Chapter 2
Malicious Networks for DDoS Attacks

Abstract In this chapter, we explore botnet, the engine of DDoS attacks, in cyberspace. We focus on two recent techniques that hackers are using to sustain their malicious networks, fast fluxing and domain fluxing. We present the mechanisms of these two techniques and also survey the detection and anti-attack methods that have been proposed against them in literature.

2.1 Introduction

Nowadays, there are numerous malicious attacks in the cyberspace. These attacks are pervasive in the Internet, and often cause great financial loss [1, 2]. Botnets are the engines behind majority of the attacks. A botnet is usually established by a botnet writer developing a program, called a bot or agent, and installing the program on compromised computers on the Internet using various techniques. All the bots from a botnet are controlled by a botmaster. The hosts running these programs are known as zombies [1, 3, 4]. For a botnet, there is one or a number of command and control (*C&C*) servers to communicate with bots and collect data from them. In order to disguise himself from legal forces, botmaster changes the URL of his *C&C* frequently, such as weekly. An excellent explanation about this could be found in [3].

Motivated by huge financial or political reward, attackers find it worthwhile to organize sophisticated botnets for use as attack tools. There are numerous types of botnets in cyberspace, such as DSNXbot, evilbot, G-Sysbot, sdbot, and Spybot [3]. On one hand, researchers have studied botnets from various perspectives, including botnet probing events [5], Internet connectivity [6], size [7], and domain fluxing [8, 9]. On the other hand, botnet owners have at their disposal state-of-the-art techniques, such as stepping stones, reflector, IP spoofing [1, 10], code obfuscation, memory encryption [11], and peer-to-peer implementation technology [10, 12, 13] to sustain their botnets and disguise their malicious activities and traces.

S. Yu, *Distributed Denial of Service Attack and Defense*, SpringerBriefs in Computer Science, DOI 10.1007/978-1-4614-9491-1_2, © The Author(s) 2014

A report from Symantec's MessageLabs shows 90.4 % of total emails were spam in June 2009. Among all spam, 83.2 % was sent through botnets. In addition, many spam emails included viruses, phishing attacks, and web-based malware. Therefore, sending spam through botnets can help to conduct further network attacks [14].

Researchers have applied signature-based methods to detect botnets for a long time. These signature-based techniques have been widely employed by some Honeynet projects, which has been discussed in [15, 16]. However, these methods cannot detect newly developed botnets as the signatures of new botnets are unknown or some botnets are polymorphic [17]. Some IRC-based approaches were developed to overcome this problem. For example, Binkley et al. [18] developed an anomaly based system combining IRC statistics and TCP work load, and Karasaridis et al. [19] applied a passive anomaly-based characterization methodology based on botnets behavior characteristics. However, these methods have high false positive rates [17].

Many researches focused on how to detect botnets or trace the botnet master. Meanwhile, many surveys reflected what had been done and summarized what future work should be. Feily et al. [20] surveyed botnet mechanisms and botnet detection techniques based on different classes they identified: signature-based, anomaly-based, DNS-based, and mining-based. They also compared and evaluated the advantages and disadvantages of some typical researches from each category.

However, in order to disguise their traces and malicious activities, botnet writers are exhausting their energy to design new strategies and mechanisms to fly under the radar. In this chapter, we discuss two recent advanced botnet mechanisms:

1. Fast Flux (FF in short): A mechanism that a set of IP addresses change frequently corresponding to a unique domain name.
2. Domain Flux (DF in short): A mechanism that a set of domain names are generated automatically and periodically corresponding to an URL of a *C&C* server.

2.2 The Fast Flux Mechanism and Detection

2.2.1 The Fast Flux Mechanism

Fast Flux (FF) is an earlier strategy employed by hackers to evade botnet detection. By IP fast flux, the mapping between multiple IP addresses and one single domain name is rapidly changing [21]. This technique makes it sophisticated to block or take down the C&C Server.

Networks that apply fast flux techniques are called fast fluxing network (FFN). Both legitimate or suspicious FFNs show almost the same characteristics, such as short TTLs and large IP pools [22]. Furthermore, fast flux can be classified into two categories: single flux and double flux.

In terms of single flux, a domain name may be resolved to different IPs in different time ranges. For example, a user accesses the same domain name twice in a short time period. For the first time, a bot sends a DNS query to the DNS server, which resolves that the corresponding IP address as IP_1. With IP_1 in place, the bot accesses a flux agent FA_1, which redirects the request to the real server "mothership". This "mothership" then processes the request and responds to FA_1. Finally, FA_1 relays the response back to the bot. After a short while, the same bot or other bots may access the same domain name again. However, the mapping between the domain name and IP_1 has been changed by hackers. As a result, a DNS server responses a different IP address, IP_2, to the name service request, and the bot uses this new address IP_2 to connect to another flux agent FA_2, which redirects the bot to the "mothership" [22].

The double flux is a more sophisticated method of counter detection compared with the single flux. It frequently changes both the flux agents and the registration in DNS servers. That is to say, in addition to fluxing their own agents, the authoritative domain name server is also a part of fluxing. This provides an additional layer of redundancy within malware networks. The fluxing nodes repeatedly register and de-register from the domain name system [21].

The fast fluxing network techniques have been abused by attackers to maintain their botnets. This is known as fast fluxing network attack (FFNA). In this case, almost all compromised computers become fluxing agents. Agents can be added or removed from the agent pool dynamically; thus, any mechanism that tends to block agents cannot take down the whole botnet [23].

2.2.2 Fast Flux Detection

Holz et al. [24] claimed that they were the first to develop a metric to detect fast flux service network (FFSN) empirically. They identified three possible parameters that could be used to distinguish normal network behaviors from that of FFSNs: the number of IP-domain mappings in all DNS lookups, the number of nameserver records in one single domain lookup, and the number of autonomous system in all IP-domain pairs. Based on these three parameters, they defined a metric, flux-score, which was a result of a linear decision function to detect FFSNs. A higher score indicated a higher fluxing degree, and vice versa. They evaluated their metric by a tenfold validation using a 2-month observation data set. Results showed that their method was able to distinguish normal network behavior from FFSN with a very low false positive probability.

There exist some limitations of these detection methods that focus on detecting domains that are related to IP addresses with short TTL in DNS query results [24,25]. In 2009, Zhou et al. [23] overcame these limitations by applying a behavior analysis model. To achieve this, they began with characterizing the behaviors of FF domains at some locations around these FF domains. Based on the analysis of those behaviors, they presented an analytical model, which showed the number of DNS

queries required to confirm an FF domain. In addition, they speeded up the detection by two schemes. The first scheme was to associate IP addresses with the queries' results from multiple DNS servers; the second scheme was to correlate queries' results with multiple possible FF domains. They also proved that the detection speed had been speeded up because of those correlation schemes. To avoid single point of failure and improve the scalability, they developed a collaborative intrusion detection architecture, LarSID, to support the distributed correlation using a peer-to-peer publish-subscribe mechanism for evidence sharing. Their results showed that their decentralized model was 16–10,000 times faster than previous centralized model [25] with the same correlation schemes [23].

Caglayan et al. [26] developed a real-time detection model for fast flux service networks (FFSN). They proposed to monitor the DNS activity of a web site at the minute level using both active and passive methods in a distributed fashion. The model included three key components: sensors, FF monitor database, and fast flux monitor (FFM). For the first key component, there were three kinds of sensors: active sensors, passive sensors, and analytic sensors. Active sensors were designed to monitor several indicating parameters including TTL, FF activity index, and footprint index. Passive sensors, however, were just functional replication of active FFM sensors by leveraging DNS replication services. Analytic sensors were mainly responsible for checking whether the IP addresses used by a certain web site existing in a blacklist.

The FF monitor database was designed to record information, such as known FFSNs, zombie IPs, collected from the sensors. By analyzing the data in the FFM database, some analytical knowledge was able to be harvested. For example, the size of FFSNs, growth rate estimation, social network of a FFSN where IP addresses were shared by diverse FFSNs, footprints of a FFSN for a given ISP in a given country, and so on. Finally, they developed a FFM classifier, which applied a Bayesian belief network to integrate multiple active and passive sensors. This classifier was then trained to calculate a prediction confidence. They demonstrated empirically how their model can generate report to assist security analysts to evaluate the security of a web site with acceptable accuracies. To improve the detection accuracy, the model calculated decisions every 10 min. The more sampling performed, the higher accuracy it obtains [26].

Perdisci et al. [27] developed a recursive DNS (RDNS) tracing methodology to detect malicious flux service networks in the wild. In their model, a sensor was deployed in front of the RDNS server, and passively monitored DNS traffic and stored information from a FF domains into a centralized data collector. Furthermore, to aim on botnets, they developed pre-filtering rules that were used to identify malicious FF networks. They considered a network as malicious FF network by four characteristics (short TTL, the change rate of the set of resolved IPs returned from each query, a large number of resolved IPs, and resolved IPs scattered across different networks). Based on these rules, they developed filters to gather the traffic they required. Besides, the filters in the sensors stored historic information. Different from most previous works, they conducted a fine-grained analysis on collected data. Firstly, they clustered domains with high relations based on their common features.

Secondly, the clusters of the domains were then classified according to the resolved IP addresses. Finally, they applied a statistical supervised learning method to build a network classifier to distinguish malicious flux services from legitimate ones. Results indicated that their model was able to distinguish and classify malicious and benign FF networks clearly.

Yu et al. [22] pointed out that one critical step of detecting botnet fast flux was to distinguish the fast fluxing attack network (FFAN) from benign fast fluxing service network (FFSN). Their idea was an improvement of the method discussed in [24], which was not able to distinguish benign ones from malicious ones. They identified FFAN by observing the agent lifespan. They showed that all agents in FFSN should keep alive almost 24/7. However, the alive time of FFAN bots is unpredictable to some extent, because the compromised computers cannot be controlled by FFAN master physically. Based on this lifespan difference between FFSN and FFAN, the authors proposed two metrics and developed a monitoring system. The first metric they defined was the average online rate (AOR), which was measured once per hour within a 24 h time interval. The AOR of FFSN should be close to 100 %. However, FFAN fluxing agents (bots) were out of attackers' control, thus the AOR was usually far bellow the AOR of FFSN. The second metric was the minimum available rate (MAR), which was the result of the number of times available out of the total measured times. For the same reason, the MAR of FFAN was far lower than that of FFSN. Based on the two metrics, they developed a flux agent monitoring system consisting mainly of four components. A digging tool was developed to gather information and add new IPs into IP record database. The second part was the agent monitor that sent HTTP request to the IPs in the IP record database and stored the responses. The third one was the IP lifespan record database storing the service status: 1 for available service, and 0 for unavailable service. The last key component was a detector differentiating FFAN from FFSN by IP lifepan records and the two metrics. Experimental results demonstrated that their system was able to distinguish FFAN from FFSN clearly because all benign FFSN had high and stable AOR and MAR, but that of the FFAN were much lower and less stable.

2.3 The Domain Flux Mechanism and Detection

Due to a single domain name, fast flux has a disadvantage of single point failure once fluxing is identified. Therefore, hackers developed a more survivable mechanism: domain flux.

2.3.1 The Domain Flux Mechanism

Stone-Gross et al. [3] pointed out that some recent botnet programs, such as Torpig, were using domain flux to sustain their botnets. Inspired by [28–30], they

discussed domain-generation techniques and provided a research report on how they cooperated with FBI to take over an advanced domain flux based botnet [3].

Domain flux is based on the idea of generating domain names through a domain generation algorithm (DGA). Both *C&C* server and its bots follow the same algorithm seeded by the same value to obtain consistent dynamic domain names. Bots try to contact the *C&C* server and other servers controlled by the botnet master according to a domain list until one DNS query succeeds. If the current domain has been blocked or suspended by authorities, bots will try to calculate other domain names using the DGA. The key idea is that the algorithm must make sure that all bots can generate domains by the same seed. Stone-gross and the co-authors revealed that Torpig calculated sub-domains using the current week and year first but independent of the current day, and then appended the top level domain (TLD). The domains generated might be "weekyear.com" or "weekyear.biz", and so on. At the same time, the bots will use these auto-generated domain names to contact the *C&C* server. If failed, bots will use the day information to calculate the "daily domain", such as "day.com" or "day.net", etc. If all these domains cannot be resolved, bots will try to use the hard-coded domain names in a configuration file as the last option [3].

2.3.2 Domain Flux Detection

It is critical for bots to phone "home" using the domain flux technique for botnet writers; it is also important for defenders to use this key component to defeat botnets. There have been sufficient work in domain flux detection and botnet takeover.

In 2009, Stone-Gross et al. conducted an in-depth research on taking over the real world Torpig botnet [3]. Taking advantage of reverse engineering on domain generation algorithm (DGA) of Torpig, they revealed that Torpig owners will not pre-register all possible domains in advance. Therefore, they registered the related domain name of *C & C* server of Torpig before the botnet owners. As a result, they took over the botnet for about 10 days. The bots of Torpig treated their server as the *C & C* server. The authors recorded many information about the botnet. They estimated the size of the Torpig by counting node identifiers (N_{id}) that were unique in Torpig. They also analyzed the advantages of the method by comparing to IP count that could be misled by DHCP. Their method was quite different from previous works in [7] and [31]. Rajab et al. [7] focused on detecting the size of IRC-based botnets by querying DNS server caches to estimate the number of bots who had connected the C&C server. Kanich et al. [31] worked on detecting the size of P2P storm networks using active probing and crawling the over-net distributed hash table (DHT).

Ma et al. [32] applied a supervised machine learning method to detect and prevent users from visiting malicious web sites based on automated URL classification. It was a lightweight model that investigated the lexical features and host-based properties of malicious URLs. The lexical features that they selected were the

http://www.springer.com/978-1-4614-9490-4

Distributed Denial of Service Attack and Defense

Yu, S.

2014, X, 97 p. 16 illus., 8 illus. in color., Softcover
ISBN: 978-1-4614-9490-4