Botnets are a major threat to all those connected to the Internet. Botnets are a network of compromised computers (or *bots*), that are controlled from a central location, to perform malicious activity. Botnet owners (*botmasters*) continue to adapt in order to hide from detection. One of the newest techniques employed by botnets is fast flux (FF).

The FF technique hides the dedicated servers hosting the malicious content. These servers, which are primarily web hosts are called *motherships*. Fast fluxing does this by utilizing the bots as proxies that forward messages to and from the motherships, effectively hiding them from traffic sniffers. By hiding these servers, botnets can stay online longer and infect more machines. This paper focuses on how to detect and mitigate botnets using FF techniques.

Chapter 1

Introduction

1.1 Introduction

Botnets are a major threat to all those connected to the Internet. They are used to distribute spam and malicious code, and perform denial of service (DOS) and phishing attacks. Through the use of fast-flux (FF) techniques, they are able to evade detection and stay running longer.

A botnet is a number of infected computers, being controlled from a central location, without their owners knowledge. These infected computers (also known as *bots*) coordinate in order to perform malicious activity and infect more hosts. Botnets vary in size and can span across domains and geographic locations. They are controlled by a central server called a *command and control* center (C&C). The C&C provides instructions to the bots, so they can act as one against one or more targets. Updates are also provided to the bots by the C&C, to keep them hidden and able to use the latest exploits. Some general botnets and all botnets utilizing FF also have deticated web hosts, called *motherships*. The motherships host the malicious or illegal website and the malicious code used for exploiting victims.

Because the bots are not owned by the *botmasters* (the people running the botnet), they are treated as unreliable and expendable. The owners of the compromised computers can shutdown their computer, disconnect it from the network, or find the malicious software and have it removed. Removing infected hosts from a botnet will not degrade performance, however blocking or removing the mothership will effectively take down the botnet. Without the mothership, a botnet has no host for its malicious software and cannot continue to distribute.

In order to keep the motherships running and undetected, the new technique of FF has been implemented. FF hides the mothership behind a group of ever changing proxies. In response to this newly developed technique, new detection methods have been developed. These methods include active and passive domain name system (DNS) monitoring (discussed in sections 1.5.1 and 1.5.1), analyzing data captured over a period of time (discussed in section 1.5.4), and analyzing domain names (discussed in section 1.5.2).

Since the FF technique relies heavily on DNS, I will describe it first. I will then briefly describe the structure and control of general botnets. I'll continue by discussing how FF works, how it is used legitimately, and then how it is utilized by botnets to evade detection. After discussing how botnets use FF, I'll describe the various techniques to discover FF botnets. In conclusion, I'll describe ways to combat FF botnets and the future direction of research.

1.2 DNS Overview

DNS is utilized by FF botnets to route victims to their malicioius sites. DNS resolves a common name, such as www.google.com, into its Internet Protocol (IP) address, such as 127.0.0.1. The IP address is how the network knows where to route packets. In its original conception DNS was designed to be static, changing the relationship between common name and IP address rarely. Dynamic DNS (DynDNS) arose from a need to alter the name to address relationship quickly. FF botnets utilize DynDNS to quickly alter to which hosts the malicious domain name resolves.

Another aspect of DNS that FF botnets take advantage of is called round-robin. If a common name has several IP addresses associated with it, DNS will sequentially select each IP as a new request comes in. For both legitimate and malicious URLs, this helps balance the traffic comming in to each server. Malicious domains use both DynDNS and round-robin to hide their motherships behind an ever changing set of proxies.

1.3 Botnet Overview

Botnets are malicious networks of compromised computers. The owners of the computers are unaware that their computer is part of this network and performing illegal activity. Botnets are structured with a central C&C which provides commands to each of the individual bots. These bots cross logical and geographic domains. They are comprised of different computers hosted on different Internet service providers (ISPs) and in different countries. Bots can either be controlled individually or as a whole.

Botnets grow through the spread of malware and other malicious code. This code is distributed via phishing scams, known remote exploits, and through other technical and social weak points. Once infected, the bot will call back to the C&C to notify that it is now active and to receive updates. This new bot will then take instructions from the C&C and begin performing malicious activity.

According to Zhu et al. (2008), there are three primary ways to control a botnet: through HTTP requests, IRC channels, and P2P protocols. IRC botnets are the simplest form of botnets and are easy to detect. These botnets can be detected by monitoring IRC channels for non-English or unusual usernames and non-English or unusual traffic in the channel. Because botnets being controlled by IRC are easy to detect, HTTP and P2P controls were developed. These two protocols can easily hide among the legitimate traffic of the same type. The FF technique (discussed in section 1.4), on which this paper focuses, is a way to enhance the already existing botnets.

1.4 Fast Flux Techniques

According to the ICANN Security and Stability Advisory Committee (2008), fast fluxing is a technique for rapidly altering the IP address that is associated with a domain name. This technique utilizes DynDNS to perform these updates. The selection of the next host to be used can be sequential, as in a round-robin; through weights, so that a currently overloaded node doesn't get more traffic; or random, which is most likely how malicious sites work. The original purpose for DynDNS and FF was for legitimate uses, such as load balancing, but botnets have borrowed the technique, so they can stay active.

1.4.1 Legitimate Uses

There are several legitimate uses for FF. One of the primary uses is for load balancing. For websites that see a lot of traffic, there needs to be a way to balance the requests between several servers. Without this balance, the web servers would become overwhelmed and not be able to respond as quickly.

A legitimate domain will tend to have a few dedicated servers for web hosting. These servers will have strong hardware that will be able to handle relatively large amounts of traffic. Even though each server can handle more than a home computer could, it still will not be able to handle the amount of traffic the website will see. One technique for balancing traffic flows between servers is through fast fluxing.

For a legitimate domain, fast fluxing is an inexpensive way to force traffic to one of its dedicated servers. The domain will advertise several IPs hosting their web content. The round-robin DNS will take care of choosing which IP to which to send the traffic. After the advertised TTL, the domain will advertise another set of IP addresses.

The reason to continue to continuously rotate out IP addresses is in case one of the servers is experiencing an abnormal situation. Advertising different IP address will allow some servers to be taken down for regular maintenance; the end user will not see any degradation in performance, since they will not be routed to that server. Rotating out IP addresses also allows a domain to remove servers that are currently overloaded with traffic, allowing the traffic to die down before rotating it back into use. Also, if a server goes down unexpectedly, a shorter TTL value will rotate it out sooner, to reduce degradation in service and allow it to be repaired.

A larger number of hosts (and hence IP addresses) and a shorter TTL increases the performance of a domain. The larger number of IP addresses allows the domain to balance among more hosts, thus each host will see a smaller amount of traffic. If a server goes down unexpectedly, a shorter TTL will remove it from service sooner and will allow servers to be repaired more regularly.

1.4.2 Use in Botnets

According to Nazario and Holt (2008), Caglayan et al. (2010), Zhang et al. (2011), and Zhou et al. (2008), botnets use FF techniques in order to hide their dedicated web hosts or motherships. A botnet using the FF technique utilizes its bots as proxies. When a victim attempts to connect to a host using a malicious URL, it will connect to one of

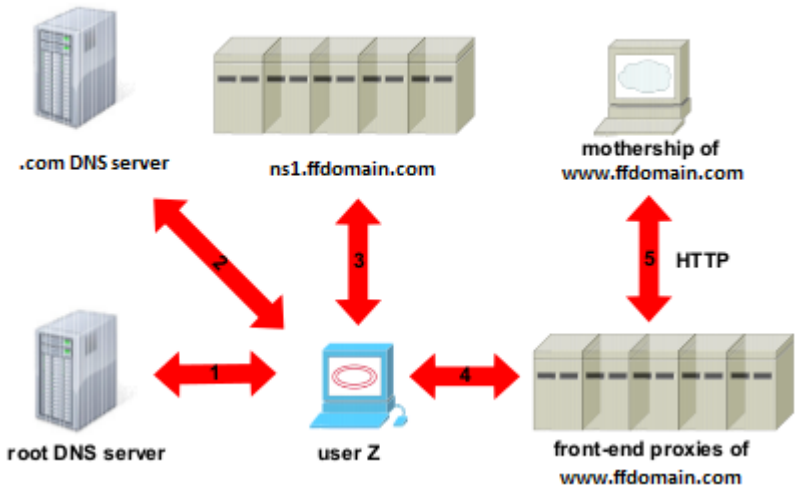

Figure 1.1: Fast flux example from Zhou et al. (2009)

the proxy servers (one of the bots) first. Then the proxy will contact the mothership, who will respond with the content of the web-page. This content is then served back to the victim through the proxy (the original bot). The victim will never know that there is a mothership, its IP address, or where it is located. This means that should a malicious domain be discovered, simple detection would only find the proxies rather than the mothership, allowing them to continue to operate.

Each of the individual bots in a botnet are assumed to be unreliable by the botmaster. This assumption is made, because the botmaster has no control over the owner of the infected computer (bot). The owner is able to shutdown their computer at any time, remove it from the network, or discover and remove the malicious software. Because the bots are considered unreliable, the botmaster will swap them out often. This way should one of the bots go offline, they will be swapped out quickly, and fewer victims will be directed to an offline bot.

Botmasters use FF to hide the mothership as well as to load balance and swap out failed bots. Since the bots are unreliable, the botmasters want to advertise as many of them as possible with a short TTL. This way the botnet will be better balanced, since each victim will be directed to a different bot; and the botnet will have little down time, since the failed bots will be swapped out quickly.

There are two primary ways in which botnets use FF techniques: single and double flux. For *single-flux* botnets, only the bots serving as proxies are fluxed. In a *double-fluxed* botnet, the botmaster has complete control over other servers that are used as *name servers* (NS), and those are also fluxed. Using the example domain of *www.ffdomain.com* and using a similar example given by Zhou et al. (2009) and their figure (figure 1.1) to show how FF works and highlight the difference between single and double flux.

First for a single-flux botnet, when a victim attempts to contact www.ffdomain.com, they will first be directed to the root DNS server, who then direcrets them to the *.com* DNS server (steps 1 and 2, in figure 1.1). This DNS server will respond with an IP address associated with the *ffdomain*. This IP address will correspond to one of many bots acting as proxies (for single flux, step 3 is skipped). The victim will then send an HTTP request to this IP (step 4) and the bot will contact the mothership, get the content, and serve it back to the victim. Each time a new victim connects, they will receive a different IP that is currently associated with *www.ffdomain.com*. After a short amount of time, all IP addresses are swapped out and a new set of bots are advertised.

For a double-flux botnet, the victim will start the same way, by contacting the root, then the *.com* DNS server (steps 1 and 2, in figure 1.1). This time, however, the root DNS server will respond with the IP address of the *ffdomain* name server, or *www.ns1.ffdomain.com*. The IP address associated with *www.ns1.ffdomain.com* is also being fluxed (step 3). From here the victim will continue as before, receiving one of many IP addresses for *www.ffdomain.com* (step 4). The name servers are controlled by the botmaster and tend to be swapped slower then proxies. They are swapped slower, because rapidly fluxing nameservers are not seen in legitimate domains. Name servers also use more resources to setup, maintain, and run than proxies, so their will be fewer of them.

1.5 Detecting Fast Flux Botnets

The largest challenge when it comes to detecting FF botnets is distinguishing them from legitimate sites using fast fluxing techniques. In this section I'll describe several different detection techniques that will help to differentiate between legitimate websites and botnets.

These techniques can be broken down into four main categories: active, passive, semantic, geolocalization, and analytic detection. Active detection queries the DNS servers for information; passive detection looks for the same information in packets traversing the network; semantic detection looks for algorithmically designed URLs, which are typical of FF botnets; geolocalization attempts to map the delay times between query and response to a geolocation; and analytic detection performs analysis on past data to attempt to discover new botnets, malicious URLs, or more components to an already known botnet.

There are also several ways in which FF botnets can deceive the current detection techniques. This deception is discussed in section 1.6.

1.5.1 Active and Passive DNS Detection

Active and passive DNS detection rely on the same data in order to detect FF botnets. Both of these methods look for data related to the *TTL* (section 1.5.1), the *FF activity index* (section 1.5.1), and the *footprint index* (section 1.5.1). The TTL is the amount of time a particular DNS record will remain in memory; the FF activity index is how many unique IPs are seen within a certain amount of time; and the footprint index is how spread out the IPs are.

Active Sensors

The primary way to perform active detection is using the *dig* command, but *nslookup* and other look-up tools can be used as well. These commands queries the DNS server for the IP addresses and TTL associated with a domain name. Caglayan et al. (2009), Zhou et al. (2009), Zhang et al. (2011), Holz et al. (2009), and Nazario and Holz (2008) all use active sensors by specifically querying the DNS servers. Each of these authors discover which domains to test by observing links in spam email and malware blacklists. The drawback to active detection is it takes at least TTL+1 for detection.

Each of these authors approach the problem a little differently. Caglayan et al. (2009) approach the problem from two directions utilizing, both active and passive methods. After ten minutes of collection, the authors used a Bayesian classifier to achieve 96.6% accuracy and 5% false positive rate. The primary drawback to this detection technique is its high false positive rate. This work also only focuses on a single sensor, but the authors also hope to create a distributed system of sensors.

Utilizing a distributed system of active sensors, Zhou et al. (2009) were able to achieve a shorter time to classify. As with both passive and active sensors, the more unique IPs that are seen, the more accurate the predictions can be. Using a distributed system of sensors, Zhou et al. (2009) were able to collect a larger amount of samples in a shorter amount of time. The authors, in their previous work [28], that after 24 hours of collection, are able to achieve a 70.2% detection rate and a 34% false positive rate. In their more current paper [27], the authors claim to achieve the same rates, and increased their speed 16 to 10,000 times faster. This increase will reduce the detection time from 24 hours to 1.5 hours to as low as 8.64 seconds. The largest drawbacks to this method are the communication overhead between nodes and the very high false positive rate.

Passive Sensors

To detect FF botnets, Perdisci et al. (2012), Huang et al. (2010), and Caglayan et al. (2009), use passive techniques. These authors setup sensors inside of networks to monitor the traffic. They collected response DNS packets parsing out the IP address, domain name, and TTL. Due to the large volume of traffic, the packets need to be filtered before processing them. This was accomplished by using the TTL values, white lists of known benign sites, and black lists of known malicious sites. Huang et al. (2010) utilized primarily the footprint index to provide detection, while Perdisci et al. (2012) and Caglayan et al. (2009) analyzed all three values (TTL, footprint index, and FF activity index), to detect FF networks.

Using only the footprint index, and 0.5 seconds of detection, Huang et al. (2010) were able to achieve 98.16% accuracy and a 0.398% false positive rate. To achieve these rates, the authors used distributed detection (multiple servers over a large area) and distributed processing. The authors used the traffic to gather the IP addresses with which the DNS servers responded. They then located the geographic location of the IP address by using the Spacial Coordinate Database, hosted on *hostip.info*. While this technique has one of the lowest false positive rates and highest accuracy, there are three drawbacks. One, it requires multiple IP address returned by a query to work, so

if just one is returned it will fail; two, if the botnet is limited to a single geographic location, it will perform poorly; and three, if the IP addresses are not found in the Spacial Coordinate Database, it cannot be included in the data.

The other authors, Perdisci et al. (2012) and Caglayan et al. (2009) use all three sensors in order to detect FF networks. Perdisci et al. (2012) also include an additional technique to classify; *domain clustering*, a technique to treat like domain names as part of the same network. Domain clustering helps to determine FF botnets, since a single botnet will utilize many similar domain names, to continue to fool victims and blacklisting. Using their technique Perdisci et al. (2012) achieve a 99.3% accuracy rate with a 0.15% false positive rate. The rates achieved by the authors are better then previous work, but the drawback to this technique is it can take up to 30 hours to detect the FF domain (up to 24 hours of collection and up to 8 hours for processing).

Time To Live (TTL)

The TTL for a FF botnet tends to be shorter than the TTL for a legitimate use of FF. The typical FF botnet will have a TTL under 300 seconds and the typical legitimate site will have a TTL from 600–3600 seconds. This measure was more useful in the past, but as botmasters are trying to mimic legitimate domains, thus making their TTL longer. Some legitimate domains have shorter TTL values as well.

This measure is not useful on its own, but is primarily used as a baseline or a way to filter DNS responses. Large TTL values can be disregarded as most FF botnets will have a value less than 3600 seconds. Thus any response with a TTL greater than this range can be disregarded.

Fast Flux Activity Index

The FF activity index refers to how many unique IP address are seen in a given time period. For a FF botnet this will be a much larger number than a legitimate site. There is no clear distinction of what this value is, so the current techniques compare detected unique values to known legitimate domains. As a greater number of unique IP addresses are seen, the higher the confidence that a FF botnet is being detected. The reason a FF botnet will advertise a larger number of unique IP addresses, is because the bots (proxies) are considered unreliable. Since the bots are unreliable, in order to have a higher average online time, the botmaster will need to advertise a large number of bots and frequently change them.

For legitimate sites the FF activity index will typically be much lower. Since legitimate sites are using dedicated hardware these sites can be sure that the servers will have longer up-times, handle more queries, and have the advantage of knowing when they will be shutdown. Because of these things, a legitimate site has less need to have as many servers. Later, in section 1.6 I will describe how a botnet can limit the number of unique IP address to fool the detection.

Footprint Index

The footprint index detects how spread the servers are between different domains and geolocations. Since a FF botnet is comprised of infected home computers, they reside in different *Autonomous System Numbers* (ASNs). An ASN is an identifier for a unique sub-network in the Internet. An ASN can be controlled by a single or multiple network operators. Caglayan et al. (2009) used a threshold of six unique ASNs before classifying a network as a botnet.

For a legitimate site, this spread is much smaller. Legitimate sites tend to house their servers, for a particular service region, in a single area on a single domain. The close proximity of the servers helps maintain the infrastructure, because there are fewer locations and networks to maintain. Later, in section 1.6 I will describe how botnets can limit there footprint to evade detection.

1.5.2 Semantic Detection

Another technique for detecting FF domains is looking at the domain name itself. Since FF domains need several domains to evade detection and need to mimic legitimate domain names to fool users, they display certain patterns. Yadav et al. (2012) attempt to detect FF networks by analyzing domain names. Their algorithm searches for specific attributes associated with algorithmically generated domain names. The attributes include: similarity to known legitimate domains, similar structure to English words, but are not found in the dictionary, the distribution of alphanumeric characters, and the distance (number of changes) from known malicious domains.

Using these domain name detection techniques, the authors claim to achieve a 100% detection rate for all test sizes (50, 100, 200, and 500 domains per test), but the false positive rate changes per test size. For the lowest set the authors used, testing 50 domain names, the average false positive rate was 27.5%; for data-sets of 100 domains, the false positive rate averages 6%, for data-sets of 200 domain names, the false positive rate averages 5%; and for data-sets of 500 domain names, the authors claim a false positive rate of averages 2%. The difference in false positive rates, per data-set size, is attributed to using the different detection techniques described in the previous paragraph.

While the authors were able to detect 100% of malicious domains, they had a relatively high false positive rate. Because of this high false positive rate, this technique cannot be the only way to detect FF botnets, since a large number of legitimate domains will also be flagged. Another drawback to this technique, is if botmasters only use English words, they can better their chances of evading detection. Because of the drawbacks, this technique is used for initial filtering of information. The previous techniques will then have less data to process and can discover FF botnets in less time.

1.5.3 Geolocalization

Geolocation is a technique for figuring out where a server is located on the globe, based on network latencies. In their paper, Castelluccia et al. (2009) present a way of tracing back where the mothership is located on the planet. Using several different clients, they

first triangulate the location of the proxy by sending packets directly to the proxy and calculating the delay times. These delay times are translated into a physical distance from the client using average network latency (distance is equal to the latency divided by the packet response time).

Once it is known where several proxies are, these are then used to triangulate the mothership. The authors (Castelluccia et al. (2009)) then access the content through these proxies via HTTP requests. They subtract the latencies to the proxies from the total latency to get the distance from the proxy to the mothership. In this way the authors triangulate the location of the mothership to within 100 km of the actual location. This technique already requires the knowledge of a FF domain.

1.5.4 Analytic Detection

Analytic detection is a way of looking at previously collected information. It attempts to narrow the search space, leading to higher detection rates and lower false positive rates. Analytic detection does this by discovering new domain names the botnet utilizes, the size of the botnet, and the structure of the botnet.

New domain names are discovered by performing reverse lookups on a known bot. The new domain names can be further analyzed to provide greater information for semantic detection (section 1.5.2). The size of a botnet can be determined through the number of unique IP addresses discovered. The size discovered will always be less than the total size as not all bots participate in FF. The size is used to help determine better thresholds for the FF activity index (section 1.5.1). The structure of the botnet is also discovered by observing the IP addresses. Using the IP addresses the location of bots and which bots tend to be more active and at what times during the day can be discovered. By analyzing the structure, better thresholds can be determined for the footprint index (section 1.5.1) and for geolocation (section 1.5.3). Using this data, the previous detection techniques can be improved upon by also narrowing the search space.

1.6 Limitation of Current Detection

The previously described detection techniques in section 1.5 can be thwarted in several different ways. Each type of detection has its own shortcomings which are discussed here.

In order to defeat the the active and passive sensors, botnets can act like legitimate domains. This technique, as discussed by Knysz et al. (2011), is dubbed a mimicry attack. In a legitimate domain, only a few IPs are advertised at a time and they tend to be located in a single region. A botnet can mimic these conditions to evade detection. Botnets can limit each set of advertised IPs to one or two ASNs and limit the number of unique IPs per advertisement. This limitation is done by repeating previously used hosts and keeping track of where hosts are located. This technique will delay detection, but won't completely evade detection, because eventually, as bots are shutdown, more unique IP address will need to be advertised. Using this mimicry technique can

adversely effect the performance of the botnet. Advertising only a few unique IP addresses increases the probability that one or more of those bots has been taken off line.

In order to defeat the semantic detection, botnets can use English words and naming conventions that will counter the detection presented by Yadav et al. (2012). However, botmasters want to mimic well known addresses to fool victims, which requires the use of similar naming while altering only a few characters. Attempting to use English words will reduce the number of domains that a botmaster can implement. This will adversely effect the performance, because once a domain is discovered, a botmaster must use a different one. With limited unique names, a botmaster can only target a specific group of victims for a short period of time.

To fool geolocation, a botnet can introduce delays into its servicing. However, doing so will delay content from reaching the victim. If end users are experiencing slow performance, they may navigate away, reducing the effectiveness of the botnet.

FF botnets are becoming more sophisticated and can more easily evade detection. However, due to the unreliable nature of nodes in a botnet, many of these techniques can reduce the performance of the botnet. Botmasters are finding ways to balance performance and evasion, which creates a constant battle with those attempting to find and block the botnets.

1.7 Defense of Fast Flux Botnets

There are only a few defenses to FF botnets, once they have been discovered. ISPs maintain a large blacklist of known malicious domains, and do not serve them to the end user. Anti-virus software integrated with browsers also provide some protection against malicious domains. Anti-virus software is also becoming more sophisticated to track and remove malicious code on end user machines. Known bots are also being constantly monitored to discover new domains that are being generated by the botnets.

1.8 Future Research

Further research and development is needed to keep up with the ever changing nature of botnets. Reducing the time of detection is critical to reducing the risk botnets pose. Most FF domains only last about one month [1], constantly changing domain names during that time. This can be achieved through more sophisticated heuristic algorithms, that don't need to analyze as much data, or through the use of distributed processing, so the same amount of data can be processed more quickly. Early detection is needed to stop these domains before too much damage is done.

Another area of research is determining in greater detail the differences between legitimate and malicious domains. As FF botnets continue to find better ways to mimic other domains, as discussed in section 1.6, the false positive rate of the current techniques will rise. Also by identifying the differences in greater detail, this will reduce the amount of time needed to locate the domains. These differences are detected by analyzing previously collected data on botnets.

One other area of research is tracking down the mothership. One technique is briefly described by Zhou (2008), but they do not discuss an algorithm or experiments. In a FF botnet, the bots (proxies) are treated as disposable, so detecting and removing them will not have a great impact on the strength of a botnet. The motherships are the hosts of the malicious software and if these are taken offline, the botnet becomes useless.

1.9 Conclusion

In conclusion there is a constant battle between FF botnets and the detection methods used. Fast fluxing is a technique to hide the servers hosting malicious content behind a wall of proxies. These proxies are fluxing in and out of use through dynamic DNS. In order to detect these malicious domains, active, passive, semantic, geolocation, and analytic sensing techniques are used. These track the potential domain names and IP addresses of the bots for these domains.

There are several techniques that FF botnets can employ to reduce the risk of detection. These include mimicking legitimate domains by advertising only a few unique IP addresses, per TTL, within only a few ASNs, using English words in their domain names, and introducing delay into the service. While each of these reduces the risk of exposure, they also adversely effect the performance of the botnet. Detection techniques need to continuously adapt in order to detect FF botnets.

Chapter 2

Methods Used

Chapter 3

Results

Chapter 4

Conclusion

Appendix A

User Manual

Appendix B

Design Documents

Appendix C

Source Code

Bibliography

[1] A. Caglayan, M. Toothaker, D. Drapaeau, D. Burke, and G. Eaton. Behavioral Patterns of Fast Flux Service Networks. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference*, pages 1–9, Jan. 2010.

[2] A. Caglayan, M. Toothaker, D. Drapeau, D. Burke, and G. Eaton. Real-Time Detection of Fast Flux Service Networks. In *Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications Technology*, pages 285 –292, march 2009.

[3] Claude Castelluccia, Mohamed Ali Kaafar, Pere Manils, and Daniele Perito. Geolocalization of Proxied Services and its Application to Fast-Flux Hidden Servers. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '09, pages 184–189, New York, NY, USA, 2009. ACM.

[4] Chia-Mei Chen, Ya-Hui Ou, and Yu-Chou Tsai. Web Botnet Detection Based on Flow Information. In *Computer Symposium (ICS), 2010 International*, pages 381–384, Dec. 2010.

[5] Chi-Chung Cheung, Man-Ching Yuen, and A.C.H. Yip. Dynamic DNS for Load Balancing. In *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference*, pages 962–965, may 2003.

[6] Hyunsang Choi, Heejo Lee, and Hyogon Kim. BotGAD: Detecting Botnets by Capturing Group Activities in Network Traffic. In *Proceedings of the Fourth International ICST Conference on Communication System Software and Middleware*, COMSWARE '09, pages 2:1–2:8, New York, NY, USA, 2009. ACM.

[7] C.J. Coit, S. Staniford, and J. McAlerney. Towards Faster String Matching for Intrusion Detection or Exceeding the Speed of Snort. In *DARPA Information Survivability Conference Exposition II, 2001. DISCEX '01. Proceedings*, volume 1, pages 367–373 vol.1, 2001.

[8] Guofei Gu, Junjie Zhang, and Wenke Lee. BotSniffer : Detecting Botnet Command and Control Channels in Network Traffic. *Technology*, 53(1):1–13, 2008.

[9] Thorsten Holz, Christian Gorecki, Konrad Rieck, and Felix C. Freiling. Measuring and Detecting Fast-Flux Service Networks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2008.

[10] Si-Yu Huang, Ching-Hao Mao, and Hahn-Ming Lee. Fast-Flux Service Network Detection Based on Spatial Snapshot Mechanism for Delay-Free Detection. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 101–111, New York, NY, USA, 2010. ACM.

[11] ICANN Security and Stability Advisory Committee. *SSA Advisory on Fast Flux Hosting and DNS*, March 2008.

[12] M. Knysz, Xin Hu, and K.G. Shin. Good Guys vs. Bot Guise: Mimicry Attacks Against Fast-Flux Detection Systems. In *INFOCOM, 2011 Proceedings IEEE*, pages 1844–1852, April 2011.

[13] C. Mazzariello. IRC Traffic Analysis for Botnet Detection. In *Information Assurance and Security, 2008. ISIAS '08. Fourth International Conference*, pages 318–323, Sept. 2008.

[14] D.K. McGrath, A. Kalafut, and M. Gupta. Phishing Infrastructure Fluxes All the Way. *Security Privacy, IEEE*, 7(5):21–28, Sept.-Oct. 2009.

[15] P. Mockapetris. Domain Names - Implementation and Specification. RFC 1035, Internet Engineering Task Force, November 1987.

[16] J. Nazario and T. Holz. As the Net Churns: Fast-Flux Botnet Observations. In *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference*, pages 24–31, Oct. 2008.

[17] R. Perdisci, I. Corona, D. Dagon, and Wenke Lee. Detecting Malicious Flux Service Networks Through Passive Analysis of Recursive DNS Traces. In *Computer Security Applications Conference, 2009. ACSAC '09. Annual*, pages 311–320, Dec. 2009.

[18] Roberto Perdisci, Igino Corona, and Giorgio Giacinto. Early Detection of Malicious Flux Networks via Large-Scale Passive DNS Traffic Analysis. *IEEE Transactions on Dependable and Secure Computing*, 99(PrePrints), 2012.

[19] William Salusky and Robert Danford. Know Your Enemy: Fast-Flux Service Networks. http://www.honeynet.org/papers/ff, July 2007. The Honeynet Project.

[20] Tao Wang and Shun-Zheng Yu. Centralized Botnet Detection by Traffic Aggregation. In *Parallel and Distributed Processing with Applications, 2009 IEEE International Symposium*, pages 86–93, Aug. 2009.

[21] Jiayan Wu, Liwei Zhang, Jian Liang, Sheng Qu, and Zhiqiang Ni. A Comparative Study for Fast-Flux Service Networks Detection. In *Networked Computing and Advanced Information Management (NCM), 2010 Sixth International Conference*, pages 346–350, Aug. 2010.

[22] Yunfeng Xu, Yansheng Lu, and Zhengbiao Guo. The Availability of Fast-Flux Service Networks. In *Mobile and Wireless Networking (iCOST), 2011 International Conference on Selected Topics*, pages 89–93, Oct. 2011.

[23] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan. Detecting Algorithmically Generated Domain-Flux Attacks With DNS Traffic Analysis. *Networking, IEEE/ACM Transactions on*, PP(99):1, 2012.

[24] Sheng Yu, Shijie Zhou, and Sha Wang. Fast-Flux Attack Network Identification Based on Agent Lifespan. In *Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International Conference*, pages 658–662, June 2010.

[25] Xiaocong Yu, Xiaomei Dong, Ge Yu, Yuhai Qin, Dejun Yue, and Yan Zhao. Online Botnet Detection by Continuous Similarity Monitoring. In *Information Engineering and Electronic Commerce, 2009. IEEC '09. International Symposium on*, pages 145–149, may 2009.

[26] Lei Zhang, Shui Yu, Di Wu, and P. Watters. A Survey on Latest Botnet Attack and Defense. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference*, pages 53–60, nov. 2011.

[27] Chenfeng Zhou, Christopher Leckie, and Shanika Karunasekera. Collaborative Detection of Fast Flux Phishing Domains. *Journal of Networks*, 4(1), 2009.

[28] C.V. Zhou, S. Karunasekera, and C. Leckie. Evaluation of a Decentralized Architecture for Large Scale Collaborative Intrusion Detection. In *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, pages 80–89, 2007.

[29] C.V. Zhou, C. Leckie, S. Karunasekera, and Tao Peng. A Self-Healing, Self-Protecting Collaborative Intrusion Detection Architecture to Trace-Back Fast-Flux Phishing Domains. In *Network Operations and Management Symposium Workshops, 2008. NOMS Workshops 2008. IEEE*, pages 321–327, April 2008.

[30] Zhaosheng Zhu, Guohan Lu, Yan Chen, Z.J. Fu, P. Roberts, and Keesook Han. Botnet Research Survey. In *Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International*, pages 967–972, 28 2008-Aug. 1 2008.