Fluxing Botnet Command and Control Channels with URL Shortening Services

Sangho Lee[a,*], Jong Kim[b]

[a] *Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Republic of Korea*
[b] *Division of IT Convergence Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Republic of Korea*

Abstract

URL shortening services (USSes), which provide short aliases to registered long URLs, have become popular owing to Twitter. Despite their popularity, researchers do not carefully consider their security problems. In this paper, we explore botnet models based on USSes to prepare for new security threats before they evolve. Specifically, we consider using USSes for alias flux to hide botnet command and control (C&C) channels. In alias flux, a botmaster obfuscates the IP addresses of his C&C servers, encodes them as URLs, and then registers them to USSes with custom aliases generated by an alias generation algorithm. Later, each bot obtains the encoded IP addresses by contacting USSes using the same algorithm. For USSes that do not support custom aliases, the botmaster can use shared alias lists instead of the shared algorithm. DNS-based botnet detection schemes cannot detect an alias flux botnet, and network-level detection and blacklisting of the fluxed aliases are difficult. We also discuss possible countermeasures to cope with these new threats and investigate operating USSes.

Keywords: Botnet, DNS, domain flux, URL shortening service

1. Introduction

As data hierarchies and command structures of Web sites become complex, long URLs have appeared. Long URLs, however, are not suitable for

*Corresponding author. Tel.: +82 54 279 2915. Fax: +82 54 279 1805
Email addresses: sangho2@postech.ac.kr (Sangho Lee), jkim@postech.ac.kr (Jong Kim)

services such as Twitter that restrict the length of messages to 140 characters or small screen devices such as smart phones. To solve the long URL problem, URL shortening services (USSes) have appeared [1]. `bit.ly` and `tinyurl.com` are the most famous USSes [2], and LongShore lists more than 600 USSes in their Web page [3].

A URL shortening service works as follows. When Alice requests a shortened URL, or *alias*, of a long URL, such as `http://long-url.com/sufficiently.html`, from `bit.ly`, it will return a shortened URL, such as `http://bit.ly/abcdef`, to Alice. She will then distribute the shortened URL to her friends via Twitter, instant messaging, or E-mail. Later, when Bob, a friend of Alice, opens the link `http://bit.ly/abcdef`, `bit.ly` will reply to Bob with a page redirection message, and then Bob will move to the long URL. The strength of USSes is that Alice can share any long URLs with her friends using shortened forms. On the other hand, USSes have an unavoidable delay owing to redirections [1, 4].

Functionally, URL shortening is similar to the domain name system (DNS) [5, 6]. DNS associates a domain name with a set of IP addresses. For instance, when a host wants to connect to `google.com`, it has to obtain an IP address of `google.com`, such as `72.14.203.103`, from a DNS name server. The differences are that 1) a USS deals with URLs but DNS deals with IP addresses, 2) many USSes do not allow modification of registered URLs but DNS allows modification of registered IP addresses, 3) a USS allows one URL for each alias but DNS allows several IP addresses for each domain name, and 4) a USS is a Web application but DNS is a core Internet service.

Many security attacks are aimed at DNS. Among them, botnets are the most serious security attack because they will become a basis of other security attacks, including E-mail spam, phishing, and distributed denial of service (DDoS) attacks. A botnet is a collection of compromised hosts that has a command and control (C&C) channel to its botmaster. Through the C&C channel, the botmaster can direct his botnet to perform various security attacks. Undoubtedly, he wants to hide his C&C channels for as long as possible to increase the lifetime of his botnet.

To hide the C&C channels, a botmaster adopts methods such as IP fast flux and domain flux [7, 8]. Fast flux is a method that frequently changes a set of IP addresses associated with a domain name. A botmaster uses this method to hide the real IP addresses of his C&C servers. In the fast flux method, even when bots use the same domain name to contact C&C servers, the IP addresses that each bot tries to connect to change frequently; thus, taking down or blocking a specific C&C server is difficult. To detect fast flux

botnets, many schemes based on active or passive DNS traffic monitoring have been proposed [9, 10, 11, 12, 13].

The fast flux method has a single-point-of-failure problem. When an investigator detects a botnet's domain name, he can obstruct the bots' attempts to connect to their C&C servers by blocking connection procedures to the corresponding domain name. To avoid this problem, recent botnets such as Conficker [14] and Torpig [15] adopt domain flux to hide their C&C servers. Domain flux is a method that frequently changes domain names that are associated with a single IP address or C&C infrastructure. Each bot runs a shared domain generation algorithm [14, 15] to compute a list of domain names and then attempts to contact the domain names in the list until it finds its C&C servers. Inputs of the algorithm will be the current date information and some numeric parameters. Some algorithms even use external information such as Twitter trends as their inputs because such information is difficult for investigators to fabricate. Since domain names change frequently, domain name blacklisting cannot prevent the domain flux method. To detect domain flux, schemes such as reputation-based domain blacklisting [16], proactive domain blacklisting by predicting new malicious names [17, 18], and activity-based malicious domain detection [19] have been proposed. These schemes are also based on DNS traffic monitoring.

Although a botmaster uses IP fast flux and domain flux to conceal his botnets, investigators eventually detect the botnets by monitoring DNS traffic because both methods rely on DNS. P2P botnets such as Storm [20, 21] and Waledac [22, 23] do not rely on DNS; thus, they can bypass DNS traffic monitoring. Managing P2P botnets, however, is more difficult than managing centralized botnets because of their decentralized structure. Consequently, a botmaster is in search for a method that can hide his C&C channels while keeping manageability.

As we pointed out, a URL shortening service (USS) is similar to DNS. Therefore, *a botmaster can use USSes instead of DNS to hide his C&C channels*. Since botmasters always try to find a new method to distribute and manage their botnets, exploration of potential threats is important to prevent them from appearing. Currently, attackers use USSes to hide real URLs of spam or phishing sites [24, 25, 26, 27, 28]. This situation implies that they already consider USSes as one of their attack vectors and thus they may seek different usages of USSes. Therefore, in this paper, we explore a new botnet with alias flux that use USSes to hide their C&C channels and discuss possible countermeasures to these new threats. The basic idea of the alias flux method is changing the aliases (shortened URLs) associated with obfuscated IP addresses of C&C servers, similar to the domain flux method.

Each bot will execute an alias generation algorithm or retrieve alias lists to obtain scheduled aliases, and then it will contact USSes with these aliases to find C&C servers. Since a botmaster registers obfuscated IP addresses of C&C servers to USSes instead of their domain names, no DNS query to find C&C servers is generated. Moreover, some USSes, such as `bit.ly`, support HTTPS. Because HTTPS cloaks the name and content of retrieved page, a traffic monitor cannot capture the retrieved aliases and obfuscated IP addresses. On the other hand, in domain flux, a traffic monitor can capture the retrieved domain names and their IP addresses because they are public information.

We summarize the contributions of this paper as follows:

- We introduce a new botnet model that uses USSes instead of DNS.
- We describe an alias flux method based on USSes, which is similar to the domain flux method of current botnets. Because it can encrypt its traffic with HTTPS, previous domain flux detection schemes cannot detect it. We also discuss possible countermeasures.
- We investigate and classify operating USSes. We also present more in-depth analysis of `bit.ly` and `goo.gl`.

We organize the remainder of this paper as follows. In Section 2, we introduce system models and assumptions. In Section 3, we explain alias flux methods to hide C&C channels. In Section 4, we analyze the alias flux botnet. In Section 5, we introduce countermeasures to the alias flux botnet. In Section 6, we discuss related work. Finally, we conclude this paper in Section 7.

2. System Models and Problem Scope

2.1. URL Shortening Service Model

We model a basic URL shortening service (USS). Our model is almost the same as existing USSes such as `bit.ly`. We assume that the domain name of our USS is `u.ss`.

When a client requests a registration for a long URL, such as `http://longurl.com/huge-page.html`, via a Web interface or API, the USS first checks whether the URL already exists in its database. If the URL does not exist, then the USS creates a *unique random alias*, stores the long URL and the alias to the database, and replies to the client with the shortened URL. The alias is a string of case-sensitive alphanumeric characters. For

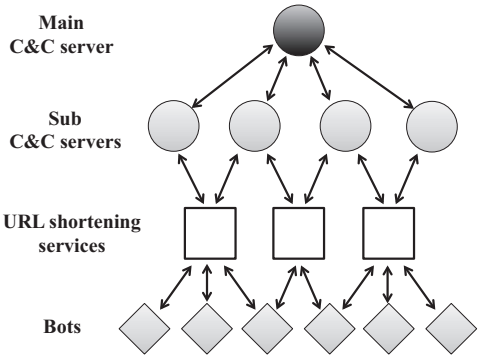


Figure 1: Presented botnet model

instance, if the USS assigns `0aeqf9` to `http://longurl.com/huge-page.html`, then `http:/u.ss/0aeqf9` is a shortened URL of `http://longurl.com/huge-page.html`. If the URL already exists in its database, then the USS replies to the client with the stored shortened URL.

Since a random alias, such as `0aeqf9`, is difficult to remember, our USS also supports *custom aliases*. For instance, when a client registers `http://longurl.com/huge-page.html` to the USS, he can give a custom alias such as `long` to the USS to create a corresponding shortened URL `http://u.ss/long`. We name a URL shortening service that supports custom aliases a *customizable URL shortening service* and a URL shortening service that only supports random aliases a *randomized URL shortening service*.

Later, when another client attempts to access the shortened URL `http://u.ss/0aeqf9` or `http://u.ss/long`, the USS responds with a redirection message to redirect the client to `http://longurl.com/huge-page.html`. Like `bit.ly`, our USS supports encrypted communications when retrieving shortened URLs. Hence, clients can hide the aliases they visit and the returned long URLs from passive traffic monitors. When a client attempts to access a shortened URL that is not registered, the USS responses to the client with a not found error. The mapping information between long URLs and shortened URLs is static; clients cannot remove or change it.

2.2. Botnet Model

We present a simple botnet model using URL shortening services (Figure 1). A *botmaster* directly manages the main C&C server. By using this server, the botmaster controls his botnets to perform various malicious activities including DDoS attacks, malware distribution, and phishing site hosting.

Sub C&C servers are proxy servers of the main C&C server. The purpose of these servers is to hide the main C&C server from investigators. To achieve this goal, the botmaster frequently changes these servers. The botmaster may use other botnets as sub C&C servers to increase their stealthiness.

Bots are machines that are infected by malware distributed by the botmaster. To receive commands and update its malware, each bot will contact with the sub C&C servers. Since the botmaster frequently changes the sub C&C servers, each bot needs to obtain the IP addresses of the changed sub C&C servers from other sources.

The considered botnet model uses URL shortening services (USSes) to solve the preceding problem. The botmaster registers the IP addresses of the sub C&C servers to USSes whenever they are changed, and then each bot connects to USSes to obtain the new IP addresses. The botmaster and each bot have to share an alias generation algorithm to precisely register and find the IP addresses.

2.3. Problem Scope

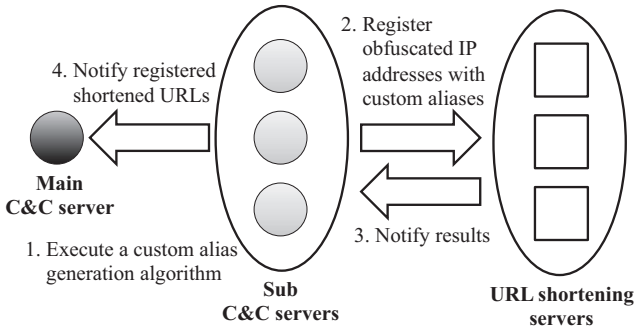
In this paper, we only consider the case where botmasters abuse public USSes instead of maintaining their own malicious USSes. Because many benign communications exist between hosts and public USSes, investigators will have more difficulties discovering and blocking C&C communications compared to botnets using the malicious USSes. In contrast, the botmaster only has restricted capabilities because he cannot modify any information stored in USSes such as the mapping information between long URLs and shortened URLs.

3. Alias Flux to Hide C&C Channels

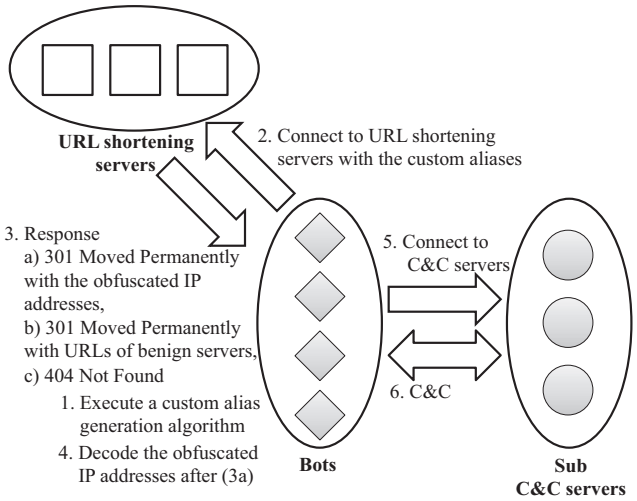
We will explore an alias flux method that hides C&C channels by using URL shortening services. The alias flux method is similar to the domain flux method but it frequently changes the aliases of IP addresses of C&C servers instead of their domain names. We explain two alias flux methods: alias flux with customizable USSes and alias flux with randomized USSes.

3.1. Alias Flux with Customizable USSes

We explain an alias flux method for USSes that support custom aliases when registering URLs. As recent botnets such as Conficker [14] and Torpig [15] use domain name generation algorithms, the alias flux botnet uses a custom alias generation algorithm that generates a set of random strings



(a) Registration of sub C&C servers to USSes



(b) Discovery of sub C&C servers via USSes

Figure 2: Alias flux with customizable USSes

composed of alphabet letters and digits. The inputs of the algorithm will be the current date information and the number of desired strings. For each period (e.g., a day, week, or month) the alias flux botnet generates α different aliases by using the algorithm. Therefore, bots can access up to α sub C&C servers during each period.

Alias flux has two phases: registering the IP addresses of sub C&C servers to USSes and discovering these IP addresses of sub C&C servers via the USSes (Figure 2). A botmaster will not only register newly added sub C&C servers but also periodically reregister the existing sub C&C servers while changing their IP addresses to bypass C&C server detection and IP address blacklisting. Whenever the botmaster registers each sub C&C server

to USSes, he obtains different aliases for the server; thus, we name this method alias flux.

There are many choices when selecting USSes for alias flux. Here, we assume that the botmaster designates u USSes and registers the IP addresses of the sub C&C servers to all of the u designated USSes. Later, when each bot attempts to find C&C servers, it will randomly select one of the designated USSes and use it.

Registration Phase: To register the sub C&C servers to the u designated USSes (Figure 2a), the botmaster executes a custom alias generation algorithm to obtain α custom aliases (1). The input given to the custom alias generation algorithm will be a future date, such as tomorrow or next week, because bots will access these aliases in the future.

Next, the botmaster registers the obfuscated IP addresses of the sub C&C servers to the USSes using the custom aliases (2). He needs IP address obfuscation to hide the IP addresses of the sub C&C servers from the USSes while keeping their forms as legitimate URLs. The botmaster can adopt various methods to obfuscate his IP addresses. For instance, the botmaster may use four different dictionaries of 256 words to represent each byte of an encrypted IPv4 address. At first, he encrypts an IP address, such as `8.8.8.8`, by using the dictionaries to represent it using four words, e.g., "Tom loves your dog." Thereafter, to convert the four words to a legitimated URL, the botmaster may use search engine queries such as `http://famous-search-engine.com/search?q=Tom+loves+your+dog`, non-existent query strings such as `http://famous-webpage.com/index.html?Tom=loves&your=dog`, or anchors such as `http://famous-webpage.com/index.html#Tom_loves_your_dog`. The botmaster can also encode multiple IP addresses as a single URL until its length becomes too long.

The botmaster will receive notifications from the USSes (3). Because someone may have already registered custom aliases generated by the algorithm to the USSes, the received notifications potentially include some error messages. In that case, the botmaster needs to re-register the refused sub C&C servers' IP addresses using different aliases. If the probability of alias collision of each USS is p_c, then he can register $(1 - p_c^u)\alpha$ sub C&C servers to at least one of the USSes. When the number of C&C servers is smaller than $(1 - p_c^u)\alpha$, he will redundantly register some sub C&C servers to the USSes to avoid 404 Not Found errors.

Finally, the main C&C server will receive the newly registered shortened URLs of the sub C&C servers (4). To register sub C&C servers for the next n periods, the botmaster has to change the URLs of the sub C&C servers slightly to obtain $n\alpha$ URLs and then register them to USSes with the next

$n\alpha$ aliases.

Discovery Phase: Each bot will periodically attempt to discover sub C&C servers (Figure 2b). First, each bot executes a custom alias generation algorithm to obtain α custom aliases (1). The input given to the custom alias generation algorithm is the current date information.

Next, each bot randomly selects one of the u USSes and one of the α custom aliases and connects to the selected USS using the selected custom alias (2). The bot will use HTTPS to communicate with the USS to hide the custom alias from network administrators and avoid network-level blacklisting. The probability that two bots connect to the same shortened URL is $(u\alpha)^{-1}$. When this probability is high, the probability of bot detection will be greater.

The USS will respond to the bot with one of three types of message: a 301 Moved Permanently message with an obfuscated IP address, a 301 Moved Permanently message with the URL of a benign server, or a 404 Not Found message. When the response is a 301 Moved Permanently message with an obfuscated IP address (3a), the bot decodes the obfuscated IP address to obtain the IP address of a sub C&C server (4). After that, the bot attempts to connect to the C&C server (5) and, if successful, the C&C phase is started (6). If the bot cannot connect to the C&C server because the server is turned off or blocked, it has to use different aliases to find other sub C&C servers.

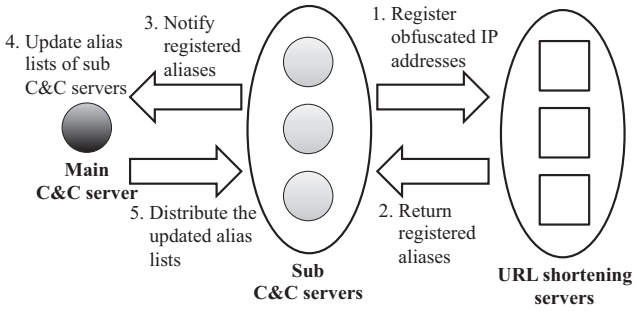
When the response is a 301 Moved Permanently message with the URL of a benign server (3b), another site already occupies the selected custom alias. In this case, the bot also has to use different aliases to find another sub C&C server.

When the response is a 404 Not Found message (3c), a bot attempted to access an alias that had not yet been registered by the botmaster. This error can occur when time synchronization between the C&C servers and the bots is broken, or when the USS blacklists the C&C servers. In the former case, the bot needs to re-synchronize its time with an external server designated by the botmaster.

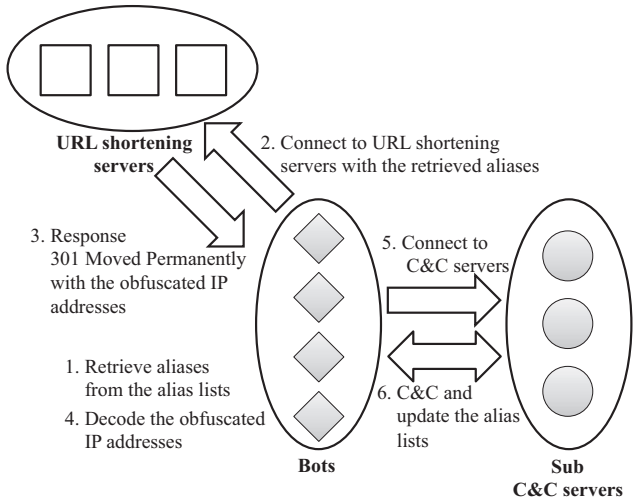
We do not specify any protocol for C&C because it is beyond the scope of this paper. The botmaster may adopt existing protocols such as IRC or HTTP, or he may use his own protocol to control his botnets.

3.2. Alias Flux with Randomized USSes

Some USSes such as `is.gd` and `ow.ly` do not support custom aliases. Therefore, we cannot use the alias flux method with customizable USSes for them. To overcome this limitation, we modify the alias flux method for use with randomized USSes. Instead of a custom alias generation algorithm,



(a) Registration of sub C&C servers to USSes



(b) Discovery of sub C&C servers via USSes

Figure 3: Alias flux with randomized USSes

we work with a list of registered aliases to deliver IP addresses of sub C&C servers to bots. Similar to alias flux with customizable USSes, the alias list includes α aliases, and these will be changed for each period. We also need to maintain u different lists for the u designated USSes. Finally, we assume that each malware binary of our botnet embeds an initial alias list to initiate C&C processes or contacts some hardcoded domains to receive the initial list, as Torpig does [15].

Registration Phase: The botmaster first registers the obfuscated IP addresses of the sub C&C servers to u randomized USSes (1) (Figure 3a).

Next, the USSes return random aliases of the registered obfuscated IP addresses to the botmaster (2). Unlike alias flux with customizable USSes,

no registration error can occur, because the USSes uniquely generate the random aliases. Thus, the botmaster can register α sub C&C servers to the u USSes. When the number of C&C servers is smaller than α, he will redundantly register some sub C&C servers to the USSes to avoid 404 Not Found errors. The main C&C server will receive the newly registered aliases of the sub C&C servers (3).

When the number of registered aliases exceeds α, the botmaster will update the alias lists (4) and distribute the updated lists to his sub C&C servers (5). To register sub C&C servers for the next n periods, the botmaster has to slightly change the URLs of the sub C&C severs to obtain $n\alpha$ URLs and then register them to USSes. In this case, the number of aliases that each C&C server and each bot need to maintain is $nu\alpha$.

Discovery Phase: To discover sub C&C servers via randomized aliases (Figure 3b), each bot randomly selects one of the u USSes and then retrieves a random alias from the alias list for the selected USS (1). After that, the bot will connect to the USS via HTTPS using the retrieved alias (2).

The USSes will respond to the bot with a 301 Moved Permanently message with an obfuscated IP address (3). Unlike the alias flux with customizable USSes, no 301 Moved Permanently message with the URL of a benign server can occur because no alias collision exists, and no 404 Not Found messages can occur because the alias lists only include registered aliases.

After receiving the response, the bot decodes the obfuscated IP address to obtain the IP address of a sub C&C server (4), and it then attempts to connect to the C&C server (5). If the connection attempt succeeds, the botmaster starts a C&C phase and may update the alias lists (6). If the connection attempt fails, the bot has to use different aliases to find another sub C&C server.

3.3. Indirect Botnet Control

By modifying the alias flux with customizable USSes method, the botmaster can distribute commands to his botnets to control them indirectly without sub C&C servers. The basic idea involves registering obfuscated commands to USSes instead of obfuscated IP addresses. We can use the same technique that we used to obfuscate IP addresses to obfuscate commands. The process is as follows (Figure 4).

First, the botmaster executes a custom alias generation algorithm to obtain α custom aliases (1), and then he registers obfuscated commands with the custom aliases to the u designated USSes (2). When obfuscating commands, the botmaster needs to attach timestamps to the commands to verify their time order of commands.

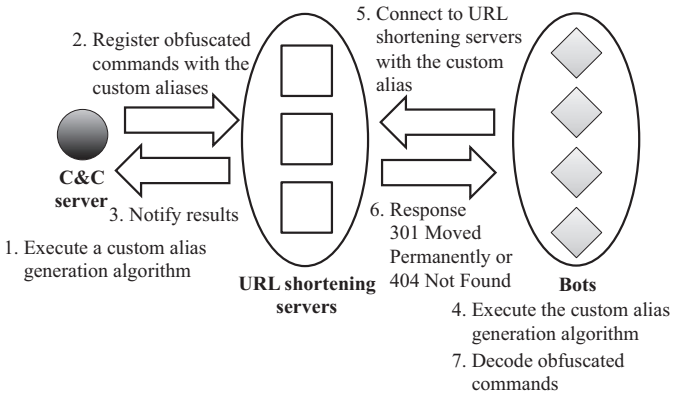


Figure 4: Indirect botnet control via customizable USSes

Next, the botmaster will receive notifications from the USSes (3). When there are some errors, the botmaster needs to re-register the refused commands with different aliases.

Later, each bot will execute a custom alias generation algorithm with the current date information to obtain α custom aliases (4). Each bot then randomly selects one of the u USSes and connects to the selected USS with all of the α custom aliases via HTTPS (5). The USS responds with redirection or error messages to the bot (6). At last, each bot will attempt to decode obfuscated commands from the responses and then execute them (7).

We may also use the alias flux with randomized USSes for method indirect botnet control. This method, however, requires out of band (OOB) channels to distribute and update alias lists. Since the advantage of indirect botnet control is that it does not need sub C&C servers, OOB channels diminish the benefit. Thus, we omit indirect botnet control with randomized USSes.

4. Analysis

4.1. Required Aliases and Discovery Trials

We analyze the number of aliases required for each bot to successfully find sub C&C servers for each period. The number of trials required for each bot to discover sub C&C servers increases when it selects already occupied aliases (alias collisions occur) or aliases of dead (turned off or blocked) sub C&C servers. The increased discovery trials will not only delay each bot's discovery of sub C&C servers but will also increase the possibility of botnet detection. Moreover, each bot requires that at least one alias be valid to

assure the reliable discovery of sub C&C servers. Therefore, a botmaster has to minimize the number of discovery trials while keeping a reasonable reliability.

First, we analyze the reliability of the discovery of sub C&C servers R_d to obtain the upper bound of the required aliases α, which is the same as the number of required sub C&C servers. R_d is the probability that at least one alias is valid. Thus, if we assume that all aliases are independent and have the same probability of invalidness p_{inv}, then

$$R_d = 1 - p_{\text{inv}}^{\alpha}, \tag{1}$$

where

$$p_{\text{inv}} = p_c + (1 - p_c)p_s, \tag{2}$$

where p_c is the probability of alias collision and p_s is the probability of the death of any given sub C&C server. p_c is related to the number of URLs already registered with each USS and p_s is related to the reliability of each sub C&C server. In alias flux with randomized USSes, p_c is always 0; thus, $p_{\text{inv}} = p_s$.

By rearranging Equation (1), we can represent α as a function of p_{inv} and R_d:

$$\alpha = \log_{p_{\text{inv}}}(1 - R_d). \tag{3}$$

For instance, when the desired R_d is smaller than 0.99 and the given p_{inv} is smaller than 0.8, α will be smaller than 20 (Figure 5a). In this case, the botmaster needs to prepare about 20 sub C&C servers.

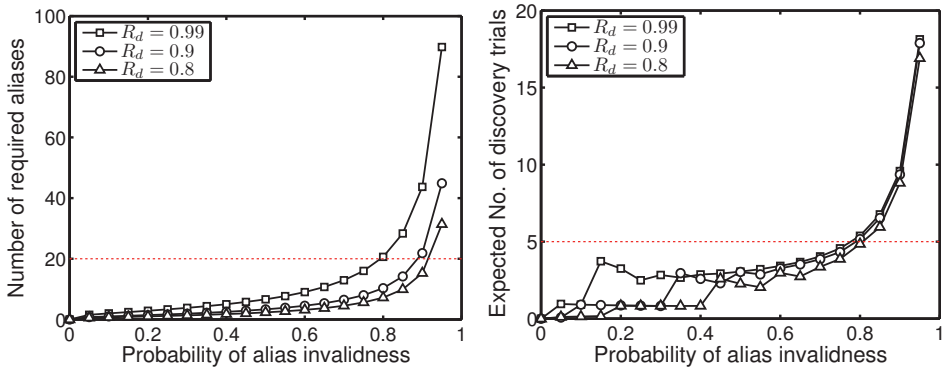
We also analyze the expected number of discovery trials. To discover an active sub C&C server, each bot will randomly select one alias from among α aliases without replacement until it discovers a valid alias. To obtain the expected number of discovery trials, we need to consider two cases: 1) at least one valid alias exists and 2) no valid alias exists.

First, we consider the case where at least one valid alias exists. According to Equation (1), the probability that at lease one valid alias exists is $1 - p_{\text{inv}}^{\alpha}$. In this case, if we let T be the number of trials required to discover a valid alias, then

$$P\{T = i\} = \left(\prod_{j=0}^{i-2} \frac{p_{\text{inv}}\alpha - j}{\alpha - j} \right) \cdot \frac{(1 - p_{\text{inv}})\alpha}{\alpha - (i-1)},$$

where $i = 1, 2, \ldots, \alpha$ (details are given in Appendix A).

Next, we consider the case where no valid alias exists. According to Equation (1), the probability that no valid alias exists is p_{inv}^{α}. In this case,



(a) (b)

Figure 5: The upper bound of the number of required aliases (a) and the expected number of discovery trials (b) according to the probability of alias invalidness (p_{inv}) and the desired reliability of discovery (R_d)

the number of discovery trials is equal to α. Therefore, the expected number of discovery trials is

$$E[T] = (1 - p_{\text{inv}}^{\alpha}) \cdot \sum_{i=1}^{\alpha} i \left(\prod_{j=0}^{i-2} \frac{p_{\text{inv}}\alpha - j}{\alpha - j} \right) \cdot \frac{(1 - p_{\text{inv}})\alpha}{\alpha - (i-1)} + p_{\text{inv}}^{\alpha}\alpha. \quad (4)$$

By applying Equation (3) to Equation (4), we can represent the expected number of discovery trials as a function of p_{inv} and R_d:

$$E[T] = R_d \cdot \sum_{i=1}^{\log_{p_{\text{inv}}}(1-R_d)} i \left(\prod_{j=0}^{i-2} \frac{p_{\text{inv}} \cdot \log_{p_{\text{inv}}}(1 - R_d) - j}{\log_{p_{\text{inv}}}(1 - R_d) - j} \right) \cdot \frac{(1 - p_{\text{inv}}) \cdot \log_{p_{\text{inv}}}(1 - R_d)}{\log_{p_{\text{inv}}}(1 - R_d) - (i-1)} + (1 - R_d) \cdot \log_{p_{\text{inv}}}(1 - R_d). \quad (5)$$

For instance, when the desired R_d is smaller than 0.99 and the given p_{inv} is smaller than 0.8, the expected number of discovery trials is smaller than five (Figure 5b). In other words, each bot needs to contact USSes an average of five times to discover a sub C&C server.

To reduce the number of discovery trials, the botmaster should reduce the desired R_d or p_{inv}. Reducing R_d is usually not desired because the small R_d means that the botmaster cannot control his botnets. To reduce p_{inv}, the botmaster needs to reduce p_c or p_s. To reduce p_c, the botmaster has to

increase the number of designated USSes u to reduce p_c as p_c^u. An increased u can also reduce the probability that two bots connect to the same shortened URL $(u\alpha)^{-1}$. On the other hand, the number of registration of sub C&C servers to USSes also increases. The botmaster can also use alias flux with randomized USSes because it has no alias collision, i.e., $p_c = 0$. This method, however, requires additional communications to update alias lists; network administrators may detect these communications. To estimate the value of p_c for `bit.ly`, we generate about 17 million random six-character aliases and check whether each random alias exists in `bit.ly`. Of the generated aliases, about 1.77 million aliases exist. Therefore, p_c for `bit.ly` is about 0.1 (details are given in Appendix B.1). To reduce this probability, a botmaster needs to use aliases longer than six characters. To reduce p_s, the botmaster has to acquire more reliable sub C&C servers. He can also reduce p_s by encoding the IP addresses of k sub C&C servers to a single URL as p_s^k. Because all approaches to reduce p_{inv} require additional costs, the botmaster needs to consider the tradeoffs.

The deaths of USSes and occurrence of 404 Not Found errors can also increase the number of discovery trials. We assume that these errors rarely occur because the botmaster will choose reliable USSes and time sources when he designs his botnets. WatchMouse [4] said that about 10 USSes have availabilities higher than 99.8%. Thus, the botmaster will select some of them to enhance the reliability of his botnets. In addition, even when some USSes or time sources are dead, each bot can independently find other USSes or time sources before discovering C&C servers. Accordingly, we omit these factors when computing the number of discovery trials.

4.2. Example Scenario

We describe how a botmaster controls a botnet by using USSes. We assume the botnet consists of $50,000$ bots, 10 sub C&C servers, and five USSes. For secrecy, the botmaster changes the IP addresses of the sub C&C servers daily. Therefore, each bot needs to contact USSes to obtain new IP addresses at least once per day. When the USSes notice that some of their pages are visited remarkably often, they may become suspicious and thus carefully analyze those pages. To avoid this, the botmaster will restrict the number of daily visits to each alias to fewer than 100, because most shortened URLs are visited fewer than 100 times in a day (details are given in Appendix B.1). If we assume that $p_{\text{inv}} = 0.5$, about three discovery trials will be expected for each bot. Therefore, each USS will receive about $30,000$ visits from about $10,000$ bots in a day. To assure fewer than 100 visits to each alias, the botmaster will register 10 sets of 30 aliases to the five USSes

for each day; each set corresponds to one of the sub C&C servers. Later, each bot randomly selects one of the 10 sets and then selects one of the 30 aliases to connect to USSes with. Alternatively, the botmaster can encode more than three sub C&C servers' IP addresses as a single URL to reduce the total number of visits and registrations. In this case, the botmaster will register 100 aliases to the five USSes and then each bot will retrieve one of them in a day. Since the lengths of most URLs registered in USSes are shorter than 200 (refer Appendix B.1), the botmaster can put several IP addresses (or commands) into a URL until its length does not exceed 200. The famous USS `bit.ly` receives more than 1.08 million visits and 0.28 million registration requests in a *nine minute period* [29]; it is difficult to distinguish tens of thousands of malicious visits and hundreds of malicious registration requests from the routine heavy traffic.

4.3. IP List vs. Alias List

If we can distribute alias lists to bots for alias flux with randomized USSes, then we can also distribute IP lists of sub C&C servers to bots. Hence, someone may argue that an IP list is better than an alias list because it is simple and does not even require USSes. However, when investigators successfully decrypt an IP list, they can block all communications to the listed IP addresses. In contrast, even when investigators successfully decrypt an alias list, they have to find the corresponding obfuscated URLs from some of the USSes and decrypt the URLs to block the corresponding IP addresses. Therefore, an alias list is more robust to investigation than an IP list.

5. Countermeasures

5.1. Detection Points and Coverage

We consider three detection points of alias flux botnets: hosts, network monitors, and URL shortening servers (Figure 6). At hosts, investigators monitor system statuses to detect malware and attempt to analyze it. However, malware can have sophisticated techniques for cloaking itself and increasing the difficulties of analysis. A network monitor can monitor and analyze traffic between bots and USSes, and between bots and sub C&C servers. Since both type of traffic can be encrypted, it is difficult to investigate messages exchanged between them. The size of the monitored network is also important in detecting malicious activities. Unlike a network monitor, a USS can monitor and analyze the encrypted traffic from bots because it is a counterpart of the bots. Moreover, it has the mapping information between the malicious aliases and obfuscated URLs. On the other hand,

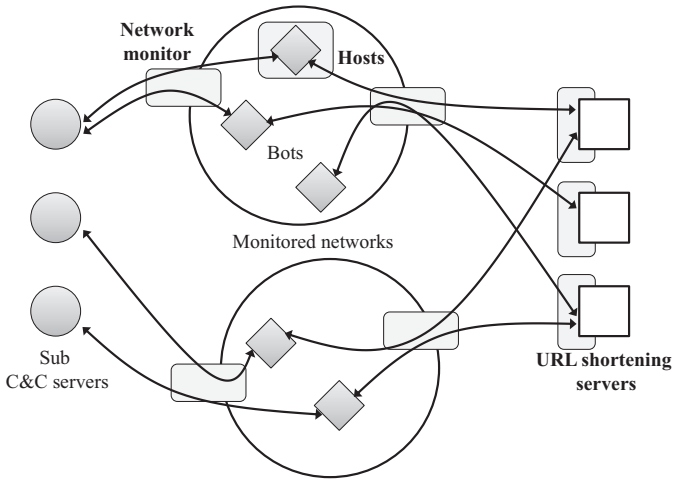


Figure 6: Detection points and coverage

since it cannot monitor traffic between the bots and the sub C&C servers, it does not know whether the real malicious activities have occurred. Moreover, since bots can randomly select their USSes and aliases, each USS has restricted monitoring coverage.

5.2. Botnet Detection in Hosts

When investigators can detect malware for botnets on hosts and successfully analyze it, they can know alias generation algorithms or alias lists. The investigators then distribute these malicious aliases to network administrators and USS providers for blacklisting. They also can discover sub C&C servers by monitoring malware's communications. However, reverse engineering is a difficult task, because there are many evasion techniques. For instance, malware authors can use virtualization obfuscators for protecting their malicious code [30]. Moreover, they can write malicious code that does not perform malicious behaviors on a virtualized environment for analysis [31]. They also can frequently update their malicious code to make the malware analysis meaningless.

5.3. Botnet Detection in Monitored Networks

Because alias flux botnets do not rely on DNS when finding C&C channels and the finding process can be secured by using HTTPS, botnet detection schemes based on DNS traffic [32, 33] cannot detect them. Likewise, DNS-based fast flux detection schemes [9, 10, 11, 12, 13] and domain flux

detection schemes [16, 17, 18, 19] cannot detect the alias flux methods. Therefore, current botnet detection schemes may not detect the C&C channel establishment of alias flux botnets.

After the C&C channels are established and the bots start to perform malicious activities, they can be detected by botnet detection schemes that rely on the correlation of malicious network activities [34, 35, 36]. These schemes, however, require that at least two bots managed by the same botmaster exist in the monitored network to check their correlation.

5.4. Botnet Detection in USSes

Verifying and Limiting URL Registrations: Currently, some USSes require user authentication, an API key, or CAPTCHA solving before shortening URLs to prevent bots' abuse of them. Some USSes also use rate limiting to prevent massive registrations of spam and phishing sites to them. These techniques can also disturb the registrations of sub C&C servers to USSes. However, they are insufficient because several techniques to evade CAPTCHAs exist [37, 38, 39] and a botmaster can create and use several bot accounts when registering his sub C&C servers to USSes. Therefore, USSes should also consider other features including rate, pattern, and correlation of URL shortening requests, as well as user reputation to detect bot accounts that have some group behavior.

Analyzing Alias Retrievals: Since bots will attempt to retrieve a shared set of aliases with a similar rate and pattern, USSes may find malicious aliases by analyzing alias retrievals. For instance, Bilge *et al.* [19] verified that malicious domains have time-based features including short life, daily similarity, repeating patterns, and access ratios. USSes can also use the time-based features to classify malicious aliases. However, because each bot's alias retrievals can be randomly distributed to several USSes, classification errors will become higher.

Blacklisting Malicious URLs and IP addresses: To prevent alias flux botnets, USSes need to determine whether the registered URLs are malicious. Suspicious URL detection schemes [24, 40, 41, 42] can be applied to detect malicious URLs such as obfuscated IP addresses and commands registered by a botmaster. The botmaster, however, may bypass this detection by using nested URL shortening [26]. This process is as follows: the botmaster finds a USS that does not check malicious URLs, registers obfuscated IP addresses or commands to the USS to obtain shortened URLs, and then registers the shortened URLs to other USSes that check malicious URLs. To prevent this evasion, USSes need to follow all redirections when checking registered URLs.

After detecting malicious URLs, the USS will blacklist the malicious URLs and the IP addresses that are used to register them. To avoid blacklisting, the botmaster may use anonymity networks such as Tor to hide his IP addresses when he registers the malicious URLs. Therefore, the IP addresses used may not belong to the botnets. Even when the IP addresses belong to the botnets, static IP address blacklisting is not enough because alias flux botnets continuously change their IP addresses. Therefore, proactive countermeasures such as reputation-based IP address blacklisting [43, 44] are required to prevent botnets from performing malicious activities.

Detecting Malicious Custom Aliases: The purpose of custom aliases for shortened URLs is to create short links that are human readable and memorable. When a custom alias is not human readable, we can assume that the alias is not meant for humans. Similar to domain name generation algorithms, the results of alias generation algorithms may not be well-formed and pronounceable words. Therefore, a malicious domain name detection scheme [18] can be applied to detect malicious custom aliases. This detection method can be applied only when USSes distinguish custom aliases from random aliases.

Monitoring Not Found Errors: The occurrence of 404 Not Found errors is one of the characteristics of alias flux botnets. Because of the randomness of custom alias generation, each bot will attempt to access nonexistent shortened URLs more frequently than normal hosts. Therefore, by checking the number of 404 Not Found errors caused by hosts, USSes may distinguish bots from normal hosts. However, botnets can bypass this detection method if they share a reliable time source or adopt rate limiting.

5.5. Collaborations to Detect Botnets

Collaboration between investigators and USS providers make botnet detection easier. For instance, when investigators give information about detected bots to the USS providers, the USS providers can provide information related to the bots such as all aliases that are accessed by the bots and the corresponding URLs to the investigators. This information can be applied to verify correlations between custom aliases and to deobfuscate obfuscated IP addresses or commands. The USS providers can also give information related to the aliases such as the IP addresses that are used to register and retrieve the aliases. This information can be employed to find undiscovered bots and sub C&C servers. However, because there is no official record of how many USSes currently exist or their security properties, collaboration is difficult.

5.6. Customizable vs. Randomized USSes

We compare alias flux with customizable USSes and randomized USSes to determine different features of them for use in detection. In a monitored network, alias flux with randomized USSes is more easily detected than alias flux with customizable USSes because it needs additional communications with sub C&C servers to update alias lists. Alias flux with customizable USSes also needs some additional communications because of alias collisions. However, the collision probability is small and the counterparts of these communications are benign servers not sub C&C servers. In a URL shortening server, in contrast, alias flux with customizable USSes is more easily to be detected than alias flux with randomized USSes because its malicious custom aliases may be distinguished from legitimate custom aliases, and it may attempt to access unregistered aliases owing to the randomness of alias generation. Since alias flux with randomized USSes uses only legitimate and registered aliases, it does not share these problems. Because many networks are already monitored but many USSes are not well prepared for security attacks, alias flux with customizable USSes seems to present a larger threat than alias flux with randomized USSes. Therefore, more considerations are required for alias flux with customizable USSes.

5.7. Do Not Allow HTTPS?

It can be argued that USSes do not have to allow HTTPS. When HTTPS is not employed, network administrators can monitor and block the aliases that each bot has accessed. Moreover, when the network administrators succeed in analyzing the malware of each bot, they can block the next aliases that each bot will use later. Because USSes are only redirection services, it seems that communication with them does not need to be secured. However, we need to consider two cases where HTTPS is required. First, when a user registers a URL to a USS by using APIs with his key, this communication should be protected to avoid revealing of the API key to attackers. If the API key is revealed, then attackers can exploit it to distribute spam messages or establish botnets. Therefore, we have to use HTTPS to protect the URL registration process. Second, when the original URL begins with `https://`, its shortened URL also needs to be accessed via HTTPS to protect the data hierarchy and command structure of the original URL from attackers. For instance, when a user directly accesses `https://company.com/2012/01/bad-profit.html`, attackers cannot eavesdrop on the page name and hierarchy that the user has accessed, i.e., `/2012/01/bad-profit.html`. On the other hand, when the user accesses the page via a shortened URL without

HTTPS, the page name and its hierarchy are revealed; thus, information leakage can occur. Therefore, we have to use HTTPS to protect URLs beginning with `https://` from attackers.

5.8. Example: Analysis of Alias Retrievals

We show how our countermeasures could be realized with an example. Among the countermeasures, we select the analysis of alias retrievals at USSes because 1) host- and network-level detections are almost the same as the previous botnet detection methods and 2) other detection methods at USSes are relatively simple and similar with the previous methods.

In this example, we assume that bots are uniformly located around the world and they try to access USSes at their boot time to obtain IP addresses of C&C servers. We generated hourly click patterns of bots based on this assumption ($10,000$ bots, 300 aliases, and $p_{\text{inv}} = 0.5$); an average number of clicks on malicious aliases follows a uniform distribution (Figure 7). However, we observed that most click probability distributions of benign aliases follow an exponential distribution (an average of 50 benign `bit.ly` URLs, Figure 7). This exponential distribution implies that most shortened URLs receive high attention at first but it gets lower as time goes on. Using this characteristic, we could distinguish malicious aliases for alias flux botnets from benign aliases. To evade this detection method, a botmaster needs to 1) control the distribution of bots around the world, e.g., the botmaster has to use bots in specific countries, or 2) control bots' access time to USSes, e.g., 40% of bots accesses USSes around 13:00 GMT, 20% of bots accesses USSes around 14:00 GMT, and so on. However, the first approach will reduce the number of controllable bots. The second approach is also challenging because whether bots are running at the scheduled time is not guaranteed. Both disadvantages are harmful to the botmaster.

6. Related Work

6.1. New Botnet Design

Many researchers have designed botnets to prepare for new security attacks before they are evolved. Many of them have proposed decentralized botnets. For instance, Hund *et al.*'s Rambot [46] uses a customized P2P module that has a credit-point system to check the reputation of each bot to deal with investigators' fake bots. Starnberger *et al.*'s Overbot [47] is based on the Kademlia distributed hash table to enhance its robustness and stealthiness. Since queries generated by bots of Overbot are similar to legitimate search requests, detection and filtering of Overbot is difficult. Nappa

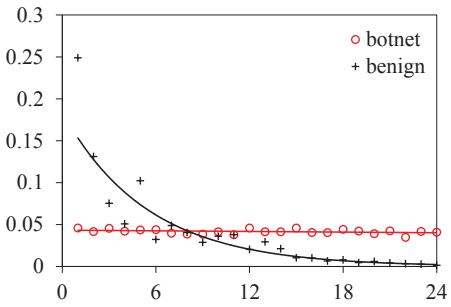


Figure 7: Hourly click probability distributions of aliases

et al.'s botnet [48] is based on Skype; because Skype traffic is difficult to filter, their botnet is also robust and stealthy. Nagaraja *et al.*'s Stegobot [49] uses a covert communication channel on a social network overlay—sharing of steganographic images between infected friends.

Most of the current botnets are still centralized because P2P-based botnets are difficult to manage. Therefore, several researchers have considered new C&C channels for centralized botnets. For instance, Singh *et al.* [50] propose an E-mail based botnet. In their botnet, each bot creates its own E-mail address and then notifies a botmaster of it. The botmaster can then control bots by sending E-mails that securely embed commands with a combination of encryption and steganography. Kartaltepe *et al.* [51] consider a Twitter-based botnet that was discovered by Nazario [52]. In this botnet, a botmaster posts tweets that embed encoded commands to his account. Bots retrieve the encoded commands by using RSS feeds of the account. Xiang *et al.* [53] also propose a mobile botnet based on microblogs and RSS feeds. Since the discussed botnets abuse popular Internet services, it is difficult to distinguish whether bots generate malicious traffic. Moreover, because of the popularity of those services, service providers require huge computation resources when investigating malicious messages and accounts.

6.2. Domain Flux Detection

Since an alias flux botnet is a variant of a domain flux botnet, domain flux detection schemes are also relevant to alias flux botnets. In domain flux, the domain names of botnets are frequently changed; thus, we need to predict new malicious domains before they are used. Felegyhazi *et al.* [17] propose a proactive domain blacklisting scheme that guesses the next malicious domain names by using the features of previous malicious domain names. Yadav *et al.* [18] also propose a scheme that can detect algorithmi-

cally generated malicious domain names. Since domain flux botnets generate domain names by using algorithms and they need to avoid domain name collisions, most of the generated domain names are not human readable and memorable; thus, they can be distinguished from domain names generated by humans. Blacklisting always produces false positives, however. To mitigate this problem, Antonakakis *et al.* [16] propose a dynamic reputation system that assigns reputation scores to new domain names according to their maliciousness. Since the preceding schemes rely on network-based features and need historical information, they cannot detect malicious domain names that are assigned to a new address space each time and never used again. To solve this problem, Bilge *et al.* [19] propose a malicious domain detection scheme that considers time-based features and does not rely on historic information on IP addresses or domains.

6.3. Suspicious URL Detection

Since an alias flux botnet uses URLs to distribute the obfuscated IP addresses of sub C&C servers or commands, suspicious URL detection schemes can be applied to detect these URLs. Several schemes [24, 40, 41, 42] have been proposed. They consider the lexical features of URLs including the length, the number of dots, and each token, and the host-based features, including owners and locations, to detect suspicious URLs.

7. Conclusion

In this paper, we have explored new botnets that use URL shortening services (USSes) to hide their existence. We explored alias flux methods that frequently change shortened URLs of C&C servers to hide the C&C channels, which is similar to the domain flux method. The strength of alias flux is that it can bypass all DNS-based fast flux and domain flux detection schemes. When designing alias flux methods, we considered both customizable USSes and randomized USSes. In addition, an indirect botnet control method based on alias flux with customizable USSes was explored. Since alias flux botnets can become real threats, both network administrators and USS providers should prepare for new threats owing to the proliferation of USSes.

[1] D. Antoniades, I. Polakis, G. Kontaxis, E. Athanasopoulos, S. Ioannidis, E. P. Markatos, T. Karagiannis, we.b: The web of short URLs, in: Int. World Wide Web Conf. (WWW), 2011.

[2] E. Schonfeld, When it comes to URL shorteners, bit.ly is now the biggest, http://techcrunch.com/2009/05/07/when-it-comes-to-url-shorteners-bitly-is-now-the- biggest (2009).

[3] LongShore, The most comprehensive and ever-growing list of URL shortening services, http://long-shore.com/services.

[4] WatchMouse, URL shorteners make the web substaintially slower; Facebooks' fb.me is slowest, http://blog.watchmouse.com/2010/03/url-shorteners-make-the-web-substantially-slower-facebooks-fb-me-is-slowest (March 2010).

[5] P. Mockapetris, Domain names - concepts and facilities, http://www.ietf.org/rfc/rfc1034.txt (1987).

[6] P. Mockapetris, Domain names - implementation and specification, http://www.ietf.org/rfc/rfc1035.txt (1987).

[7] The Honeynet Project & Research Allicance, Know your enemy: Fast-flux service networks, http://old.honeynet.org/papers/ff/fast-flux.html (2007).

[8] G. Ollmann, Botnet communication topologies, http://www.dis.arkansas.gov/security/Documents/WP_Botnet_Communications_Primer_(2009-06-04).pdf (2009).

[9] T. Holz, C. Gorecki, K. Rieck, F. C. Freiling, Measuring and detecting fast-flux service networks, in: Network and Distributed System Security Symp. (NDSS), 2008.

[10] J. Nazario, T. Holz, As the net churns: Fast-flux bonet observations, in: Int. Conf. Malicious and Unwanted Software (Malware), 2008.

[11] E. Passerini, R. Paleari, L. Martignoni, D. Bruschi, FluXOR: Detecting and monitoring fast-flux service networks, in: Conf. Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), 2008.

[12] R. Perdisci, I. Corona, D. Dagon, W. Lee, Detecting malicious flux service networks through passive analysis of recursive DNS traces, in: Annual Computer Security Applications Conf. (ACSAC), 2009.

[13] A. Caglayan, M. Toothaker, D. Drapeau, D. Burke, G. Eaton, Behavioral patterns of fast flux service networks, in: Hawaii Int. Conf. System Sciences (HICSS), 2010.

[14] P. Porras, H. Saïdi, V. Yegneswaran, A foray into Conficker's logic and rendezvous points, in: USENIX Workshop Large-Scale Exploits and Emergent Threats (LEET), 2009.

[15] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, G. Vigna, Your botnet is my botnet: Analysis of a botnet takeover, in: ACM Conf. Computer and Communications Security (CCS), 2009.

[16] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, N. Feamster, Building a dynamic reputation system for DNS, in: USENIX Security Symp., 2010.

[17] M. Felegyhazi, C. Kreibich, V. Paxson, On the potential of proactive domain blacklisting, in: USENIX Workshop Large-Scale Exploits and Emergent Threats (LEET), 2010.

[18] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, S. Ranjan, Detecting algorithmically generated malicious domain names, in: Internet Measurement Conf. (IMC), 2010.

[19] L. Bilge, E. Kirda, C. Kruegel, M. Balduzzi, EXPOSURE: Finding malicious domains using passive DNS analysis, in: Network and Distributed System Security Symp. (NDSS), 2011.

[20] J. B. Grizzard, V. Sharma, C. Nunnery, B. Kang, D. Dagon, Peer-to-peer botnets: Overview and case study, in: USENIX Workshop Hot Topics in Understanding Botnets (HotBots), 2007.

[21] T. Holz, M. Steiner, F. Dahl, E. Biersack, F. Freiling, Measurements and mitigation of peer-to-peer-based botnets: A case study on Storm worm, in: USENIX Workshop Large-Scale Exploits and Emergent Threats (LEET), 2008.

[22] G. Sinclair, C. Nunnery, B. B. Kang, The Waledac protocol: The how and why, in: Int. Conf. Malicious and Unwanted Software (Malware), 2009.

[23] C. Nunnery, G. Sinclair, B. B. Kang, Tumbling down the rabbit hole: Exploring the idiosyncrasies of botmaster systems in a multi-tier botnet infrastructure, in: USENIX Workshop Large-Scale Exploits and Emergent Threats (LEET), 2010.

[24] D. K. McGrath, M. Gupta, Behind phishing: An examination of phisher modi operandi, in: USENIX Workshop Large-Scale Exploits and Emergent Threats (LEET), 2008.

[25] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, B. Y. Zhao, Detecting and charaterizing social spam campaigns, in: Internet Measurement Conf. (IMC), 2010.

[26] C. Grier, K. Thomas, V. Paxson, M. Zhang, @spam: The underground on 140 characters or less, in: ACM Conf. Computer and Communications Security (CCS), 2010.

[27] Z. Chu, S. Gianvecchio, H. Wang, S. Jajodia, Who is tweeting on Twitter: Human, bot, or cyborg?, in: Annual Computer Security Applications Conf. (ACSAC), 2010.

[28] G. Stringhini, C. Kruegel, G. Vigna, Detecting spammers on social networks, in: Annual Computer Security Applications Conf. (ACSAC), 2010.

[29] H. Mason, A data-driven look at the realtime web ecosystem, in: Web 2.0 Expo San Francisco, 2010.

[30] M. Sharif, A. Lanzi, J. Giffin, W. Lee, Automatic reverse engineering of malware emulators, in: IEEE Symp. on Security and Privacy (Oakland), 2009.

[31] X. Chen, J. Andersen, Z. M. Mao, M. Bailey, J. Nazario, Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware, in: IEEE Int. Conf. Dependable Systems and Networks (DSN), 2008.

[32] H. Choi, H. Lee, Identifying botnets by capturing group activities in DNS traffic, Computer Networks 56 (2012) 20–33.

[33] J. Morales, A. Andre Al-Bataineh, S. Xu, R. Sandhu, Analyzing DNS activities of bot processes, in: Int. Conf. Malicious and Unwanted Software (Malware), 2009.

[34] G. Gu, J. Zhang, W. Lee, BotSniffer: Detecting botnet command and control channels in network traffic, in: Network and Distributed System Security Symp. (NDSS), 2008.

[35] G. Gu, R. Perdisci, J. Zhang, W. Lee, BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection, in: USENIX Security Symp., 2008.

[36] T.-F. Yen, M. K. Reiter, Traffic aggregation for malware detection, in: Conf. Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), 2008.

[37] P. Golle, Machine learning attacks against the Asirra CAPTCHA, in: ACM Conf. Computer and Communications Security (CCS), 2008.

[38] J. Yan, A. S. E. Ahmad, A low-cost attack on a Microsoft CAPTCHA, in: ACM Conf. Computer and Communications Security (CCS), 2008.

[39] B. B. Zhu, J. Yan, Q. Li, C. Yang, J. Liu, N. Xu, M. Yi, K. Cai, Attacks and design of image recognition CAPTCHAs, in: ACM Conf. Computer and Communications Security (CCS), 2010.

[40] J. Ma, L. K. Saul, S. Savage, G. M. Voelker, Beyond blacklists: Learning to detect malicious web sites from suspicious URLs, in: ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD), 2009.

[41] J. Ma, L. K. Saul, S. Savage, G. M. Voelker, Identifying suspicious URLs: An application of large-scale online learning, in: Int. Conf. Machine Learning (ICML), 2009.

[42] D. Canali, M. Cova, G. Vigna, C. Kruegel, Prophiler: A fast filter for the large-scale detection of malicious web pages, in: Int. World Wide Web Conf. (WWW), 2011.

[43] S. Sinha, M. Bailey, F. Jahanian, Shades of grey: On the effectiveness of reputation-based "blacklists", in: Int. Conf. Malicious and Unwanted Software (Malware), 2008.

[44] J. Zhang, P. Porras, J. Ullrich, Highly predictive blacklisting, in: USENIX Security Symp., 2008.

[45] SelenumHQ, Selenium web application testing system, `http://seleniumhq.org`.

[46] R. Hund, M. Hamann, T. Holz, Towards next-generation botnets, in: European Conf. Computer Network Defence (EC2ND), 2008.

[47] G. Starnberger, C. Kruegel, E. Kirda, Overbot - a botnet protocol based on Kademlia, in: Int. Conf. Security and Privacy in Communication Networks (Securecomm), 2008.

[48] A. Nappa, A. Fattori, M. Balduzzi, A. M. Dell, L. Cavallaro, Take a deep breath: a stealthy, resilent and cost-effective botnet using Skype, in: Conf. Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), 2010.

[49] S. Nagaraja, A. Houmansadr, P. Piyawongwisal, V. Singh, P. Agarwal, N. Borisov, Stegobot: a covert social network bot, in: Information Hiding Conf. (IH), 2011.

[50] K. Singh, A. Srivastava, J. Giffin, W. Lee, Evaluating Email's feasibility for botnet command and control, in: IEEE Int. Conf. Dependable Systems and Networks (DSN), 2008.

[51] E. J. Kartaltepe, J. A. Morales, S. Xu, R. Sandhu, Social network-based botnet command-and-control: Emerging threats and countermeasures, in: Int. Conf. Applied Cryptography and Network Security (ACNS), 2010.

[52] J. Nazario, Twitter-based botnet command channel, `http://asert.arbornetworks.com/2009/08/twitter-based-botnet-command-channel` (2009).

[53] C. Xiang, F. Binxing, Y. Lihua, L. Xiaoyi, Z. Tianning, Andbot: Towards advanced mobile botnets, in: USENIX Workshop Large-Scale Exploits and Emergent Threats (LEET), 2011.

Appendix A. Additional Explanations of Section 4.1

Let assume that we have α aliases and each alias has the same probability of invalidness p_{inv}. In other words, we have $V = (1 - p_{\text{inv}})\alpha$ valid aliases and $W = p_{\text{inv}}\alpha$ invalid aliases. A bot will select one of the aliases until it detects a valid alias. When the bot selects an invalid alias, the bot will remove it from the alias list (i.e., without replacement). The probabilities that a bot

selects a valid alias at its first to fourth discovery trials are

$$
\begin{aligned}
P\{T=1\} &= \frac{V}{\alpha}, \\
P\{T=2\} &= \frac{W}{\alpha} \cdot \frac{V}{\alpha-1}, \\
P\{T=3\} &= \frac{W}{\alpha} \cdot \frac{W-1}{\alpha-1} \cdot \frac{V}{\alpha-2}, \\
P\{T=4\} &= \frac{W}{\alpha} \cdot \frac{W-1}{\alpha-1} \cdot \frac{W-2}{\alpha-2} \cdot \frac{V}{\alpha-3}.
\end{aligned}
$$

These equations can be generalized as

$$
P\{T=i\} = \left(\prod_{j=0}^{i-2} \frac{W-j}{\alpha-j} \right) \cdot \frac{V}{\alpha-(i-1)},
$$

where $i = 1, 2, \ldots, \alpha$, which is the same as the equation we explained in Section 4.1. The above equation is valid when $1 \leq V \leq \alpha$, whose probability is $1 - p_{\text{inv}}^{\alpha}$. If $V = 0$, whose probability is p_{inv}^{α}, the number of discovery trials is α. Therefore, the expected number of discovery trials is

$$
E[T] = (1 - p_{\text{inv}}^{\alpha}) \cdot \sum_{i=1}^{\alpha} iP\{T=i\} + p_{\text{inv}}^{\alpha}\alpha,
$$

which is the same as Equation (4).

Appendix B. Investigation of USSes

Many USSes support both a Web interface and APIs for shortening URLs. A botmaster may prefer APIs, owing to the simplicity of automatic registration of his sub C&C servers to USSes. For automatic registration via the Web interface, the botmaster may use Web automation tools, such as Selenium [45] or custom browsers. Some USSes provide URL shortening APIs that only support random aliases; for example, `is.gd` uses the format `http://is.gd/api.php?longurl={URL}`. By sending a GET request of this URL to `is.gd`, a requester can obtain a random alias of `URL`. To prevent malicious and massive URL shortening requests by attackers, some USSes request a user identifier and key to access their APIs, such as `bit.ly` requiring `http://api.bit.ly/v3/shorten?login={ID}&apiKey={Key}&longUrl={URL}`. Thus, the botmaster has to create several fake accounts to maliciously access their APIs. Some USSes support URL shortening APIs that allow customizable

aliases; for example, `tinyurl.com` allows `http://tinyurl.com/create.php?alias={Alias}&url={URL}`. By using these APIs, a requester can associate Alias with URL. Almost all of these APIs limit the number of requests to prevent malicious and massive URL shortening requests. Therefore, the botmaster has to employ several hosts to avoid this rate limiting. Other USSes, such as `bit.ly`, support custom aliases via their Web interfaces; but their shortening APIs do not allow custom aliases. Thus, a botmaster may need to use automation tools or custom browsers to register his sub C&C servers with custom aliases to `bit.ly`.

Many USSes also support APIs to retrieve the original URLs of shortened URLs without redirection. Since their APIs are different, embedding all of their APIs into each bot makes it complex. Accordingly, we assume that each bot will directly access shortened URLs.

When a bot attempts to discover sub C&C servers via a USS, it first sends a GET request to the USS, such as `http://{DomainName}/{Alias}`, where `DomainName` is the domain name of the USS and `Alias` is a generated custom alias or retrieved random alias. Later, when it receives a 301 Moved Permanently message from the USS, it will parse the Location header of the 301 Moved Permanently message to obtain an obfuscated URL and then decode the URL to obtain the IP address of a sub C&C server.

Instead of 301 Moved Permanently, some USSes use other response codes such as 302 Found, 303 See Other, and 307 Temporary Redirect for URL redirection. Since all of these adopt the Location header to deliver the original URL, there is no difference for the bot. Some other USSes, however, use other URL redirection methods based on frame tags, iframe tags, meta refresh tags, or JavaScript. Moreover, some USSes only provide a preview or link to the original page. In these cases, each bot has to parse the corresponding fields of received HTML documents to retrieve the IP address of the sub C&C server.

As of July 21, 2011, we obtained 706 USSes from LongShore's list [3] and `tiny-url.info`. Among them, we found 15 USSes that support HTTPS. We investigated and classified them according to whether authentication is required, custom aliases are available, and which redirection methods are used (Table B.1). For a botmaster, HTTPS is essential to cloak each bot's C&C discovery traffic. He also prefers USSes without authentication because he does not need to maintain fake accounts or pass CAPTCHA tests for authentication. Moreover, the availability of custom aliases will give additional attack routes to him. Among the 15 USSes, a botmaster may prefer the four USSes that do not require authentication and support custom aliases, which are `6ble.com`, `crum.bs`, `gkurl.us`, and `smll.fr`. Moreover,

Table B.1: Classification result of USSes that support HTTPS

Redirection	No Auth. required		Auth. required		
method	Custom/Random	Random	Custom/Random	Random	Total
301	6ble.com gkurl.us	lurl.no o-x.fr tiny.pl tllg.net ux.nu	bit.ly	su.pr	9
302	crum.bs smll.fr	linkee.com	referer.us twi.im		5
preview				j2j.de	1
Total	4	6	3	2	15

when he uses random aliases and can bypass authentication, he can use all 15 of the USSes. For instance, bit.ly only checks the validity of an E-mail address when creating an account. Thus, a botmaster can create an arbitrary number of fake accounts when he owns several E-mail addresses.

Appendix B.1. In-depth Analysis on bit.ly and goo.gl

For an in-depth analysis, we crawled 3,969,907 shortened URLs from bit.ly and goo.gl from January 25 to April 17, 2011 and analyzed them. For crawling, we randomly generated 25,033,952 aliases whose lengths are four to six characters. Then, we accessed the APIs of bit.ly and goo.gl to discover the corresponding long URLs. For bit.ly, we generated five- and six-character aliases because bit.ly currently provides six-character aliases for users and we wanted to inspect the present and past values. By the same logic, we generated four- and five-character aliases for goo.gl because it currently provides five-character aliases for users. From our data sets, we verified that more than half of the five-character aliases of bit.ly and almost all of the four-character aliases of goo.gl were already occupied. Therefore, bit.ly and goo.gl currently provide six- and five-character aliases for users because still fewer than 11% and 5%, respectively, of aliases with those lengths were occupied (Table B.2). In addition, goo.gl announces removed links that violated security or legal rules through its APIs. Thus, we examined the ratio of removed aliases to all registered aliases; goo.gl has removed about 20% of five-character aliases and 30% of six-character aliases.

Table B.2: Summary of collected data from bit.ly and goo.gl

USS (alias length)	Number of shortened URLs			Portion (%)	
	Removed (R)	Exist (E)	Accessed (A)	$\frac{R}{R+E}$	$\frac{R+E}{A}$
`bit.ly` (5)	-	1,035,269	1,970,522	-	52.54
`bit.ly` (6)	-	1,772,944	17,021,947	-	10.42
`goo.gl` (4)	244,679	996,098	1,244,905	19.72	99.67
`goo.gl` (5)	72,264	165,596	4,796,578	30.38	4.96

Lexical Features: We analyzed three lexical features of the four million URLs from `bit.ly` and `goo.gl`: the length of the URLs, the number of dots in the hostnames of the URLs, and the number of tokens in the paths and queries. First, we examined the distribution of URL lengths. The lengths are distributed between 1 and 5463 characters (we ignore schemes of URLs, such as `http://`, when we measure their lengths). Of the examined lengths, about 76.2% are shorter than 100 and about 19.4% are between 100 and 200 (Figure B.8a). Since most URLs are shorter than 200, URLs longer than 200 are unusual and need to be investigated. We even found that some shortened URLs, such as `bit.ly/9JcN2k`, redirected to long character strings instead of valid URLs.

Since a `bit.ly` shortened URL with six-characters produces an alias with a length of 13, benign users will not shorten the URLs with lengths that are less than or equal to 13. Nevertheless, we found shorter than 13 character URLs in about 1.2% of the crawled URLs; most of them are already shortened URLs. This result implies that someone recursively performs URL shortening to avoid blacklisting, which is known as nested URL shortening [26]. For instance, when a user clicks a link to `bit.ly/ctrKUV`, he will be redirected to `ow.ly/20ny9` and then be redirected to a spam page.

Next, we checked the number of dots in the hostnames of the URLs. The number of dots is distributed between 0 and 45. Among them, about 0.2% have no dot, about 24.6% have one dot, and about 65.7% have two dots in their hostnames (Figure B.8b). The URLs that have no dot in their hostname include `localhost` and meaningless strings. In addition, some shortened URLs, such as `goo.gl/LrJeh`, are associated with a suspicious URL that has 45 dots.

We also checked the number of tokens in the paths and queries of the URLs. We split the path and query by using special characters including '#', '/', '?', '.', '=', '-', and '_'. The number of tokens is distributed between 0 and 1308. Among them, about 88.6% have fewer than 16 tokens in their

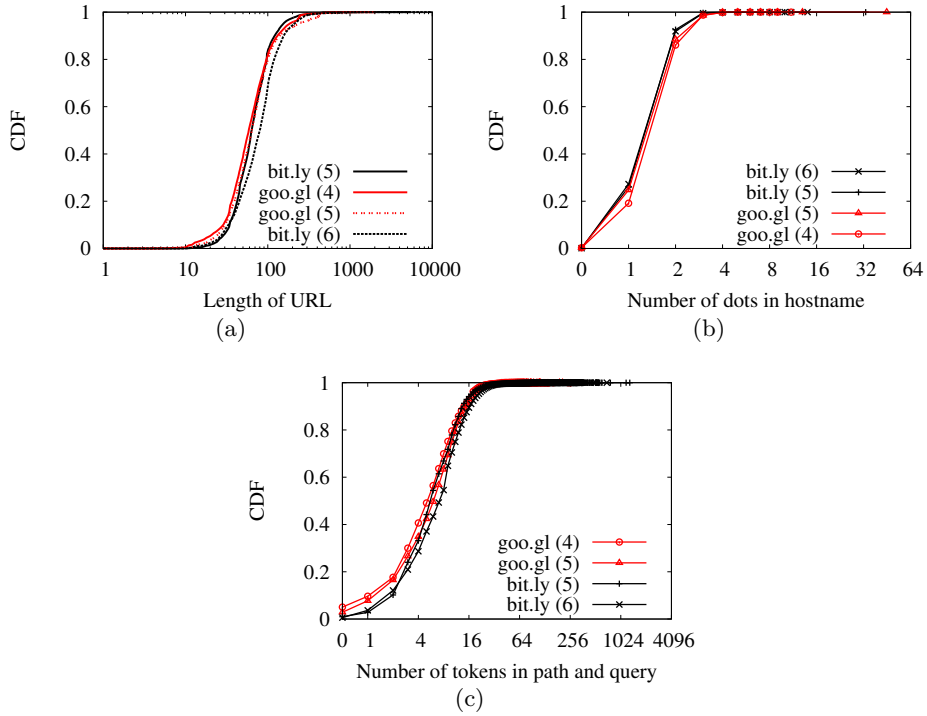


Figure B.8: Lexical features of URLs in bit.ly and goo.gl

path and query (Figure B.8c).

Daily Clicks: We checked the number of daily clicks of the crawled shortened URLs of `goo.gl` using its API for checking click statistics. Among the 1,161,694 `goo.gl` URLs, 3,950 were clicked from 1 to 39,236 times in a day during the crawling period. About 97.5% of the 3,950 URLs were clicked fewer than 100 times in a day (Figure B.9). Therefore, the URLs clicked more than 100 times are unusual and thus should be investigated.

Connection Status: We checked the connection statuses of the 3,969,907 crawled URLs. We attempted to connect to them during April 22–26, 2011. We could access about 77.8% of the URLs (we followed all redirections). The remaining URLs led us to unreachable hosts or nonexistent pages. Since malicious hosts and pages usually have short lifetimes, USSes need to investigate who created and distributed URLs with errors.

Protocols: We checked which protocols are mostly shortened by `bit.ly` and `goo.gl`. In our data sets, about 99.1% of URLs were HTTP and about 0.9% of URLs were HTTPS. Other protocols include FTP, RTSP, ITMS for

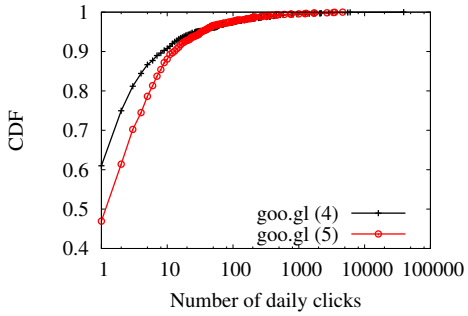


Figure B.9: Daily clicks of goo.gl's URLs visited at least once

Table B.3: Connection status of URLs in bit.ly and goo.gl

Response	Number	Portion (%)
No response	312,648	7.88
Success (2XX)	3,089,933	77.83
Client error (4XX)	518,785	13.07
Server error (5XX)	48,541	1.22

iTunes Music Store, and MARKET for Android Market (Table B.4).

Table B.4: Protocols of URLs in bit.ly and goo.gl

Protocol	Number	Portion (%)
HTTP	3,935,794	99.141
HTTPS	33,964	0.855
Others	149	0.004

IP Address Instead of Domain Name: Some malicious URLs contain IP addresses instead of domain names [24, 40, 41, 42]. We examined the number of URLs containing IP addresses in our data set. We found 16,067 URLs containing IP addresses, comprising about 0.4% of the crawled URLs.