A Survey of Bots Used for Distributed Denial of Service Attacks

Vrizlynn L. L. Thing, Morris Sloman, and Naranker Dulay
Department of Computing, Imperial College London,
180 Queen's Gate, SW7 2AZ, London, United Kingdom.
{vlt, mss, nd}@doc.ic.ac.uk
WWW home page: http://www.doc.ic.ac.uk

Abstract. In recent years, we have seen the arrival of Distributed Denial-of-Service (DDoS) open-source bot-based attack tools facilitating easy code enhancement, and so resulting in attack tools becoming more powerful. Developing new techniques for detecting and responding to the latest DDoS attacks often entails using attack traces to determine attack signatures and to test the techniques. However, obtaining actual attack traces is difficult, because the high-profile organizations that are typically attacked will not release monitored data as it may contain sensitive information. In this paper, we present a detailed study of the source code of the popular DDoS attack bots, Agobot, SDBot, RBot and Spybot to provide an in-depth understanding of the attacks in order to facilitate the design of more effective and efficient detection and mitigation techniques.

1 Introduction

In recent years, professionalism in Internet crime has advanced with the aid of open source attack tools, higher bandwidth connections and higher processing power of desktop workstations. Distributed Denial-of-Service (DDoS) attacks on high profile organizations are becoming prevalent and have received considerable media attention [1, 2]. A recent survey [3] of 36 tier 1, tier 2 and hybrid IP network operators in North America, Europe and Asia indicated that DDoS attacks remain the foremost concern for the large network operators, with 64% indicating that DDoS attacks are the most significant operational security issue they face.

Lately, DDoS attacks have been used by extortionists and business rivals against websites of banking and financial companies, online gambling firms, web retailers and government [4-8] to cripple their operations. These attacks are launched from a large pool of compromised computers in homes, education, business and government organizations. These compromised computers, referred to as bots, typically connect automatically to a remote Internet Relay Chat (IRC) server to enable remote control

by the attacker to form a *botnet* [9, 10]. Botnets are used for generating spam emails, viruses, worms as well as DDoS attacks.

In the past, typical botnet sizes were as large as hundreds of thousands [11, 12], but, a recent report [13] has shown botnets to have "slimmed" down to an average of 20,000 in order to be less visible and make detection more difficult. It also showed that blacklisted or worn-out botnets were being resold for DDoS attacks as these did not use email or viruses and so would not be caught by the blacklists or signature-based antivirus products. A relatively small botnet comprising a few thousand bots can seriously damage a victim's website or server as their combined bandwidth (e.g. 1000 x each uplink bandwidth of 128kbps = 125 Mbps) can be higher than the Internet connection bandwidth of many corporate systems.

Developing new techniques for detecting and responding to DDoS attacks often entails using attack traces to determine attack signatures and to test the techniques. However, obtaining actual attack traces can be very difficult, particularly for the latest attacks, because the high-profile organizations which are typically attacked will not release monitored data as it may contain sensitive information. In addition, they often do not want to publicly admit to being attacked as this can damage their reputation. Analysis of the way bots behave in terms of the types of attacks they can generate, how they generate data within an attack message, the target port addresses they attack, how they generate legitimate or spoofed source addresses, can be used to formulate attack signatures and anomaly detection algorithms.

In this paper, we present a detailed study of the source code of the popular DDoS attack bots. The availability of open source for bots and their modular design has led to thousands of variants of the popular ones which require very frequent updates of signature based anti-virus products to try to prevent infections and can outwit signature-based attack detection techniques. Analyzing the attack tools based on their source code enables a more in-depth understanding and presents a clearer picture of the attacks rather than studying the attack traces. We obtained the bot source code from hacker web and forum sites. We also discuss the implications of our findings on well-known DDoS mitigation techniques and emphasize the need to acquire an understanding of the attacks before being able to design and develop more effective and efficient mitigation techniques.

Section 2 of the paper presents the related work discussing botnets. In Section 3, we describe 4 popular DDoS bots, namely Agobot, SDBot, RBot and Spybot. In Section 4, we discuss our findings and the implications on DDoS mitigation techniques. Section 5 concludes the paper.

2 Related Work

The evolution of botnets has resulted in them becoming the latest most prevalent threat on the Internet and so has resulted in significant research in the network security community to develop detection and response techniques.

A Symantec white paper [14] discusses the design, coding and structure of the source code of popular bots and looks at how they have evolved with enhancement in network propagation, communication encryption and polymorphism. Observations

on botnet activities, collected using Honeypots and mwcollect is described in [15]. 180 botnets were tracked over 5 months to observe the coordinated activities within the botnets. Preventive mechanisms by identification of the activities and infiltration of the botnets to stop their operations, are proposed. In [16], an overview of the origins and structure of botnets is presented. It used data from the Internet Motion Sensor project [17] and Honeypot [18] to demonstrate the dangers of botnets due to their increase in number and their ability to exploit common system vulnerabilities such as the DCOM RPC [19] and LSASS [20]. Botnet detection by correlating data to pinpoint bots and botnet communications is also discussed. In [21], the authors studied the source code of popular bots and classified them according to their design and implementation characteristics, commands and control protocol, mechanisms to manipulate bots, propagation mechanisms, available vulnerabilities exploit, malware delivery mechanisms, obfuscation and detection evasion mechanisms. However, we could not find any existing reports providing a thorough understanding of the inner working and characteristics of the DDoS attack tools used in bots. Therefore, we conduct an in-depth study on these tools in this paper.

3 DDoS Bots

We studied the DDoS source code of 4 popular bots, namely Agobot, SDBot, RBot and Spybot [22-24] and present the details of the attacks in this section. These botnets have a few hundred to thousand variants due to multiple authors working to enhance the exploitation, propagation and communication code. We chose the version with the most advanced DDoS attack tools.

3.1 Agobot

Agobot is one of the most popular bots with the Anti-Virus vendor, Sophos [24], listing over 600 different versions. Variants of Agobot include Gaobot, Nortonbot, Phatbot and Polybot. The source code that we studied is the widely available "current" version of Phatbot, written in C++ and provides cross platform capabilities. The bot is structured in a modular way and allows new attacks to be easily added. Of all the bots studied, this has the most comprehensive set of DDoS attack tools, with the following attack commands:

- ddos.synflood <host> <time> <delay> <port>
- ddos.udpflood <host> <port> <time> <delay>
- ddos.httpflood <url> <number> <referrer> <delay> <recursive>
- ddos.phatsyn <host> <time> <delay> <port>
- ddos.phaticmp <host> <time> <delay>
- ddos.phatwonk <host> <time> <delay>
- ddos.targa3 <host> <time>
- ddos.stop

In all the above attacks, host is the IP address of the victim, time is the duration of the attack in secs, delay is the interval in msecs between sending attack packets,

and port is the victim's destination port. Other dynamic or attack specific parameters are presented as follows.

In the synflood attack, if port = 0, a random port number from 1000 to 10000 will be generated for each attack packet, otherwise, the one provided will be used. In the IP header, the identification (ID) field is set to 1 and the Time-to-Live (TTL) to 128. In the TCP header, the SYN flag is set and the window size is set to 16384. The TCP sequence number for each attack packet is formed by performing a binary OR on 2 32-bit randomly generated numbers (with one been left shifted by 16 bits). For each attack packet sent, each byte of the source IP address is randomly generated from 0 to 255 and the source port is randomly generated from 1000 to 2000.

In the udpflood attack, if port = 0, the destination port number will be randomly generated from 1 to 65535 for every attack packet. The 2-byte network prefix (i.e. x1.x2) of the source IP address (i.e. x1.x2.x3.x4) is initialized to that of the attacking node (i.e. local address y1.y2.y3.y4) or "255.255" if an error occurs while retrieving the local address. y2, y3 and y4 of the local address are stored in 3 counters and incremented in nested loops for each packet. The counter for y4 is incremented and reset to 1 after 254 and the counter for y3 is incremented. The counter for y3 is reset to 0 after 254 and the counter for y2 is incremented (also reset to 0 after 254 is reached). The above mentioned process is used to generate x2 of the source IP address. x1, which is equal to y1 remains the same for all the packets. x3 and x4 of the source IP address are randomly generated from 0 to 253 and 1 to 253, respectively. The data portion of the packet is 256 bytes and is filled with the character 'A'. The attack packet source port is a random number from 1000 to 2000.

In the httpflood attack, url is the web address to be accessed and number is the number of requests to be made to the specified address. Referrer is provided by the attacker and used in the http request. If the delay = 0, a random delay in the range of 1 msec to 24 hours is generated at the end of each cycle of a request (including recursive requests) for a URL. If recursive = 0, only the URL is accessed, otherwise, a recursive request on the page's resources is performed.

In the phatsyn attack, the destination port number is randomized from 0 to 65535 for each attack packet if the one provided is 0. The SYN and URG flags in the TCP header are set. For each attack packet, the ID and TTL fields in the IP header are randomly generated from 1024 to 65535 and 200 to 255, respectively, while the TCP source port, ACK number, window size and URP (offset for computing sequence number of last byte of urgent data) field are randomly generated from 0 to 65535. The TCP sequence number is formed by adding 2 randomly generated numbers from 0 to 65535 with one number left shifted by 8 bits. The 2-byte network prefix of the source IP address is set to that of the victim's, while the lower 2 bytes are randomly generated from 1 to 254 for each attack packet.

In the phaticmp attack, the destination port is hard-coded to 0. For each attack packet, the Type-of-Service (TOS), ID, more fragmentations, total length and TTL fields in the IP header are set to 4, 1234, 1, 0 and 255, respectively. The type and code fields in the ICMP header are randomly generated from 0 to 17 and 0 to 14, respectively. The 2-byte network prefix of the source IP address is set to that of the victim's, while the lower 2 bytes are randomly generated from 1 to 254.

In the phatwonk attack, 28 victim's ports (i.e. 1025, 21, 22, 23, 25, 53, 80, 81, 88, 110, 113, 119, 135, 137, 139, 143, 443, 445, 1024, 1433, 1500, 1720, 3306, 3389,

5000, 6667, 8000, 8080) are scanned to discover open ones. The port numbers of the open ports are placed in an array of size 28. The destination port to be used in the entire attack is chosen from the array of open ports or randomly generated if any of the entries is 0. The selection process ends after 28 iterations or when the attack duration time has expired, whichever happens first. 1 TCP SYN packet and 1023 TCP ACK packets are sent to the victim per inner loop for the attack duration. The TOS, ID and TTL fields in the IP header are set to 8, a random number from 1024 to 65535, and 255, respectively, each time the outer loop is run. In each run, the most significant 2 bytes of the source IP address are randomly generated from 1 to hexadecimal FFFE and the least significant 2 bytes are set to that of the victim's IP address. Assume that x1.x2.x3.x4 represents the IP address, then, x1.x2 remains the same as the victim's IP address while x3 ranges from 1 to 254 and x4 ranges from 0 to 255. The source port, window size, sequence number and data offset fields in the TCP header are set to a random number from 0 to 65535, 16384, an addition of 2 random numbers from 0 to 65535 with one been left shifted by 8 bits, and 5, respectively, for each run of the outer loop. In the inner loop, the IP ID field and the TCP sequence number are incremented for each of the 1024 attack packets.

In the targa3 attack, the destination port number is set to 666. The IP header protocol field and fragmentation offset field are randomly chosen from a set of 14 (i.e. 0, 1, 2, 4, 6, 8, 12, 17, 22, 41, 58, 255, random number from 0 to 254) and 10 integers (i.e. 0, 0, 0, 8192, 4, 6, 16383, 1, random number from 0 to 8099), respectively, for each packet sent. The last integer in each array is randomly generated. The source IP address has the 24-bit network prefix set to that of the attacking node. The last byte of the address is randomly generated from 0 to 254. The TOS, ID and TTL fields in the IP header are set to 4, a randomly generated number from 0 to RAND_MAX (based on the compiler) and 255, respectively.

Lastly, the stop command allows the synflood, udpflood and httpflood attacks to be stopped if they are running.

3.2 SDBot

SDBot is another popular bot with over 1800 variants. The widely available version is 0.5b, but only comes with ping and udp flooding tools, whereas the "SYN Flood Edition" includes TCP SYN flooding attacks. SDBot is written in C++ and targets Windows systems. The DDoS commands are as follow:

- udp <host> <number> <packet size> <delay> <port>
- ping <host> <number> <packet size> <delay>
- syn <host> <port> <time>

In the above attacks, host is the victim IP address, number is the number of attack packets to send, packet size is the size of each attack packet in bytes, delay is the interval in msecs between every attack packet sent and is set to 1 if < 1, port is the victim's destination port number and time is the duration of the attack in secs.

In the udp attack, the code restricts the port number to be from 1 to 65535. The data contents in the packets are filled with randomly generated bytes from 0 to 254. The actual size of the packet is randomized by subtracting a random number ranging from 0 to 9 from the packet size parameter provided, for each attack packet.

In the ping attack, the ICMP.DLL API is used. The ICMP Echo Request messages are used as the attack packets. The packet size is restricted to be ≤ 65535.

In the syn attack, the bot's registry entries and executables are removed from the system if a syn attack fails and the REMOVE_NONSYNNERS macro is defined. However, it is commented out of the code. The source IP address is initialized by adding the victim's address (as unsigned long integer) to 256 and a random number from 0 to 511. The ID and TTL fields in the IP header are set to 1 and 128, respectively. The SYN flag is set in the TCP header and the window size is set to 16384. For each attack packet, the source IP address is incremented by 1 and the TCP source port is randomly generated from 1000 to 2000. The TCP sequence number is formed by performing a binary OR on 2 randomly generated numbers (with one been left shifted by 16 bits).

3.3 RBot

RBot has over 1600 variants. It is also written in C++ and targets Windows systems. The version we studied is the one with the LSASS exploit and master password for scanning and compromising Optix servers. The DDoS commands include:

- ddos.syn/ddos.ack/ddos.random <host> <port> <time>
- synflood/syn <host> <port> <time>
- tcpflood/tcp <type> <host> <port> <time> [-r]
- icmpflood/Icmp <host> <time> [-r]
- pingflood/ping <host> <number> <size> <delay>
- udpflood/udp <host> <number> <size> <delay> <port>

In the above attacks, host, port, time, number, and delay have the same meaning as for the SDBot, size is the size of each attack packet in bytes, and if the optional parameter 'r' is provided, source IP address spoofing is used.

The ddos.syn, ddos.ack and ddos.random attacks exist in the same code module. For the ddos.syn attack, the ACK number in the TCP header is set to 0 and the SYN flag is set. For the ddos.ack attack, the ACK number in the TCP header is set to 0 and the ACK flag is also set. For the ddos.random attack, the ACK number in the TCP header is set to a random number from 0 to 2 and the SYN or ACK flag is set based on a probability of 0.5. In the attack packets, the ID and TTL fields in the IP header are set to 1 and 128, respectively. The source IP address is initialized by adding the victim's address (in the unsigned long integer format) to 256 and a random number from 0 to 511. It is then incremented by 1 for each packet. The TCP source port is randomized from 1000 to 2000 and the sequence number is formed by performing a binary OR on 2 randomly generated numbers (with one been left shifted by 16 bits). The TCP window size is set to 16384.

The synflood or syn attack code is based on the one in SDBot. However, the non-synners remover code is not implemented here.

In the tcpflood or tcp attack, the parameter type allows the attacker to specify a "syn", "ack" or "random" TCP attack. It has the same settings of flag and ACK number based on the attack type as in the ddos.syn/ddos.ack/ddos.random attacks. The ID and TTL fields in the IP header are set to 1 and 128, respectively. If the parameter 'r' is used, source address spoofing is performed. Otherwise, the real

source address of the attacking host will be used. The spoofed source IP address is generated by adding 4 randomly generated numbers with the 2nd, 3rd and 4th number left shifted by 8, 16 and 24 bits, respectively. Each number is in the range of 0 to RAND_MAX. The TCP source port is randomized from 0 to 1024 and the sequence number is set to the hexadecimal number 12345678. The TCP destination port is randomized from 0 to 1024 if port = 0. The TCP window size is set to 512.

In the icmpflood or icmp attack, the source IP address is generated similarly to that in the above tcpflood/tcp attack. The destination port is set to 0. The ID and TTL fields in the IP header are set to 1 and 128, respectively. For each attack packet, the ICMP type and code are random numbers from 0 to 255. The ICMP ID number is randomly generated from 1 to 240, and the ICMP sequence number is set to 1. The data portion is filled with bytes randomly generated from 0 to 254.

The pingflood or ping attack code is based on the one in the SDBot, and is similar in characteristics and functions used (i.e. ICMP.DLL API).

The udpflood or udp attack is similar to the one in the SDBot.

3.4 Spybot

Spybot is written in C and also affects Windows systems. It has over 200 variants currently and the version we studied is 2.0. It has more spreading abilities than the original version written by the author known as Mich. The DDoS commands are:

- syn <host> <port> <delay> <number>
- spoofdsyn <host> <port> <delay> <number>
- ping <host> <port> <delay> <number>

In the above attacks, the parameters have the same meaning as for the SDBot.

In the syn attack, socket connections are made to the victim and closed after the connection attempts and delay is forced to a minimum of 5 msec. The source IP address is not spoofed and the source port is randomly generated by the system.

In the spoofdsyn attack, delay is forced to a minimum of 5 msec. The ID and TTL fields in the IP header are set to 1 and 128, respectively. For each packet, each byte of the source IP address is randomized from 0 to 254. The SYN flag is set in the TCP header and the window size is set to 16384. The TCP source port is randomly generated from 1000 to 2000. The TCP sequence number is formed by performing a binary OR on 2 randomly generated numbers (with one been left shifted by 16 bits).

In the ping attack, the ICMP.DLL API is used. ICMP Echo Request messages are used as the attack packets. The destination port number is set to 65500 if greater and delay is forced to a minimum of 1 msec. Each byte of the data is set to the integer 37.

4 Analysis and Discussions

4.1 Bot Features

Most of the tools provide source IP address spoofing (either in whole or in part) and randomization of the source ports, destination ports, other header fields such as the

TCP sequence number, and the data contents of the attack packets. With a high degree of randomization, it makes mitigation such as dropping the traffic difficult due to the problem of accurately identifying the signature or pattern of the attack packets. However, if there is no restriction on the randomization of fields, this will result in more anomaly values appearing and so easing the detection of the presence of attacks. For example, performing partial source IP address spoofing reduces the randomness of the attack packets. However, if the addresses produced are within the safe range of legitimate addresses, this will reduce the chance of triggering a defense alarm mechanism. On the other hand, randomization without restrictions of destination port numbers or IP protocol types would raise alarms due to obvious anomalies such as hitting closed ports or unassigned network services. Thus, a high degree of randomization eases detection of the presence of an ongoing attack (e.g. through packet sampling). However, mitigation by means of checking the validity of each individual packet and dropping them are more difficult without a common identifiable signature.

All the attack tools that perform source IP address spoofing have different ways of forming the address from the randomly generated numbers. However, all of them prevent setting the final byte to 255 which will translate to a broadcast address. Source port randomization, though provided, is not really necessary as it will be randomly generated anyway if socket binding is not performed. Ranges of destination port numbers generated include 1000 to 10000, 0 to 65535, and 1 to 65535. However, some of these ports are still unassigned with only 0 to 1023 in the "Well-known ports" range. Therefore, most of these ports will most likely be closed at the victim. Randomization of the IP identification and fragmentation offset fields are most likely used to deter mitigation. However, it is not very useful since providing a value of 0 would allow the attack traffic to mix in well with the legitimate traffic as most Internet traffic does not require fragmentation. IP Time-to-Live field randomization could hide the actual hop counts traversed by the packets though it is not particularly useful since hop counts could not reveal the exact location of the attacking host anyway.

Agobot, SDBot and RBot all support SYN flood attacks, but RBot's ddos.random attack is the most dangerous SYN attack tool as it can randomly generate SYN and ACK packets thereby circumventing mitigation techniques which try to correlate TCP SYN and ACK according to the protocol characteristics. Next in line would be Agobot's ddos.phatsyn as it set the URG flag which allows the packet to have a high priority. When the TCP/IP stack at the server sees a packet with the URG flag set, it is duty bound to stop what it is doing and immediately send this packet to the server. RBot's ddos.syn and SDBot/RBot's syn simply provide standard SYN packets flooding with partially spoofed source IP address and randomized sequence numbers. The last five tools are Agobot's ddos.synflood and Spybot's spoofdsyn, which spoofs all 4 bytes of the source IP addresses, RBot's tcpflood syn and random, which fixes the TCP sequence number for all the packets to hexadecimal number 12345678, and Spybot's syn, which performs connection and disconnection attempts of sockets and does not provide source IP address spoofing.

For the UDP flood tools, we have the Agobot's ddos.udpflood and SDBot/RBot's udp. However, Agobot's ddos.udpflood filled its data with the character 'A' which simiplifies signature-based detection. SDBot/RBot's udp tool randomized its data but

only during initialization. It also randomizes its packet size but it has no source IP address spoofing. It is possible that future versions will combine the advantageous features of both tools while also randomizing the data contents for each attack packet which would make detection difficult.

For the ICMP flood tools, we have the Agobot ddos.phaticmp and RBot icmpflood. Agobot's ddos.phaticmp is slightly superior to RBot's as it limits its ICMP type spoofing from 0 to 17 and code spoofing from 0 to 14, instead of 0 to 255 in RBot. The Type-of-Service flag is set to 4 for route selection to maximize throughput if supported. In ICMP, type 1, 2 and 7 are not assigned and most types have no code at all. Spoofing an invalid type or type/code combination would therefore trigger the DDoS detection alarm. However, the chance of Agobot triggering an alarm is less than for RBot due to its type and code spoofing restrictions, though attack signature-based detection is slightly easier for Agobot due to this restriction and the fact that the identification field is fixed to the value of 1234.

SDBot/RBot and Spybot ping tools simply provide ICMP Echo Request messages flooding. It does not have any source IP address spoofing capability and is similar in function to a common ping, though Spybot's is distinguishable from its data contents which have the value 37. Agobot is the only one with HTTP requests flooding tool to emulate legitimate requests of resources from web servers. Source IP address spoofing is not used since information of subsequent resources to be retrieved has to be known to continue the recursive attacks. It also makes the attack indistinguishable from legitimate requests.

Agobot's ddos.phatwonk attack has the advantage of scanning for a list of ports to check if they are open before attempting to flood it with SYN and ACK packets. However, a balance of 0.5 probability of generating either SYN or ACK packets would reduce anomalies rather than the 1 SYN followed by 1023 ACK packets for each round of flooding. In Agobot's targa3 attack, the destination port is set to 666, which is the designated port for a popular multiplayer PC game, Doom. However, the list of IP protocol types to use will raise anomalies as only the TCP and UDP network services are typically supported for port 666. RBot's ddos.ack and tcpflood ack attack tools simply flood the victim with TCP ACK packets. However, RBot's tcpflood ack has the same disadvantage as its tcpflood syn and random whereby the TCP sequence number is set to hexadecimal number 12345678.

The main purpose of the above analysis and discussion is not to advise on how to enhance attack tools to circumvent current mitigation techniques, but to raise the awareness to the network security research community that making changes to improve on the attack tools is possible and easy. When designing and developing network security products, we have to bear in mind the need to foresee the future attacks that the attackers would be able to come up with to "challenge" the mitigation techniques and systems.

4.2 Implications on Mitigation Techniques

We see that source IP address spoofing remains a security issue. Although it could be postulated that since bots are used in current DDoS attacks, tracing the source of

attack does not lead to the attack controller, so source spoofing is not really needed. However, managing bots is not an easy task for attackers and often attackers maintain ownership of their botnets to rent out for fees. Therefore, they want to make sure the bots are not traced and neutralized. Thus, source address spoofing is still used in the DDoS attack tools to deter detection. Ingress filtering removes any traffic from a customer site to the Internet which has invalid source addresses i.e. not within the range allocated to the customer. Egress filtering on traffic from the Internet to a customer site discards traffic with "illegitimate" source addresses such as private/reserved IP addresses or addresses within the domain of the customer site. Although ingress and egress filtering [25, 26] is performed, it is not universally applied and so does not completely prevent DDoS attacks with spoofed source addresses. In addition some attack tools circumvent this filtering by spoofing source addresses from within the network of the bot. [27] is a technique used to infer hop count information from the Time-to-Live value in the IP header to determine if source IP address spoofing has been performed and thus detect if the traffic is legitimate or not. However, in the case of internal source address spoofing, it would fail to tell the difference since the hop count would not differ greatly from the legitimate source. Backscatter analysis [28] also proves that source address spoofing is indeed still widely used in current attacks while [29] shows that spoofing remains a serious problem to Internet security.

In [30], a DDoS TCP SYN flooding detection mechanism, SYN-dog, was proposed based on the protocol behavior of the TCP SYN – SYN/ACK pairs to detect source IP address spoofing, which is used in TCP SYN flood attacks. The non-parametric Cumulative Sum (CUSUM) method [31] was applied to make the scheme insensitive to site and access pattern. SYN-dog was meant to be implemented near the flooding sources as with a spoofed source address, a TCP SYN packet sent out to a server would not result in receiving a SYN/ACK packet. However, we noticed in the attack source code that it is possible for attackers to send out randomized SYN or ACK packets, imitating the three-way handshake. Therefore, this mechanism will not work at the victim end as there is unlikely to be much variation in the number of SYN and ACK packets seen by the victim or within the network.

5 Conclusion

In this paper, we presented a detailed study of the functionalities of the popular DDoS attack bots, namely Agobot, SDBot RBot and Sybot. We found that analyzing the attack tools based on their source code to give an in-depth understanding of the attacks is better than studying attack traces which are difficult to obtain. The information presented on the attack tools can be used to design both detection and attack mitigation techniques.

One of the most important characteristics is the degree of randomization of addresses, protocol fields and data contents. Greater randomization can ease detection as more anomalies are generated but can make mitigation more difficult as specifying packet signatures for filtering becomes harder. We have also given a

comparison between the attack tools in the bots and provided a view of possible enhancements on the tools in the foreseeable future.

We have shown that well-known DDoS mitigation techniques can be easily bypassed. For example, partial source IP address spoofing circumvents ingress and egress filtering and hop count filtering. Randomization of SYN and ACK packet generation makes some SYN flood detection mechanisms ineffective. Therefore, we see the need to acquire an understanding of the attacks before being able to design and develop more effective and efficient mitigation techniques.

As the modular design and open source nature make modifications and implementation of additional features easy for the bot authors, there will always be a race between the attackers, and network security providers to see who can be the most innovative. Therefore, it is important that network security products are able to get a grasp on the latest attack tools in use today and possibly in the future, and incorporate learning techniques and adaptive mechanisms to provide timely responses to the new variants of attack tools.

6 Acknowledgements

We gratefully acknowledge the support from the EU funded Diadem Distributed Firewall FP6 IST-2002-002154. This research is continuing through participation in the International Technology Alliance sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence.

7 References

1. Diane E. Levine and Gary C. Kessler, "Chapter 11 - Denial of Service Attacks, Computer Security Handbook, 4th Edition", Editors - Seymour Bosworth, Michel E. Kabay, 2002.
2. K. J. Houle and G. M. Weaver, "Trends in Denial of Service Attack Technology", Oct. 2001, CERT Coordination Center, http://www.cert.org/archive/pdf/DoS_trends.pdf.
3. Arbor Networks, "Worldwide ISP Security Report", Sept. 2005.
4. Federal Bureau of Investigation, "THE CASE OF THE HIRED HACKER: Entrepreneur and Hacker Arrested for Online Sabotage", http://www.fbi.gov/page2/april05/hiredhacker041805.htm, Apr. 2005.
5. Dawn Kawamoto, "Blackmailers try to black out Million Dollar Homepage", CNET News, http://news.zdnet.com/2100-1009_22-6028131.html, Jan. 2006.
6. BBC Technology News, "Hacker threats to bookies probed", http://news.bbc.co.uk/1/hi/technology/3513849.stm, Feb. 2004.
7. Ashlee Vance, "Man admits to eBay DDoS attack", http://www.theregister.co.uk/2005/12/28/ebay_bots_ddos/, Dec. 2005.
8. Jan Libbenga, "Dutch hackers sentenced for attack on government sites", The Register, http://www.theregister.co.uk/2005/03/16/dutch_hackers_sentenced/, Mar. 2005.
9. Basudev Saha and Ashish Gairola, "Botnet: An Overview", CERT-In White Paper, CIWP-2005-05, Jun. 2005.

10. Laurianne McLaughlin, "Bot Software Spreads, Causes New Worries", IEEE Distributed Systems Online, Jun. 2004.
11. Drew Cullen, "Dutch smash 100,000-strong zombie army", http://www.theregister.co.uk/2005/10/07/dutch_police_smash_zombie_network/, Oct. 2005.
12. Joris Evers, "'Bot herders' may have controlled 1.5 million PCs", ZDNet News, http://news.zdnet.com/2100-1009_22-5906896.html, Oct. 2005.
13. Dawn Kawamoto, "Bots slim down to get tough", CNET News, Nov. 2005.
14. John Canavan, "The Evolution of Malicious IRC Bots", Virus Bulletin Conference, Oct. 2005.
15. Felix C. Freiling, Thorsten Holz, and Georg Wicherski, "Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks", 10th European Symposium on Research in Computer Security (ESORICS 2005), Sept. 2005.
16. Evan Cooke, Farnam Jahanian, and Danny McPherson, "The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets", USENIX SRUTI: Steps to Reducing Unwanted Traffic on the Internet Workshop, Jul. 2005.
17. Michael Bailey, et al., "The Internet Motion Sensor: A distributed blackhole monitoring system", Network and Distributed System Security Symposium (NDSS), Feb. 2005.
18. The Honeynet Project, "Know you enemy: Tracking botnets", https://www.honeynet.org/papers/bots/, Mar. 2005.
19. Microsoft, "DCOM RPC vulnerability", http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx, Jul. 2003.
20. Microsoft, "LSASS vulnerability", http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx, Apr. 2004.
21. Paul Barford and Vinod Yegneswaran, "An Inside Look at Botnets", To appear in Series - Advances in Information Security, Springer, 2006.
22. McAfee Threat Center, http://vil.nai.com.
23. Symantec, http://www.symantec.com.
24. Sophos, http://www.sophos.com.
25. T. Killalea, "Recommended Internet Service Provider Security Services and Procedures", IETF BCP 46, RFC 3013, Nov. 2000.
26. P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", IETF BCP 38, RFC 2827, May 2000.
27. Cheng Jin, Haining Wang, and Kang G. Shin, "Hop-count filtering: an effective defense against spoofed DDoS traffic", 10th ACM Conference on Computer and Communications Security, Oct. 2003.
28. David Moore, et al., "Inferring Internet Denial-of-Service Activity", ACM Transactions on Computer System (TOCS), May 2006, **24**(2), pp. 115-139.
29. Robert Beverly and Steven Bauer, "The Spoofer Project: Inferring the Extent of Source Address Filtering on the Internet", USENIX SRUTI: Steps to Reducing Unwanted Traffic on the Internet Workshop, Jul. 2005.
30. Yu-Shun Wang, Danlu Zhang, and Kang G. Shin, "SYN-dog: Sniffing SYN Flooding Sources", 22nd IEEE International Conference on Distributed Computing Systems, Jul. 2002.
31. B. E. Brodsky and B. S. Darkhovsky, *"Nonparametric Methods in Change-point Problems"*. 1993: Kluwer Academic Publishers.