorrent was captured from a virtual machine that had bittorrent (P2P downloading software) installed on it. However, we could not make sure that non-P2P traffic is blocked even when we used a tool called QQ computer management to limit the traffic caused by other applications (some traffic is caused by the advertisements which are built in bittorrent). Thus, there can be a chance that the trace file Bittorrent contains a small part of non-P2P traffic. But, such traffic should be non-malicious. We also captured two traffic files from our lab in two days. The daily network traffic file "Normal1" and "Normal2 " were obtained from the network interface of my own PC, and the traffic includes e-mails, web browsing, gaming, downloading, etc. These two traffic files are considered as pure non-malicious traffic. Since the machines in our lab are well protected by firewall, antivirus products and professional people. In

addition, we also have a normal traffic file from ISCX's testbed which is declared non-malicious [42].

It should be noticed that all the traffic files described above are all pure traffic, that is, each file contains only either normal traffic or P2P botnet traffic. It is good for our experiments with the labelled data. As the machine techniques we applied in our proposed approach are supervised learning techniques which need labelled data for training. If the datasets are not labelled, it is impossible to train the classifiers and to evaluate the classifiers as well. However, the traffic files cannot be fed to DT or SVM directly, that is, we need to convert the network traffic files listed in Table 4.3 into feature vectors.

As Figure 4.1 shows, we connected two PCs with a cable. Machine A had windows 7 installed, which was used to capture the raw traffic through the network interface sent by machine B. Machine B installed Ubuntu to replay all the traffic files listed above one by one with Tcpreplay. Tcpreplay can be used to replay the traffic from the traffic file with the exactly same content and speed when it was captured before. The feature extraction component described in Section 3.2.1 installed on Machine A was used to convert the captured raw traffic into conversation-based feature vectors. For each traffic file, all feature vectors converted from this file were stored in a corresponding file (having the same name as the traffic file, but with different file extension). As all the traffic in the traffic files is pure, the instances were labelled while extracting the features. That means, each feature vector was related to a la-

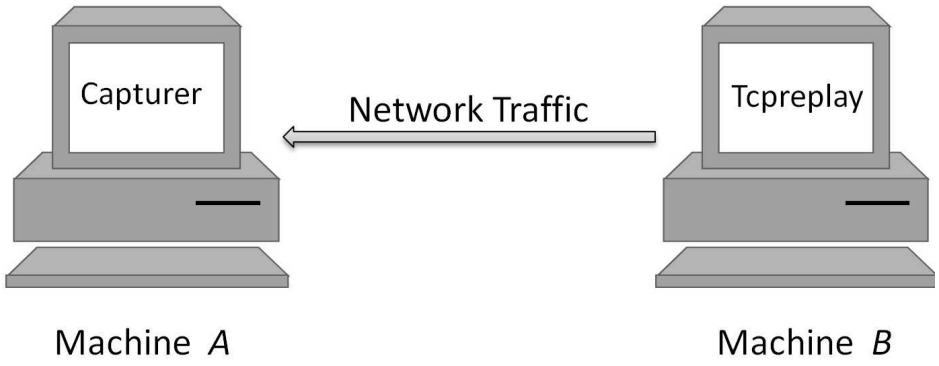


Figure 4.1: Traffic replay

bel indicated the class the instance belonged to. In the training datasets, the labels were used to build classification models, because DT and SVM both are supervised machine-learning techniques which need labelled instances for training. In the test datasets, the labels were used for evaluating the classification results. Finally, seven feature vector files with labels were generated from the seven traffic files in Table 4.3.

After feature extraction, there was one file contained only Storm instances called STORM and one file contained Waledac instances called WALEDAC. And there were five files contained normal instances, we merged them into one large file called NORMAL which consisted of normal instances generated by different non-malicious traffic files. Finally, there are three files: NORMAL, STORM and WALEDAC.

We randomly picked instances from the three files to construct training datasets and test datasets. As our experiments wanted to prove that we can

detect unknown P2P botnet. We not only used Waledac to detect Storm, but also used Storm to detect Waledac. Tables 4.4 generated by pseudo code 4 show the 20 datasets used as unknown P2P botnet in order to test the classifiers. The first column of each table indicates the names of the datasets used in our experiments. Each dataset consisted of both normal instances and P2P botnet instances. For instance, dataset "1WN_test" consisted of both P2P botnet instances from WALEDAC and normal instances from NORMAL. It should be pointed out that there is no overlap between different datasets. We deleted an instance as soon as it was put into one dataset, so it cannot exist in some other datasets simultaneously. Therefore, after this step, the three files: NORMAL, STORM, WALEDAC did not contain the removed instances which were existed in the 20 datasets we created. In order to express clearly, we named these three files: newNORMAL, newSTORM and newWALEDAC after creating the 20 test datasets. These three files is going be used to generate training datasets later.

4.3.1 Reasonable ratio between normal data and P2P botnet data in training datasets

For a training dataset, the ratio between P2P botnet instances and normal instances could influence the performance of a classifier, which built upon the training dataset. Thus we had to find a reasonable ratio before we start feature selection and P2P botnet detection. With the data files we mentioned in

Table 4.4: Test datasets

(a) Test datasets: Waledac & Normal

Dataset	Waledac	Normal	Total
1WN_test	493	1786	2279
2WN_test	488	1771	2259
3WN_test	498	1779	2277
4WN_test	499	1794	2293
5WN_test	492	1780	2272
6WN_test	483	1780	2263
7WN_test	494	1789	2283
8WN_test	485	1789	2274
9WN_test	492	1781	2273
10WN_test	488	1783	2271

(b) Test datasets: Storm & Normal

DataSet	Storm	Normal	Total
1SN_test	573	1788	2361
2SN_test	568	1777	2345
3SN_test	569	1774	2343
4SN_test	573	1786	2359
5SN_test	564	1777	2341
6SN_test	564	1780	2344
7SN_test	564	1779	2343
8SN_test	563	1782	2345
9SN_test	567	1782	2349
10SN_test	566	1784	2350

Algorithm 4 Pseudo code of generating test datasets

Input:

normal dataset $NORMAL$;
p2p botnet dataset $P2PBOTNET$;
the normal data volume in each test dataset n;
the P2P botnet data volume in each test dataset b;

Output:

new normal dataset $newNORMAL$;
new P2P botnet dataset $newP2PBOTNET$;
test datasets $BN\_test[]$;

1: **for** $i = 1$ to 10 **do**
2: **while** $BN\_test[i] \leq n$ KB **do**
3: randomly select instance $\mathcal{I}$ from $NORMAL$;
4: add $\mathcal{I}$ into $BN\_test[i]$;
5: delete $\mathcal{I}$ from $NORMAL$;
6: **end while**
7: **while** $BN\_test[i] \leq (n + b)$ KB **do**
8: randomly select instance $\mathcal{I}$ from $P2PBOTNET$;
9: add $\mathcal{I}$ into $BN\_test[i]$;
10: delete $\mathcal{I}$ from $P2PBOTNET$;
11: **end while**
12: **end for**
13: $newNORMAL$=$NORMAL$;
14: $newP2PBOTNET$=$P2PBOTNET$;
15: **return** $BN\_test[]$;

Section 4.3, namely newNORMAL, newSTORM and newWALEDAC, different training dataset groups are generated. We applied different ratio to these training dataset groups. The ratios between P2P botnet data and normal data are 1:9; 2:8; 3:7; 4:6; 5:5; 6:4 and 7:3. We set the volume of P2P botnet instances to a fixed size: 50 KB. Therefore, the volume of normal instances are 450 KB, 200 KB, 117 KB, 75 KB, 50 KB, 35 KB, 22 KB corresponding to the ratios. The test datasets we used in our experiments are listed in

Table 4.5: Training datasets

(a) Training datasets contain 50 KB Waledac instances & 50 KB Storm instances

Name	Waledac	Normal	Total
1WN	448	497	945
2WN	445	485	930
3WN	455	489	944
4WN	443	489	932
5WN	447	488	935

(b) Training datasets contain 50 KB Storm instances & 50 KB Storm instances

Name	Storm	Normal	Total
1SN	439	559	998
2SN	448	553	1001
3SN	439	570	1009
4SN	436	572	1008
5SN	446	565	1011

Table 4.4.

4.3.1.1 50 KB:50 KB as an example

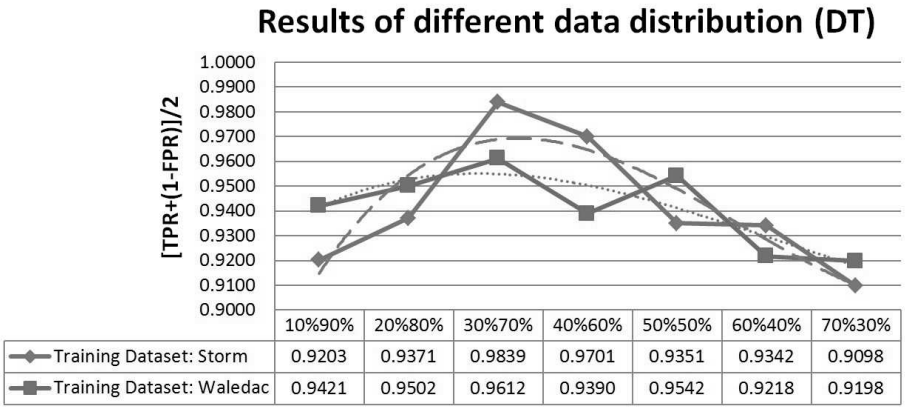
As an example, the training datasets we used contains the same number of P2P botnet data and normal data. The training datasets list in Table 4.5 consist of instances from newNORMAL, newSTORM and newWALEDAC as described in Section 4.3. The datasets in Table 4.5(a) were used to construct decision tree classifiers, and then the test datasets in Table 4.4(b) were used to test these decision tree classifiers in order to evaluate the performance of the classifier derived from 50 KB P2P botnet data and 50 KB normal data. The metrics we used in this evaluation is [TPR+(1-FPR)]/2. Since we used every test dataset to test every decision tree classifier, finally, we had 50 results in this evaluation. The average value was used to represent the performance of a decision tree classifier, which is 0.95418 (See Table 4.6). Table 4.6 also shows another three results we obtained from training datasets

Table 4.6: Performance of the datasets with 50 KB P2P botnet instances and 50 KB normal instances

Training data	Test data	Method	[TPR+(1-FPR)]/2
Table 4.5(b)	Table 4.4(a)	DT	0.93509
Table 4.5(b)	Table 4.4(a)	SVM	0.84374
Table 4.5(a)	Table 4.4(b)	DT	0.95418
Table 4.5(a)	Table 4.4(b)	SVM	0.88332

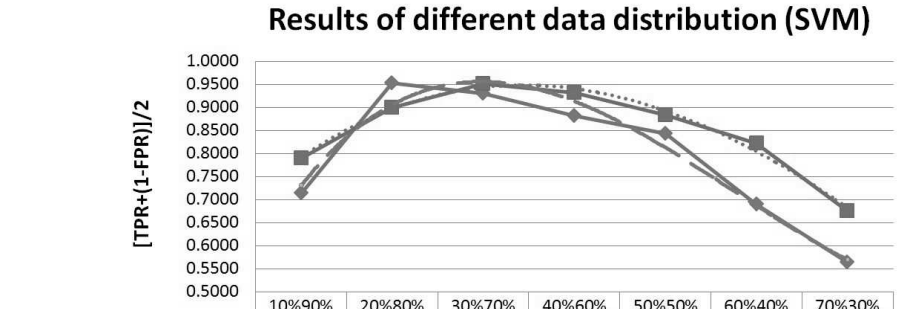
with 50 KB P2P botnet data and 50 KB normal data.

The same procedures were performed to all dataset groups with different ratios between P2P botnet instances and normal instances. Figures 4.2 and 4.3 show the results (the two percentages (eg.10%90%) represent the data ratio between P2P botnet data and normal data in a training file). Investigated into the dash lines (fitting the data in to a line) in both figures, we may come to a conclusion it is better to construct the training datasets with 30% P2P botnet data and 70% normal data. On the other hand, the results derived from DT classifiers are better than the corresponding results obtained from SVM classifiers, especially when the P2P botnet data is much less than normal data or normal data is less then P2P botnet data. The data points at the left of the fitting lines have relatively poor performance may caused by the unbalanced training datasets. Meanwhile, the data points at the right side of the fitting lines represents the performance of corresponding training datasets. Their performances are also not good, the reason behind it may be the training datasets do not have enough data to construct correct classifiers. To sum up, the result would be applied to the subsequent experiments.



	10%90%	20%80%	30%70%	40%60%	50%50%	60%40%	70%30%
Training Dataset: Storm	0.9203	0.9371	0.9839	0.9701	0.9351	0.9342	0.9098
Training Dataset: Waledac	0.9421	0.9502	0.9612	0.9390	0.9542	0.9218	0.9198

Figure 4.2: Results of different data distribution (DT)



	10%90%	20%80%	30%70%	40%60%	50%50%	60%40%	70%30%
Training Dataset: Storm	0.7141	0.9541	0.9306	0.8817	0.8437	0.6905	0.5640
Training Dataset: Waledac	0.7906	0.8996	0.9516	0.9329	0.8833	0.8220	0.6766

Figure 4.3: Results of different data distribution (SVM)

4.4 Feature selection results

According the results from Section 4.3.1, we believe good results might be achieved if the data distribution of the train datasets is 3: 7 (P2P botnet: normal). Therefore, we generate the datasets shown in Table 4.7 to train classifiers.

4.4.1 Waledac as a known p2p botnet

As mentioned before, we need to validate that our proposed approach is able to detect unknown P2P botnets. Thus, in this scenario, we used Waledac as a known P2P botnet and used Storm as an unknown P2P botnet, that is only the datasets derived from Waledac (Table 4.7(a)) were used in the feature selection process. However, the instances obtained from Storm should not appear in this phase.

4.4.1.1 Feature selection for Decision Tree

We used information gain-based feature selection algorithm and information gain ratio-based feature selection algorithm to select feature subsets for DT. In this step, we used InfoGainAttributeEval and GainRatioAttributeEval from WEKA (A collection of machine learning algorithms for data mining tasks) which implemented the two algorithms. Figure 4.4 shows the detail of the features' information gain according to the table in Appendix A. The X-axis represents the i_{th} feature in the feature list 3.1 and the Y-axis indicates

Table 4.7: Training datasets

(a) Training datasets contain 50K Waledac instances and 117K Normal instances

Name	Waledac	Normal	Total
1WN	490	1041	1531
2WN	491	1041	1532
3WN	488	1039	1527
4WN	489	1035	1524
5WN	478	1041	1519
6WN	496	1038	1534
7WN	482	1041	1523
8WN	487	1048	1535
9WN	490	1052	1542
10WN	495	1055	1550

(b) Training datasets contain 50K Storm instances and 117K Normal instances

Name	Storm	Normal	Total
1SN	568	1045	1613
2SN	574	1043	1617
3SN	565	1049	1614
4SN	562	1055	1617
5SN	575	1033	1608
6SN	566	1036	1602
7SN	571	1046	1617
8SN	574	1040	1614
9SN	568	1040	1608
10SN	560	1040	1600

the information gain of each feature in each training dataset. Similarly, Figure 4.5 gives the detailed information gain ratio for the features.

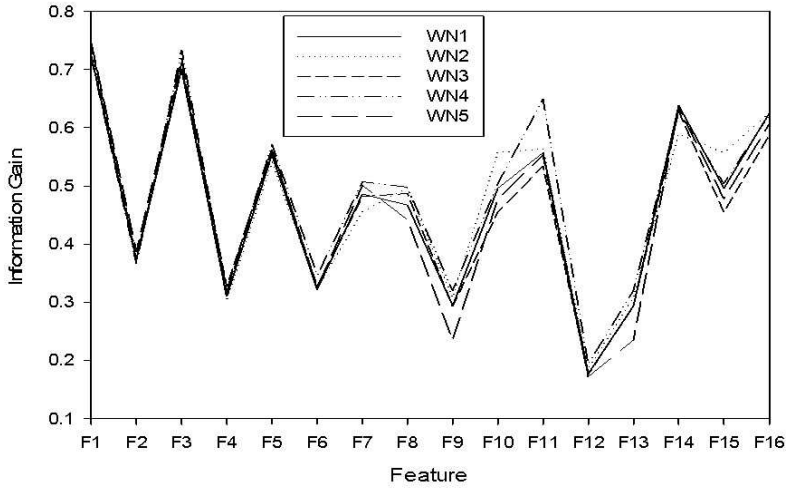


Figure 4.4: Information gain

Table 4.8: Ranked feature list based on information gain

InfoGain																
$1WN$	1[1]	3	14	16	11	5	15	10	7	8	2	6	4	13	9	12
$2WN$	1	3	16	14	11	10	15	5	8	7	2	6	9	13	4	12
$3WN$	1	3	14	16	5	11	8	7	15	10	2	6	4	9	13	12
$4WN$	3	1	11	14	16	5	7	15	10	8	2	6	9	13	4	12
$5WN$	1	3	14	16	5	11	7	10	15	8	2	6	4	13	9	12
	informative										less informative					

[1] 1: feature F_1 in Table 3.1.

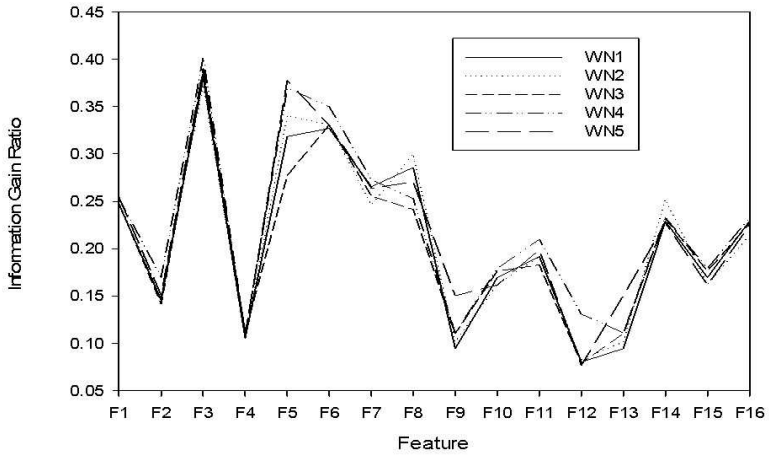


Figure 4.5: Information gain ratio

Based on the information gain and information gain ratio, tables 4.8 and 4.9) show the ranked features lists after applying the algorithms to the five training datasets (1WN to 5WN). The first column shows the names of the datasets used in feature selection procedure. Each row shows a ranked feature list obtained from a dataset by applying a feature selection algorithm.

We can see from the tables, for the five datasets, the ranked feature lists generated by InfoGainAttributeEval are very similar. The same can be said

Table 4.9: Ranked feature list based on information gain ratio

GainRatio																
$1WN$	3[1]	6	5	8	7	1	16	14	11	15	10	2	4	13	9	12
$2WN$	3	5	6	8	14	1	7	16	11	15	10	2	4	13	9	12
$3WN$	3	6	5	7	1	8	14	16	11	10	15	2	4	13	9	12
$4WN$	3	5	6	7	1	8	14	16	11	10	15	2	12	9	13	4
$5WN$	3	5	6	8	7	1	14	16	11	15	10	2	9	13	4	12
	informative												less informative			

[1] 3: feature F_3 in Table 3.1.

for GainRatioAttributeEval. Moreover, for each dataset, the two ranked feature lists from different algorithms have some similarity. It is clear that, the last four features are the same in each ranked feature list. They are the least informative features among all features.

Two groups of feature subsets were yielded from each group of ranked feature lists, as discussed in Chapter 3. In this case, if a feature were to appear 4 or 5 times (we used 5 training datasets), it would be included in the feature subset. Finally, we derived four groups of feature subsets called GainRatioCFB, GainRatioFirst$X$, InfoGainCFB and InfoGainFirst$X$, respectively. Both GainRatioFirst$X$ and InfoGainFirst$X$ contains 14 feature subsets (Can be seen from Table 4.10 and Table 4.11). GainRatioCFB and InfoGainCFB both had 128 (2^7) feature subsets. As outlined in Chapter 3, all features in the TOP part are definitely used in the subsets, and the features in the BOTTOM part would not show up in the subsets. Hence, each combination of the features in MIDDLE part yielded a subset with the TOP features. When we compared GainRatioCFB with InfoGainCFB, we found that the features

Table 4.10: First$X$ features derived with information gain

InfoGain	Feature Subset
First2	F_1,F_3
First3	F_1,F_3
First4	F_1,F_3,F_{14},F_{16}
First5	F_1,F_3,F_{14},F_{16}
First6	F_1,F_3,F_5,F_{11},F_{14},F_{16}
First7	F_1,F_3,F_5,F_{11},F_{14},F_{16}
First8	F_1,F_3,F_5,F_{11},F_{14},F_{16}
First9	F_1,F_3,F_5,F_7,F_{10},F_{11},F_{14},F_{15},F_{16}
First10	F_1,F_3,F_5,F_7,F_8,F_{10},F_{11},F_{14},F_{15},F_{16}
First11	F_1,F_2,F_3,F_5,F_7,F_8,F_{10},F_{11},F_{14},F_{15},F_{16}
First12	F_1,F_2,F_3,F_5,F_6,F_7,F_8,F_{10},F_{11},F_{14},F_{15},F_{16}
First13	F_1,F_2,F_3,F_5,F_6,F_7,F_8,F_{10},F_{11},F_{14},F_{15},F_{16}
First14	F_1,F_2,F_3,F_5,F_6,F_7,F_8,F_{10},F_{11},F_{13},F_{14},F_{15},F_{16}
First15	F_1,F_2,F_3,F_4,F_5,F_6,F_7,F_8,F_9,F_{10},F_{11},F_{13},F_{14},F_{15},F_{16}

in BOTTOM parts were almost identical as evidenced by Table 4.12. This similarity might be caused by the attribute ranking criterion used in these two feature selection algorithms; the basic element in the attribute ranking criterion is information entropy. And the intrinsic information used in GainRatioAttributeEval brings the difference between these two algorithms. Some evaluations should be performed to find a best feature subset from the feature subsets we generated in previous processes. The SN_test datasets were still not used in this phase because we treated Storm as an unknown P2P botnet. The feature subsets were thus evaluated only between the WN datasets. The first five WN datasets were used for training and the rest of the WN datasets were used for testing. To be more specific, dataset "6WN" was used to test the classifier built by "1WN", dataset "7WN" was used to test

Table 4.11: First$X$ features derived with information gain ratio

GainRatio	Feature Subset
First2	F_3
First3	F_3,F_5,F_6
First4	F_3,F_5,F_6
First5	F_3,F_5,F_6,F_7
First6	F_1,F_3,F_5,F_6,F_7,F_8
First7	F_1,F_3,F_5,F_6,F_7,F_8,F_{14}
First8	F_1,F_3,F_5,F_6,F_7,F_8,F_{14},F_{16}
First9	F_1,F_3,F_5,F_6,F_7,F_8,F_{11},F_{14},F_{16}
First10	F_1,F_3,F_5,F_6,F_7,F_8,F_{11},F_{14},F_{16}
First11	F_1,F_3,F_5,F_6,F_7,F_8,F_{10},F_{11},F_{14},F_{15},F_{16}
First12	F_1,F_2,F_3,F_5,F_6,F_7,F_8,F_{10},F_{11},F_{14},F_{15},F_{16}
First13	F_1,F_2,F_3,F_5,F_6,F_7,F_8,F_{10},F_{11},F_{14},F_{15},F_{16}
First14	F_1,F_2,F_3,F_5,F_6,F_7,F_8,F_{10},F_{11},F_{13},F_{14},F_{15},F_{16}
First15	F_1,F_2,F_3,F_4,F_5,F_6,F_7,F_8,F_9,F_{10},F_{11},F_{13},F_{14},F_{15},F_{16}

Table 4.12: TOP & MIDDLE & BOTTOM parts with different feature selection methods

Method	TOP	MIDDLE	BOTTOM
InfoGain	F_1,F_3,F_{14},F_{16}	F_2,F_5,F_7,F_8,F_{10},F_{11},F_{15}	F_4,F_6,F_9,F_{12},F_{13}
GainRatio	F_3,F_5,F_6,F_7	F_1,F_8,F_{10},F_{11},F_{14},F_{15},F_{16}	F_2,F_4,F_9,F_{12},F_{13}

Table 4.13: Top five feature subsets for InfoGainFirst$X$ group(DT)

InfoGainFirstX						
	[TPR+(1-FPR)]/2					
Feature Subset	1WN 6WN	2WN 7WN	3WN 8WN	4WN 9WN	5WN 10WN	Average
1[1],2,3,5,6,7,8,10,11,14,15,16	0.9892	0.9877	0.9826	0.9935	0.9855	0.9877
1,2,3,5,6,7,8,10,11,13,14,15,16	0.9892	0.9877	0.9826	0.9935	0.9855	0.9877
1,2,3,4,5,6,7,8,9,10,11,13,14,15,16	0.9892	0.9877	0.9826	0.9935	0.9855	0.9877
1,2,3,5,7,8,10,11,14,15,16	0.9776	0.9877	0.9893	0.9935	0.9855	0.9867
1,3,5,7,8,10,11,14,15,16	0.9776	0.9877	0.9893	0.9935	0.9835	0.9863

[1] 1: feature F_1 in Table 3.1.

the classifier built by "2WN", and so on. This step was carried out to find the top five feature subsets in each group. (Notice that the top five feature subsets can be more than five feature subsets, because there can be some feature subsets have the same [TPR+(1-FPR)]/2 value in the fifth place). Table 4.13 shows the top five feature subsets in group InfoGainFirst$X$.

The results show that no matter which group, the results of the top five feature subsets are very close. It is obvious that there are some common features which cause the similarity among the results. Based on the evaluation of feature subsets in each group, four final candidate feature subsets were constructed by combining different groups' results in Table 4.13 and the tables in Appendix B.

DT_First$X$ is a candidate feature subset that derives from GainRatioFirst$X$ and InfoGainFirst$X$. Consider the top five feature subsets from these two groups together. Let n be the total number of these feature subsets, and if a feature were to appear more than $n/2$ times, then it was added into

Table 4.14: Four candidate feature subsets for DT

Feature Subset Name	Feature Subset
DT_First$X$	$F_1, F_2, F_3, F_5, F_6, F_7, F_8, F_{10}, F_{11}, F_{14}, F_{15}, F_{16}$
DT_CFB	$F_1, F_3, F_5, F_6, F_7, F_8, F_{11}, F_{14}$
DT_InfoGain	$F_1, F_2, F_3, F_5, F_7, F_8, F_{10}, F_{11}, F_{14}, F_{15}, F_{16}$
DT_GainRatio	$F_1, F_3, F_5, F_6, F_7, F_8, F_{10}, F_{11}, F_{14}, F_{15}$

DT_First$X$. The other four final feature subsets were created in the same way. DT_CFB comes from GainRatioCFB and InfoGainCFB; DT_GainRatio comes from GainRatioFirst$X$ and GainRatioCFB; DT_InfoGain comes from InfoGainFirst$X$ and InfoGainCFB. The four candidate feature subsets are shown in Table 4.14.

It is easy to find out that features F_1,F_3,F_5,F_7,F_8,F_{11} and F_{14} are the common features in every feature subsets. We can infer that these features play an important role in training, and these features are the basic features which can lead to a good result. Moreover, with the help of the remaining features, a better performance can be achieved. We also noticed that features F_4,F_9,F_{12} and F_{13} does not appear in the four candidate feature subsets. The reason is that, no matter which feature selection algorithm was used, these four features are always in the BOTTOM part.

Among these feature subsets, the best one should be selected and used in detection. The SN_test datasets can be used to decide the best feature subset from the four candidate feature subsets. For each candidate feature subset, we used 1WN∼5WN datasets to build five classifiers and tested the classifiers with SN_test datasets one by one. As we have 5 classifiers and 10

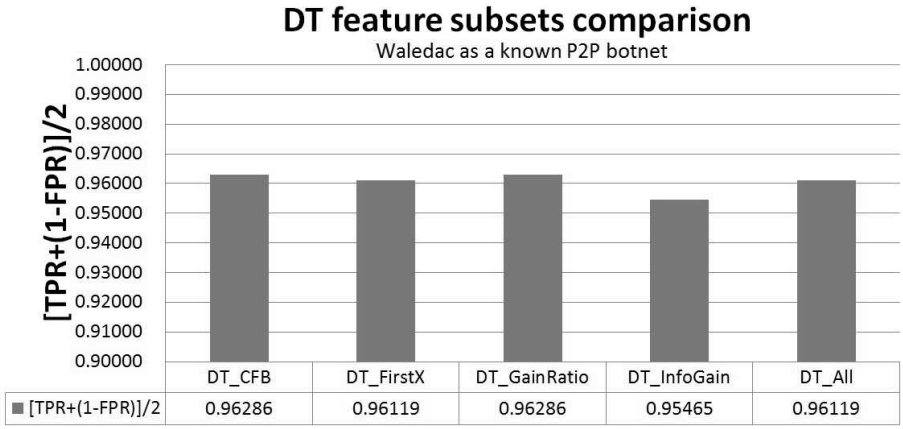


Figure 4.6: DT feature subsets comparison (Waledac as known P2P botnet)

SN_test datasets, each feature subset has 50 results, the average result was calculated to avoid the bias. Therefore, the average results were used to evaluate the candidate feature subsets. A final feature subset which would be used for the decision tree in the P2P botnet detection phase was determined by comparing the results. The results are shown in Figure 4.6, which show that DT_GainRatio and DT_CFB are the best feature subsets. On the other hand, feature subset DT_InfoGain has the worst performance. According to the discussion in Chapter 3, this is not surprising since the decision tree we used is J48 from WEKA. J48 is an implementation of C4.5 that uses information gain ratio to build decision tree. That is why the features generated by GainRatioAttributeEval are better than those generated by InfoGainAttributeEval.

In conclusion, taking into account all the results shown in Figure 4.6 , we maintain that the feature subset DT_GainRatio (F_1, F_3, F_5, F_6, F_7, F_8, F_{10},

Table 4.15: Four candidate feature subsets for SVM

Feature Subset Name	Feature Subset
SVM_First$X$	$F_2, F_3, F_4, F_5, F_7, F_8, F_9, F_{10}, F_{11}, F_{13}, F_{15}, F_{16}$
SVM_CFB	$F_4, F_5, F_7, F_9, F_{10}, F_{11}, F_{13}, F_{15}$
SVM_ReliefF	$F_3, F_4, F_5, F_7, F_8, F_9, F_{10}, F_{11}, F_{13}, F_{15}, F_{16}$
SVM_SVM-RFE	$F_2, F_4, F_5, F_7, F_9, F_{10}, F_{11}, F_{12}, F_{13}, F_{15}$

F_{11}, F_{14}, F_{15}) and DT_CFB(F_1, F_3, F_5, F_6, F_7, F_8, F_{11}, F_{14}) are the best feature subsets to be used for decision tree in the next stage: decision fusion.

4.4.1.2 Feature selection for SVM

The feature selection processes for SVM are quite similar with DT. The different parts are the feature selection algorithms. For SVM, we used ReliefFAttributeEval in WEKA which is an implementation of ReliefF we described in Section 3.2.2.2. A SVM-RFE algorithm was implemented in Matlab and was used to perform SVM-RFE feature selection in our feature selection stage. Table 4.15 shows the four candidate feature subsets for SVM. With the similar procedure as decision tree did, we evaluated the four candidate feature subsets and the results are shown in Figure 4.7.

In conclusion, considering all of the results above, by using all 16 features, the experiments achieved the best results. But, we assert that the feature subset SVMFirxtX ($F_2, F_3, F_4, F_5, F_7, F_8, F_9, F_{10}, F_{11}, F_{13}, F_{15}, F_{16}$) is the most suitable feature subset to be applied to SVM in the subsequent experiments if time is token into account. The time comparison will be discussed in detail in 4.5.1.

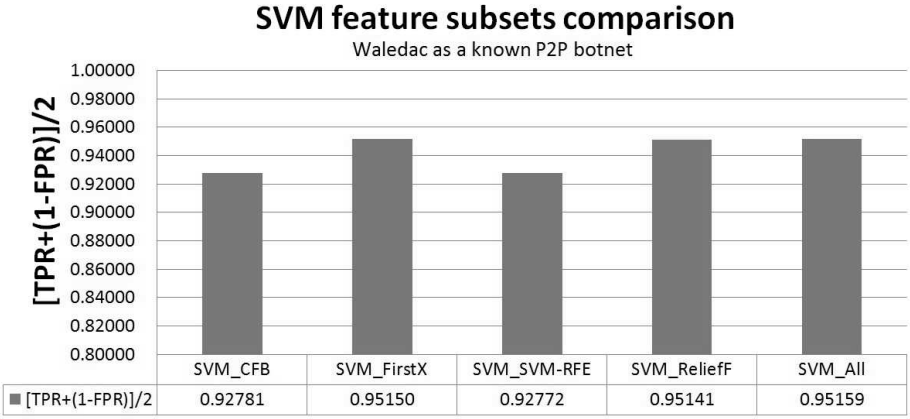


Figure 4.7: SVM Feature Subsets Comparison (Waledac as known P2P botnet)

4.4.2 Storm as a known P2P botnet

In Section 4.4.1, we treated Waledac as a known P2P botnet and used Storm as an unknown P2P botnet to find the best way to obtained a appropriate feature subset. The results shows that both DT_GainRatio and DT_CFB give the best performance for DT; Whereas, SVM_First$X$ shows the best performance for SVM.

With the same procedure in Section 4.4.1, we also did the experiments that treat Storm as a known P2P botnet. The results in these experiments indicated that the feature subset DT_GainRatio is still the best feature subset, however, DT_CFB is not as good as DT_GainRatio this time. SVM_First$X$ also shows good performance. The results are shown in Figures 4.8 and 4.9 Taking everything into consideration, we can safely infer that the feature subset DT_GainRatio is suitable for DT and the feature subset SVM_First$X$

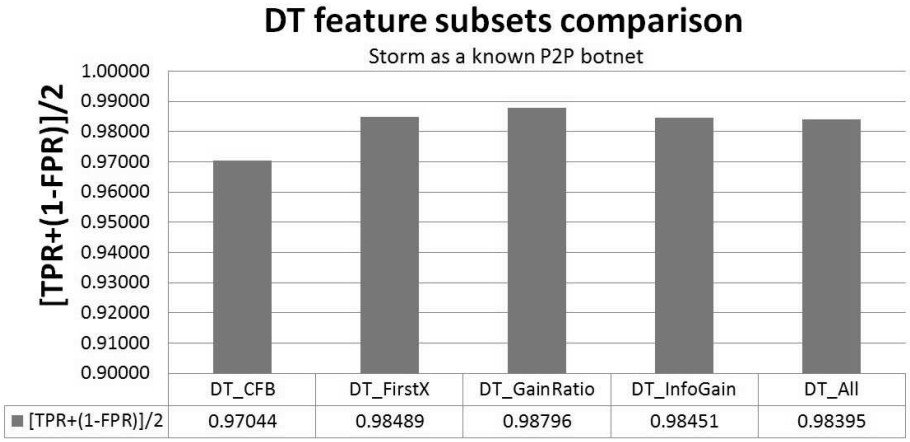


Figure 4.8: DT feature subsets comparison (Storm as a known P2P botnet)

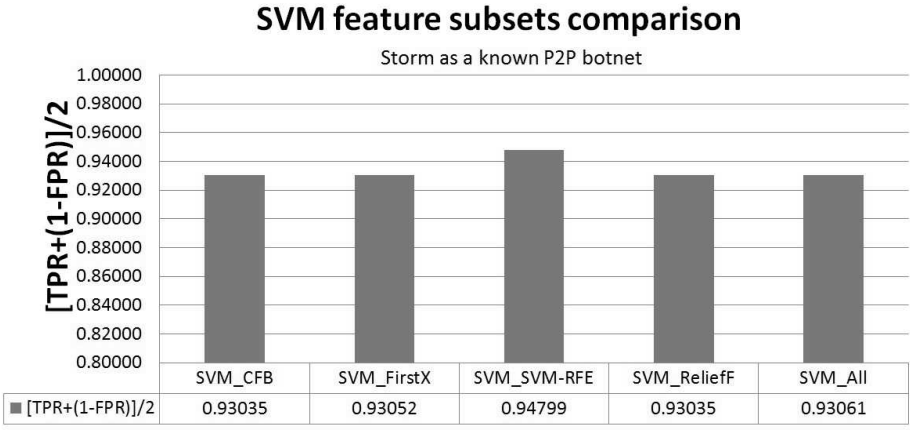


Figure 4.9: SVM feature subsets comparison (Storm as a known P2P botnet)

has the best performance for SVM. These two feature subsets were going to be used in detection component.

4.5 Detection results

The feature selection for decision tree and support vector machine are given in previous sections. In the detection phase, we used all the datasets listed in Table 4.7(a) and Table 4.4(b) (Waledac is used as a known P2P botnet and Storm used as an unknown P2P botnet). However, the instances should be reconstructed according to the feature selection results. A filter was applied before using the datasets, and all the unnecessary features were removed from the original 16-dimensional feature instances. The reduced dimensional feature vectors are represented below:

- Selected feature subset for decision tree:
  $<F_1, F_3, F_5, F_6, F_7, F_8, F_{10}, F_{11}, F_{14}, F_{15}, Label>$
- Selected feature subset for support vector machine:
  $<F_2, F_3, F_4, F_5, F_7, F_8, F_9, F_{10}, F_{11}, F_{13}, F_{15}, F_{16}, Label>$

4.5.1 Detection with DT and SVM

This section shows the P2P botnet detection experiments and results by using decision tree and support vector machine individually. J48 from WEKA was used as an implementation of decision tree. While libsvm was used as an

Table 4.16: Detection results with individual technique

	DT_GainRatio	DT_All	SVM_First$X$	SVM_All
TPR	0.93764	0.93454	0.96510	0.96404
FPR	0.01191	0.01215	0.06210	0.06086
[TPR+(1-FPR)]/2	0.96286	0.96119	0.95150	0.95159
Precision	0.96316	0.96254	0.83234	0.83520
Accuracy	0.97591	0.97498	0.94448	0.94516

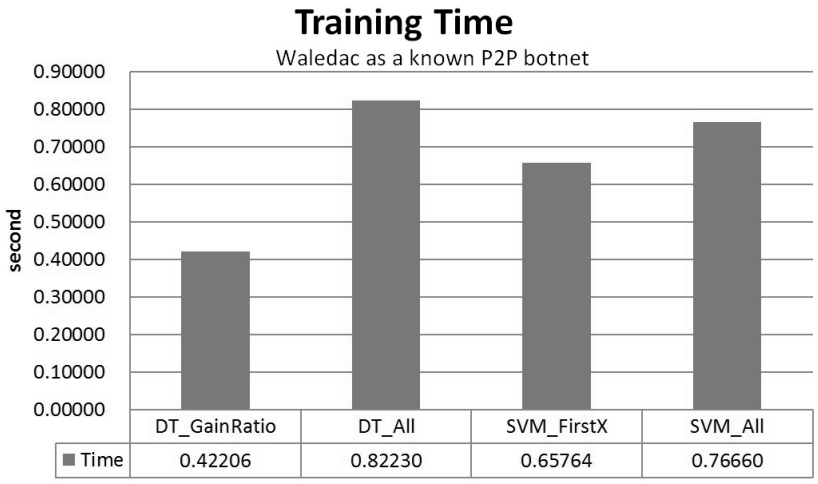
implementation of SVM. All classifiers were trained with 1WN∼5WN (in Table 4.7(a)) with the corresponding feature subsets. On the other hand, the test datasets 1SN_test∼10SN_test are listed in Table 4.4(b). For each machine learning technique with specific features, there are 50 results. The average value is used to represent the detection ability of a machine learning technique with specific features. Table 4.16 gives the results. All the metrics were evaluated. The lowest TPR is 93% and the worst FPR is 6%, meaning both SVM and DT have good performance in detecting P2P botnet with conversation-based feature vectors.

As all the experiments in this phase used the exactly same training datasets and test datasets. The results in Table 4.16 are comparable. Decision tree applied to selected features is better than decision tree used all 16-dimensional features in terms of all the metrics. However, SVM had a different situation, TPR had an improvement by using selected features and the FPR was slightly higher than using 16-dimensional features. However, SVM with selected features and SVM with all 16 features had a equivalent performance. When we compared DT and SVM, it shows that decision tree is better than

SVM in terms of all the metrics except TPR.

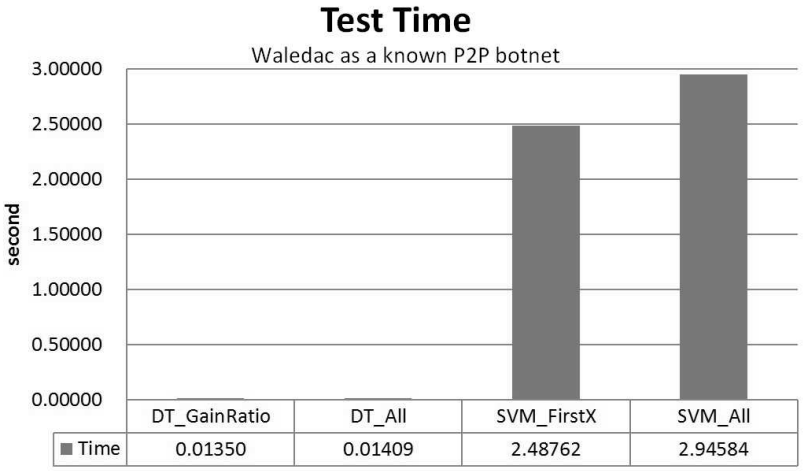
Also, we carried out experiments to compare the training time and test time. These experiments were conducted in a Windows 7 machine with 3.00 GB RAM and 2.66 GHz processor. We not only considered the training time, but also took the test time into account. Training time refer to the total time of training 5 training datasets (1WN∼5WN), test time refer to the total time of using 5 classifiers to test 10 test datasets (1SN_test∼10SN_test). We ran the experiments 15 times and the average time was calculated for evaluation. J48 and libsvm were used as the implementations of decision tree and support vector machine, respectively.

Figure 4.10 offers the experimental results. The comparison of training time is shown in Figure 4.10(a) (Appendix C.1 shows the details). For decision tree, the training time with selected features is 0.422 second. However, if we trained 5 training datasets with all 16 features; it takes 0.822 second which was almost twice of the training time with selected features. The training time of SVM also reduced after omitting some less informative features. But, it does not save so much time as decision tree. This may caused by the different mechanisms of constructing classifiers and the different number of features. As we know, SVM is much complicated than DT. But, we noticed that there was no big difference between the training time of DT and SVM. The reason is that, when we train a DT classifier, all the training instances are needed. However, SVM only needs a small portion of the training instances which called support vectors.



(a) Training time comparison

Figure 4.10: Time comparison



(b) Test time comparison

Figure 4.10: Time comparison

Figure 4.10(b) gives the results of test time when use different techniques

with different features (Appendix C.2 shows the details). Both DT and SVM saved time when selected features were used. The time different between DT and SVM is significant huge. The test time DT used was almost 0.015 second for testing, while SVM uses more than 2 seconds. According the discussions in Chapter 3, decision tree classifies an instance only according a set of rules. But SVM needs to assert the instance belongs to which side. Thus, regarding to test, DT is much simpler than SVM. That's why SVM needs more time to test an instance than DT does.

Consider the training time, test time, and the performance, we believe we made a right decision to use feature subset DT_GainRatio for decision tree. And use the selected feature subset SVM_First$X$ for SVM, because it can save time while the performance is slightly worse than SVM_All which is acceptable.

4.5.2 Decision fusion

The previous experiments give good results by using decision tree and SVM with conversation-based feature vectors. The true positive rate of decision tree are above 93%, and the false positive rates of decision are extremely low. Similarly, SVM's true positive rates are over 96%, and its highest false positive rates are about 6%. According to these results, we believe that our conversation-based P2P botnet detection can work well by applying decision tree and SVM.

However, there is always space for improvement. The mechanisms that de-

cision tree and SVM used to build classifiers are totally different. Therefore, combining the results from decision trees and SVM may improve the performance of detection. We use 10 classifiers (according to the 5 training datasets, we have 5 DT classifiers and 5 SVM classifiers) in a fusion function to make use of the results from both DT and SVM in order to make a better decision.

For most of the test instances, different classifiers can assign them with the right target value. It is evidenced by Figures 4.11 and 4.12, the training datasets are 1WN∼5WN and the test dataset consists of 200 normal instance from 1SN_test and 200 bot instances from 1SN_test.

Figure 4.11 shows the classification results of the 400 test instances with 5 SVM classifiers. The Y-axis represents the distance between an instance to the hyperplane of a SVM classifier. The first 200 test instances are normal instances, thus the distances should be greater than 0; If the distance of a normal instance is less than zero, then it is a misclassification. Comparing the results of the 5 SVM classifiers, most of the results are correct and similar. From another perspective, the hyperplanes of these 5 classifiers are close to each other. Meanwhile, Figure 4.12 proves that 5 DT classifier also have similar results when classifying the 400 test instances.

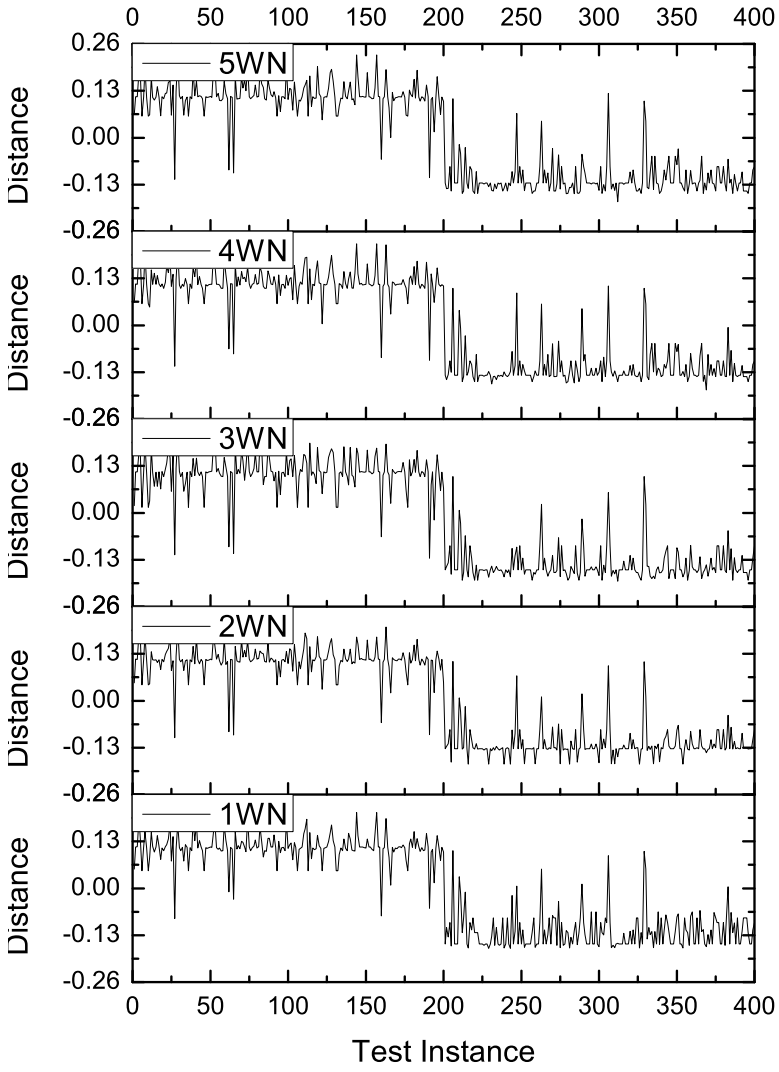


Figure 4.11: SVM classification results

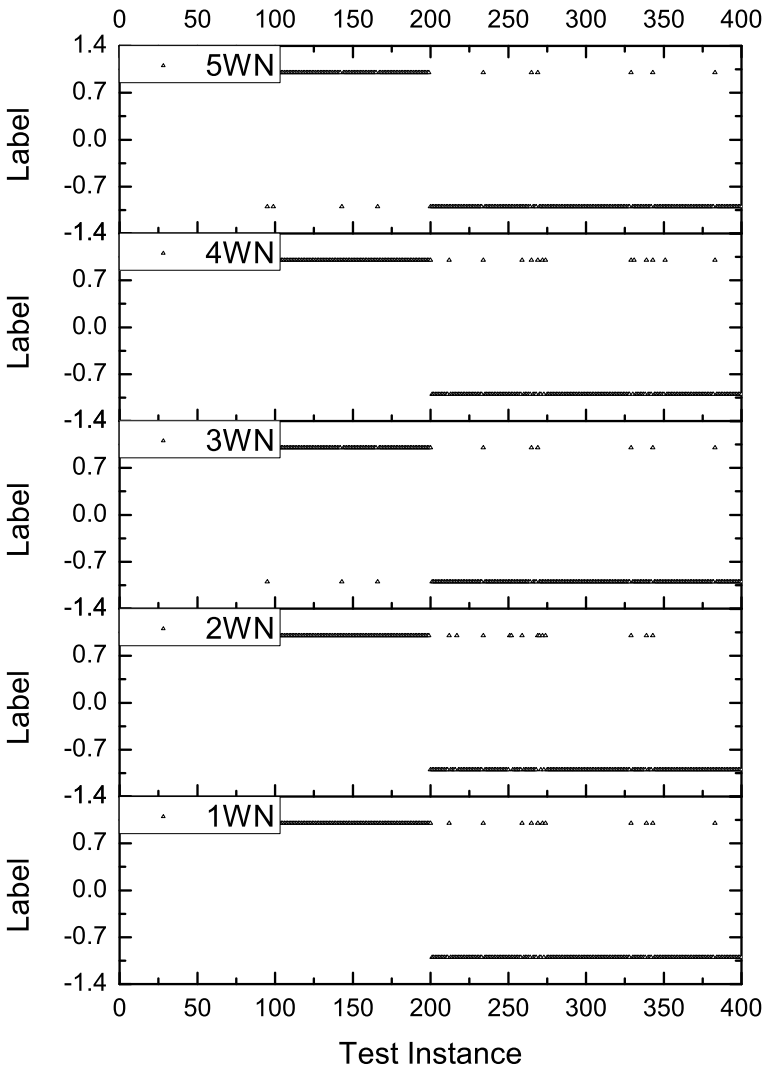


Figure 4.12: Decision tree classification results

However, there still exist a small amount of test instances, that different classifiers have different opinions on them. If we enlarge an area of Figure 4.11, we can see in Figure 4.13 that three of five SVM classifiers classified instance

#289 into normal class and two of them considered this instance as a bot instance. To overcome this kind of situation, the probabilities are used to help to make a final decision. The probabilities represent the confidences for each decision, and we choose the class, which has the highest probability as the final decision. Algorithm 5 shows the algorithm used for decision fusion based on 10 classifiers (5 decision tree classifiers and 5 SVM classifiers) and the probabilities we described in Chapter 3.

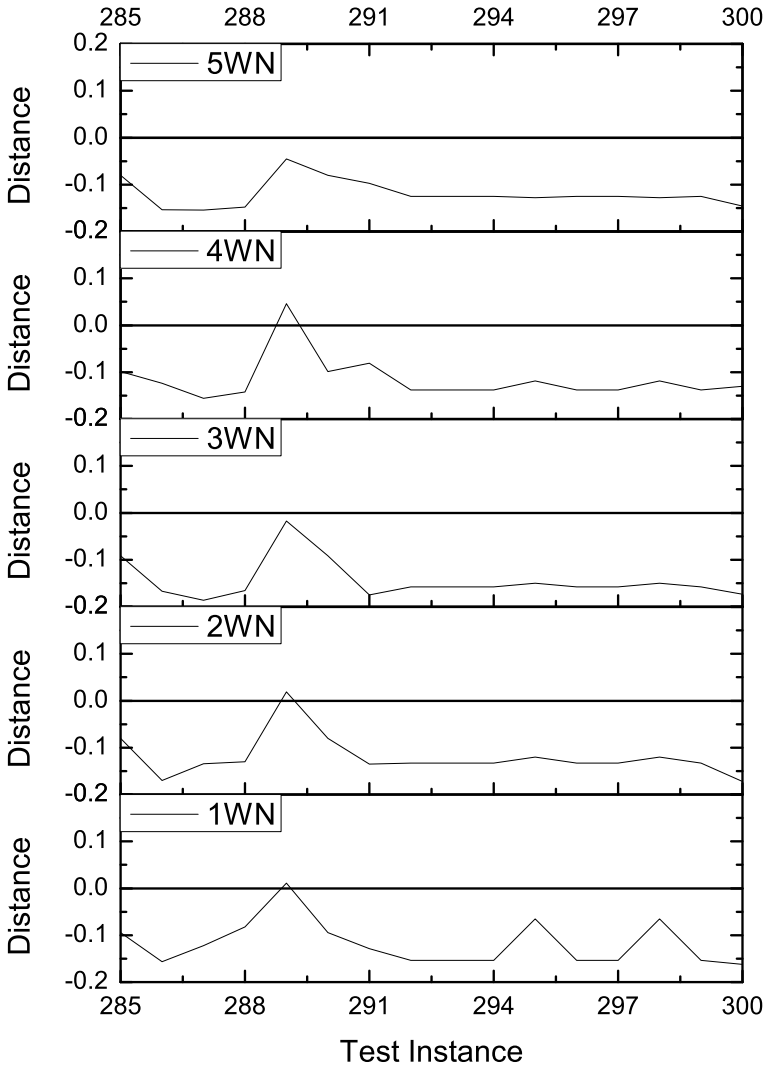


Figure 4.13: SVM classification results

Algorithm 5 Pseudo code of Decision Fusion

Input:

the training datasets D_i;
a test instance T_i;

Output:

the target value of the test instance c;

```
for i = 1 to n do
    build classifiers DT_i and SVM_i;
end for
initialize B = 0 to represent the number of the classifiers classify T_i as a botnet instance;
initialize N = 0 to represent the number of the classifiers classify T_i as a normal instance;
for traverse all classifiers do
    if classify T_i as botnet then
        B++;
    else
        N++;
    end if
end for
if B-N ≥ x then
    c = botnet;
else if N-B ≥ x then
    c = normal;
else
    p_b =sum of the probabilities from the classifiers that classify T_i as a botnet instance;
    p_n =sum of the probabilities from the classifiers that classify T_i as a normal instance;
    if p_b/B ≥ p_n/N then
        c = botnet;
    else
        c = normal;
    end if
end if
return c;
```

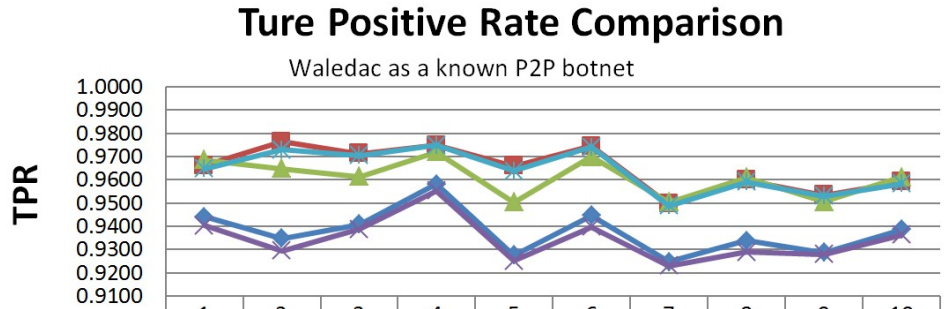
4.5.3 Comparison

As we discussed in P2P botnet detection component, the detection performance is improved by feature selection firstly; Then decision fusion based on probabilities is applied to make the detection component perform better.

Figure 4.14 shows that the results for all the metrics discussed in Section 4.2. Generally speaking, we have a good detection result. The TPR results are basically over 92% and the FPR results are all lower than 8%. Moreover, [TPR-(1-FPR)]/2 results, precision results and accuracy results are all good. Although SVM almost had the highest TPR, from the perspective of accuracy, Figure 4.14(e) shows that SVM's accuracy is not as good as DT and Fusion (stand for the results after fusion algorithm). It is reasonable, because there are more normal instances than botnet instances in the test datasets and the FPR results of SVM are worse than DT and Fusion. Based on the results, this thesis come to a conclusion that conversation-based P2P botnet detection have a good performance.

Considering all the metrics, the performance of decision tree is improved by using selected feature subset. Although the improvement is not that significant, the time complexity is reduced. However, for SVM, the improvement only can be seen from TPR results. Thus, we can infer that all 16 features can contribute to the performance of SVM. But we still believe that the feature selection processes are needed. Because that the time complexity is decreased by using selected feature subset. In Section 4.5.2, we combines the results from different classifiers. Comparing the result after fusion with

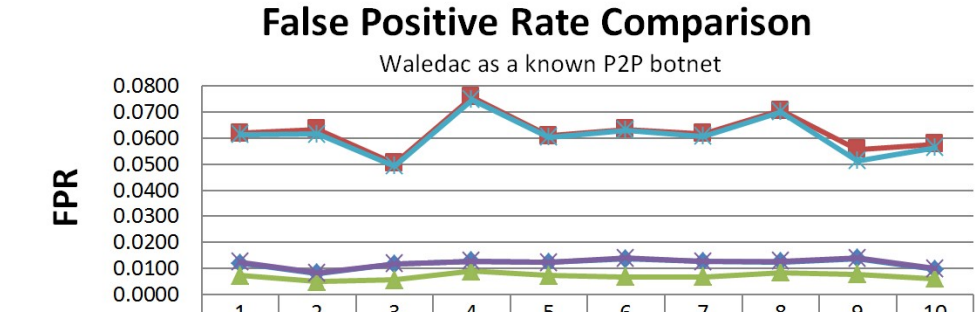
individual machine learning technique, the TPR results of fusion are a little worse than SVM. But fusion has a outstanding performance regarding to the other four metrics. To sum up, taking all the metrics used in our experiments into account, the thesis concludes that our feature selection processes can reduce the time complexity without negative influence on the detection performance. Moreover, our decision fusion model can improve the performance by combining decision tree and support vector machine.



	1	2	3	4	5	6	7	8	9	10
DT_GainRatio	0.9442	0.9349	0.9409	0.9581	0.9277	0.9447	0.9248	0.9339	0.9287	0.9385
SVM_FirstX	0.9661	0.9764	0.9712	0.9749	0.9660	0.9745	0.9496	0.9599	0.9534	0.9590
Fusion	0.9686	0.9648	0.9613	0.9721	0.9504	0.9699	0.9504	0.9609	0.9506	0.9611
DT_All	0.9403	0.9296	0.9388	0.9553	0.9252	0.9397	0.9230	0.9290	0.9280	0.9364
SVM_All	0.9647	0.9729	0.9705	0.9749	0.9638	0.9741	0.9489	0.9591	0.9531	0.9583

(a) True positive rate comparison

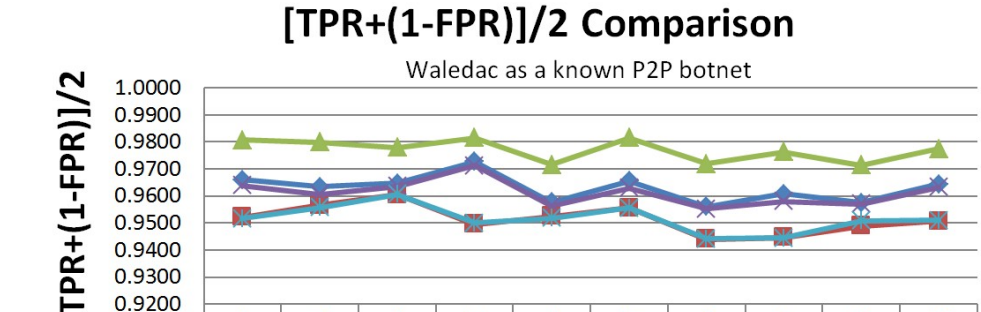
Figure 4.14: Metrics comparison (Waledac as a known P2P botnet)



	1	2	3	4	5	6	7	8	9	10
DT_GainRatio	0.0121	0.0082	0.0116	0.0128	0.0123	0.0137	0.0126	0.0125	0.0138	0.0096
SVM_FirstX	0.0619	0.0634	0.0503	0.0757	0.0609	0.0634	0.0616	0.0706	0.0557	0.0576
Fusion	0.0073	0.0051	0.0056	0.0090	0.0073	0.0067	0.0067	0.0084	0.0079	0.0062
DT_All	0.0125	0.0083	0.0118	0.0129	0.0124	0.0140	0.0127	0.0128	0.0140	0.0100
SVM_All	0.0614	0.0616	0.0493	0.0747	0.0603	0.0630	0.0607	0.0700	0.0513	0.0563

(b) False positive rate comparison

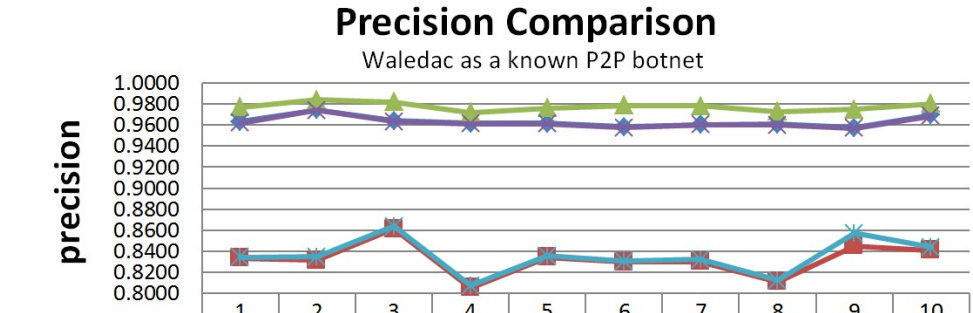
Figure 4.14: Metrics comparison (Waledac as a known P2P botnet)



	1	2	3	4	5	6	7	8	9	10
DT_GainRatio	0.9660	0.9633	0.9647	0.9727	0.9577	0.9655	0.9561	0.9607	0.9575	0.9644
SVM_FirstX	0.9521	0.9565	0.9604	0.9496	0.9525	0.9555	0.9440	0.9446	0.9489	0.9507
Fusion	0.9807	0.9799	0.9778	0.9816	0.9715	0.9816	0.9718	0.9763	0.9714	0.9775
DT_All	0.9639	0.9606	0.9635	0.9712	0.9564	0.9628	0.9552	0.9581	0.9570	0.9632
SVM_All	0.9517	0.9557	0.9606	0.9501	0.9518	0.9555	0.9441	0.9446	0.9509	0.9510

(c) ((TPR+(1-FPR))/2 comparison

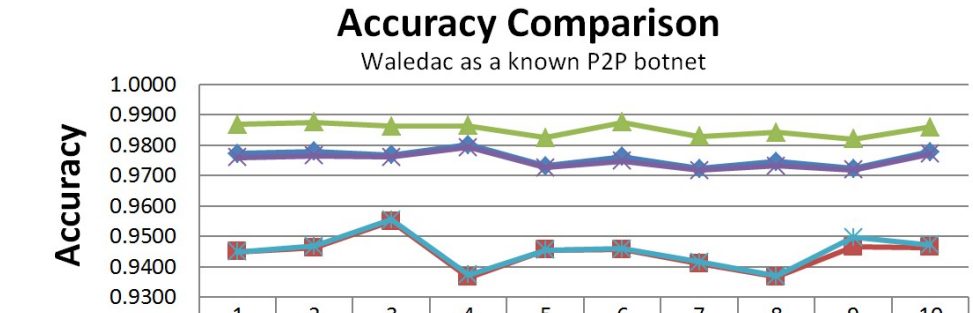
Figure 4.14: Metrics comparison (Waledac as a known P2P botnet)



	1	2	3	4	5	6	7	8	9	10
DT_GainRatio	0.9632	0.9744	0.9641	0.9615	0.9616	0.9583	0.9605	0.9609	0.9575	0.9697
SVM_FirstX	0.8336	0.8313	0.8612	0.8053	0.8345	0.8298	0.8302	0.8113	0.8453	0.8409
Fusion	0.9771	0.9838	0.9820	0.9721	0.9763	0.9785	0.9781	0.9730	0.9747	0.9802
DT_All	0.9620	0.9740	0.9634	0.9612	0.9614	0.9574	0.9602	0.9599	0.9570	0.9687
SVM_All	0.8344	0.8348	0.8636	0.8074	0.8355	0.8305	0.8322	0.8124	0.8570	0.8441

(d) Precision comparison

Figure 4.14: Metrics comparison (Waledac as a known P2P botnet)



	1	2	3	4	5	6	7	8	9	10
DT_GainRatio	0.9773	0.9780	0.9769	0.9802	0.9733	0.9763	0.9723	0.9747	0.9723	0.9779
SVM_FirstX	0.9449	0.9463	0.9549	0.9366	0.9456	0.9457	0.9411	0.9367	0.9465	0.9464
Fusion	0.9869	0.9876	0.9863	0.9864	0.9825	0.9876	0.9829	0.9842	0.9821	0.9860
DT_All	0.9760	0.9766	0.9762	0.9794	0.9726	0.9748	0.9718	0.9732	0.9720	0.9771
SVM_All	0.9449	0.9468	0.9555	0.9373	0.9455	0.9459	0.9416	0.9370	0.9498	0.9472

(e) Accuracy comparison

Figure 4.14: Metrics comparison (Waledac as a known P2P botnet)

4.5.4 Concluding remarks

This chapter conducted the experiments to evaluate the proposed approached we described in Chapter 3. Feature selection component and P2P botnet detection component were implemented and evaluated. The results were compared and it was concluded that our conversation-based P2P botnet detection approach has a good performance and better results are achieved by using decision fusion. The next chapter concludes the thesis and gives several possible points that can be made to improve or extend this work in the future.

Chapter 5

Conclusions and Future work

Chapter 1 has introduced network security and our contributions in this thesis. In Chapter 2, related background information are provided and previous related research are discussed. The proposed P2P botnet detection approach is presented in detail in Chapter 3. Evaluation of our proposed approach is given in Chapter 4. In this chapter, some conclusions of this thesis are presented and several possible future work is suggested.

5.1 Conclusion

Since the advent of the Internet, network security has always been a major concern of the Internet users. Currently, botnets are the most serious challenges in this field in recent years. The detection approach proposed in this thesis deals with the most dangerous botnet: P2P botnet. Our detection ap-

proach is based on the features that extracted from 30-second conversations between two IPs. These features are simply derived from the headers of the packets, thus, our detection approach can be used to detect encrypted traffic as well. The feature selection methods are used to remove the less informative features which may influence the results. The results shown in Chapter 4 reveals that, DT has better results with the selected features. Although the results of SVM are a little worse, the time complexity is reduced because it uses fewer features than the original feature vector. It is a trade off between time and classification performance. To sum up, both DT and SVM have desirable performance. The true positive rates are all above 92% and the false positive rates are all lower than 8%. Therefore, we have every reason to believe that our conversation-based P2P botnet detection approach works well with DT and SVM.

In order to improve the detection performance, a fusion process is proposed to use the probabilities based on the classifier models. To be more specific, for decision tree, the probability model is created according to the structure of DT. On the other hand, the probability model for SVM is based on the hyperplane and the probability is derived depending on the distance from an instance to the hyperplane. The experiments demonstrated that the decision fusion process can indeed improve the performance of the classifiers.

5.2 Future work

There are several points that can be made to improve or extend this work in the future:

1. **Introduce more machine learning techniques**

   We applied two machine learning techniques in our work. There are still many novel machine learning techniques that can be used to detect P2P botnets. For example, naïve bayes, random forest and so on. If more machine learning techniques are introduced to our decision fusion process, better results may be achieved, because the machine learning techniques are based on different theories.

2. **Use different feature selection algorithms**

   Although the feature selection algorithms we used in our thesis can help us to reduce the dimension of the feature vectors without greatly decreasing the performance. Some other feature selection algorithms can be proposed in the future in order to obtain a better feature subset.

3. **Find more informative features**

   Based on P2P conversations and review of the literature, we originally created 16 features. It is possible to find more informative conversation-based features by deeply analyzing the traffic generated by P2P botnets. These features may be helpful for improving the performance one step further.

4. **Update the classifiers when detecting in real time**

   New P2P botnet emerge from time to time. It is necessary to retrain the classifiers in order to be able to detect new P2P botnets.

Bibliography

[1] M. Feily, A. Shahrestani, and S. Ramadass, "A survey of botnet and botnet detection," in *Proceedings of the 2009 Third International Conference on Emerging Security Information, Systems and Technologies*, ser. SECURWARE '09, June 2009, pp. 268–273.

[2] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling, "Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm," in *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, ser. LEET'08, Apr. 2008, p. 9.

[3] E. Cooke, F. Jahanian, and D. McPherson, "The zombie roundup: Understanding, detecting, and disrupting botnets," in *Prodeedings of the first Workshop on Steps to Reducing Unwanted Traffic on the Internet (STRUTI)*. USENIX, July 2005, pp. 39–44.

[4] P. Wang, S. Sparks, and C. Zou, "An advanced hybrid peer-to-peer botnet," vol. 7, no. 2, Apr. 2010, p. 2.

[5] D. Dittrich and S. Dietrich, "Discovery techniques for P2P botnets," Tech. Rep., Sep. 2008.

[6] J. Goebel and T. Holz, "Rishi: identify bot contaminated hosts by IRC nickname evaluation," in *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, ser. HotBots'07, Apr. 2007, p. 8.

[7] W. Lu, G. Rammidi, and A. A. Ghorbani, "Clustering botnet communication traffic based on n-gram feature selection," *Comput. Commun.*, vol. 34, no. 3, pp. 502–514, Mar. 2011.

[8] K. Chiang and L. Lloyd, "A case study of the rustock rootkit and spam bot," in *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, ser. HotBots, Apr. 2007.

[9] N. Daswani and M. Stoppelman, "The anatomy of clickbot.a," in *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, ser. HotBots, Apr. 2007, p. 11.

[10] D. Dong, Y. Wu, L. He, G. Huang, and G. Wu, "Deep analysis of intending peer-to-peer botnet," in *Grid and Cooperative Computing, 2008. GCC '08. Seventh International Conference on*, Oct. 2008, pp. 407–411.

[11] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Revised Papers from the*

First International Workshop on Peer-to-Peer Systems, ser. IPTPS '01. London, UK, UK: Springer-Verlag, Mar. 2002, pp. 53–65.

[12] N. Ianelli and A. Hackworth, "Botnets as a vehicle for online crime," *CERT Coordination Center Technical Report*, Dec. 2005.

[13] P. Bächer, T. Holz, M. Kötter, and G. Wicherski, "Know your enemy: Tracking botnets," Published on the Web, Mar. 2005. [Online]. Available: http://www.honeynet.org/papers/bots/

[14] "Shadowserver," http://www.shadowserver.org/wiki/pmwiki.php/Stats/DDoSHistorical#toc3, 2011.

[15] A. Ramachandran and N. Feamster, "Understanding the network-level behavior of spammers," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, Aug. 2006.

[16] J. P. John, A. Moshchuk, S. D. Gribble, and A. Krishnamurthy, "Studying spamming botnets using botlab," in *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, ser. NSDI'09, Apr. 2009, pp. 291–306.

[17] M. Bailey, E. Cooke, F. Jahanian, Y. Xu, and M. Karir, "A survey of botnet technology and defenses," in *Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications Technology*, Mar. 2009, pp. 299–304.

[18] W. Strayer, D. Lapsely, R. Walsh, and C. Livadas, "Botnet detection based on network behavior," in *Botnet Detection*, ser. Advances in Information Security, W. Lee, C. Wang, and D. Dagon, Eds. Springer US, vol. 36, pp. 1–24.

[19] N. Provos, "Honeyd-a virtual honeypot daemon," in *10th DFN-CERT Workshop, Hamburg, Germany*, vol. 2, Feb. 2003.

[20] E. Balas and C. Viecco, "Towards a third generation data capture architecture for honeynets," in *Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC*, June 2005, pp. 21–28.

[21] T. H. Felix C. Freiling and G. Wicherski, "Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks," in *In Proceedings of 10th European Symposium on Research in Computer Security, ESORICS*, Apr. 2005, pp. 319–335.

[22] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proceedings of the 17th conference on Security symposium*, ser. SS'08, June 2008, pp. 139–154.

[23] M. Masud, T. Al-khateeb, L. Khan, B. Thuraisingham, and K. Hamlen, "Flow-based identification of botnet traffic by mining multiple log files,"

in *Distributed Framework and Applications, 2008. DFmA 2008. First International Conference on*, Oct. 2008, pp. 200–206.

[24] Y. Al-Hammadi and U. Aickelin, "Behavioural correlation for detecting P2P bots," in *Future Networks, 2010. ICFN '10. Second International Conference on*, Jan. 2010, pp. 323–327.

[25] A. Nummipuro, "Detecting P2P-controlled bots on the host," *TKK T-110.5290, Seminar on Network Security, Aalto University, Espoo, Helsinki*, Oct. 2007.

[26] F. Giroire, J. Chandrashekar, N. Taft, E. Schooler, and D. Papagiannaki, "Exploiting temporal persistence to detect covert botnet channels," in *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, ser. RAID '09, Sep. 2009.

[27] M. Szymczyk, "Detecting botnets in computer networks using multi-agent technology," in *Proceedings of the 2009 Fourth International Conference on Dependability of Computer Systems*, ser. DEPCOS-RELCOMEX '09, July 2009, pp. 192–201.

[28] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Using machine learning techniques to identify botnet traffic," in *In 2nd IEEE LCN Workshop on Network Security*, Nov. 2006, pp. 967–974.

[29] W. Lu, M. Tavallaee, G. Rammidi, and A. A. Ghorbani, "BotCop: An online botnet traffic classifier," in *Communication Networks and Ser-*

vices Research Conference, 2009. CNSR '09. Seventh Annual, May 2009, pp. 70–77.

[30] W. Lu and A. A. Ghorbani, "Botnets detection based on IRC-Community," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, Dec. 2008, pp. 1–5.

[31] M. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham, "A practical approach to classify evolving data streams: Training with limited amount of labeled data," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ser. ICDM '08, Dec. 2008, pp. 929–934.

[32] X. Dong, F. Liu, X. Li, and X. Yu, "A novel bot detection algorithm based on API call correlation," in *Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on*, vol. 3, Shaodong, China, Aug. 2010, pp. 1157–1162.

[33] D. Liu, Y. Li, Y. Hu, and Z. Liang, "A P2P-botnet detection model and algorithms based on network streams analysis," in *Future Information Technology and Management Engineering (FITME), 2010 International Conference on*, vol. 1, Oct. 2010, pp. 55–58.

[34] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of relieff and rrelieff," *Mach. Learn.*, vol. 53, no. 1-2, pp. 23–69, Oct. 2003.

[35] "Kdnuggets," http://www.kdnuggets.com/polls/2011/algorithms-analytics-data-mining.html, 2011.

[36] C. Ferri, P. A. Flach, and J. Hernandez-Orallo, "Improving the AUC of probabilistic estimation trees," in *Proceedings of the 14th European Conf. on Machine Learning*. Springer, Sep. 2003, pp. 121–132.

[37] I. Alvarez, S. Bernard, and G. Deffuant, "Keep the decision tree and estimate the class probabilities using its decision boundary," in *Proceedings of the 20th international joint conference on Artifical intelligence*, ser. IJCAI'07, 2007, pp. 654–659.

[38] S. Baker and B. Kramer, "Peirce, Youden, and receiver operating characteristic curves," *The American Statistician*, vol. 61, no. 4, pp. 343–346, 2007.

[39] "French chapter of honeynet," http://www.honeynet.org/chapters/france.

[40] G. Szabó, D. Orincsay, S. Malomsoky, and I. Szabó, "On the validation of traffic classification algorithms," in *Proceedings of the 9th international conference on Passive and active network measurement*, ser. PAM'08, Apr. 2008, pp. 72–81.

[41] "LBNL enterprise trace repository," http://www.icir.org/enterprise-tracing, 2005.

[42] "Information security center of excellence," http://www.iscx.ca/dataset, 2011.

Appendix A

Information gain and information gain ratio

A.1 Information gain values of five training datasets

Table A.1: Information gain values of five training datasets

Information Gain					
Feature	1WN	2WN	3WN	4WN	5WN
F_1	0.726	0.746	0.744	0.723	0.748
F_2	0.373	0.379	0.373	0.366	0.387
F_3	0.701	0.714	0.711	0.734	0.719
F_4	0.31	0.302	0.327	0.32	0.317

F_5	0.554	0.541	0.564	0.572	0.561
F_6	0.322	0.324	0.327	0.346	0.325
F_7	0.486	0.456	0.482	0.507	0.501
F_8	0.467	0.497	0.487	0.498	0.442
F_9	0.294	0.308	0.293	0.32	0.235
F_{10}	0.496	0.557	0.455	0.504	0.478
F_{11}	0.557	0.564	0.534	0.649	0.553
F_{12}	0.177	0.186	0.176	0.194	0.172
F_{13}	0.294	0.308	0.293	0.32	0.235
F_{14}	0.638	0.589	0.628	0.628	0.635
F_{15}	0.496	0.557	0.455	0.504	0.478
F_{16}	0.625	0.625	0.587	0.622	0.606

A.2 Information gain ratio values of five training datasets

Table A.2: Information gain ratio values of five training datasets

Information Gain Ratio					
Feature	1WN	2WN	3WN	4WN	5WN

F_1	0.247	0.248	0.2476	0.253	0.254
F_2	0.146	0.149	0.142	0.170	0.151
F_3	0.381	0.373	0.387	0.402	0.390
F_4	0.106	0.105	0.112	0.110	0.109
F_5	0.318	0.341	0.277	0.370	0.378
F_6	0.327	0.331	0.331	0.350	0.330
F_7	0.266	0.246	0.256	0.273	0.264
F_8	0.286	0.300	0.242	0.253	0.271
F_9	0.095	0.102	0.110	0.111	0.150
F_{10}	0.170	0.162	0.176	0.179	0.163
F_{11}	0.192	0.192	0.183	0.210	0.199
F_{12}	0.080	0.083	0.079	0.131	0.077
F_{13}	0.095	0.102	0.110	0.111	0.150
F_{14}	0.229	0.252	0.232	0.233	0.227
F_{15}	0.170	0.162	0.176	0.179	0.162
F_{16}	0.230	0.214	0.227	0.233	0.225

Appendix B

Top five feature subsets for each group (DT)

B.1 Top five feature subsets for InfoGainCFB group (DT)

Table B.1: Top five feature subsets for InfoGainCFB group(DT)

InfoGainCFB						
	[TPR+(1-FPR)]/2					
Feature Subset	1WN 6WN	2WN 7WN	3WN 8WN	4WN 9WN	5WN 10WN	Average
1,2,3,8,14,16	0.996	0.992	0.988	0.998	0.985	0.992
1,3,8,10,11,14,16	0.995	0.992	0.990	0.998	0.984	0.992

1,3,8,11,14,15,16	0.995	0.992	0.990	0.998	0.984	0.992
1,3,8,11,14,16	0.996	0.992	0.989	0.998	0.984	0.992
1,3,8,10,11,14,15,16	0.995	0.992	0.990	0.998	0.982	0.991

B.2 Top five feature subsets for GainRatioFirst*X* group (DT)

Table B.2: Top five feature subsets for GainRatioFirst$X$ group(DT)

GainRatioFirst$X$						
	[TPR+(1-FPR)]/2					
Feature Subset	1WN 6WN	2WN 7WN	3WN 8WN	4WN 9WN	5WN 10WN	Average
1 ,2,3,5,6,7,8,10,11,14,15,16	0.989	0.988	0.983	0.994	0.986	0.988
1,2,3,5,6,7,8,10,11,13,14,15,16	0.989	0.988	0.983	0.994	0.986	0.988
1,2,3,4,5,6,7,8,9,10,11,13,14,15,16	0.989	0.988	0.983	0.994	0.986	0.988
1,3,5,6,7,8,14	0.989	0.988	0.983	0.994	0.986	0.987
1,3,5,6,7,8,10,11,14,15,16	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,11,14,16	0.989	0.988	0.983	0.994	0.985	0.987
1,3,5,6,7,8	0.989	0.988	0.983	0.994	0.984	0.987

B.3 Top five feature subsets for GainRatioCFB group (DT)

Table B.3: Top five feature subsets for GainRatioCFB group(DT)

GainRatioCFB						
	[TPR+(1-FPR)]/2					
Feature Subset	1WN 6WN	2WN 7WN	3WN 8WN	4WN 9WN	5WN 10WN	Average
1,3,5,6,7,8,11	0.989	0.988	0.983	0.994	0.985	0.988
1,3,5,6,7,8,11,16	0.989	0.988	0.983	0.994	0.985	0.988
1,3,5,6,7,8,14,15	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,15	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,14	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,10	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,10,11,14,16	0.989	0.988	0.9826	0.994	0.984	0.987
1,3,5,6,7,8,10,11,14,15	0.989	0.988	0.9826	0.994	0.984	0.987
1,3,5,6,7,8,10,11,15,16	0.989	0.988	0.9826	0.994	0.984	0.987

1,3,5,6,7,8,10,11,14,15,16	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,11,14,15,16	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,11,14,16	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,11,14,15	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,11,15,16	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,10,11,16	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,10,11,15	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,10,11,14	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,10,14,15	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,10,15	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,10,14	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,10,11	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,11,15	0.989	0.988	0.983	0.994	0.984	0.987
1,3,5,6,7,8,11,14	0.989	0.988	0.983	0.994	0.984	0.987

Appendix C

Training time and test time

C.1 Training time

Table C.1: Training time comparison

Experiment	DT		SVM	
	DT_GainRatio	DT_All	SVM_First$X$	SVM_All
1	0.423	0.816	0.73511	0.98653
2	0.434	0.814	0.64431	0.78639
3	0.418	0.826	0.65485	0.74222
4	0.417	0.820	0.639	0.752
5	0.418	0.828	0.628	0.759
6	0.425	0.814	0.645	0.790
7	0.427	0.805	0.636	0.746

8	0.424	0.859	0.696	0.780
9	0.424	0.918	0.697	0.786
10	0.419	0.810	0.636	0.718
11	0.417	0.804	0.650	0.728
12	0.412	0.804	0.635	0.729
13	0.422	0.806	0.648	0.727
14	0.419	0.810	0.633	0.732
15	0.430	0.801	0.688	0.736
Average	0.422	0.822	0.658	0.767

C.2 Test time

Table C.2: Test time comparison

Experiment	DT		SVM	
	DT_GainRatio	DT_All	SVM_First$X$	SVM_All
1	0.013	0.015	2.36500	2.72222
2	0.013	0.013	2.38850	2.79110
3	0.013	0.014	2.418	2.828
4	0.013	0.013	2.443	2.907
5	0.013	0.013	2.469	3.057
6	0.013	0.014	2.464	2.915
7	0.014	0.017	2.468	2.943

8	0.013	0.013	2.483	2.805
9	0.014	0.015	2.495	2.989
10	0.015	0.013	2.569	2.989
11	0.014	0.018	2.513	3.040
12	0.013	0.013	2.550	3.035
13	0.014	0.013	2.559	3.026
14	0.013	0.013	2.574	3.128
15	0.014	0.014	2.555	3.015
Average	0.014	0.014	2.488	2.946

Vita

Candidate's full name: Shaojun Zhang
Place & date of birth: Jiangsu, China. Oct. 1, 1987

University attended:
September 2011 - January 2013
Faculty of Computer Science
University of New Brunswick
Fredericton, New Brunswick, Canada

September 2010 - June 2013
College of Software Engineering
Southeast University
Suzhou, Jiangsu, China

Bachelor of Software Engineering
September 2006 - June 2010
College of Software Engineering
Southeast University
Nanjing, Jiangsu, China

Publications:
Shaojun Zhang, Ali Ghorbani, "Conversation-based P2P Botnet Detection with Decision Fusion," Poster at the 9th Research Exposition of UNB Faculty of Computer Science, April 2012, Fredericton, NB, Canada.