ARTICLE IN PRESS

Computer Networks xxx (2011) xxx–xxx

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

A fuzzy pattern-based filtering algorithm for botnet detection

Kuochen Wang [a], Chun-Ying Huang [b,*], Shang-Jyh Lin [a], Ying-Dar Lin [a]

[a] Department of Computer Science, National Chiao Tung University, 1001 University Road, Hsinchu 30010, Taiwan
[b] Department of Computer Science and Engineering, National Taiwan Ocean University, 2 Pei-Ning Road, Keelung 20224, Taiwan

ARTICLE INFO

Article history:
Received 25 December 2010
Received in revised form 22 April 2011
Accepted 29 May 2011
Available online xxxx

Keywords:
Botnet
Fuzzy pattern recognition
Network security
Real trace analysis

ABSTRACT

Botnet has become a popular technique for deploying Internet crimes. Although signature-based bot detection techniques are accurate, they could be useless when bot variants are encountered. Therefore, behavior-based detection techniques become attractive due to their ability to detect bot variants and even unknown bots. In this paper, we propose a behavior-based botnet detection system based on fuzzy pattern recognition techniques. We intend to identify bot-relevant domain names and IP addresses by inspecting network traces. If domain names and IP addresses used by botnets can be identified, the information can be further used to prevent protected hosts from becoming one member of a botnet. To work with fuzzy pattern recognition techniques, we design several membership functions based on frequently observed bots' behavior including: (1) generate failed DNS queries; (2) have similar DNS query intervals; (3) generate failed network connections; and (4) have similar payload sizes for network connections. Membership functions can be easily altered, removed, or added to enhance the capability of the proposed system. In addition, to improve the overall system performance, we develop a traffic reduction algorithm to reduce the amount of network traffic required to be inspected by the proposed system. Performance evaluation results based on real traces show that the proposed system can reduce more than 70% input raw packet traces and achieve a high detection rate (about 95%) and a low false positive rates (0–3.08%). Furthermore, the proposed FPRF algorithm is resource-efficient and can identify inactive botnets to indicate potential vulnerable hosts.

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 is increasing [1], most Internet users are still unaware of what is going on and how serious the problem is. Many of these users' computers are easily compromised by bot malware and then become members of botnets. Since bot malware usually does not affect regular uses of compromised computers, bot masters or bot herders can control these compromised computers remotely and ask them to carry out malicious activities, such as sending SPAMs, launching distributed denial of service (DDoS) attacks, and stealing personal private information.

Bot detection systems can be classified into two categories, i.e., signature-based and behavior-based systems. Although a signature-based system is accurate, it has the following drawbacks. First, signature-based systems is not possible to detect unknown bots. Second, a string signature is for a specific bot. When a bot has a variant, even it behaves similar, string signatures cannot work for it. Hence, the false negative rates may increase when new bots are developed. Third, as the number of bot variants increases, the false positive rates may increase as well. This is because an extremely large database containing all

* Corresponding author.
E-mail addresses: kwang@cs.nctu.edu.tw (K. Wang), chuang@ntou.edu.tw (C.-Y. Huang), szlin@cs.nctu.edu.tw (S.-J. Lin), ydlin@cs.nctu.edu.tw (Y.-D. Lin).

1389-1286/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2011.05.026

identified bots' signatures may accidentally match benign software. Finally, it is possible for a bot to bypass signature-based checks by using code obfuscation techniques.

On the contrast, behavior-based systems try to identify bot activities by using observed particular bot behavior. If well tuned, behavior-based systems are able to perform similar to signature-based systems in terms of detection rates. In addition, a behavior-based system does not need to maintain a signature database to detect bots. Such a system can be much more lightweight than a signature-based system.

In this paper, we propose a behavior-based system to detect malicious domain names and IP addresses used by botnets. The contribution of the paper is threefold. First, we propose an effective traffic reduction algorithm to reduce the amount of traffic that is required to be checked by a bot detection system. Second, we propose a generic framework to detect botnets based on fuzzy pattern recognition techniques. Third, we make in-depth observations to bot activities and then design proper membership functions to detect bots in a monitored network. Evaluation results show that the proposed bot detection system 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 paper is organized as follows. Section 2 overviews botnet behavior and reviews related work. Section 3 formally defines the main problem and sub-problems the proposed system is going to resolve and explains the core of the proposed system, which is named the fuzzy pattern recognition-based filtering (FPRF) algorithm. Section 4 presents the experiment environment and results. Finally, a concluding remark is given in Section 5.

2. Background and related work

2.1. Overview of botnet behavior

The working scenario of a botnet can be classified into two phases. One is the *infection* phase and another is the *attack* phase, as shown in Fig. 1. In the infection phase, a bot herder tries to expand the size of its army of bots. The bot herder commands existing bots to compromise more users' computers. There are many techniques to compromise a computer such as exploiting software vulnerabilities and social engineering. Once a targeted host is compromised, remote controllable software, which is downloaded from a binary-download server, is installed and launched so that the information about the compromised computer is reported to the bot herder. In the attack phase, a bot herder sends commands to compromised hosts, i.e., the bots. On receipt of the commands, each bot launches various tasks 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.

2.2. Existing solutions to traffic reduction

It is common that the traffic generated by bots are mixed with regular network traffic. To improve botnet detection efficiency, existing researches often reduce the amount of input traffic by filtering out bot-irrelevant traffic. Hence, a bot detection algorithm is able to concentrate only on bot traffic. A good traffic reduction algorithm may improve the overall system performance. However, if not well designed, it could increase false negative rates and/or false positive rates. Some common criteria used to filter out input traces are listed below [2–5]:

- Eliminate all port-scan activities
  Although port-scan is an essential step to compromise a remote host, it is not used when bots are communicating with each other. Therefore, it is possible to filter out port-scan activities without degrading the bot detection performance. Some port-scan packets have specific patterns. For example, a TCP port-scan packet with both the SYN flag and the RST flag are set.
- Ignore peer-to-peer traffic
  If a detection system focuses only on IRC botnets, it often filters out P2P traffic and hence gets a significant traffic reduction rate. However, such a system cannot detect P2P bots. There are already a lot of systems to identify peer-to-peer traffic, such as [6–8].
- Skip short lived flows
  Filter out flows containing only a few packets or lasting only for a few seconds. These flows do not correspond to bots that are standing by at the ready [2].
- Filter out based on black lists and white lists
  If the source or the destination address of a packet is well-known, it is often not necessary to check it. Hence, the packet can be safely ignored.

Although traffic reduction brings benefits, the above criteria also raise some concerns. First, ignoring P2P traffic eliminates the possibility of identifying P2P bots. Second, skipping short lived flows may cause failures in detecting inactive botnet traffic. An inactive bot is a bot that is not able to connect to its command and control (C&C) server either temporarily or persistently. An inactive bot should also be detected so that the network can be protected if the bot becomes active again. Third, when black lists and white lists are used to reduce input traffic, the lists must be well managed and make sure that the lists are up-to-date. This is because a benign host may be compromised and then turned into a malicious one at any time. If a list is not updated accordingly, the malicious traffic involved with a newly compromised host may be incorrectly ignored.

2.3. Related work

Kolbitsch et al. [9] proposed a host-based malware detection system. They used 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 false positive rates.

Please cite this article in press as: K. Wang et al., A fuzzy pattern-based filtering algorithm for botnet detection, Comput. Netw. (2011), doi:10.1016/j.comnet.2011.05.026

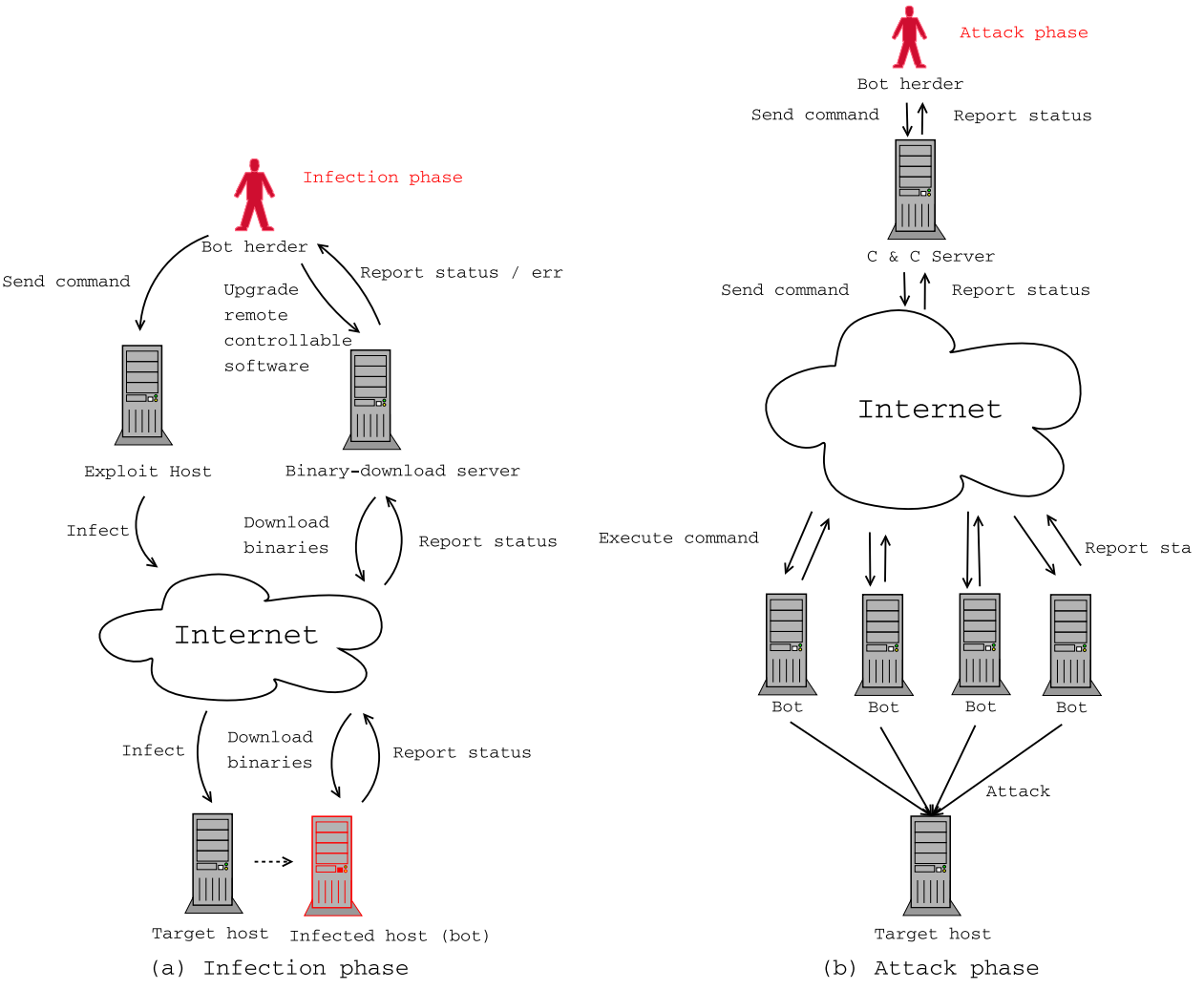

Fig. 1. Botnet working scenario: the infection and the attack phases.

Livadas et al. [2] developed a system to detect C&C traffic of IRC botnets. The system leverages machine learning techniques. The system contains two stages. In the first stage, it extracts several per-flow traffic attributes including flow duration, maximum initial congestion window, and average byte counts per packet. In the second stage, it uses a Bayesian network classifier to make the classification balanced between false negative rates and false positive rates. However, the false positive rate is still high (~15.04%). Sadhan et al. [10] also tried to identify botnet C&C traffic and they observed that this type of traffic appears 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. [11] proposed a botnet detection mechanism solely based on monitoring of DNS traffic. In an aggregated network trace, they found that DNS queries sent from bots can be easily grouped together by similarities of DNS requests and hence is able to be used to detect bot activities. Variants of bots can be detected as well. Gu et al. [5] proposed the "Bot-Sniffer." They identified bot hosts based on spatial–temporal correlation of collected network traces. They used bots collected from real world, reimplemented bots, and self-produced bots to evaluate their solution. Although the results showed a high detection rate and low false positive rates, the number of evaluated real bots is very limited (only 1 bot).

3. The proposed fuzzy pattern recognition filtering algorithm for botnet detection

3.1. Problem statement

The goal of the proposed system is to identify domain names and IP addresses used by botnet C&C servers. The identified domain names and IP addresses can be classified into two categories, i.e., *active* and *inactive*. Active domain names or IP addresses can be used to reach C&C servers and hence they are used to report information, send feedbacks, and retrieve commands. On the contrary, inactive

Please cite this article in press as: K. Wang et al., A fuzzy pattern-based filtering algorithm for botnet detection, Comput. Netw. (2011), doi:10.1016/j.comnet.2011.05.026

domain names and IP addresses were previously used to contact C&C servers but they are currently inactive for some reasons. A domain name may become inactive if its DNS server in charge has no mapping records. An IP address may become inactive because the corresponding host is shutdown or off-line. However, it is possible that an inactive one becomes active again. Therefore, it is important for us to identify and maintain both active and inactive records. To achieve the goal, the proposed system targets on three sub-problems. They are:

(a) *Traffic reduction:* It is common that input raw packet traces contain many different types of packets. Since most of them are not relevant to botnet detection, they should be filtered out. With an accurate and efficient traffic reduction algorithm, it enables a botnet detection system to run in a more efficient way.
(b) *Feature extraction:* We observed that bots usually operate with particular behavior. Some of the behavior is distinguishable from normal behavior and hence features of the behavior can be extracted to detect bots. An ideal feature should be applicable to as many bots as possible.
(c) *Pattern recognition:* Once distinguishable features of bots are extracted, we need a good pattern recognition technique to identify bots based on the extracted features. A good candidate should be able to correctly classify input traffic. The technique cannot be too complex. It must be efficient so that a decision can be made within a short period of time. The pattern recognition technique must have high detection rates and low false positive rates.

3.2. Overview of the proposed algorithm

The proposed fuzzy pattern recognition filtering (FPRF) algorithm for botnet detection is shown in Fig. 2. There are three stages in the algorithm: traffic reduction, feature extraction, 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 domain names and IP addresses based on extracted features. The most important issue that affects the effectiveness of the proposed algorithm is what features are used to detect bots' behavior. Hence, we make observations on real bots and find that the following phenomena are the most common ones:

- *Generate failed domain name system (DNS) queries*: It is common that a bot has a built-in domain name list of all possible C&C servers. However, since C&C servers could be shutdown or off-line at any time, a DNS query about an unreachable C&C server generates a failed DNS response.
- *Have similar DNS query intervals*: If a DNS query about a C&C server fails, a bot may lookup either the same domain name again or the next domain name available in the built-in domain name list. To prevent it from affecting the user, a bot often makes another attempt after sleeping for a period of time. Therefore, the time intervals between successive DNS queries may be similar.
- *Generate failed network flows*: In addition to use a built-in domain name list, a bot may use a built-in list of IP addresses as well. Similar to the DNS query cases, if a bot tries to contact a unreachable C&C server, the process fails. It is possible that a bot successfully obtains an IP address from a DNS server but the computer associated with the obtained IP address is shutdown or off-line. This is because a C&C server may be installed on a compromised user's computer, there is no guarantee that the C&C server is always on-line.
- *Have similar payload sizes for different network flows*: If a C&C server can be reached successfully, a bot then downloads commands provided by bot herders from the C&C server. However, it is common that a command keeps unchanged for a period of time. This is because bot herders are not able to predict the exact time that bots read the provided commands. Hence, to make sure the number of bots that read the commands is sufficiently large, bot herders would not frequently change the commands. As a result, the payload sizes of bot relevant network flows for the same family of bots are

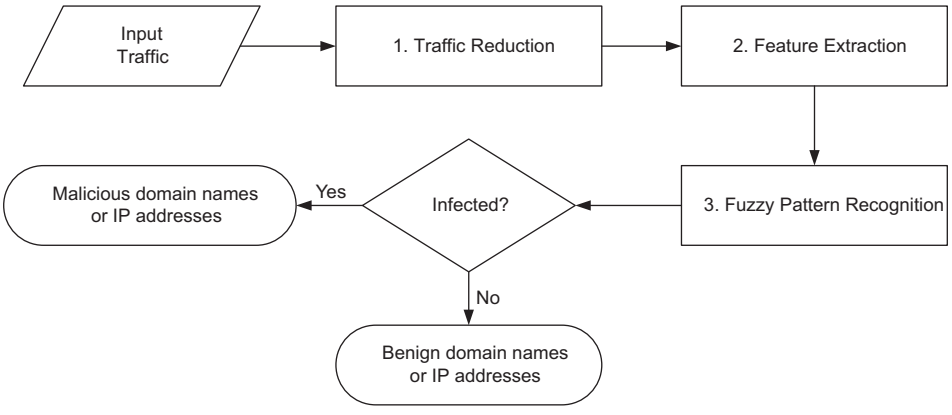

Fig. 2. Fuzzy pattern recognition filtering (FPRF) algorithm for identifying botnet domain names and IP addresses.

Please cite this article in press as: K. Wang et al., A fuzzy pattern-based filtering algorithm for botnet detection, Comput. Netw. (2011), doi:10.1016/j.comnet.2011.05.026

similar. Note that the payload size for a TCP and an UDP network flow is counted in different ways. The payload size for a TCP flow is the cumulated value of payload sizes of all involved packets. On the contrast, the payload size for a UDP flow is counted separately based on the datagram size of each UDP message.

The proposed solution is able to detect bots based on the above preliminary observations. To work with the fuzzy pattern recognition techniques, we have to design at least one *membership function* for each observed behavior. Membership functions can be altered, removed from, or added to the system so that the system is able to adapt itself to new bots' behavior. In the rest of this section, we discuss how the features are collected and used to detect bot relevant domain names and 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 hence increases the overall system performance. However, if a filter eliminates data improperly, bot detection rates could decrease. Therefore, criteria for traffic reduction must be carefully considered.

In the proposed solution, we use only one intrinsic traffic reduction filter, as shown in Fig. 3. To prevent botnets from being detected, it is common for bots to dynamically retrieve the IP addresses of C&C servers. A bot herder is able to register several domain names and asks the bots to look up the IP addresses of these domain names. As a result, bots need to send DNS queries frequently to get the IP addresses currently being used by C&C servers.

Since bots' activities often start with DNS queries, this characteristic can be used to filter out bot-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.

3.4. Feature extraction

As we mentioned before, bots activities often start with DNS queries. The procedure of a bot's execution is shown in Fig. 4. If the domain name of a C&C server cannot be resolved or the resolved IP addresses are unreachable (offline hosts or invalid IP addresses), the bot is inactive. On the contrary, if one of the resolved IP addresses is valid and a bot is able to connect to the C&C server, it is an active bot. As mentioned in Section 2.2, we classify bots into two types, i.e., active and inactive bots. It is not difficult to distinguish active and inactive bots. An active bot is always able to establish connections with one C&C server. On the other hand, an inactive bot could receive a number of DNS failure messages and it is not able to reach a C&C server. Therefore, we extract features from DNS queries and network flows and then the extracted features are used to detect C&C server addresses.

3.4.1. Feature extracted from DNS packets

Based on the bot traces we collected, we find that DNS queries from bots are usually periodical. These periodical DNS queries can be further classified into two types. The first type is a single fixed interval pattern, as shown in Fig. 5. The interval between any two successive DNS queries for a single bot is fixed. In Fig. 5, the interval pattern is $\{15, 15, 15, \ldots\}$ seconds. The second type is an interleaved interval pattern, as shown in Fig. 6. The interleaved interval pattern shown in Fig. 6 is $\{\{1,1,2,4\},\{1,1,2,4\},\ldots\}$ seconds.

3.4.2. Feature extracted from network flows

After a DNS query is successful, a bot then tries to connect to its C&C server. Fig. 7 shows the relationship between DNS queries and network flows. We can see that DNS queries are followed by a number of network requests. We can also see that the network requests are initiated periodically.

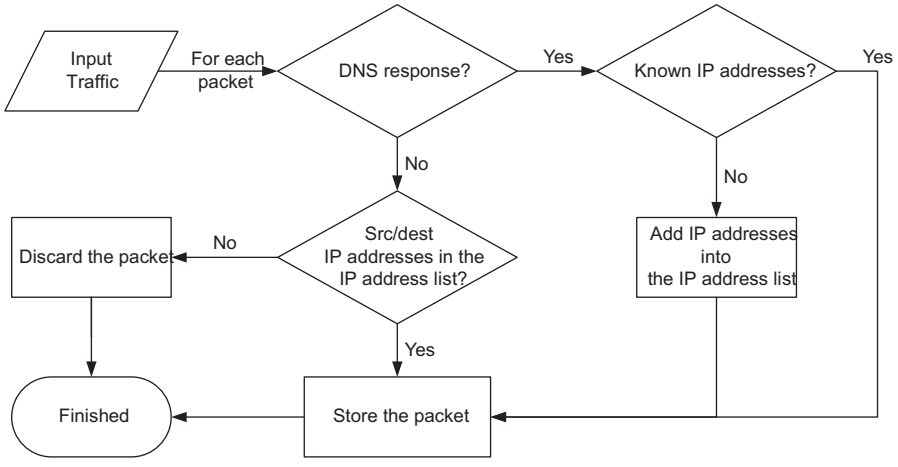

Fig. 3. The procedure of traffic reduction.

Please cite this article in press as: K. Wang et al., A fuzzy pattern-based filtering algorithm for botnet detection, Comput. Netw. (2011), doi:10.1016/j.comnet.2011.05.026

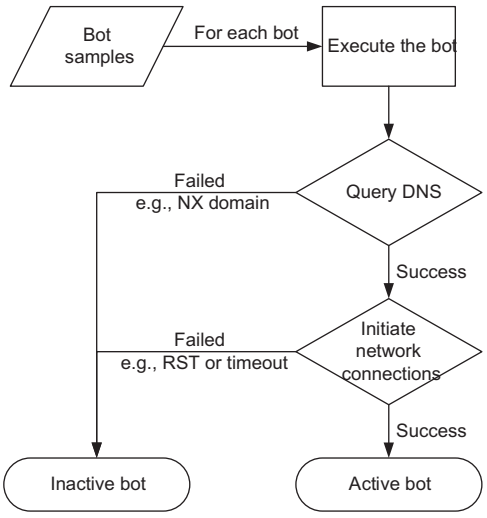

Fig. 4. The procedure of a bot's execution.

3.5. Fuzzy pattern recognition

To prevent themselves from being detected, bots often try to simulate human-like behavior. To resolve this problem, we use fuzzy pattern recognition to detect bots. In the proposed fuzzy pattern recognition based filter (FPRF) algorithm, there are two phases, i.e., the DNS phase and the network flow phase, as shown in Fig. 8. In the DNS phase, we detect a bot based on DNS features. Suppose a DNS query is made for domain name D and a corresponding DNS response returns an associated IP address A. If D is identified as a malicious domain name, both D and A will be treated as malicious. However, if D is neither an active bot nor an inactive bot, all subsequent network packets associated with the returned IP address A are then passed to the network flow phase. In the network flow phase, we detect a bot based on network flow features. If the IP address A does not belong to an inactive or active bot, the IP address is benign. Both the two phases identify malicious domain names and IP addresses based on fuzzy pattern recognition. The features are collected in the feature extraction stage and then the max membership principle is applied to the features, as shown in Fig. 9, to identify malicious ones.

3.5.1. The DNS phase

In the DNS phase, for each identified domain name, we define a feature vector $x = (\alpha, \beta, \gamma)$ for the domain name, where

- Suppose the maximum time interval between two successive DNS queries is less than n seconds. We define α as a fixed size set that contains n counters, i.e., $\alpha = \{a_1, a_2, \ldots, a_n\}$. Each counter in α has an initial value of zero. Given a segment of a network trace containing m DNS queries, the time intervals between two successive DNS queries can be extracted and then form a sequence $S = \{s_1, s_2, \ldots, s_{m-1}\}$. For each time interval s_j $(1 \leqslant j \leqslant m-1)$ in S, we calculate i as $\lceil s_j \rceil$ if and only if s_j is less than or equal to n and then increase the counter a_i by 1. Therefore, if s_j is greater than n, no counter is increased.
- β is the total number of DNS responses.
- γ is the number of failed DNS responses.

In this phase, we define the following three states and each state has its own membership function:

(a) Inactive malicious DNS query
We assume that a DNS query about an inactive malicious domain name usually gets a failed DNS response. Therefore, more failed DNS responses

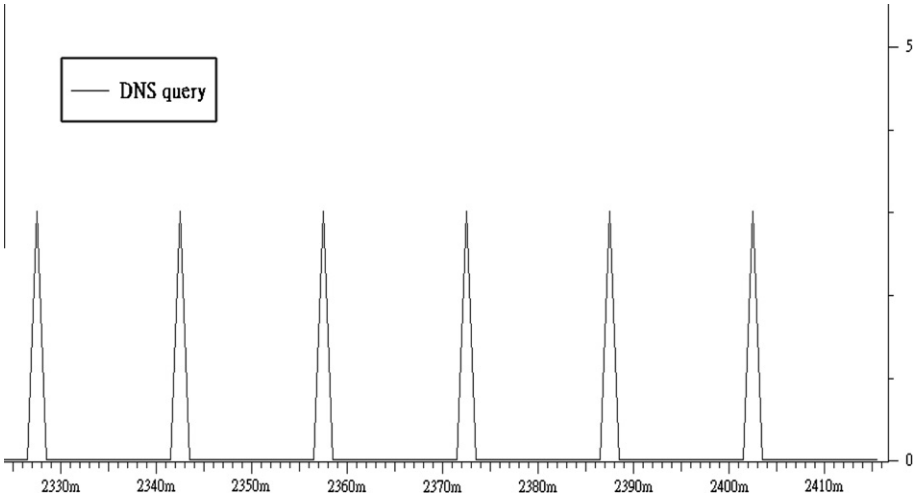

Fig. 5. The distribution of botnet DNS query packets generated by the "Trojan-Spy.Win32.Zbot.aaak" (a.k.a. ZeusBot) bot: single fixed interval pattern (x-axis: time in minutes; y-axis: number of DNS packets).

Please cite this article in press as: K. Wang et al., A fuzzy pattern-based filtering algorithm for botnet detection, Comput. Netw. (2011), doi:10.1016/j.comnet.2011.05.026

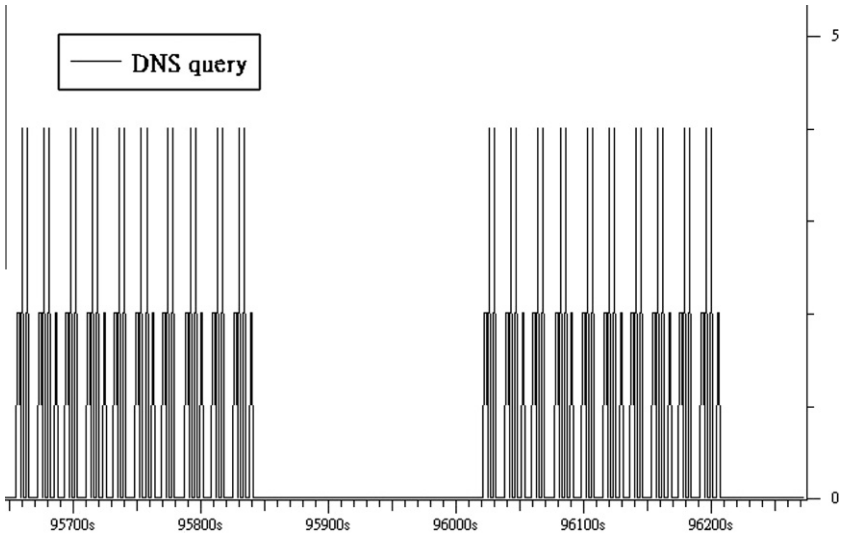

Fig. 6. The distribution of botnet DNS query packets generated by the "Backdoor.Win32.SmallBot.c" bot: interleaved intervals pattern (*x*-axis: interval in seconds; *y*-axis: number of DNS query packets).

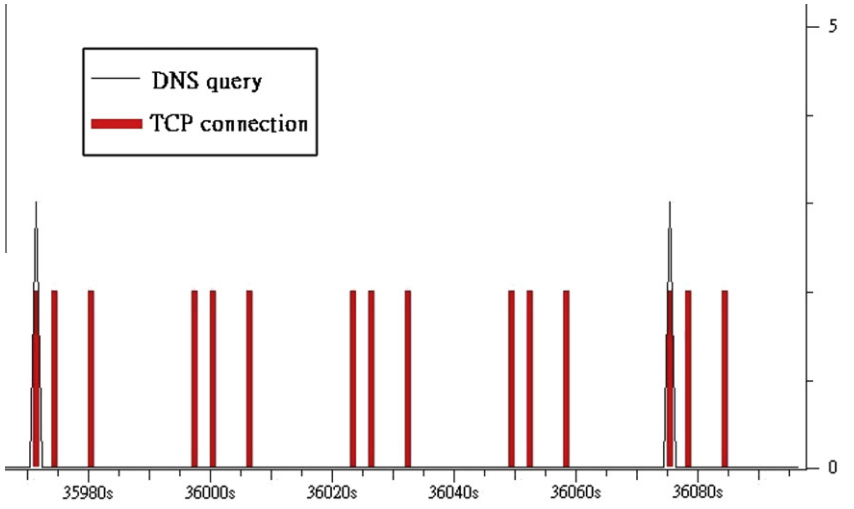

Fig. 7. The distribution of DNS query and network packets generated by the "Win32.SpyBot" (*x*-axis: interval in seconds; *y*-axis: number of DNS queries or network packets (TCP)).

should lead to a higher membership value. Based on the observation, we define a membership function X_1 which is used to calculate the probability of being an inactive malicious DNS query. The function X_1 is defined as

$$X_1(x) = 1 - \frac{\beta - \gamma}{\beta}. \quad (1)$$

(b) Malicious DNS query
Since malicious DNS queries usually have similar time intervals. If most DNS queries for an identified domain name have similar time intervals, it could be a malicious domain name. We define a membership function X_2 to calculate the probability of contacting a malicious domain name. Hence, we define function X_2 as

$$X_2(x) = \begin{cases} \frac{\max(\alpha)}{\sum \alpha}, & \sum \alpha \geqslant \rho, \\ 0, & \text{otherwise}. \end{cases} \quad (2)$$

In the equation, we define a threshold ρ. If the number of observed DNS queries is less than ρ, we believe that the identified domain name is benign and thus X_2 has a value of zero.

(c) Normal DNS query
We define a membership function X_3 to calculate the probability of being a normal DNS query. If an identified domain name has no failed DNS response, low query frequency, and diverse time intervals, it would be a benign domain name. Therefore, the function X_3 is defined as

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

Please cite this article in press as: K. Wang et al., A fuzzy pattern-based filtering algorithm for botnet detection, Comput. Netw. (2011), doi:10.1016/j.comnet.2011.05.026

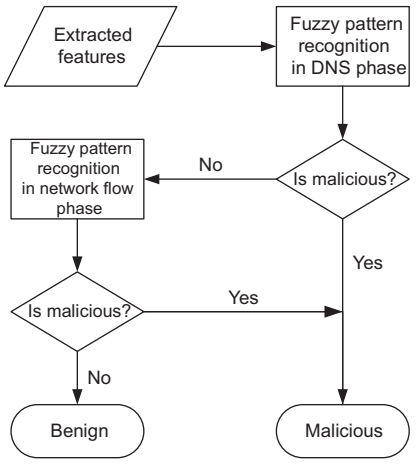

Fig. 8. The procedure of the fuzzy pattern recognition stage.

3.5.2. The network flow phase

Similar to the DNS phase, we also define a feature vector $x = (\alpha, \beta, \gamma)$ for each destination IP address identified in network flows, where

- Suppose the maximum time interval between a request and its response is less than n seconds. We define α as a fixed size set that contains n counters, i.e., $\alpha = \{a_1, a_2, \ldots, a_n\}$. Each counter in α has an initial value of zero. Given a segment of a network trace containing m request-response pairs, the time intervals between a request and the corresponding response can be measured and then form a sequence $S = \{s_1, s_2, \ldots, s_m\}$. For each time interval s_j $(1 \leqslant j \leqslant m)$ in S, we calculate i as $\lceil s_j \rceil$ and then increase the counter a_i by 1. Note that if a request does not have a corresponding response, no counter is increased. In addition, if s_j is greater than n, no counter is increased as well.
- β is the total number of network requests.
- Suppose the maximum payload size is less than b bytes. We define γ as a fixed size set that contains $b + 1$ counters, i.e., $\gamma = \{r_0, r_1, r_2, \ldots, r_b\}$. Each counter in γ has an initial value of zero. Given a segment of a network trace containing t network flows, the payload size of each network flow is extracted and form a sequence $P = \{p_1, p_2, \ldots, p_t\}$. For each payload size p_j $(1 \leqslant j \leqslant t)$ in P, we set i to p_j and then increase the counter r_i by 1 if $i \leqslant b$.

In this phase, we also define the following three states and their corresponding membership functions:

(a) Inactive malicious IP address
We assume that if an IP address receives many requests but does not respond, it is highly probable that the destination IP address is an inactive malicious IP address. We define a membership function X_1 to calculate the probability of being an inactive malicious IP address:

$$X_1(x) = \begin{cases} 1, & \sum \alpha = 0 \quad \text{and} \quad \beta \geqslant \sigma, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

In the equation, σ is a threshold for the number of retries. When a destination IP address has been reconnected for more than σ times, the destination IP address is treated as malicious.

(b) Malicious IP address
Since computers with malicious IP addresses, e.g., C&C servers, often provide the same commands to bots, it can be observed that connections to these malicious IP addresses would have similar payload sizes. Without counting a payload size of zero, we define a membership function X_2 to calculate the probability of being a malicious IP address:

$$X_2(x) = \begin{cases} \frac{\max(\gamma)}{\beta - r_0}, & \beta - r_0 \geqslant \rho, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

We believe that bots always try to reach malicious IP addresses as possible as they could. If the number of network flows established with the destination IP address is less than a threshold ρ, the IP addresses is treated as benign and thus X_2 has a value of zero.

(c) Normal IP address
We define a membership function X_3 to calculate the probability of being a normal IP address. If a destination IP address has no failed network flows and the payload sizes are diverse, it would be a benign address. Therefore, the function X_3 is defined as

$$X_3(x) = 1 - \max\{X_1(x), X_2(x)\}. \tag{6}$$

4. Evaluation

4.1. Trace collection

To generate real botnet traces, we collect malicious binaries using honeytrap [12], store binary sequences injected into honeytraps in a share folder, and then run executable bots in an unpatched Windows XP SP3 operating system installed in VirtualBox virtual machines [13], as shown in Fig. 10. We obtain 100 bot binary sequences from the honeytraps. However, only 44 of them are complete

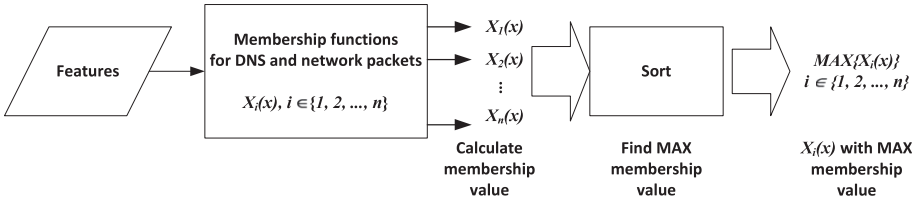

Fig. 9. Fuzzy pattern recognition based on maximum membership principle.

Please cite this article in press as: K. Wang et al., A fuzzy pattern-based filtering algorithm for botnet detection, Comput. Netw. (2011), doi:10.1016/j.comnet.2011.05.026

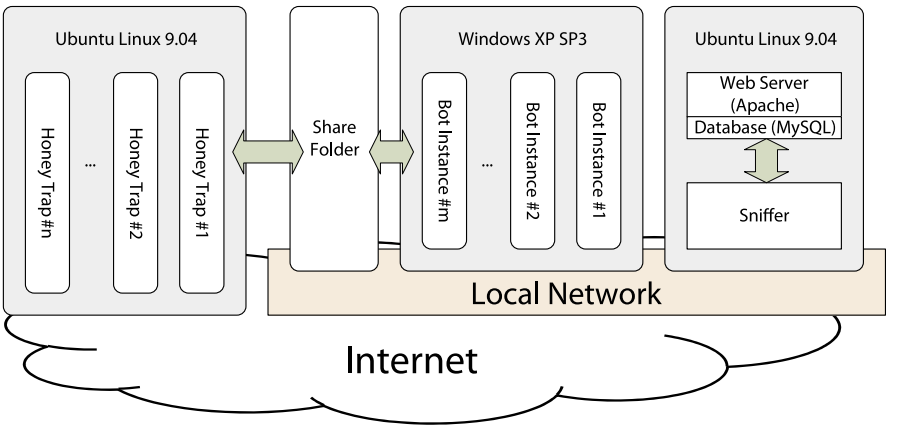

Fig. 10. Experimental environment for botnet traces collection.

Table 1
List of the 44 bots used to generate malicious network traces.

Backdoor.IRC.Botnut.c	Backdoor.IRC.ChanBot.a	Backdoor.IRC.Lambot
Backdoor.IRC.Strobot	Backdoor.PHP.Buzbot.a	Backdoor.PHP.ShellBot.e
Backdoor.Win32.Bashbot.c	Backdoor.Win32.Bitbot.a	Backdoor.Win32.Cubot.a
Backdoor.Win32.FatBot.m	Backdoor.Win32.HareBot.eo	Backdoor.Win32.Leetbot.b
Backdoor.Win32.LiteBot	Backdoor.Win32.LiteBot.a	Backdoor.Win32.Mocbot.a
Backdoor.Win32.MTBot.a	Backdoor.Win32.Nepoe.mi	Backdoor.Win32.Robobot.aa
Backdoor.Win32.RXBot.a	Backdoor.Win32.ShBot.a	Backdoor.Win32.Sikbot.a
Backdoor.Win32.Slackbot.b	Backdoor.Win32.SmallBot.c	Backdoor.Win32.SmallBot.e
Backdoor.Win32.VBbot.ac	Backdoor.Win32.XBot.a	Email-Worm.VBS.Bugbot.a
Email-Worm.VBS.Rodybot	Net-Worm.Win32.Kolabc.hdu	Net-Worm.Win32.Padobot
P2P-Worm.Win32.SpyBot.ad	P2P-Worm.Win32.SpyBot.ag	P2P-Worm.Win32.Sybot.a
Packed.Win32.Krap.b	Trojan-Downloader.Win32.Small.anhv	Trojan-Dropper.Win32.MultiJoiner.l3.b
Trojan-Dropper.Win32.SpoofBot	Trojan-Mailfinder.Win32.AIMBot.a	Trojan-Proxy.Win32.Chubot
Trojan-Spy.Win32.Zbot.aaak	Trojan.Win32.Spabot.ad	(Unknown)
Virus.DOS.Gobot.2097	Virus.DOS.Gobot.2099	

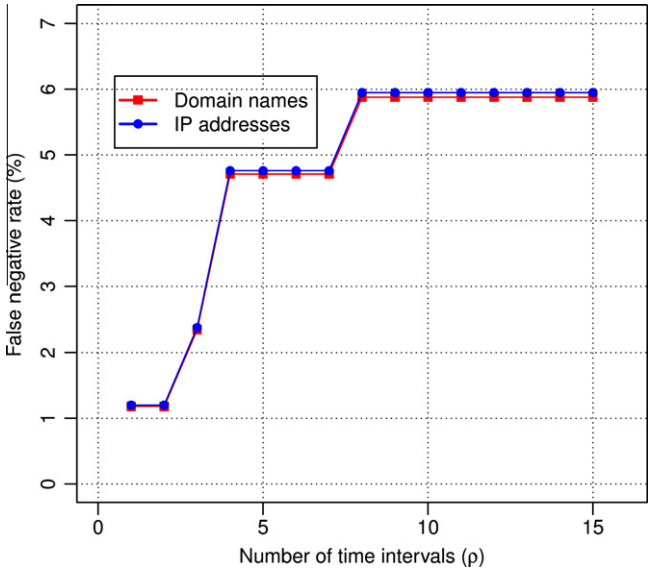

Fig. 11. Malicious domain names and IP addresses: false negative rate vs. ρ.

executables. Therefore, the traces are collected only for the 44 executable bots. Table 1 shows the name of the 44 bot binaries used to generate malicious network traces. All input and output network traces of the virtual machines are captured and stored in a MySQL database. Each bot is run for 48 h. Both packet headers and complete packet

Table 2
Botnet traces statistics and detection rates (for $\rho = 7$).

Number of bots	44
Number of malicious domain names	85
Number of malicious IP addresses	84
Number of DNS packets	294,385
Number of TCP packets	675,164
Detection rate of malicious domain names	95.29%
Detection rate of malicious IP addresses	95.24%

Table 3
Statistics of active/inactive malicious domain names and IP addresses (for $\rho = 7$).

	Domain names	IP addresses
Active	35	45
Inactive	33 (w/ IP addresses) 17 (w/o IP addresses)	39
Total	85	84

payloads are stored for further analysis. In addition to collect traffic generated by bots, we also collect real network traces from a campus network. We install a network sniffer at an edge router of a well-managed network of a laboratory. There are around 20 users inside the lab during work hours. The collected normal traces are generated by commonly used network applications and services including on-line chatting, peer-to-peer file sharing, web browsing, and remote management. We use normal traces to evaluate the false positive rates of the proposed solution.

4.2. Numeric results

We use captured real botnet traces to evaluate the proposed FPRF algorithm. In Fig. 11, we run different thresholds of ρ for botnet traces to check the false negative rates (FNR) of malicious domain names and IP addresses. As ρ increases, FNR also increases. This is because some bots generate very few packets and the involved domain names or malicious IP addresses would not be detected if ρ is too large. Statistics for the detection of malicious domain names and IP addresses are shown in Table 2. The evaluation result shows that FPRF has a high detection rate, i.e., 95.29% and 95.24% for malicious domain names and IP addresses, respectively. Table 3 shows the detailed numbers of detected malicious domain names and IP addresses. There are 85 and 84 malicious domain names and IP addresses identified, respectively. Among all identified records, we also find that there are only 35 active malicious domain names and 45 active malicious IP addresses.

We also collect 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 application traces including IRC, HTTP, and P2P traffic generated by different users. In Fig. 12, we use different ρ for normal traces and check the FPR of detecting malicious IP addresses. As ρ increases, the FPR decreases. This is because more benign packets are examined and the corresponding IP addresses can be determined to be malicious or not. In Table 4, we show that the FPRF algorithm achieves high reduction rates and low FPRs of malicious domain names and IP addresses.

In our last experiment, we choose $\rho = 7$ to have a better performance. According to Figs. 11 and 12, it is a trade-off between false negative and false positive rates. When we set ρ to 7, the FPRF algorithm has the detection rates of 95.29% and 95.24% for malicious domain names and IP addresses, respectively, as shown in Table 2. Note that the FPRF algorithm has false positive rates ranging from 0% to 3.08% for benign domain names and IP addresses, as shown in Table 4. It is worth to note that the threshold ρ is chosen based on the empirical data, i.e., the captured malicious and normal traces. We believe that if the amount

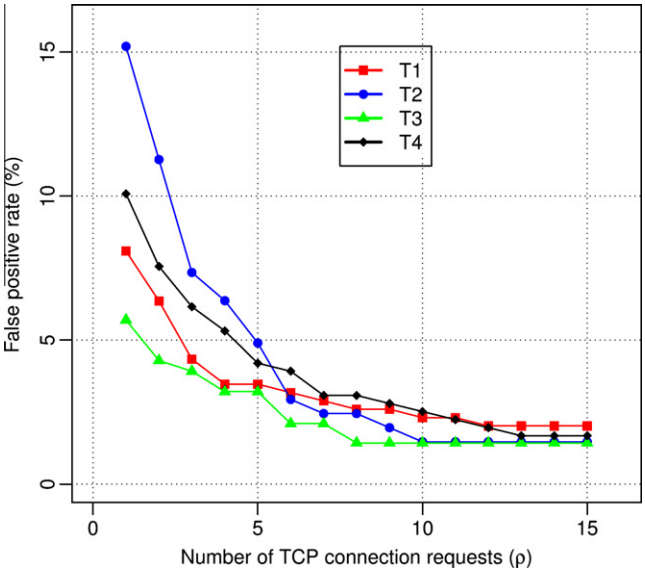

Fig. 12. Benign IP addresses false positive rate vs. ρ.

Please cite this article in press as: K. Wang et al., A fuzzy pattern-based filtering algorithm for botnet detection, Comput. Netw. (2011), doi:10.1016/j.comnet.2011.05.026

Table 4
Normal traces statistics and FPRs of benign domain names and IP addresses (for $\rho = 7$).

Trace ID	T1	T2	T3	T4
Reduction rate	71.8%	94.2%	93.7%	71.8%
Number of DNS packets	1,507	567	2,155	3,483
Number of TCP packets	249,527	37,624	44,298	43,667
Duration	48 h	7 h	16 h	13 h
FPR of malicious domain names	0.4%	0.0%	3.7%	0.9%
FPR of malicious IP addresses	2.89%	2.45%	2.10%	3.08%

Table 5
Comparison of behavior-based botnet detection methods.

Approach	Livadas et al. [2]	Gu et al. [5]	The proposed FPRF
Core technique	Machine learning	Spatial–Temporal correlation (statistic-based)	Fuzzy pattern recognition
Bot samples	1 Re-implememted in lab traffic	8 1: Real bot/real traffic 3: Re-implemented 2: Self-developed 2: Botnet IRC log	44 Real bots/real traffic
Rate of traffic reduction	N/A (4 filters)	N/A (2 filters)	More than 70% (1 filter)
Inactive bots detection	No	No	Yes
True positive rate	92%	100%	95%
False positive rate	11–15%	0–6%	0–3.08%

of empirical data is sufficient, the chosen value can be widely applied to different networks.

Table 5 compares the proposed FPRF with the other two existing botnet detection methods [2,5]. Based on the core techniques used, the computation cost of the proposed solution would be much lower than the other two methods. The proposed fuzzy pattern recognition-based filtering (FPRF) algorithm requires only basic arithmetics, while machine leaning and statistic based algorithms requires either high-dimensional vector processing or complicated arithmetics. To make sure that the proposed solution works with real world bots, we also evaluate the algorithm with 44 real bot samples to generate malicious traffic. In addition to have high traffic reduction rates, high true positive rates, and low false positive rates, the result also shows that the proposed solution performs well with real world botnet traces. Readers should notice that the comparison of detection accuracies is evaluated independently by developers of each compared solution. Evaluation with different traces may lead to different results. Nevertheless, we believe that evaluation with real traces would be better than with re-implemented and self-generated traces.

5. Conclusion

In this paper, we propose an extensible fuzzy pattern recognition-based filtering (FPRF) algorithm for botnet detection. Based on common bot host behavior observed from DNS and TCP traffic, our FPRF algorithm is divided into three stages: (1) traffic reduction: reduce input raw packet traces and speed up the processing of bots specific activities; (2) feature extraction: extract features from the reduced input packet traces; and (3) fuzzy pattern recognition: with extracted features, detect bot-relevant malicious domain names and IP addresses based on the maximum membership principle. We use a number of real bots to generate botnet traces to evaluate the proposed FPRF algorithm. Experimental results show that the proposed FPRF has high detection rates of 95.29% and 95.24% for malicious domain names and malicious IP addresses, respectively. The experimental results based on normal traces also show a high traffic reduction rate of over 70% and low false positive rates (0–3.08%). Both results show that the FPRF algorithm is not only efficient but also highly accurate. In addition, the FPRF algorithm can detect inactive botnets, which can be used to identify potential vulnerable hosts.

Acknowledgment

This research was supported in part by National Science Council under the grants NSC 99-2221-E-009-081-MY3 and by Taiwan Information Security Center at NTUST (TWISC@NTUST) under the grants NSC 100-2219-E-011-002. We would also like to thank the anonymous reviewers for their valuable and helpful comments.

References

[1] B. McCarty, Botnets: big and bigger, IEEE Security and Privacy 1 (4) (2003) 87–90.
[2] C. Livadas, R. Walsh, D. Lapsley, W.T. Strayer, Usilng machine learning technliques to identify botnet traffic, in: Proceedings of the 31st IEEE Conference on Local Computer Networks, IEEE, 2006, pp. 967–974.
[3] W.T. Strayer, R. Walsh, C. Livadas, D. Lapsley, Detecting botnets with tight command and control, in: Proceedings of the 31st IEEE Conference on Local Computer Networks, 2006, pp. 195–202.
[4] R. Walsh, D. Lapsley, W.T. Strayer, Effective flow filtering for botnet search space reduction, in: Proceedings of the 2009 Cybersecurity Applications and Technology Conference for Homeland Security, 2009, pp. 141–149.
[5] G. Gu, J. Zhang, W. Lee, Botsniffer: Detecting botnet command and control channels in network traffic, in: Proceedings of Network and Distributed System Security Symposium, 2008.

Please cite this article in press as: K. Wang et al., A fuzzy pattern-based filtering algorithm for botnet detection, Comput. Netw. (2011), doi:10.1016/j.comnet.2011.05.026

[6] T. Karagiannis, A. Broido, M. Faloutsos, K. Claffy, Transport layer identification of P2P traffic, in: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, 2004, pp. 121–134.
[7] M. Pernyi, T.D. Dang, A. Gefferth, S. Molnr, Identification and analysis of peer-to-peer traffic, Journal of Communications 1 (7) (2006) 36–46.
[8] I. Dedinski, H. Meer, L. Han, L.Mathy, D.P. Pezaros, J.S. Sventek, X.Y. Zhan, Cross-layer peer-to-peer traffic identification and optimization based on active networking, in: Active and Programmable Networks: Proceedings of IFIP TC6 7th International Working Conference, 2009, pp. 13–27.
[9] C. Kolbitsch, P.M. Comparetti, C. Kruegel, E. Kirda, X. Zhou, X. Wang, Effective and efficient malware detection at the end host, in: Proceedings of 18th USENIX Security Symposium, USENIX Association, 2009, pp. 351–366.
[10] B. Sadhan, J.M.F. Moura, D. Lapsley, Periodic behavior in botnet command and control channels traffic, in: Proceedings of the 28th IEEE Conference on Global Telecommunications, IEEE Press, 2009, pp. 2157–2162.
[11] H. Choi, H. Lee, H. Lee, 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, pp. 715–720.
[12] T. Werner, Honeytrap - a dynamic meta-honeypot daemon. <http://honeytrap.carnivore.it/>, 2009.
[13] Oracle, Virtualbox. <http://www.virtualbox.org/>.

Kuochen Wang received the B.S. degree in Control Engineering from the National Chiao Tung University, Taiwan, in 1978, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Arizona in 1986 and 1991, respectively. He is currently a Professor/Director in the Institute of Networking Engineering, National Chiao Tung University. He was a Deputy Director of the Computer and Network Center at this university from June 2007 to July 2009. He was a Visiting Scholar in the Department of Electrical Engineering, University of Washington from July 2001 to February 2002. From 1980 to 1984, he was a Senior Engineer at the Directorate General of Telecommunications in Taiwan. He served in the army as a second lieutenant communication platoon leader from 1978 to 1980. His research interests include wireless networks, network security, mobile cloud computing, and power management for multimedia portable devices.

Chun-Ying Huang received the B.S. degree in Computer Science from National Taiwan Ocean University in 2000 and the M.S. degree in Computer Information Science from National Chiao-Tung University in 2002. He received the Ph.D. degree in Electrical Engineering from National Taiwan University in 2007. Since then he took the one-year compulsory military service as a second lieutenant. From August 2008, he joined the Computer Science and Engineering Department at National Taiwan Ocean University as an assistant professor. His current research interests focus on various aspects of computer networks and network security, including key management, attack mitigation, intrusion detection, and traffic analysis. Dr. Huang is a member of IEEE and ACM.

Shang-Jyh Lin received the B.S. degree in Computer Science and Engineering from the Yuan Ze University, Taiwan, in 2008, and the M.S. degree in Computer Science from the National Chiao Tung University in 2010. He was a TA in the Computer Center, Department of Computer Science, National Chiao Tung University from July 2008 to June 2010. His research interests include computer networks, network security, and web applications.

Ying-Dar Lin is Professor of Computer Science at National Chiao Tung University (NCTU) in Taiwan. He received his Ph.D. in Computer Science from UCLA in 1993. He spent his sabbatical year as a visiting scholar at Cisco Systems in San Jose in 2007–2008. Since 2002, he has been the founder and director of Network Benchmarking Lab (NBL, www.nbl.org.tw), which reviews network products with real traffic. He also cofounded L7 Networks Inc. in 2002, which was later acquired by D-Link Corp. His research interests include design, analysis, implementation, and benchmarking of network protocols and algorithms, quality of services, network security, deep packet inspection, P2P networking, and embedded hardware/software co-design. His work on "multi-hop cellular" has been cited over 470 times. He is currently on the editorial boards of IEEE Communications Magazine, IEEE Communications Surveys and Tutorials, IEEE Communications Letters, Computer Communications, and Computer Networks. He is publishing a textbook "Computer Networks: An Open Source Approach" with Ren-Hung Hwang and Fred Baker through McGraw-Hill in February 2011.

Please cite this article in press as: K. Wang et al., A fuzzy pattern-based filtering algorithm for botnet detection, Comput. Netw. (2011), doi:10.1016/j.comnet.2011.05.026