

國立交通大學

網路工程研究所

碩士論文

基於模糊識別演算法之僵尸網路偵測方法



A Fuzzy Pattern Recognition-based

Filtering Algorithm for Botnet Detection

研究生：林上智

指導教授：王國禎 博士

中華民國九十九年六月

基於模糊識別演算法之僵尸網路偵測方法
A Fuzzy Pattern Recognition-based
Filtering Algorithm for Botnet Detection

研究生：林上智

Student : Shang-Jyh Lin

指導教授：王國禎

Advisor : Kuochen Wang

國立交通大學

網路工程研究所



Submitted to Institute of Network Engineering

College of Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2010

Hsinchu, Taiwan, Republic of China


中華民國九十九年六月

基於模糊識別演算法之僵屍網路偵測方法

學生：林上智 指導教授：王國禎 博士

國立交通大學網路工程研究所

摘要



由於目前僵屍網路的盛行，造成分散式阻斷攻擊、垃圾郵件及釣魚郵件散佈、非法儲存或偷取智慧財產等網路犯罪的問題。傳統的字串比對偵測方法容易有誤判及漏判的問題。為了解決這些問題，在本論文中，我們提出一種基於模糊識別演算法之僵屍網路偵測方法，簡稱 FPRF，來分析網路流量以偵測僵屍網路。我們是根據網路封包行為模式來做僵屍網路之分析與偵測。FPRF 分成三個階段，第一個階段是利用僵屍網路的特性來過濾掉不需檢查的封包。第二個階段則是取出封包流量的特徵。在最後階段我們利用模糊辨識方法來偵測僵屍網路。為了評估此方法的有效性，我們收集了真實僵屍網路流量及校園正常流量來評量我們的方法。實驗結果顯示，我們提出的 FPRF 對

於殭屍網路的流量辨識正確率高達 95% 以上，且對於正常網路流量只有 0 ~ 3.08% 的誤判率。此外，封包縮減率達 70% 以上，如此可達到快速而有效的辨識率。

關鍵詞：殭屍網路、模糊識別、網路安全、真實流量。



A Fuzzy Pattern Recognition-based Filtering Algorithm for Botnet Detection

Student : Shang-Jyh Lin

Advisor : Dr. Kuochen Wang

Department of Computer Science

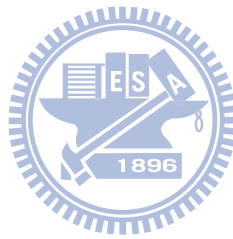
National Chiao Tung University

Abstract

Botnets become a popular technique for deploying Internet crimes. Existing botnet detection methods using string signature matching usually get high false positive rate (FPR) and low true positive rate (TPR). Therefore, the behavior-based detection method becomes a major way for botnet detection. In this thesis, we propose a behavior-based botnet detection method using a *fuzzy pattern recognitions-based filtering* (FPRF) algorithm. The proposed FPRF extracts bot features first and then recognizes botnets based on collected bot behaviors. In this algorithm, there are three stages. The *traffic reduction* stage is to reduce input raw packet traces for speeding processing. The *feature extraction* stage is used to extract features from the reduced input packet traces. The *fuzzy pattern recognition* stage has two phases. First, the *DNS* (domain name system) *phase* analyzes features of DNS packets. If a domain name (DN) is determined to be malicious, the corresponding DN and its associated IP address(es) will be marked without going to the next phase. Second, the *TCP connection phase* analyzes features of TCP connection packets. The associated IP addresses will be marked if TCP connection packets are malicious. Performance evaluation results based on real traces show that with features extracted from raw

network traces, the proposed FPRF can reduce input raw packet traces by over 70%, while achieve a high TPR (95%) and a low FPR (0 ~ 3.08%). Unlike two representative methods, Livadas and Gu, we used real botnet traffic and only one traffic reduction filter for evaluation. Furthermore, FPRF is resource-efficient so that on-line botnet detection based on FPRF can be incorporated to hosts.

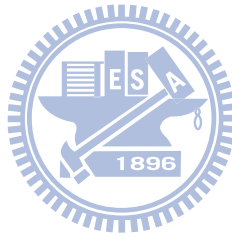
Keywords: Botnet, fuzzy pattern recognition, network security, real trace.



Acknowledgements

Many people have helped me with this thesis. I deeply appreciate my thesis advisor, Dr. Kuochen Wang, for his intensive advice and guidance. I would like to thank all the members of the *Mobile Computing and Broadband Networking Laboratory* (MBL) for their invaluable assistance and suggestions and *Network Benchmarking Lab* (NBL) for their botnet samples. The support by the National Science Council under Grant NSC 97-3114-E-009-001 is also gratefully acknowledged.

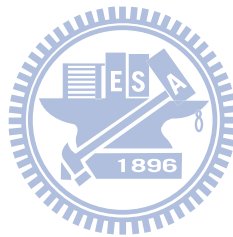
Finally, I thank my family for their endless love and support.



Contents

Abstract (in Chinese)	i
Abstract	iii
Contents	vi
List of Figures	viii
List of Tables	ix
Abbreviations List	1
Chapter 1 Introduction	2
Chapter 2 Background and Related Work	4
2.1 Overview of botnet behaviors	4
2.2 Existing methods for traffic reduction	6
2.3 Related work	7
Chapter 3 Proposed Fuzzy Pattern Recognition Filtering Algorithm	9
3.1 Problem statement.....	9
3.1.1 The main problem.....	9
3.1.2 The sub-problems	9
3.2 Design of a botnet detection algorithm.....	10
3.3 Traffic reduction.....	11
3.4 Feature extraction.....	13
3.4.1 DNS packets	14
3.4.2 TCP connection packets	15
3.5 Fuzzy pattern recognition	16
3.5.1 DNS phase	18
3.5.2 TCP connection phase	19
Chapter 4 Evaluation	21

4.1 Traces collection	21
4.2 Test results for botnet traces.....	22
Chapter 5 Conclusion	26
5.1 Concluding remarks	26
5.2 Future work.....	26
References.....	27

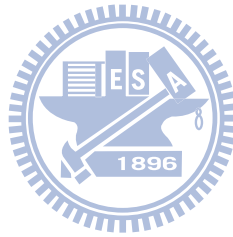


List of Figures

Figure 1. Botnet working scenario: infection and attack phases.....	5
Figure 2. Fuzzy pattern recognition filtering (FPRF) algorithm for identifying botnet DNs/IP addresses.	11
Figure 3. The traffic reduction procedure.	12
Figure 4. The procedure of bot activation.....	13
Figure 5. The distribution of botnet DNS query packets: single fixed interval pattern [x: minutes; y: number of DNS packets].....	14
Figure 6. The distribution of botnet DNS query packets: multiple fixed intervals pattern [x: seconds; y: number of DNS packets].....	15
Figure 7. The distribution of DNS query and TCP connection packets [x: seconds y: number of DNS query or TCP connection packets].....	15
Figure 8. Packets distribution of botnet traffic [x: minutes; y: number of packets]....	16
Figure 9. Bytes distribution of botnet traffic [x: minutes; y: number of bytes].	16
Figure 10. Procedure of the fuzzy pattern recognition stage.	17
Figure 11. Fuzzy pattern recognition based on MAX membership principle.	18
Figure 12. Experimental environment for botnet traces collection.....	21
Figure 13. DNs/IP addresses false negative rate vs. ρ	22
Figure 14. IP addresses fault positive rate vs. ρ	24

List of Tables

Table 1. Botnet traces statistics and detection rates ($\rho = 7$).	23
Table 2. Statistics of active/inactive malicious DNS/IP addresses ($\rho = 7$).	23
Table 3. Normal traces statistics and DNS/IP addresses's fault positive rate ($\rho = 7$). ..	24
Table 4. Comparison of behavior-based botnet detection methods.....	25



Abbreviations List

The list contains the main abbreviations used throughout this thesis.

C&C: Command and control

DDoS: Distributed denial of service

DN: Domain name

DNS: Domain name system

FN: False negative

FNR: False negative rate

FP: False positive

FPR: False positive rate

FPRF: Fuzzy pattern recognition filtering

TP: True positive

TPR: True positive rate



Chapter 1

Introduction

A botnet, or the army of bots (zombies), is comprised of more than thousands or tens of thousands of compromised computers. Although statistics show that the number of botnets are increasing [6], most Internet users are still unaware of what is going on and how serious the problem is. Many of these users' computers are easily got infected by bot malwares and then become a botnets' members. Since bot malwares usually do not affect regular uses of infected computers, bot masters, also called bot herders, can control these infected computers remotely and ask them to carry out malicious activities such as sending SPAMs [5], launching distributed denial of service (DDoS) attacks, and stealing personal privacy information.

Bot detection solutions can be classified into two categories, i.e., *signature-based* and *behavior-based* solutions. Although a signature-based solution has a high precision, it has the following drawbacks. First, signature-based solutions cannot detect unknown bots. Second, a string signature is for a specific bot. When a bot has bot variants, even with the same malicious behaviors, the string signature does not work for these variants. That is, different string signatures are needed to detect different bot variants. Hence, the false negative rate (FNR) may increase when new bots are developed. Third, as the number of distinct bots increases, the false positive rate (FPR) may be increased as well. This is because an extremely large database containing all identified bots' signatures may accidentally match benign software. Finally, a bot is able to easily bypass signature-based checks by using techniques such as code obfuscations and mutations.

On the contrast, behavior-based solutions try to identify bot activities by using

observed particular bot behaviors. If well tuned, behavior-based solutions are able to perform better than signature-based solutions in terms of detection rates. In addition, to detect unknown bots, a behavior-based solution does not need to maintain a signature database.

In this thesis, we propose a behavior-based solution to detect malicious domain names (DNs) and IP addresses used by botnets. The contribution of the thesis is three-fold. First, we propose an effective input traffic reduction algorithm to reduce the amount of input traffic that is required to be checked by the bot detection algorithm. Second, we make in-depth observations to bot activities and then extract proper behavior features to detect bots in the internet. Third, a bot detection algorithm based on fuzzy pattern recognitions is proposed. Evaluations show that the proposed bot detection solution has good detection rates. In addition, it is able to detect various types of bots including IRC, HTTP, and peer-to-peer (P2P) bots.

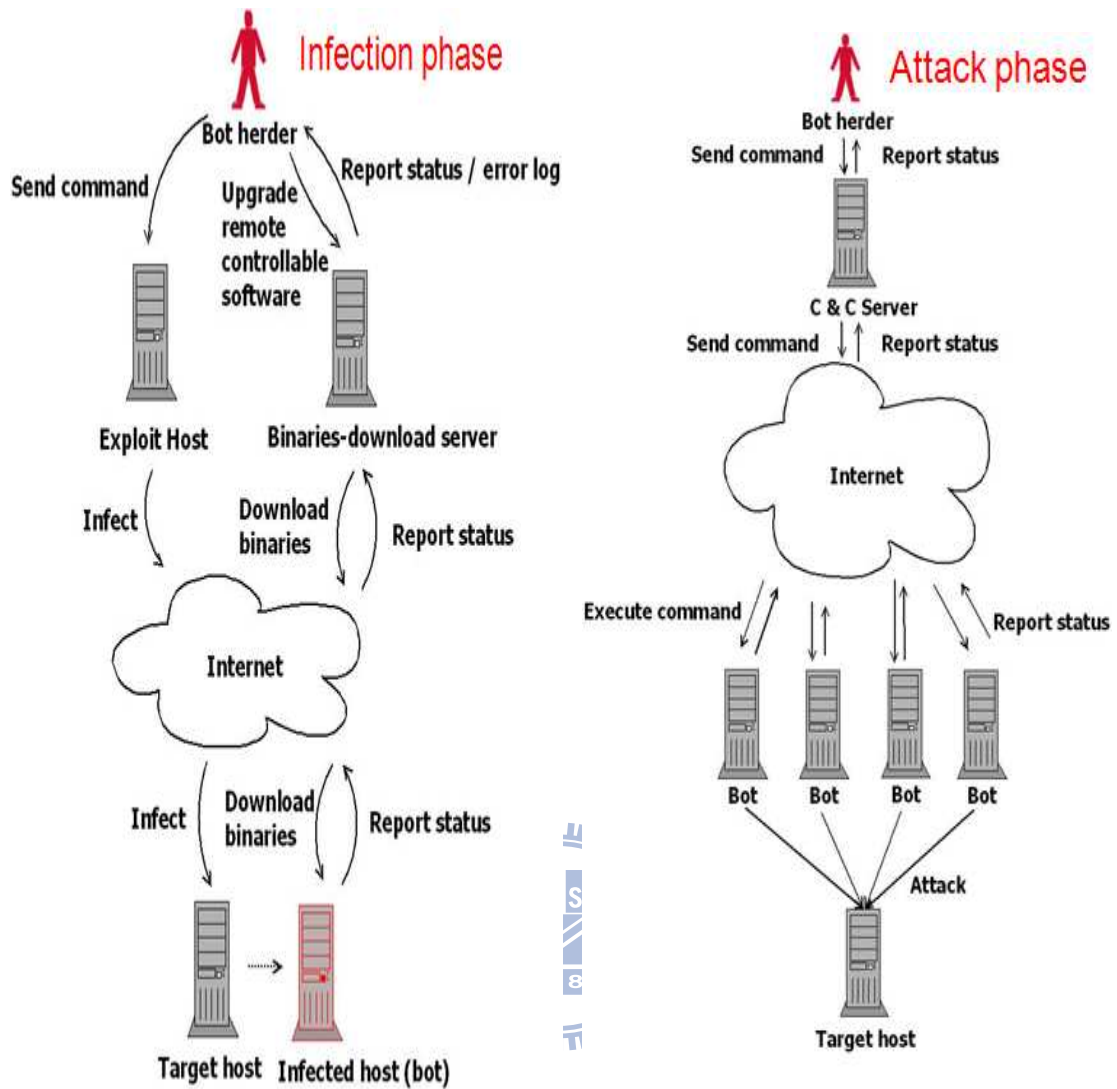
The rest of this thesis is organized as follows. Chapter 2 overviews of botnet behaviors and reviews related work. Chapter 3 formally defines main problem and sub-problems the proposed solution is going to resolve and explains the design of the proposed solution, which is named as the fuzzy pattern recognition-based filtering (FPRF) algorithm. Chapter 4 presents the experiment environment and discusses experimental results. Finally, a concluding remark and future work are given in Chapter 5.

Chapter 2

Background and Related Work

2.1 Overview of botnet behaviors

The working scenario of a botnet can be classified into two phases. One is the *infection phase* and the other is the *attack phase*, as shown in Figure 1. In the infection phase, as shown in Figure 1(a), a bot herder sends a command and tries to intrude and compromise a regular user's computer and make it become one member of the botnet. There are many methods to intrude a computer, such as exploiting software/hardware vulnerabilities and social engineering. Once the intrusion is successful, the infected computer sends its status to the bot herder and tries to install a remote controllable software, which was downloaded from a binaries-download server. The binaries-download server is responsible for reporting infected hosts status, error log and receiving the upgraded software. In the attack phase, a bot herder sends commands to compromised hosts, i.e., the bots. On receipt of the commands, the bots act based on the instructions embedded in the commands. A bot herder is therefore able to ask bots to collect valuable information, report botnet status, and launch attacks to target hosts.



(a) Bot herder infects a normal host.

(b) Bot herder sends commands to infected hosts (bots).

Figure 1. Botnet working scenario: infection and attack phases.

2.2 Existing methods for traffic reduction

To improve botnet detection efficiency, existing researches have tried to reduce the amount of input traffic by filtering out bot-irrelevant traffic. A good input traffic reduction algorithm is able to reduce the data to be processed and can also reduce a FPR. Some common criteria used to filter out input traces are listed as follows :

- *Eliminate all port-scanning activities* [4][14][15]

A TCP packet containing only SYN and RST flags are filtered out.

- *Ignore P2P* [4][14][15]

If a detection solution focuses only on IRC botnets, it often filters out P2P traffic and hence gets a significant traffic reduction rate. However, such a solution cannot detect P2P bots.

- *Skip short lived flows* [4][14][15]

Filtered out flows of only a few packets or a few seconds. These do not correspond to bots that are standing by “at the ready” [4].

- *Filter out based on black and white lists* [9]

If the source or destination address of a packet is well-known, it is often not necessary to check it. Hence, these packets can be safely ignored.

The above criteria raise some concerns. First, ignoring P2P traffic removes the possibility of identifying P2P bots. Second, skipping short lived flows may cause failures in detecting inactive botnet traffic. An inactive bot is a host that was compromised, but it cannot connect to its command and control (C&C) server temporarily, due to a C&C server’s IP address that is unavailable or a C&C server that is out of bot herder control. If a bot can connect to its C&C server, it is an active bot. Third, extra efforts are needed to maintain black and white lists, and one needs to keep track and update

the state of each member in the black (white) list. For example, a host in the black list may clean up its botnet program someday, but it is still in the black list. A host in the white list may have the similar situation.

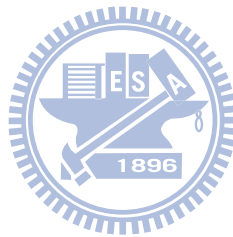
2.3 Related work

Kolbitsch et al. [3] proposed a host-based malware detection method. They use behavior graphs to match a stream of system call invocations and determine whether a program is malicious or not. The method has to be trained before it can be used. Although it is possible to detect variants of a malware program, it may fail on detection of unknown malware. The evaluation showed that the detection rate is only 64% and there is no report on FPR. In addition, its target applications only involve Explore, Email software, putty and text-editor. Its detection method might fail in other applications.

Livadas et al. [4] developed a machine learning technique to identify the C&C traffic of IRC-based botnets. The solution focuses only on IRC traffic. In the first stage, they extract traffic attributes, including flow duration, maximum initial congestion window, average bytes per packet for a flow, and so on. In the second stage, they use a Bayesian network classifier to make the classification achieve the best trade-off between FNR and FPR. However, the FPR (15.04%) is still high. Their results can be improved by using more training data. AsSadhan et al. [8] tried to identify botnet C&C traffic and they found that this type of traffic can be observed periodically. However, the observation was made for a simulated bot, not a real world bot. The validness of the results needs to be further examined by using real world bots.

Choi et al. [12] focused on features of DNS traffic only. They proposed a botnet detection mechanism that monitors DNS traffic to detect botnets, which form a group

activity in DNS queries simultaneously sent by distributed bots [12]. Their anomaly-based botnet detection mechanism can detect the variants of bots by looking at their group activities in DNS traffic [12]. Evaluation showed that a botnet can be detected by a similarity value. Gu et al. [9] proposed a “botsniffer.” They used features of spatial-temporal correlation and similarity and statistical algorithms to detect botnets [9]. They used real traces, reimplement botnet traces, and self-produced botnet traces to evaluate the proposed solution and the result showed high detection rate and low FPR, but their real traces sample are not large.



Chapter 3

Proposed Fuzzy Pattern Recognition Filtering Algorithm

3.1 Problem statement

3.1.1 The main problem

Given a network packet trace, the goal of the proposed solution is to identify the domain names (DNs) and IP addresses used by botnet C&C servers. These DNs and IP addresses can be classified into two categories, i.e., *active* and *inactive*. Active DNs or IP addresses can be used to reach C&C servers and hence are used to report information, send feedback, and receive commands. On the contrast, inactive DNs and IP addresses can be used to reach C&C servers but they do not work for some reasons. A domain name may be inactive if the DNS server in charge has no mapping record. An IP address may be inactive because the host is shutdown. It is possible that an inactive one becomes active again. Therefore, it is important for us to identify and maintain both categories of records.

3.1.2 The sub-problems

- *Traffic reduction*

Input raw packet traces contain many different types of packets, but most of them are not relevant to botnet detection. In other words, packets irrelevant to botnet detection could be filtered. With a high traffic reduction rate, it makes a botnet detection algorithm run in a more efficient way.

- *Feature extraction*

Bots usually operate with particular behaviors. These behaviors are usually different from regular users' behaviors and hence features of these behaviors can be extracted for bots detection. An ideal feature is that it is applicable to as many bots as possible and is not just limited for a specific bot.

- *Pattern recognition*

Once distinguishable features are extracted, we need a good pattern recognition solution to identify bots based on the extracted features. A good solution should be able to correctly classify input traffic. The solution cannot be too complex. It must be efficient so that a classification can be made within a short period of time. In addition, it must have a high true positive rate (TPR) and low false positive rate (FPR). Although the FPR often increases with the TPR, it is a trade-off between the two rates.

3.2 Design of a botnet detection algorithm

The proposed *fuzzy pattern recognition filtering* (FPRF) algorithm for botnet detection is shown in Figure 2. There are three stages in the algorithm: *traffic reduction*, *extraction feature* and *fuzzy pattern recognition*. First, input traffic is passed to the traffic reduction stage. Then, filtered packets are passed to the feature extraction stage. Finally, the fuzzy pattern recognition stage is used to detect malicious DNS and IP addresses based on extracted features.

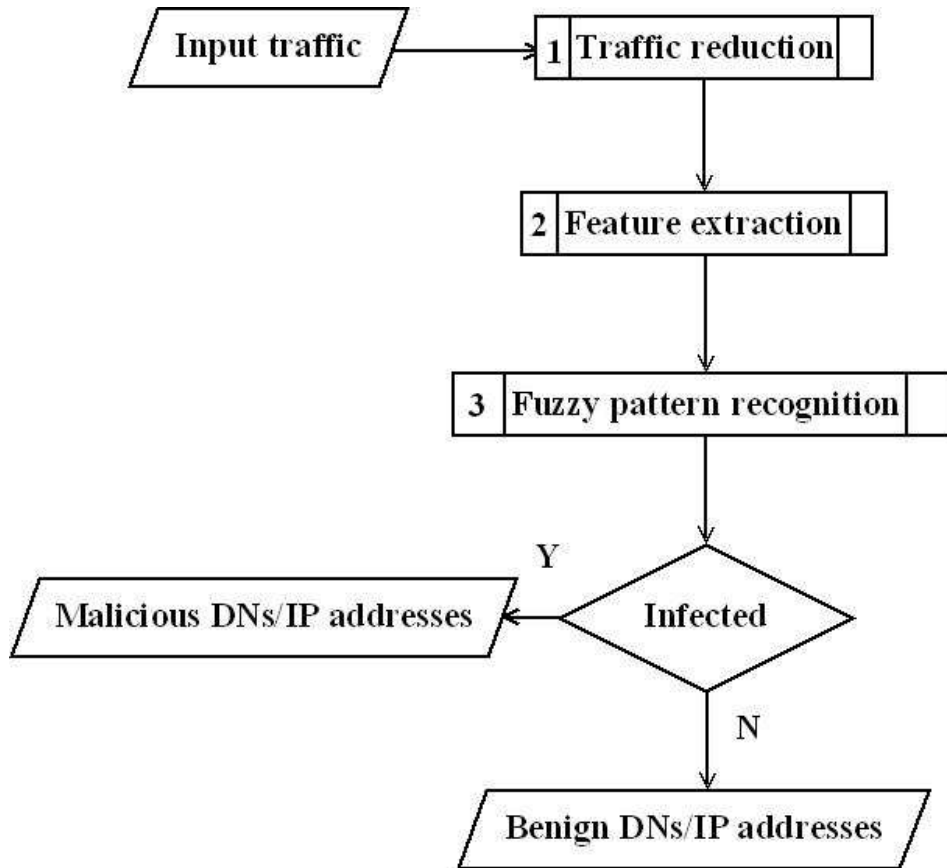


Figure 2. Fuzzy pattern recognition filtering (FPRF) algorithm for identifying botnet DNs/IP addresses.

3.3 Traffic reduction

We have discussed the pros and cons of several existing traffic reduction methods in Section 2.2. It is true that a good traffic reduction filter can reduce the data needed to be processed and also increase classification accuracy. However, if a filter eliminates data improperly, the bot detection rate may be affected. In addition, if too many filters are used, the traffic reduction time may be too long. Therefore, the use of traffic reduction filters must be carefully considered.

In the proposed solution, we use only one intrinsic traffic reduction filter, as shown in Figure 3. To prevent botnets from being detected, it is common for bots to dynamically retrieve the IP addresses of C&C servers. In addition, a bot herder may

register several DNSs and asks the bots to look up the IP addresses of these DNSs. As a result, bots need to send domain name system (DNS) queries frequently to get IP addresses which are currently being used by C&C servers.

Since bots activities always involve DNS queries, this characteristic can be used to filter out botnet-irrelevant traffic. Based on this feature, we check DNS query and response packets and put returned IP addresses from the DNS into an IP address list. A packet is sent to the feature extraction stage if and only if its source or destination address is listed in the IP address list.

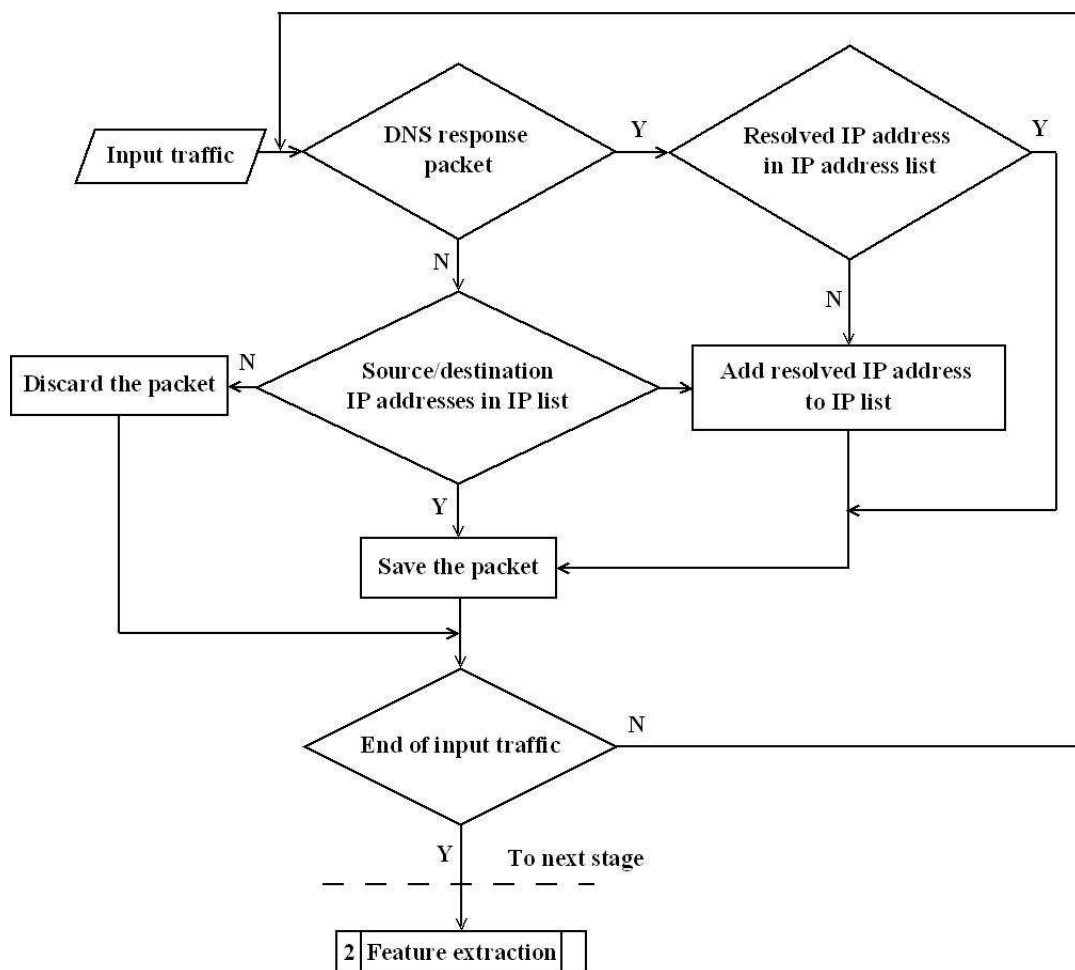


Figure 3. The traffic reduction procedure.

3.4 Feature extraction

A bot activity often starts by sending a DNS query, as shown in Figure 4. If a DN cannot be resolved or the resolved IP addresses cannot be used to connect to C&C server (invalid IP addresses), the bot is inactive. On the contrary, if one of the resolved IP addresses is valid and the bot is able to connect to the C&C server, it is an active bot.

As mentioned in section 2.2, we divided bots into two types: active and inactive bots. For an active bot, it can establish malicious connections that involve malicious behaviors. As to an inactive bot, malicious connections cannot to establish due to no DNS record or the resolved IP addresses unable to be used to make TCP connections. Both types of bots are malicious and can be detected and identified by our algorithm.

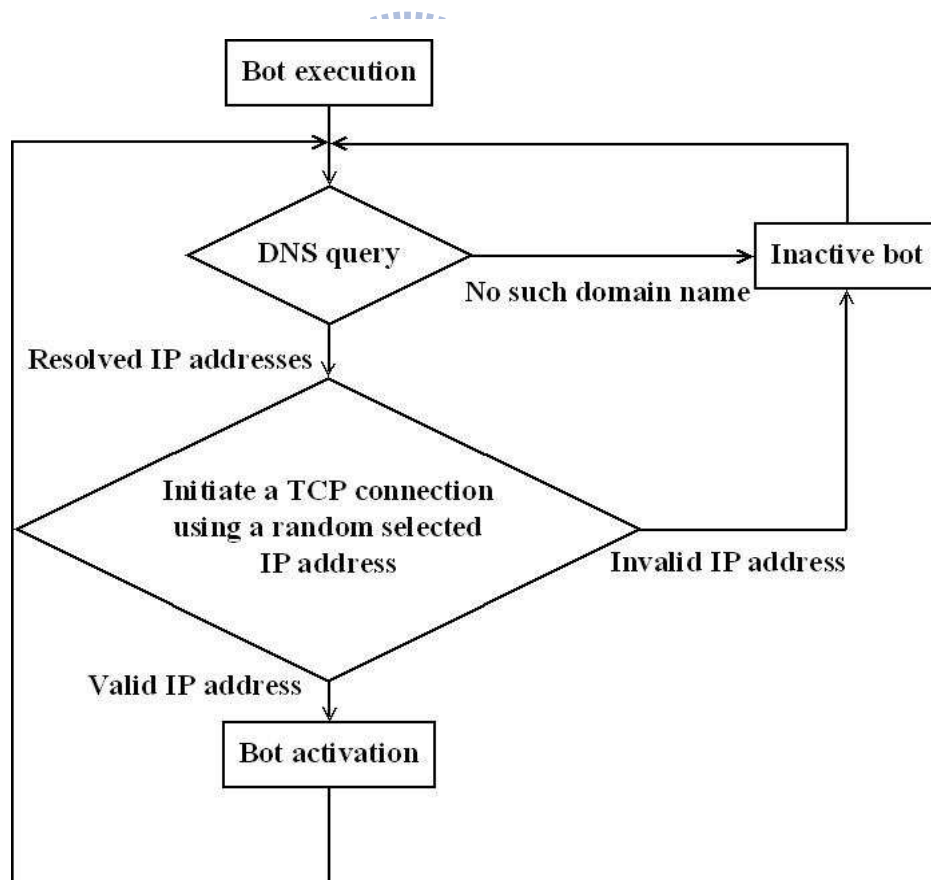


Figure 4. The procedure of bot execution.

3.4.1 Feature extraction from DNS packets

As we mentioned before, bots activities always involve DNS queries. Based on the bot traces we collected, we found that DNS queries from bots are often periodical. These periodical DNS queries can be further classified into two types. The first type is a single fixed interval pattern, as shown in Figure 5. The interval between any two DNS query packets for a single bot is fixed. In Figure 5, the interval pattern is {15, 15, 15, ...}. The second type is a multiple fixed interval pattern, as shown in Figure 6. The multiple fixed interval pattern shown in Figure 6 is { {1, 1, 2, 4}, {1, 1, 2, 4}, ...}. In Figure 7 shows the relationship between DNS queries and TCP connection packets.

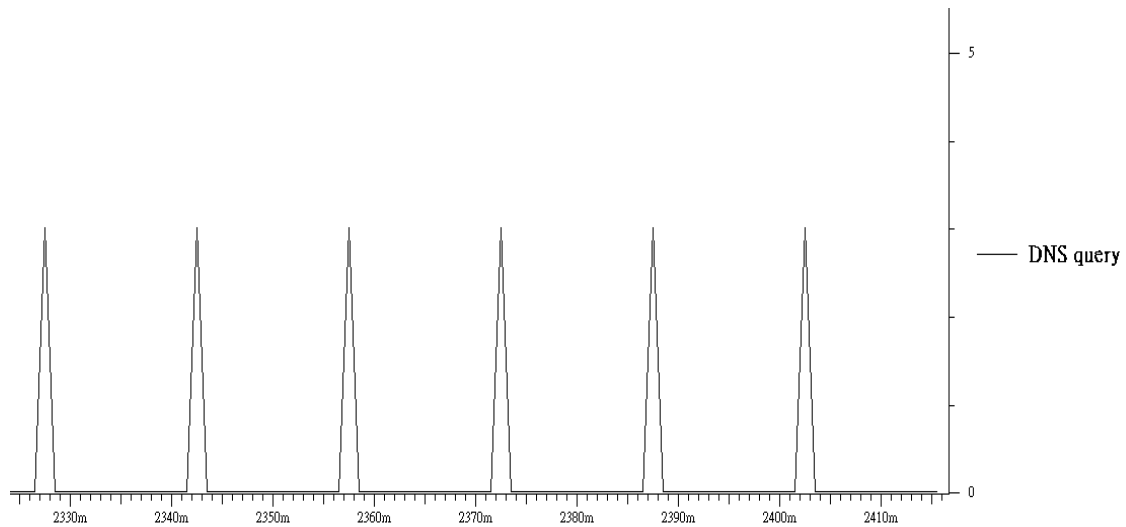


Figure 5. The distribution of botnet DNS query packets: single fixed interval pattern.

[x axis: minutes; y axis: number of DNS packets]

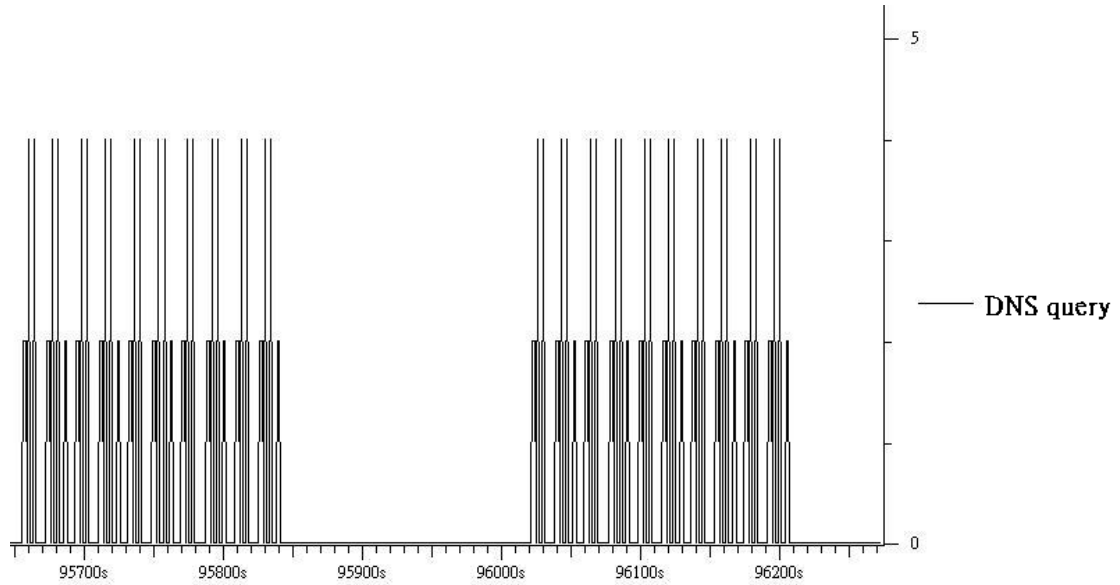


Figure 6. The distribution of botnet DNS query packets: multiple fixed intervals pattern. [x : seconds; y : number of DNS packets]

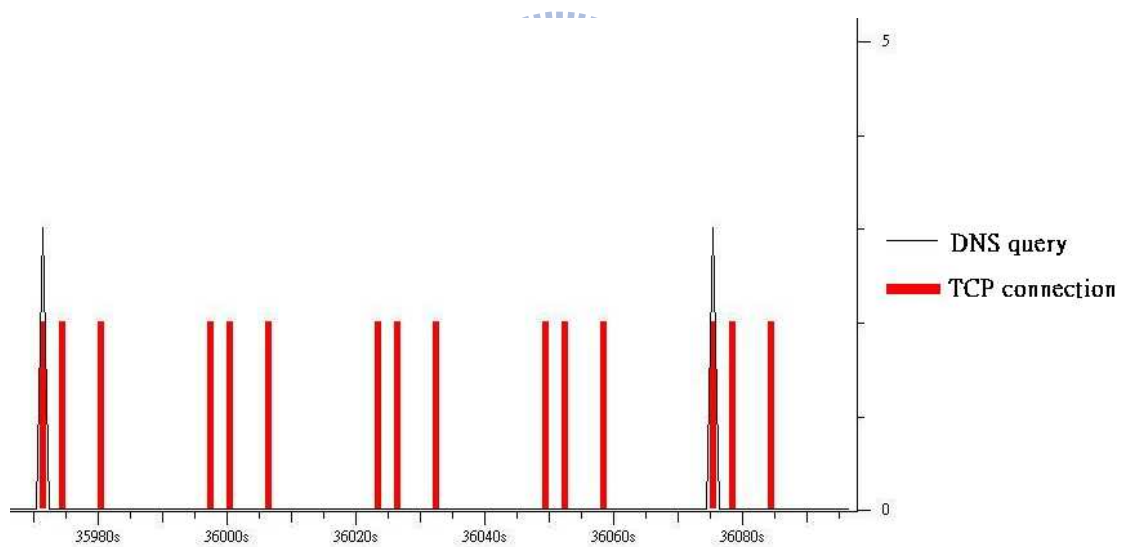


Figure 7. The distribution of DNS query and TCP connection packets. [x : seconds y : number of DNS query or TCP connection packets]

3.4.2 Feature extraction from TCP connection packets

Figures 8 and 9 show the packets and bytes distributions of botnet traffic, respectively. Both figures show the packets and bytes distributions of botnet traffic are periodical.

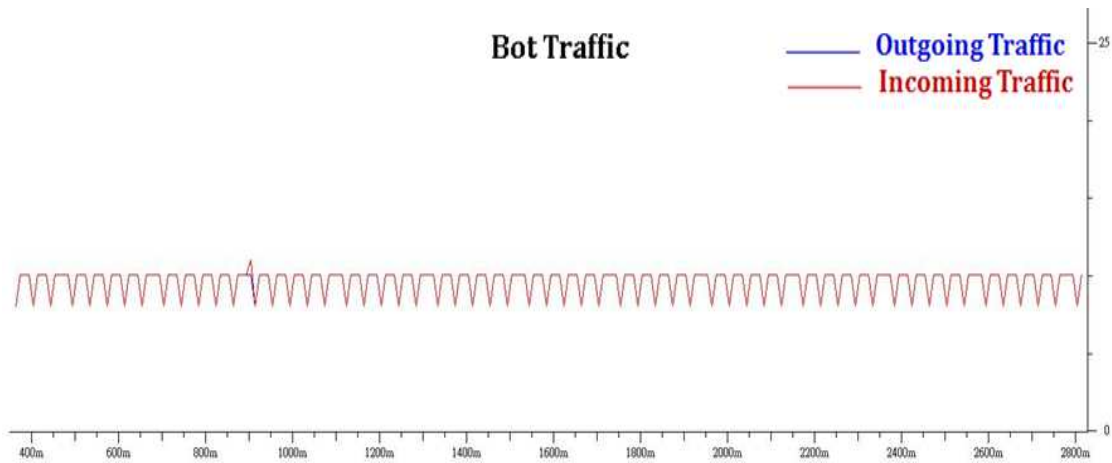


Figure 8. Packets distribution of botnet traffic. [x : minutes; y : number of packets]

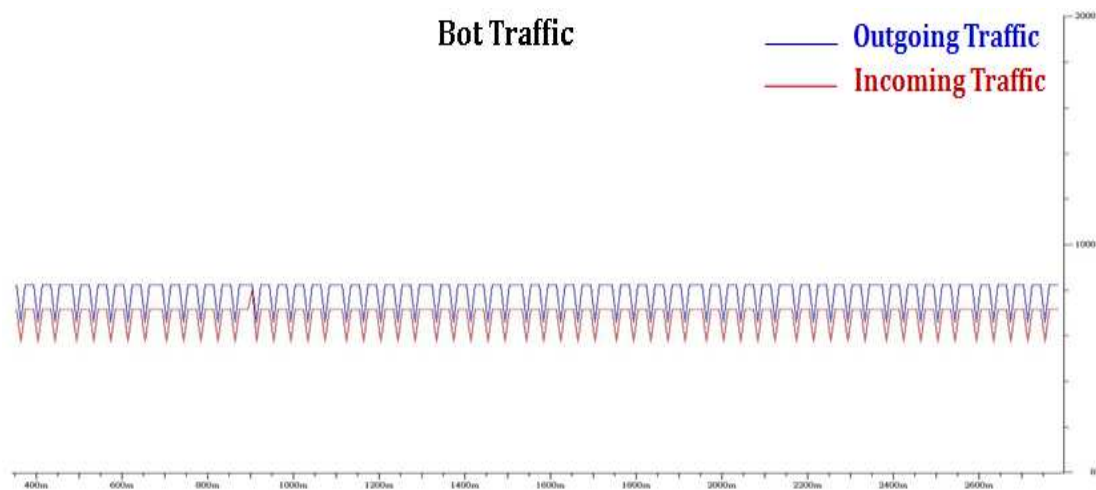


Figure 9. Bytes distribution of botnet traffic. [x : minutes; y : number of bytes]

3.5 Fuzzy pattern recognition

To prevent from being detected, bots often try to simulate human-like behaviors. To resolve this problem, we use fuzzy pattern recognition to detect bots.

In the proposed FPRF algorithm, there are two phases, *DNS phase* and *TCP connection phase*, as shown in Figure 10. In the DNS phase, we detect a bot based on DNS features. If a DN does not belong to an inactive or active bot, the DN is passed to the TCP connection phase. In the TCP connection phase, we detect a bot based on TCP connection features. If an IP address does not belong to an inactive or active bot,

the IP address is benign. If a DN failed to pass in the DNS phase, the associated IP addresses are usually also malicious. We do not need to pass these IP addresses to the TCP connection phase for filtering. Both phases use fuzzy pattern recognition to classify bot and non-bot behaviors based on the MAX membership principle, as shown in Figure 11. Each membership function corresponds to a state and has a membership value.

The FPRF algorithm will find a MAX membership value, and the DN or IP address is in the state associated with this membership value.

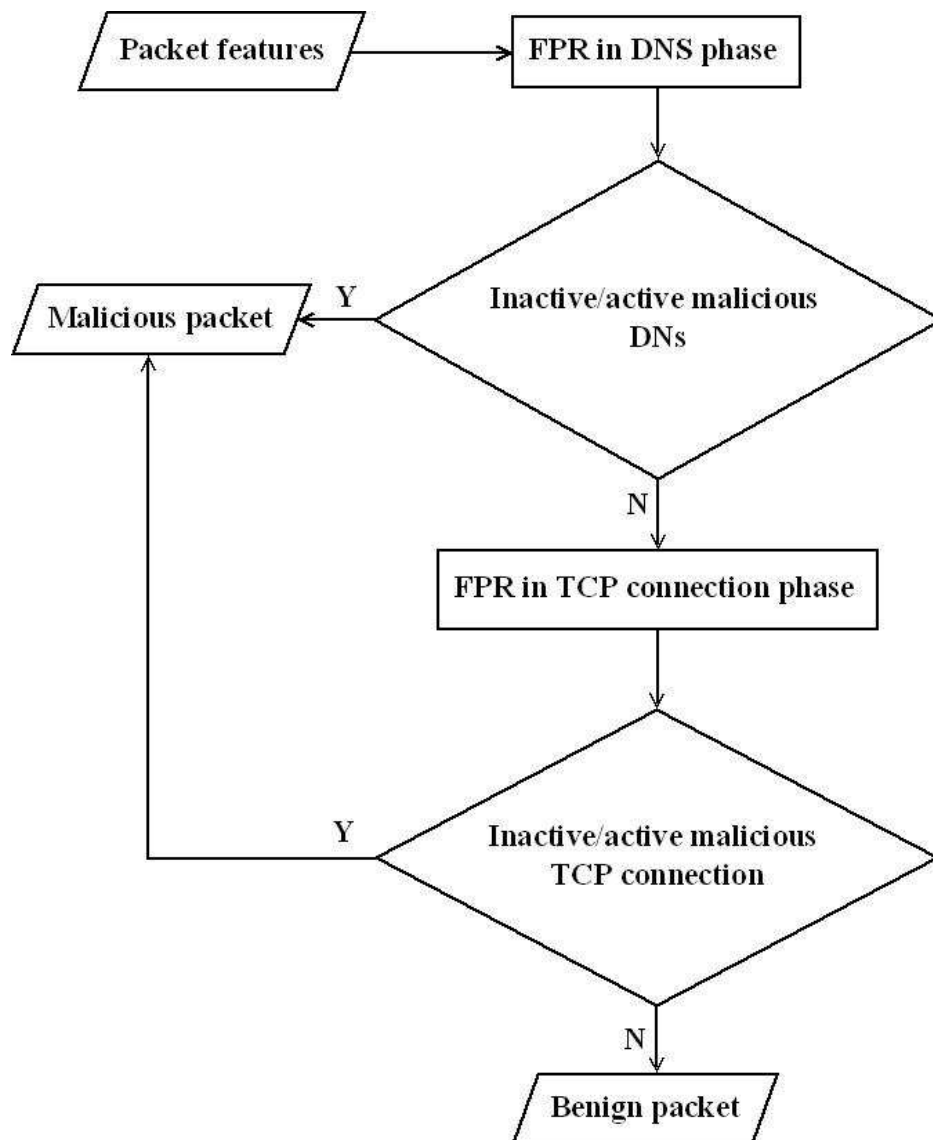


Figure 10. Procedure of the fuzzy pattern recognition stage.

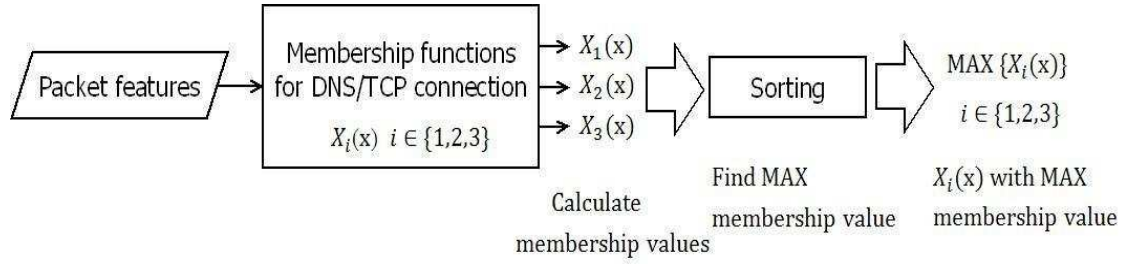


Figure 11. Fuzzy pattern recognition based on maximum membership principle.

3.5.1 DNS phase

In the DNS phase, we define a packet features vector $x = (\alpha, \beta, \gamma)$, where α is a set of number of occurrences for a specific interval time between the current response packet and the next request packet, $\alpha_i \in \alpha, 0 \leq i \leq n$, where i is the length of the interval time, and n is the length of the MAX time interval; β is the number of DNS responses; γ is the number of failed DNS responses. In this phase, we define the following three states and their associated membership functions

- *Inactive malicious DNS query*

An inactive malicious DNS query usually receives a failed DNS response. More failed DNS responses lead to a higher membership value. We define a membership function X_1 for calculating the probability of being an inactive malicious DNS query. By observation, we chose 3 for threshold σ .

$$X_1(x) = \begin{cases} 1 - \frac{|\beta - \gamma|}{\beta}, & \gamma \geq \sigma \\ 0, & \text{otherwise} \end{cases}$$

σ : the threshold for number of failed DNS responses (1)

- *Malicious DNS query*

A malicious DNS query usually has the same interval time. If the DNS query interval time is fixed, it might be a malicious DNS query. We define a membership function X_2 for calculating the probability of being a malicious DNS query.

$$X_2(x) = \begin{cases} \frac{\text{MAX}(\alpha_i)}{\sum_{k=0}^n (\alpha_k)} & \text{for } i \in \{0, 1, 2, \dots, n\}, \\ 0 & , \quad \sum_{k=0}^n (\alpha_k) \cong \rho \\ & \text{otherwise} \end{cases}$$

ρ : the threshold for the number of time intervals between the current response packet and the next request packet (2)

- *Normal DNS query*

We define a membership function X_3 for calculating the probability of being a normal DNS query.

$$X_3(x) = 1 - \text{MAX}\{X_1(x), X_2(x)\} \quad (3)$$

3.5.2 TCP connection phase

In the TCP connection phase, we define a packets features vector $x = (\alpha, \beta, \gamma)$, where α is a set of number of occurrences for a specific interval time between the current response packet and the next request packet, $\alpha_i \in \alpha, 0 \leq i \leq n, n$ is belong to maximum time interval; β is the number of TCP connection requests; γ is a set of payload sizes, $\gamma_i \in \gamma, 0 \leq i \leq n, n$ is belong to maximum payload size. In this phase, we define the following three states and their associated membership functions.

- *Inactive malicious IP address*

If a TCP connection sent many requests but did not receive any response, it is highly probable that it is associated with an inactive malicious IP address.

We define a membership function X_1 for calculating the probability of

being an inactive malicious IP address. By observation, we chose 3 for threshold σ .

$$X_1(x) = \begin{cases} 1, & \sum_{k=0}^n (\alpha_k) = 0, \beta \geq \sigma \\ 0, & \text{otherwise} \end{cases}$$

σ : the threshold for number of TCP connection request packets (4)

- *MaliciousIP address*

Malicious IP addresses are usually associated with similar packet payload sizes. We ignore packets with zero payload size. We define a membership function X_2 for calculating the probability of being a malicious IP address.

$$X_2(x) = \begin{cases} \frac{\text{MAX}(\gamma_i)}{\beta - \gamma_0} & \text{for } i \in \{1, 2, \dots, n\}, \quad \beta - \gamma_0 \geq \rho \\ 0 & , \quad \text{otherwise} \end{cases}$$

ρ : the threshold for number of TCP connection request packets (5)

- *Normal IP address*

We define a membership function X_3 for calculating the probability of being a normal IP address.

$$X_3(x) = 1 - \text{MAX}\{X_1(x), X_2(x)\} \quad (6)$$

Chapter 4

Evaluation

4.1 Traces collection

To generate real botnet traces, we ran unpatched Windows XP SP3 in a virtual machine using Ubuntu and Virtualbox, and executed 100 real bot samples inside a Windows environment (HoneyTrap). We used a share folder to manage honeytraps, as shown in Figure 12. Both input and output network traffic of the virtual machine were recorded and stored in a database via DBInserter. Among 100 bots, only 44 bots had network traces. The rest of bots were not executable. The botnet traces were recorded for 48 hours. Both the packet header and complete packet payload were stored for further botnet analysis.

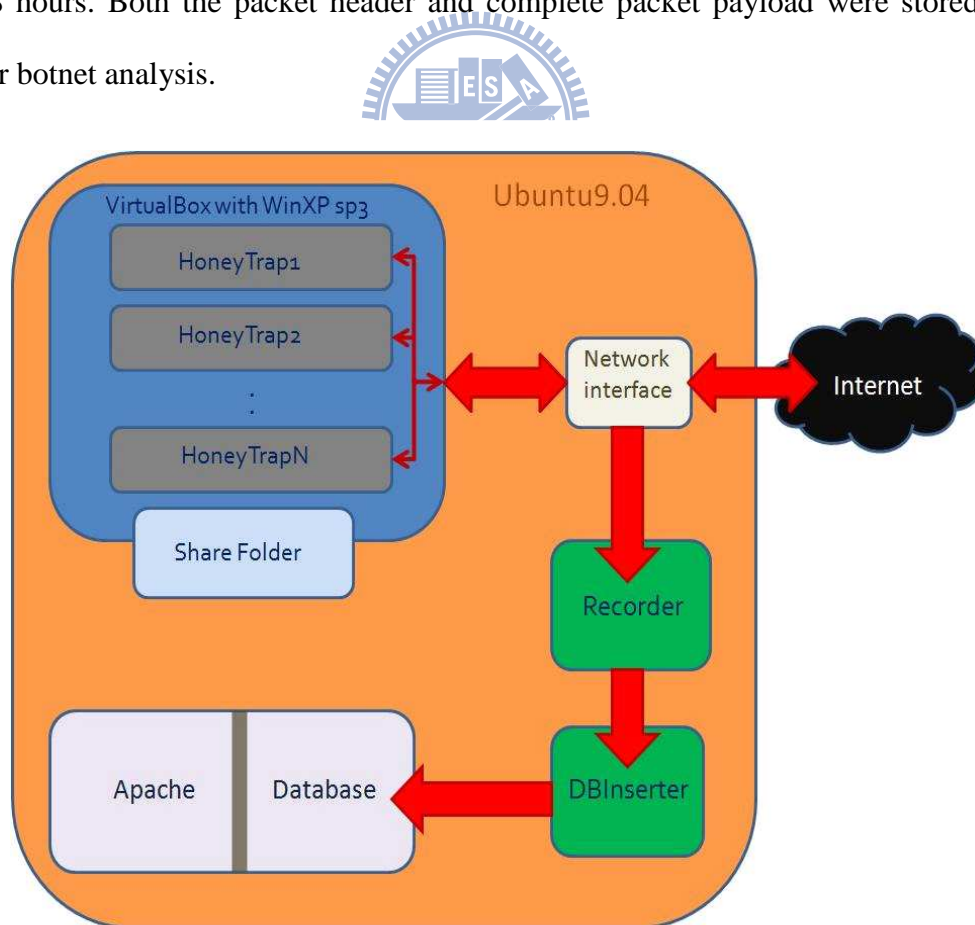


Figure 12. Experimental environment for botnet traces collection.

4.2 Test results for botnet traces

We used real botnet traces to evaluate our FPRF algorithm. In Figure 13, we ran different thresholds of ρ for botnet traces to check the false negative rates (FNRs) of malicious DNS/IP addresses. As ρ increases, FNR also increases. This is because some malicious IP addresses only contained very few packets, and these malicious IP addresses would not be detected if ρ is set too large. Statistics of the botnet traces and the bot DNS/IP addresses detection results are shown in Table 1. The evaluation result shows that FPRF has a high malicious DNS/IP addresses detection rate (95.29% / 95.24%). In Table 2, we detailed the statistics of active/inactive malicious DNS/IP addresses. We found 35 active malicious DNSs and 45 active malicious IP addresses.

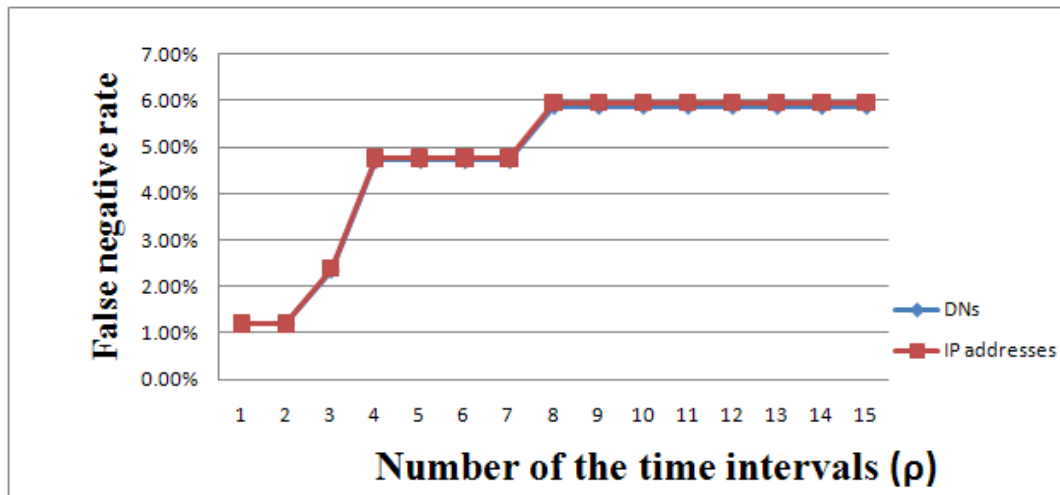


Figure 13. Malicious DNS/IP addresses false negative rate vs. ρ .

Table 1. Botnet traces statistics and detection rates (for $\rho = 7$).

Number of bots	Number of malicious DNAs	Number of malicious IP addresses	Number of DNS packets	Number of TCP connection packets	Malicious DNAs detection rate	Malicious IP addresses detection rate
44	85	84	294,385	675,164	95.29%	95.24%

Table 2. Statistics of active/inactive malicious DNAs/IP addresses (for $\rho = 7$).

Inactive malicious DNAs (without IP addresses)	Inactive malicious DNAs	Inactive malicious IP addresses
17	33	39
Active malicious DNAs		Active malicious IP addresses
35		45

We also collected four normal traces (T1 through T4) to evaluate the FPRF's traffic reduction rate and the false positive rate (FPR). These traces contain various types of benign applications from different users, e.g., IRC, HTTP, and P2P. In Figure 14, we ran the threshold of ρ for these normal traces to check the FPR of malicious IP addresses. As ρ increases, the FPR decreases. This is because more benign packets were examined and the corresponding IP addresses can be determined to be malicious or not. In Table 3, we found that the FPRF achieves high reduction rates and low FPRs of malicious DNAs and IP addresses.

In our experiments, we chose $\rho = 7$ to have better performance. According to

Figures 13 and 14, it is a trade-off between high and low ρ values. For $\rho = 7$, the FPRF had the detection rates of 95.29% / 95.24% in malicious DNS/IP addresses, respectively, as shown in Table 1, and the FPRs of 0 ~ 3.08 % in benign DNS/IP addresses, as shown in Table 3.

Table 3. Normal traces statistics and FPRs of benign DNS/IP addresses (for $\rho=7$).

Test Trace	Reduction rate	Number of DNS packets	Number of TCP connection packets	Duration	FPR of malicious DNS	FPR of malicious IP addresses
T1	71.8%	1,507	249,527	48 hours	0.4%	2.89%
T2	94.2%	567	37,624	7 hours	0%	2.45%
T3	93.74%	2,155	44,298	16 hours	3.67%	2.1%
T4	96.38%	3,483	43,667	13 hours	0.89%	3.08%

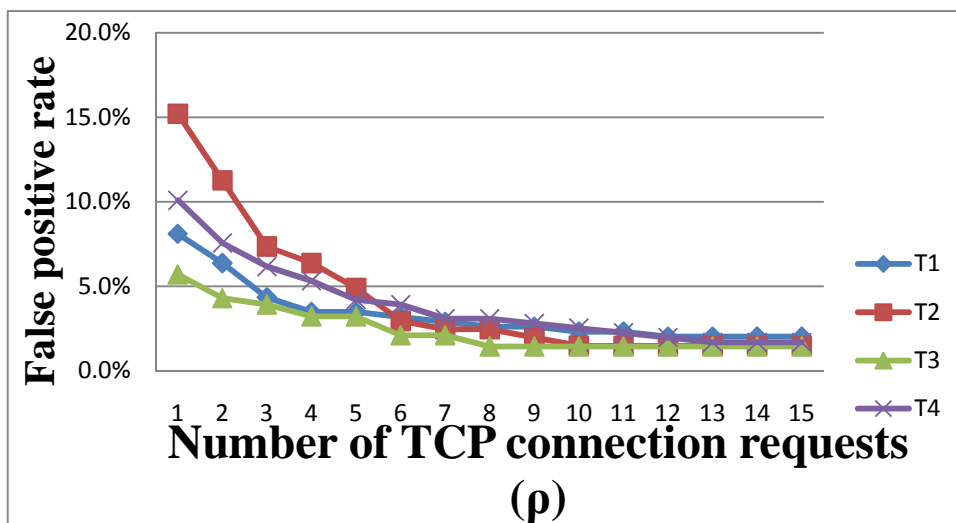


Figure 14. Benign IP addresses fault positive rate vs. ρ .

In Table 4, we compare the proposed FPRF and the other two existing botnet detection methods. Unlike these two existing methods, FPRF not only used 44 real bot samples to generate real botnet traffic and use only one reduction filter for normal traces, but also can detect inactive bots.

Table 4. Comparison of behavior-based botnet detection methods.

Approach	FPRF (proposed)	Livadas [4]	Gu [9]
Basic idea	Fuzzy pattern recognition	Machine learning technique	Spatial-temporal correlation in network traffic and utilizing statistical algorithm [9]
Botnet trace	Real botnet traffic	Lab traffic with reimplementation of a bot without packet payload	Real traffic: 1 bot Reimplementation: 3 bots Written by themselves: 2 bots Botnet IRC logs: 2 bots
Traffic reduction rate	Over 70% (with 1 filter)	N/A (with 4 filters)	N/A (with 2 filters)
Inactive bot detection	Yes	No	No
True positive rate	95%	92%	100%
False positive rate	0 ~ 3%	11 ~ 15%	0 ~ 6
Bot samples	44	1	8
Categories of bot samples	IRC + HTTP + P2P bots	IRC bots	IRC + HTTP bots

Chapter 5

Conclusion

5.1 Concluding remarks

In this thesis, we have presented a fuzzy pattern recognition-based filtering (FPRF) algorithm for botnet detection. Our FPRF algorithm is divided into three stages: (1) traffic reduction: reducing input raw packet traces for speeding up the processing of bots specific activities, (2) feature extraction: extracting features from the reduced input packet traces, and (3) fuzzy pattern recognition: using features as an input pattern and fuzzy membership functions based on the maximum membership principle to detect bots. We have used real bots to generate real botnet traces for evaluating the proposed of FPRF. Experimental results based on real botnet traces have shown that the proposed FPRF has high detection rates of 95.29% in malicious domain names detection and 95.24% in malicious IP addresses detection. The experimental results based on normal traces have also shown a high traffic reduction rate of over 70% and low false positive rates (0 ~ 3%) in malicious domain names and IP addresses detection. This means that FPRF is not only efficient but also accurate. In addition, FPRF can detect inactive botnets, which may become malicious but have no any malicious action up to now.

5.2 Future work

In our FPRF scheme, 48-hour botnet traces were examined for botnet detection. The future work will focus on how to achieve on-line botnet detection while still maintaining a high detection rate and a low false positive rate based on the proposed FPRF.

References

- [1] S. Gianvecchio, M. Xie, Z. Wu, and H. Wang, "Measurement and classification of humans and bots in internet chat," in *Proceedings of USENIX 17th conference on Security symposium*, 2008.
- [2] "The Honeynet Project," [Online]. Available: <http://www.honeynet.org/>.
- [3] C. Kolbitsch, P. Comparetti, C. Kruegel, E. Kirda, X. Zhou, and X. Wang, "Effective and efficient malware detection at the end host," In *Proceedings of USENIX 18th conference on Security symposium*, 2009.
- [4] C. Livadas, B. Walsh, D. Lapsley and W. Timothy Strayer, "Using Machine Learning Techniques to Identify Botnet Traffic," *Local Computer Networks, Proceedings 31st IEEE Conference*, 2006.
- [5] "Botnet Statistics from the Security Labs team at M86 Security," [Online]. Available: http://www.m86security.com/trace/Bot_statistics.asp/.
- [6] B. McCarty, "Botnets: big and bigger," *Security & Privacy, IEEE*, 2003.
- [7] "Shadowserver Foundation," [Online]. Available: <http://www.shadowserver.org/>.
- [8] B. AsSadhan, José M. F. Moura and D. Lapsley, "Periodic Behavior in Botnet Command and Control Channels Traffic," in *IEEE GLOBECOM*, 2009.
- [9] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting botnet command and control channels in network traffic," in the *15th Annual Network and Distributed System Security Symposium*, 2008.
- [10] "Tcpcap/Libpcap," [Online]. Available: <http://www.tcpdump.org/>.
- [11] "Wireshark," [Online]. Available: <http://www.wireshark.org/>.
- [12] H. Choi, H. Lee, H. Lee and H. Kim, "Botnet Detection by Monitoring Group Activities in DNS Traffic," in *Proceedings of the 7th IEEE International Conference on Computer and Information Technology*, 2007.

- [13] G. Gu, V. Yegneswaran, P. Porras, J. Stoll, and W. Lee, "Active Botnet Probing to Identify Obscure Command and Control Channels," *Annual Computer Security Applications Conference*, 2009.
- [14] W. Timothy Strayer, R. Walsh, C. Livadas and D. Lapsley, "Detecting Botnets with Tight Command and Control," in *Proceedings of the 31st IEEE Conference on Local Computer Networks*, 2006.
- [15] R. Walsh, D. Lapsley and W. Timothy Strayer, "Effective Flow Filtering for Botnet Search Space Reduction," in *Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security*, 2009.

