Fast-Flux Bot Detection in Real Time

Ching-Hsiang Hsu[1], Chun-Ying Huang[2], and Kuan-Ta Chen[1]

[1] Institute of Information Science, Academia Sinica
[2] Department of Computer Science and Engineering,
National Taiwan Ocean University

Abstract. The fast-flux service network architecture has been widely adopted by bot herders to increase the productivity and extend the lifespan of botnets' domain names. A fast-flux botnet is unique in that each of its domain names is normally mapped to different sets of IP addresses over time and legitimate users' requests are handled by machines other than those contacted by users directly. Most existing methods for detecting fast-flux botnets rely on the former property. This approach is effective, but it requires a certain period of time, maybe a few days, before a conclusion can be drawn.
In this paper, we propose a novel way to detect whether a web service is hosted by a fast-flux botnet *in real time*. The scheme is unique because it relies on certain intrinsic and invariant characteristics of fast-flux botnets, namely, 1) the request delegation model, 2) bots are not dedicated to malicious services, and 3) the hardware used by bots is normally inferior to that of dedicated servers. Our empirical evaluation results show that, using a passive measurement approach, the proposed scheme can detect fast-flux bots in a few seconds with more than 96% accuracy, while the false positive/negative rates are both lower than 5%.

Key words: Botnet, Request delegation, Document fetch delay, Processing delay, Internet measurement, Supervised classification

1 Introduction

A botnet is a collection of compromised Internet hosts (a.k.a. bots), that have been installed with remote control software developed by malicious users. Such software usually starts automatically when a parasite host boots. As a result, malicious users (a.k.a. bot herders), can coordinate large-scale Internet activities by controlling the bots (the victims). Bot herders always attempt to compromise as many hosts as possible. According to the report of the FBI's "Operation Bot Roast" project [7], more than one million victim IP addresses had been identified on the Internet by the end of 2007, and the number continues to increase. Botnets allow bot herders to engage in various malicious activities, such as launching distributed denial of service (DDoS) attacks, sending spam mails [24], hosting phishing sites [13], and making fraudulent clicks [5]. Statistics show that botnets yield great economic benefits for bot herders [15, 16]; for example, Gartner [8]

estimated that the economic loss caused by phishing attacks alone is as much as US$3 billion per year.

To help legitimate users avoid malicious services (mostly in the form of websites) hosted on a bot, researchers and practitioners have investigated ways to determine whether a host is part of a botnet [9–11, 20]. If a bot is detected, the host owner can remove the remotely controlled software by using malicious software removal tools, or the network ISP can disconnect the bot if the host owner does not take appropriate action. Obviously, bot herders take countermeasures to keep their botnets alive and productive. Particularly, the Fast-Flux Service Network (FFSN) architecture has been used to *increase the productivity* and *extend the lifetime of domain names* linked to the bots.

Usually, a bot herder applies for a domain name for each of his bots and distributes the domain names (normally in the form of URLs) via various channels, such as spam mails or web blogs. However, if a machine is in down time, the bot cannot be controlled and the URL will be temporarily unavailable. Moreover, control of the bot may be lost due to removal of the malicious software. In this case, the bot herder will not gain any more benefits from the domain name unless it is re-mapped to another IP address (of another bot).

An FFSN-based botnet (called a *fast-flux botnet* for short), solves the abovementioned problems because of two architectural innovations: 1) the mapping between domain names and IP addresses, and 2) the way legitimate users' requests are processed.

- First, in a fast-flux botnet, *a domain name is mapped to a number of IP addresses* (possibly hundreds, or even thousands) rather than a single IP address. As a result, if the mapping is handled properly, i.e., a domain name is always resolved to a controllable and live bot, the productivity (in terms of the access rate of malicious services) will be higher than that of a traditional botnet. In addition, if it is known that a bot has been detected, the domain name's link to the bot can be terminated immediately so that their relationship cannot be discovered.
- Second, *legitimate users' requests are indirectly handled by other machines called motherships, rather than the bots the users contact.* In other words, when a legitimate user accesses a service provided by a fast-flux botnet via a URL, the bot that the URL connects to and receives requests from does not handle the requests itself. Instead, it serves as a proxy by delegating the requests to a mothership, and then forwarding the mothership's responses to the user. By so doing, bot herders can update a malicious service (and the content it offers) anytime because they have more control over the mothership and the number of mothership nodes is relatively small compared to that of bots. In addition, since malicious services do not reside on bots, it is easier for bot herders to reduce the footprint of the malicious software so that it is less likely to be detected by anti-malware solutions.

To obscure the link between a domain name and the IP addresses of available bots, fast-flux botnets often employ a strategy that resolves a domain name

— Returned DNS records at time t —

```
;; ANSWER SECTION:
f07b42b93.com.   300   IN A   68.45.212.84
f07b42b93.com.   300   IN A   68.174.233.245
f07b42b93.com.   300   IN A   87.89.53.176
f07b42b93.com.   300   IN A   99.35.9.172
f07b42b93.com.   300   IN A   116.206.183.29
f07b42b93.com.   300   IN A   174.57.27.8
f07b42b93.com.   300   IN A   200.49.146.20
f07b42b93.com.   300   IN A   204.198.77.248
f07b42b93.com.   300   IN A   207.112.105.241
f07b42b93.com.   300   IN A   209.42.186.67
```

— Returned DNS records at time t+300 second —

```
;; ANSWER SECTION:
f07b42b93.com.   300   IN A   64.188.129.99
f07b42b93.com.   300   IN A   69.76.238.227
f07b42b93.com.   300   IN A   69.225.51.55
f07b42b93.com.   300   IN A   76.10.12.224
f07b42b93.com.   300   IN A   76.106.49.207
f07b42b93.com.   300   IN A   76.127.120.38
f07b42b93.com.   300   IN A   76.193.216.140
f07b42b93.com.   300   IN A   99.35.9.172
f07b42b93.com.   300   IN A   200.49.146.20
f07b42b93.com.   300   IN A   204.198.77.248
```

Fig. 1. An example of how a fast-flux botnet rapidly changes the mapping of IP addresses to its domain names. These two consecutive DNS lookups are 300 seconds apart.

to different sets of IP addresses over time. For example, we observed that the malicious service `f07b42b93.com`, which hosts a phishing webpage that deceives users by getting them to reveal their iPhone serial numbers, adopts this strategy. As shown in Fig. 1, during a DNS query at time t, the domain's DNS server replies with 10 `A` records, any of which will lead users to the phishing webpage. The short time-to-live (TTL) value, i.e., 300 seconds, indicates that the records will expire after 300 seconds, so a new DNS query will then be required. At t+300 seconds, we re-issued the same query and obtained another set of IP addresses. In total, there are 19 IP addresses with one duplication in the two sets, which indicates that the bot herder currently owns a minimum of 19 bots. The duplication could occur because the DNS server returns IP addresses randomly, or the bot herder does not have enough bots and cannot provide any more unseen IP addresses. A single fast-flux botnet domain name may be resolved to a huge number of IP addresses. For example, we observed a total of $5,532$ IP addresses by resolving the domain name `nlp-kniga.ru` between October 2009 and March 2010. The larger the IP address pool, the higher will be the "productivity" of

the botnet. As a result, the link between any two bots that serve the same bot herders will be less clear, which is exactly what the bot herders desire.

A number of approaches have been proposed to detect fast-flux botnets. By definition, a fast-flux botnet domain name will be resolved to different IP addresses over time because 1) bots may not be alive all the time; and 2) bot herders want the links between the bots to be less obvious. Therefore, most studies rely on the number of IP addresses of a domain name by actively querying a certain domain name [3, 12] or passively monitoring DNS query activities for a specific period [25] (normally a few days). This approach is straightforward and robust; however, the time required to detect bots is simply too long. A bot herder may only require a few minutes to set up a new domain name and a malicious service to deceive legitimate users; therefore, we cannot spend a few days trying to determine whether a certain domain hosts malicious services. *In order to fully protect legitimate users so that they do not access malicious services unknowingly, we need a scheme that can detect whether a service is hosted by a fast-flux botnet in real time.*

In this paper, we propose such a scheme. The key features of the scheme are as follows:

1. The scheme can work in either a *passive* or an *active* mode. In the passive mode, it works when users are browsing websites; while in the active mode, it can also issue additional HTTP requests and thereby derive more accurate decisions. Irrespective of the mode used, the scheme can determine whether a website is hosted by a fast-flux bot within a few seconds with a high degree of accuracy.
2. The scheme relies on certain intrinsic and invariant characteristics of fast-flux botnets, namely, i) the request delegation model; ii) bots have "owners," so they may not be dedicated to malicious services; and iii) the network links of bots are not normally comparable to those of dedicated servers. Among the characteristics, the first one exists by definition; while the other two, fortunately, cannot be manipulated by bot herders. Consequently, bot herders cannot implement countermeasures against the scheme.
3. The scheme does not assume that a fast-flux botnet owns a large number of bots (IP addresses). Thus, even if a botnet only owns a few bots, as long as it adopts the "request delegation" architecture, our scheme can detect it without any performance degradation.

The remainder of this paper is organized as follows. In Section 2, we discuss existing solutions for detecting fast-flux botnets. The intrinsic properties of fast-flux botnets are analyzed in Section 3. The proposed solution is introduced in Section 4. Section 5 evaluates the proposed solution. Section 6 considers practical issues related to the proposed solution. Section 7 contains some concluding remarks.

2 Related Work

To the best of our knowledge, the Honeynet project [22] was the first research to study the abuse of fast-flux botnets. The authors explained the hidden operations of botnets by giving examples of both single and double fast-flux mechanisms. Single fast-flux mechanisms change the `A` records of domains rapidly, while double fast-flux techniques change both the `A` records and the `NS` records of a domain frequently.

Holz et.al. [12] monitored domain name service (DNS) activities over a seven-week period and proposed a fast-flux botnet domain name detection scheme based on the fluxy-score. The score is computed by counting the number of unique `A` records in all DNS lookups, the number of `NS` records in a single DNS lookup, and the number of unique autonomous system numbers (ASNs) for all DNS `A` records. A number of detection schemes [14, 17, 18, 25] detect fast-flux botnet domain names by monitoring how frequently a domain name changes its corresponding IP addresses. However, these solutions often have to observe DNS activities for a long time (months). Although the observation period can be reduced by using both active and passive monitoring techniques [3], the approach still needs several minutes along with the help of a data center to determine whether a domain name is controlled by a botnet.

The proposed fast-flux botnet detection scheme is fundamentally different from all previous approaches. Since DNS-based detectors often require a long time to identify fast-flux botnets, the proposed solution does not monitor DNS activities. Instead, it relies on several basic properties that are measured at the network level with a short period of time. As a result, it can detect fast-flux botnets accurately and efficiently.

3 Intrinsic Characteristics of Fast-Flux Bots

In this section, we consider the intrinsic characteristics of fast-flux bots, which serve as the basis of the proposed detection method described in Section 4. Since these characteristics are intrinsic and invariant, they are common to fast-flux bots. Therefore, bot herders cannot manipulate them in order to evade detection by the proposed scheme.

3.1 Request Delegation

As mentioned in Section 1, a fast-flux bot does not process users' requests itself. Instead, it acts as a proxy by delegating requests to the mothership, and then forwards responses from the mothership to the users. The purpose of this design is twofold: 1) to protect the mothership from being exposed or detected; and 2) to avoid having to replicate malicious services and content to every bot, which would increase the risk of being detected and also slow down the collection of fraudulent information (e.g., obtaining users' confidential data via phishing). The request delegation design is illustrated in Fig. 2. When a client sends a

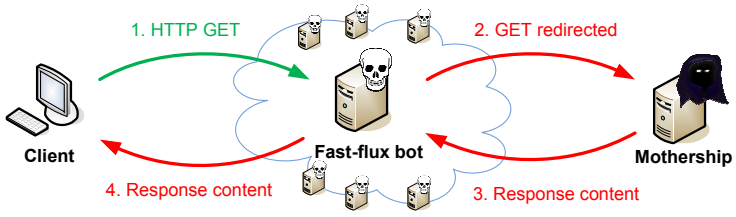

Fig. 2. An example of how a fast-flux botnet delivers malicious content secretly to a client.

request to a fast-flux bot, the request is redirected to a mothership node, as shown in the figure. The node processes the request (mostly by reading a static webpage from a hard disk), and sends the response to the bot. The bot, as a proxy, forwards the response to the requester as if it had handled the request itself.

Because of this design, a client may perceive a slightly longer delay between issuing a request and receiving the response when the "service provider"[1] is a fast-flux bot. The increase in the response time is roughly the same as the message forwarding delay between the bot and the mothership. As long as the request delegation model is employed, technically, the increase in response time cannot be avoided.

3.2 Consumer-Level Hardware

Bot herders expand their collection of bots by compromising as many computers as possible. Most botnets are comprised of residential PCs [23]. One reason is that such PCs are not well-maintained normally; e.g., the anti-virus software may be out-of-date and/or the operating system and applications may not be patched. Residential PCs are normally equipped with consumer-level hardware and usually connect to the Internet via relatively low-speed network links, e.g., ADSL and a cable modem. As a result, compared to dedicated web servers, like those of Google and Yahoo, most bots have relatively low computation power and network bandwidth to access the Internet, which may cause the following phenomena:

- Because of a bot's relatively low computation power, the message forwarding operation at a bot may experience significant delays if any foreground application is running at the same time (see the next subsection).
- Because of a bot's relatively low network bandwidth, and the fact that residential network links are normally shared by a number of users (e.g., users in the same building), it is likely that significant network queuing will occur.

[1] We use the term "service provider" because, although a fast-flux bot is the service provider from the end-user's viewpoint, the actual service is provided by the mothership behind the bot.

This will induce variable queuing time and make a request's response time more fluctuating.

Obviously, bot herders cannot alter the level of a bot's equipment for network bandwidth access. For this reason, we consider such characteristics intrinsic and the phenomena are unalterable by external parties; in other words, longer message forwarding delays and more variable network delays should be widely observable in fast-flux botnets.

3.3 Uncontrollable Foreground Applications

Ideally, bot herders should be able to control bots via remote control software; however, bots are not controlled *exclusively* by bot herders: They are personal computers that may be used by the owners at the same time. For example, a bot may be serving phishing webpages for bot herders at exactly the same time that the PC owner is playing an online game or watching a movie. This possibility indicates that foreground applications run by bot owners and background malicious processes run by bot herders may compete for computing resources, such as the CPU, memory, disk space, and network bandwidth. In other words, if the workloads of bot owners and bot herders compete for resources, the performance of both applications may suffer.

This characteristic implies that the delay incurred by the message forwarding operation at a bot, i.e., the time taken to forward a user's request to the mothership and the time taken to forward the mothership's response to the user, may vary according to the instantaneous foreground workload on the bot. This effect would be especially significant if a bot's computation power is low (due to consumer-level hardware). In this case, any foreground workload would slow the above message forwarding operation, so a high level of variability in message forwarding delays will be observed.

Bot herders cannot avoid this situation because malicious software would be easily detected if it affects the performance of bot owners' foreground applications. More specifically, if a bot herder's malicious software requests a high priority for computation, bot owners may notice that the performance of their foreground applications deteriorates and run a scan, which would detect and remove the malicious software.

3.4 Summary

In Table 1, we list the characteristics that are intrinsic to fast-flux bots, and also compare fast-flux bots with dedicated severs and traditional bots (i.e., bots that malicious services are running on, but they do not delegate users' requests). It is clear that dedicated servers do not have any of the characteristics of fast-flux bots. Traditional bots, on the other hand, are similar to fast-flux bots, except that they do not delegate requests.

The effects of these intrinsic characteristics are also summarized in Table 1. Because of these properties, we expect to see long delays in fetching documents

Table 1. Comparison of the intrinsic characteristics of bots (dedicated servers, traditional bots and fast-flux bots).

	Dedicated servers	Traditional bots	Fast-flux bots	Consequence
Requests delegated	✗	✗	✔	Long delays in fetching documents
Comsumer-level hardware	✗	✔	✔	Low bandwidth & variable network delays
Uncontrollable foreground tasks	✗	✔	✔	Long processing delays

(called document fetch delays hereafter), variable network queuing delays, and long processing delays when users make requests to a malicious service hosted by a fast-flux bot. Measuring the three types of delay form the basis of our fast-flux bot detection scheme, which we discuss in detail in the next section.

4 The Proposed Solution

In this section, we introduce the proposed solution for detecting fast-flux bots. Our scheme assumes that bot herders exploit the bots to execute web-based malicious services, e.g., phishing pages or other types of fraudulent webpages. Specifically, the malicious software on the bots includes a HTTP server that listens to TCP ports 80/443 and accepts HTTP/HTTPS requests. Before describing the proposed scheme, we explain the rationale behind our design:

- *Realtimeness*. We expect the scheme to be able to detect fast-flux bots in real time, e.g., within a few seconds, so that we can prevent legitimate users from proceeding with malicious services in time.
- *Robustness*. We expect that the scheme will not be dependent on the signatures of certain botnet implementations. The scheme must be signature-independent in order to cope with updates from existing botnets as well as new, unknown botnet implementations without degrading the detection performance.
- *Lightweight*. We expect the scheme to be as lightweight as possible so that it can be deployed on any type of device without using too many computing resources.

Given the above guidelines, we propose *a real-time, signature-less, and lightweight detection scheme* for fast-flux bots based on their intrinsic characteristics (cf. Section 3). Under the scheme, if a client tries to download webpages from a web server suspected of being a fast-flux bot, the scheme will monitor the packet exchanges between the client and the server and issue additional HTTP requests if necessary. The decision about whether the server is part of a fast-flux botnet

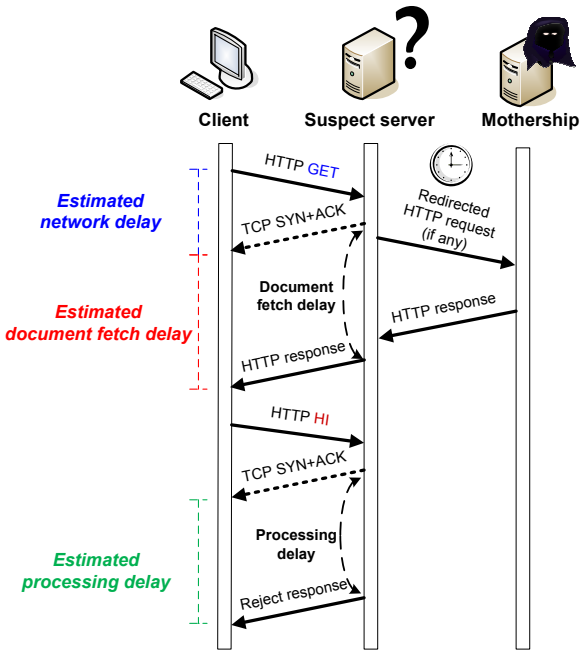

Fig. 3. The measurement techniques used to estimate network delays, processing delays, and document fetch delays based on HTTP requests.

is based on measurements of the packet transmission and receipt times observed at the client. We call the web server that the client sends HTTP requests to a "suspect server" or simply a "server." However, the machine may only be a proxy, so it does not handle HTTP requests itself (e.g., in the case of fast-flux bots).

Next, we define the three delay metrics used to determine whether a suspect server is a fast-flux bot.

1. **Network delay (ND)**: The time required to transmit packets back and forth over the Internet between the client and the server.
2. **Processing delay (PD)**: The time required for the server to process a dummy HTTP request that does not incur any additional computation and I/O operations.
3. **Document fetch delay (DFD)**: The time required for the server to fetch a webpage (either from a hard disk or from a back-end mothership).

Network delays occur at the network-level, while the processing delays occur at the host-level (i.e., at the suspect server). Document fetch delays are more complicated in that they may occur at the host-level only (at the suspect server) if the request delegation model is not employed, or they may arise if the server delegates received requests to a mothership via the Internet. In the latter case,

DFDs involve host-level delays (at the suspect server and the mothership node) and network-level delays (between the suspect server and the mothership node). The measurement techniques used to estimate the three types of delay are shown in Fig. 3. We discuss the techniques in detail in the following sub-sections.

4.1 Network Delay Measurement

Network delay (ND) is defined as the difference between the time a client sends out the first TCP SYN packet to the suspect server and the time the client receives the corresponding TCP SYN+ACK packet from the server. By using this estimate, a TCP connection only yields one network delay sample. To collect more samples, when appropriate, our scheme temporarily disables the persistent connection option in HTTP 1.1, which ensures a separate TCP connection for each HTTP request; thus, the number of ND samples will be the same as the number of HTTP requests.

4.2 Processing Delay Measurement

Measuring processing delays (PD) at the suspect server is not straightforward because HTTP does not support such operations naturally. We need a HTTP command that will respond to the client without contacting the back-end mothership (if any), irrespective of whether the suspect server is a fast-flux bot. For this purpose, we attempted to make the following requests:

1. Valid HTTP requests with methods other than `GET`, e.g., `OPTIONS` and `HEADER` methods.
2. HTTP requests with an invalid version number.
3. HTTP requests with incomplete headers.
4. HTTP requests with an undefined method, e.g., a nonsense `HI` method.

Our experiments showed that most fast-flux bots still contacted their mothership in the first three scenarios. On the other hand, most of them rejected HTTP requests with undefined methods directly by sending back a HTTP response, usually with the status code 400 (Bad Request) or 405 (Method Not Allowed).

Consequently, we estimate the processing delay at the server by subtracting the network round-trip time from the application-level message round-trip time. Specifically, assuming AD is the difference between the time a client sends out a HTTP request with an undefined method and the time the client receives the corresponding HTTP response (code 400 or 405), then a PD sample is estimated by subtracting ND (the network delay) from AD.

4.3 Document Fetch Delay Measurement

We define the document fetch delay (DFD) as the time required for the suspect server to "fetch" a webpage. Since the fetch operation occurs at the server side, we cannot know exactly what happens on the remote server. Thus, we employ

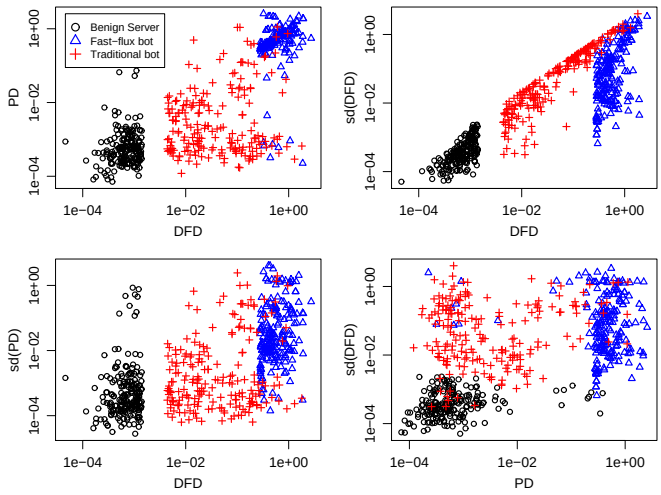

Fig. 4. Scatter plots of processing delays, document fetch delays, and their respective standard deviations. Both the x- and y-axis are in log scale.

the following simple estimator. Assuming RD is the difference between the time a client sends out a successful HTTP GET request and the time the client receives the corresponding HTTP response (code 200), then a DFD sample is estimated by subtracting ND (the network delay) from RD. Figure 4 shows the distribution of DFD, PD and their respective standard deviations measured for benign servers, traditional bots, and fast-flux bots.

4.4 Decision Algorithm

In this sub-section, we explain how we utilize the three delay metrics in our decision algorithm.

- The objective of measuring network delays is to capture the level of network congestion between a client and the suspect server. As per Section 3.2, the ND and sd(ND), where sd(·) denotes the standard deviation, tend to be (relatively) large if the suspect server is a fast-flux bot rather than a benign, dedicated web server.
- The processing delay helps us determine the server's workload and the required computation power. If there are other workloads on the server, the estimated processing delays would be high and fluctuate over time. Thus, as per Section 3.2 and Section 3.3, the PD and sd(PD) tend to be large if the suspect server is a fast-flux bot.
- The document fetch delay indicates how much time the server takes to fetch a webpage. Because of the request delegation model (Section 3.1), DFD and sd(DFD) tend to be large if the suspect server is a fast-flux bot.

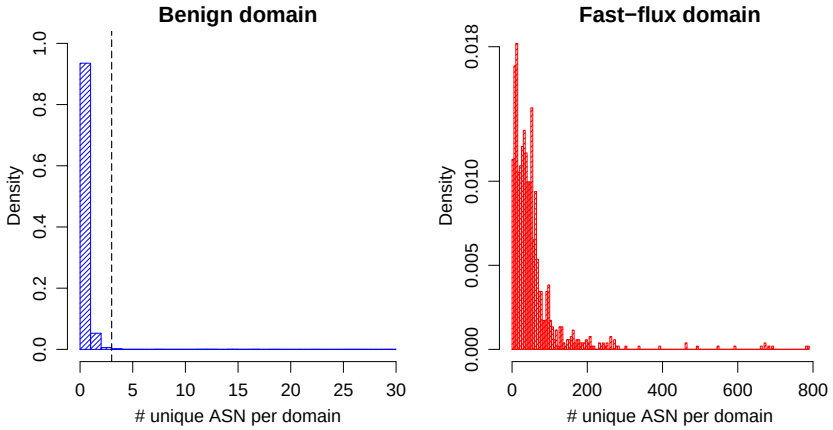

Fig. 5. The distribution of unique autonomous system numbers (ASNs) per domain name in the dataset of benign servers and fast-flux bots.

For a suspect server, we collect six feature vectors (ND, PD, DFD, and their respective standard deviations), each of which contains n elements assuming n HTTP GET requests are issued. For the PD samples, another n HTTP requests with an undefined method must also to be issued.

To determine whether a suspect server is a fast-flux bot, which is a binary classification problem, we employ a supervised classification framework and use linear SVM [4] as our classifier. A data set containing the delay measurement results for both benign web servers and web servers hosted on fast-flux bots is used to train the classifier. When a client wishes to browse pages on an unknown website, our scheme collects the delay measurements and applies the classifier to determine whether the suspect server is part of a fast-flux botnet.

5 Methodology Evaluation

In this section, we evaluate the performance of the proposed fast-flux bot detection scheme. First, we describe the data set and examine whether the derived features differ significantly according to the type of suspect server. Then, we discuss the detection performance of the scheme and consider a passive use of the scheme.

5.1 Data Description

To evaluate the performance of the proposed scheme in real-life scenarios, we need a set of URLs that legitimate users can browse. Our dataset contains the following three categories of URLs, which point to different kinds of servers:

– *Benign servers*: The top 500 websites listed in the Alexa directory [1].

Table 2. The trace used to evaluate the detection performance of the proposed scheme.

Host type	#domain	#IP address	#session	#connection
Benign servers	500	3,121	60,936	565,466
Traditional bots	16,317	9,116	79,694	943,752
Fast-flux bots	397	3,513	213,562	726,762

- *Traditional bots*: URLs that appear in the PhishTank database [19] with suspicious fast-flux domains removed (see below).
- *Fast-flux bots*: URLs that appear in the ATLAS Global Fast Flux database [2] and the FastFlux Tracker at `abuse.ch` [6].

Between January and April 2010, we used `wget` to retrieve the URLs in our dataset at hourly intervals. During the web page retrieval process, we ran `tcpdump` to monitor all the network packets sent from and received by the client. After retrieving each web page, we sent out a HTTP request with the undefined method "HI" to measure the processing delays that occurred at the suspect server, as described in Section 4.2.

We found that some URLs in the PhishTank database actually point to fast-flux bots, and some URLs listed as pointers to fast-flux bots may actually point to traditional bots. Therefore, after collecting the data, we performed a post hoc check based on the number of distinct autonomous system numbers (ASNs). Figure 5 shows the distributions of distinct ASNs of benign domain names and fast-flux domain names over the trace period. Nearly all the benign domain names were associated with three or fewer ASNs, while most fast-flux domain names were associated with many more ASNs over the three-month period. Based on this observation, we set 3 ASNs as the threshold to determine whether or not a domain name was associated with a fast-flux botnet. Thus, if a domain name was reported as a non-fast-flux bot, but it was associated with four or more ASNs (or vice versa), we regarded the domain name as questionable. We simply removed such domain names from our trace to ensure its clarity and correctness. In addition, if a URL was unavailable due to domain name resolution failures, packet unreachable errors, HTTP service shutdown, or removal of corresponding web services for 10 successive attempts, we removed it from the dataset.

The three-month trace is summarized in Table 2, where a connection refers to a TCP connection, a session refers to a complete web page transfer (including the HTML page and its accessory files, such as images and CSS files). As we turned off the HTTP 1.1 persistent connection option in order to acquire more samples for the delay metrics (cf. Section 4), the number of connections is much higher than that of sessions because a web page often contains several accessory files (maybe even dozens). Figure 6 shows the top 10 (out of 19) top-level domains and the top 20 (out of 127) countries associated with the observed fast-flux bots.

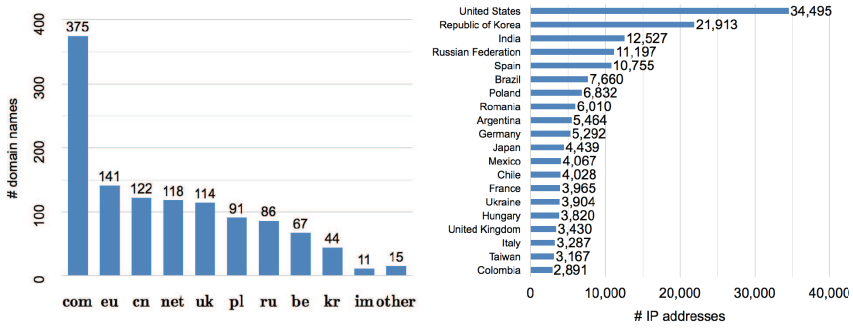

Fig. 6. (a) The top 10 top-level domains and (b) the top 20 countries associated with the fast-flux domain names in our dataset.

5.2 A Closer Look at the Derived Features

We now examine whether the empirical delay measurements derived during web browsing can be used to distinguish between fast-flux bots and benign servers. First, we investigate whether, as expected, consumer-level hosts incur higher and more variable processing delays and more variable network delays (cf. Section 3.4). To do this, we use a common technique that infers whether a host is associated with dial-up links, dynamically configured IP addresses, or other low-end Internet connections based on the domain name of reverse DNS lookups [21]. For example, if a host's domain name contains strings like "dial-up," "adsl," and "cable-modem," we assume that the host is for residential use and connects to the Internet via relatively slow links. Figure 7 shows the distributions of the six features for normal and consumer-level hosts. The plots fit our expectation that consumer-level hosts of fast-flux botnet incur more variable network delays, longer processing delays, and more variable processing delays than those of dedicated servers. In addition, we consider that the longer and more variable document fetch delays are due to lower computation power and longer disk I/O access latency on the consumer-level hosts.

Figure 8 shows the distributions of the six features for benign servers, traditional bots, and fast-flux bots. Clearly, fast-flux bots lead to much higher magnitudes for all six features compared with the other two server categories, manifesting the effects of the intrinsic characteristics of fast-flux bots. The magnitudes of the six features of traditional bots are generally lower than those of fast-flux bots, but higher than those of benign servers except for the standard deviation of network delays. We believe this is because benign servers usually have more visitors than the other two categories of servers; therefore, network links to benign servers tend to be busy and it is more likely that a slightly higher degree of network queuing and delay variations will be observed.

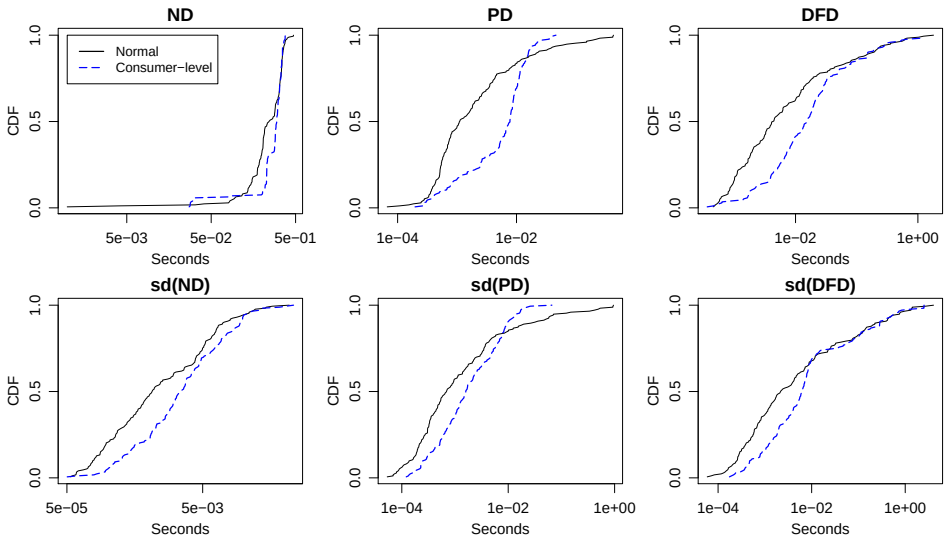

Fig. 7. The cumulative distribution functions of network delays, processing delays, and document fetch delays, and their respective standard deviations of normal and consumer-level hosts were measured based on 5 probes.

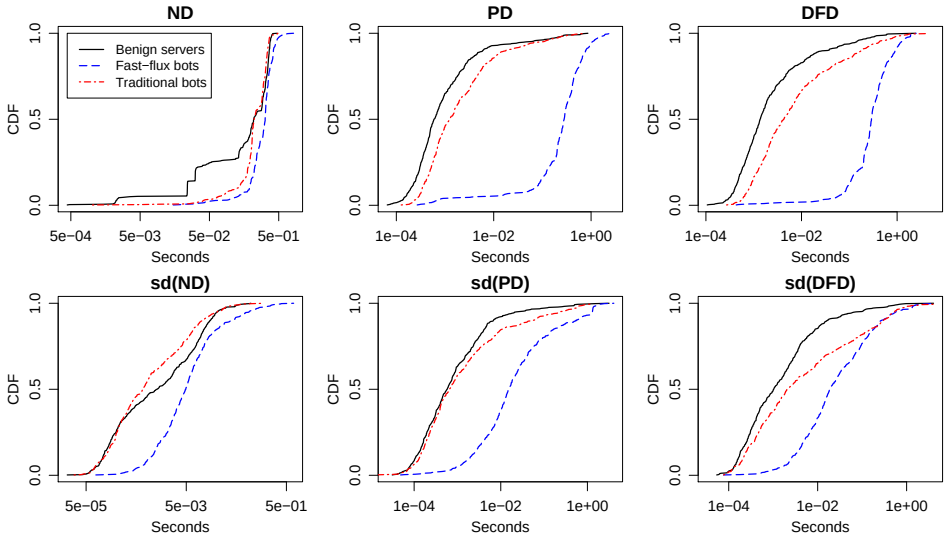

Fig. 8. The cumulative distribution functions of network delays, processing delays, and document fetch delays, and their respective standard deviations of three server categories were measured based on 5 probes.

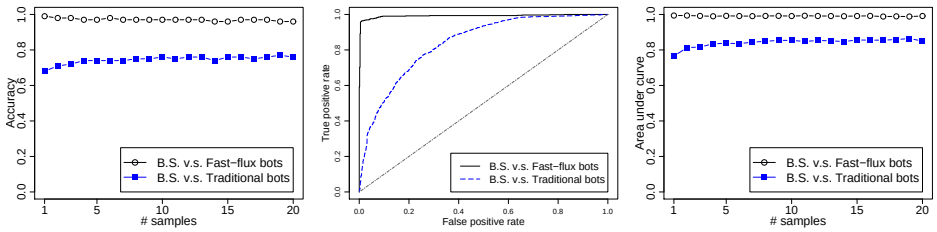

Fig. 9. (a) The relationship between the classification accuracy and the number of samples; (b) the ROC curve of the SVM classifier using 5 probes; and (c) the relationship between the area under the curve and the number of probes.

5.3 Detection Performance

The graphs in Figure 8 confirm that the six features we derived may vary significantly according to the type of web server a user browses. In this sub-section, we perform supervised classification using SVM based on the derived six features.

Although we focus on the detection of fast-flux bots, we also include traditional bots in our evaluation. This is because, according to our analysis (Section 3), traditional bots also behave differently to benign servers in terms of most of the defined delay metrics. We perform two types of binary classification using SVM, namely, benign servers vs. fast-flux bots and benign servers vs. traditional bots. Figure 9(a) shows the relationship between the classification accuracy and the number of samples observed (which may vary according to the number of accessory files of webpages), where the accuracy is derived using 10-fold cross validation. The results show that our scheme achieves more than 95% accuracy when we try to distinguish fast-flux bots from benign servers, even when only one sample (i.e., the TCP connection) is observed. We find that it is more difficult to distinguish between benign servers and traditional bots because the classification accuracy is only 70%–80%; however, the accuracy rate increases when more samples are observed.

Figure 9(b) shows the ROC curves of the two types of classification based on 5 samples. The area under the curve (AUC), which distinguishes between benign servers and fast-flux bots, is 0.993; hence, the proposed detection scheme performs almost perfectly in this scenario. The AUC degrades to 0.83 when we try to classify traditional bots from benign servers, which implies that our detection scheme can detect traditional bots with a moderate degree of accuracy. As the number of samples may affect the classification performance, we plot the relationship between the AUC and the number of samples in Fig. 9(c). The graph shows that the detection performance remains nearly constant regardless of the number of samples used for fast-flux bot detection (the AUC is always higher than 0.99). In contrast, the number of samples is more important when we try to detect traditional bots, as the AUC increases above 0.8 if more than 10 samples are observed before classification is performed.

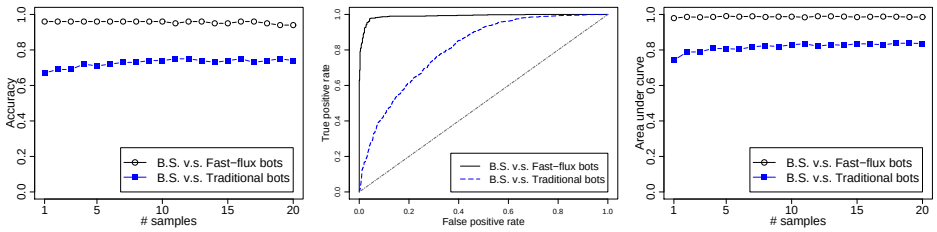

Fig. 10. (a) The relationship between the classification accuracy and the number of samples, (b) the ROC curve of the SVM classifier using 5 probes, and (c) the relationship between the area under the curve and the number of probes in the passive mode.

5.4 Passive Mode

The network delay and document fetch delay can be measured by passive measurements when users are browsing webpages, but an active approach must be used to measure the processing delay (i.e., by sending HTTP requests with an undefined method). Since active measurements incur additional overhead, to keep our method lightweight whenever possible, we consider that a "passive mode" would be quite useful when traffic overhead is a major concern.

In the passive mode, instead of using all six features, we only use the average and standard deviations of network delays and document fetch delays in the supervised classification. The classification accuracy is plotted in Fig. 10. We observe that the classification between fast-flux bots and benign servers is hardly affected by the removal of the "active features," i.e., processing delays and their standard deviations, except when the number of samples is quite small. We believe this indicating that document fetch delays already serve as a powerful indicator for distinguishing the two server categories. On the other hand, the classification accuracy between benign servers and traditional bots is slightly affected by the removal of active features, as processing delays play an important role in distinguishing between the two types of servers. The ROC curves and the AUCs of different numbers of samples shown in Figure 10 also indicate that the passive mode of our scheme yields accurate detection results, especially when a fast-flux-bot detection method is required.

6 Discussion

In this section, we discuss several issues that are worth investigating further.

6.1 Content Delivery Network

One concern raised in a previous work on fast-flux bot detection [17] is that content delivery networks (CDNs) share a similar property with fast-flux botnets;

that is, the nodes in CDNs and fast-flux botnets are associated with multiple IP addresses rather than a single IP address. This leads to confusion if a fast-flux botnet detection scheme is based on a number of IP addresses (or autonomous systems) that are associated with a certain domain name [3,12,17,25]. However, this is not a problem in our proposed method because it does not count the number of IP addresses.

6.2 Proxy Server

Although proxy servers also employ the request delegation model, we argue that the proposed scheme does not confuse fast-flux bots with proxy servers. The reason is that proxy servers are clearly visible to the end users, and the users' clients are aware that they are fetching web documents from a web server with the help of a proxy server. On the other hand, a fast-flux bot does not pretend to be a proxy server because the HTTP proxy protocol does not hide the identity of back-end web servers unless a transparent proxy is used; therefore, the mothership nodes will be revealed, which is a situation that bot herders strive to avoid. Furthermore, if a transparent proxy is used, the proposed method will not be affected because the roles in the request delegation model are different. This is because the suspect servers contacted by users do not delegate requests to others; instead, the request-delegation operation is performed by a hidden man-in-the-middle (i.e., a transparent proxy server), which may only reduce document fetch delays. Therefore, proxy servers along the paths between users and suspect servers will not be detected as fast-flux bots.

6.3 Deployment

Our scheme can be deployed in a number of approaches. First, because of its lightweightness, it can run on end-users' machines, such as personal computers or even mobile devices. In this case, it can be implemented as a browser add-on or stand-alone software that monitors users' web browsing activities and warns users when they are browsing a website hosted by fast-flux bots.

Second, it may be more convenient if the scheme is deployed at a gateway router to protect all the users in a local area network. Since the transmission latency between a gateway router and a host is usually negligible, the delay metrics measured on the router would be roughly the same as those measured on users' computers. Therefore, we can simply monitor all outgoing HTTP requests, measure the delays, and notify users if the measurements indicate that a certain HTTP request has been sent to a fast-flux bot. We consider this to be an efficient way to deploy the proposed detection scheme to protect legitimate users.

6.4 Limitations

Although the proposed detection scheme achieves high accuracy, as shown by the results in Section 5, it has some limitations. Recall that fast-flux bots are

normally equipped with consumer-level hardware and connect to the Internet with (relatively) narrower network links. The proposed scheme may fail in the following cases:

1. A bot herder may compromise powerful servers and incorporate them into a fast-flux botnet.
2. A benign server may not be equipped with high-level hardware like the dedicated web servers provided by Internet service providers.

In the first case, we believe that bots with consumer-level hardware would still dominate because high-level and high-connectivity servers are normally well-maintained and patched; hence, they are less likely to be infected and controlled by malicious software. If this should happen, we would observe short processing delays at the suspect server. The second case may occur when web servers are set up for amateur and casual use. Then, we would observe long and variable processing delays and network delays when users access webpages via such web servers. In both cases, as our method relies on all three intrinsic characteristics in the active mode (or two in the passive mode) rather than a single characteristic, a compromised server could still be detected using other characteristics, especially the "long document fetch delay" property.

7 Conclusion

We have proposed a novel scheme for detecting whether a web service is hosted by a fast-flux botnet in real time. Evaluations show that the proposed solution achieves a high detection rate and low error rates. Unlike previous approaches, our scheme does not assume that a fast-flux botnet owns a large number of bots (IP addresses). Thus, even if a botnet only owns a few bots, as long as it adopts the "request delegation" architecture, the proposed scheme can detect the botnet without any performance degradation.

In addition to being efficient and robust, the proposed solution is lightweight in terms of storage and computation costs. Therefore, it can be deployed on either fully fledged personal computers or resource-constrained devices to provide Internet users with complete protection from botnet-hosted malicious services.

8 Acknowledgment

This research was supported in part by National Science Council under the grant NSC 97-2218-E-019-004-MY2 and by Taiwan Information Security Center at NTUST (TWISC@NTUST) under the grant NSC 99-2219-E-011-004.

References

1. Alexa: Alexa the web information company, `http://www.alexa.com`

2. ATLAS: Arbor networks, inc., http://atlas.arbor.net/
3. Caglayan, A., Toothaker, M., Drapeau, D., Burke, D., Eaton, G.: Real-time detection of fast flux service networks. In: Proceedings of the Cybersecurity Applications & Technology Conference for Homeland Security-Volume 00. pp. 285–292 (2009)
4. Chang, C., Lin, C.: Libsvm: a library for support vector machines (2001)
5. Click Forensics, I.: Botnets accounted for 42.6 percent of all click fraud in Q3 2009 (2009), http://www-staging.clickforensics.com/newsroom/press-releases/146-botnets-accounted.html
6. dnsbl.abuse.ch: abuse.ch fastflux tracker (2010), http://dnsbl.abuse.ch/fastfluxtracker.php
7. FBI: Over 1 million potential victims of botnet cyber crime (2007), http://www.fbi.gov/pressrel/pressrel07/botnet061307.htm
8. Gartner: Gartner survey shows phishing attacks escalated in 2007; more than $3 billion lost to these attacks (2007), http://www.gartner.com/it/page.jsp?id=565125
9. Gu, G., Perdisci, R., Zhang, J., Lee, W.: BotMiner: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In: Proceedings of the 17th USENIX Security Symposium (2008)
10. Gu, G., Porras, P., Yegneswaran, V., Fong, M., Lee, W.: BotHunter: Detecting malware infection through IDS-driven dialog correlation. In: Proceedings of the 16th USENIX Security Symposium. pp. 167–182 (2007)
11. Gu, G., Zhang, J., Lee, W.: BotSniffer: Detecting botnet command and control channels in network traffic. In: Proceedings of the 15th Annual Network and Distributed System Security Symposium (2008)
12. Holz, T., Gorecki, C., Rieck, K., Freiling, F.: Measuring and detecting fast-flux service networks. In: Proceedings of the Network & Distributed System Security Symposium (2008)
13. Ianelli, N., Hackworth, A.: Botnets as a vehicle for online crime. CERT Coordination Center (2005)
14. McGrath, D., Kalafut, A., Gupta, M.: Phishing infrastructure fluxes all the way. IEEE Security & Privacy pp. 21–28 (2009)
15. Moore, T., Clayton, R.: Examining the impact of website take-down on phishing. In: Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit (2007)
16. Namestnikov, Y.: The economics of Botnets (2009)
17. Nazario, J., Holz, T.: As the net churns: Fast-flux botnet observations. In: International Conference on Malicious and Unwanted Software, MALWARE (2008)
18. Passerini, E., Paleari, R., Martignoni, L., Bruschi, D.: FluxOR: detecting and monitoring fast-flux service networks. Detection of Intrusions and Malware, and Vulnerability Assessment pp. 186–206 (2008)
19. PhishTank: http://www.phishtank.com
20. Shadowserver: http://www.shadowserver.org
21. Spamhaus: http://www.spamhaus.org
22. The Honeynet Project: Know your enemy: Fast-flux service networks (2007)
23. The Honeynet Project: Know your enemy: Tracking botnets (2008)
24. TRACELabs, M.: Marshal8e6 security threats: Email and web threats (2009)
25. Zhou, C., Leckie, C., Karunasekera, S., Peng, T.: A self-healing, self-protecting collaborative intrusion detection architecture to trace-back fast-flux phishing domains. In: Proceedings of the 2nd IEEE Workshop on Autonomic Communication and Network Management (2008)