International Journal of Communication Networks and Security

Volume 1 | Issue 2 Article 13

July 2011

An Implementation Of Botnet Detection Algorithm For Grid Networks

G.Pradeep Reddy
JNTUA COLLEGE OF ENGINEERING ANANTAPUR, ANDHRA PRADESH,INDIA,
pradeepreddy006@gmail.com

A.Ananda Rao
JNTUA COLLEGE OF ENGINEERING ANANTAPUR, ANDHRA PRADESH,INDIA, akepogu@gmail.com

Recommended Citation

Reddy, G.Pradeep and Rao, A.Ananda (2011) "An Implementation Of Botnet Detection Algorithm For Grid Networks," *International Journal of Communication Networks and Security*: Vol. 1 : Iss. 2 , Article 13.
DOI: 10.47893/IJCNS.2011.1024
Available at: https://www.interscience.in/ijcns/vol1/iss2/13

An Implementation Of Botnet Detection Algorithm For Grid Networks

G.Pradeep Reddy, A.Ananda Rao
JNTUA COLLEGE OF ENGINEERING
ANANTAPUR, ANDHRA PRADESH,INDIA
E-mail : pradeepreddy006@gmail.com, akepogu @gmail.com

Abstract—Grid is an emerging technology that aims at utilizing resources efficiently and effectively, A botnet is a collection of infected computers and the common attacks are A Distributed denial of service attack (DDOS) is any type of attack on a networking structure to disable a server from servicing its clients. Attacks range from sending millions of requests to a server in an attempt to slow it down, flooding a server with large packets of invalid data, to sending requests with an Invalid or spoofed ip address.

A botnet is taking action on the client itself via IRC Channels without the hackers having to login to the clients computer. In this paper we show the implementation and analysis of three main types of attack: Ping of Death, TCP SYN Flood, and Distributed DOS. The Ping of Death attack will be simulated against a Microsoft Windows xp, computer. The TCP SYN Flood attack will be simulated against a Microsoft Windows 2007 IIS FTP Server. Distributed DOS will be demonstrated by simulating a distribution zombie program that will carry the Ping of Death attack.

This paper focuses on improving the efficiency of the system performance over the network by implementing algorithm, It demonstrate the potential damage from DOS attacks and analyze the ramifications of the damage.

Keywords-Network Security, DDoS Attacks, Collaborative Change-Point Detection, Internet Infrastructure, Collaboration Grids, Community Networks, Peer-to-Peer Systems, and Internet Service Provider, Denial of Service Attack (DOS), TCP SYN Flood.

I. INTRODUCTION

Community networks and Grid systems can be large or small, ranging from local-area to wide-area networks. They form the backbone infrastructure for building multi-site computing clusters, collaboration Grids, P2P systems, web services, enterprise Grids, or any ISP-based core networks for community services. Community networks and collaboration Grids are often formed under a federation of IT administrators. Cooperative computing and high degree of resource sharing are expected in such networked systems.

Denial of services attacks (DOS) is a constant danger to web sites. DOS has received increased attention as it can lead to a severe lost of revenue if a site is taken offline for a substantial amount of time. There are many types of denial of service attacks but two of the most common are Ping of Death and TCP SYN Flood. We have chosen to implement these two techniques and add Distributed DOS (DDOS) as well.

In a Ping of Death attack, a host sends hundreds of ping requests (ICMP Echo Requests) with a large or illegal packet size to another host in attempt to knock it offline or to keep it so busy responding with ICMP Echo replies that it cannot service its clients.

A TCP SYN Flood attack takes advantage of the standard TCP three-way handshake by sending a request for connection with an invalid return address.

In this paper we demonstrate DDOS by creating a worm like program that installs programs on remote machines to attack a particular server. These attackers listen in the background for a message from a master program that will tell these attackers to launch a DOS attack against a machine. DDOS attacks are difficult to stop because they can be coming from anywhere in the world. We will implement a DDOS attack by launching the Ping of Death implementation against a victim computer from several other workstations.

We propose a collaborative change-detection scheme to solve this problem. Using the NS-2 simulator, we carried out intensive experiments to verify the effectiveness of our new DDoS defense system. Under different type of flooding attacks with variant flooding rate, our scheme is capable of detecting the start of DDoS attack quickly with high accuracy. Another impressive advantage is the small false positive alarm rate experienced. Treating Internet traffic as stochastic process, sequential change-point detection technique was developed to detect the start of flooding DDoS attacks The typical change point detection methodologies are hindered by lack of accurate statistical model to describe the pre-change and post-change traffic distributions.

This paper is organized as follows. Section -2 provides provides information about *TCP SYN FOOD IMPLEMENTATION* and Section -3 provides FLOODING PATTERENS OF DDOS ATTACKS, Section-4 provides algorithmic approach that we can avoid the botnet from the multiple vendors so we propose an Attack Pattern Recognition algorithm which reduces the threats and it increases the efficiency of the vendor specific internet traffic. Finally section-5 provides MULTI-DOMAIN DDOS ATTACK DETECTION.

II. TCP SYN FOOD IMPLEMENTATION

When hosts need to establish communications via the TCP transport protocol, they must do a session initiation, which consists of a three-way handshake:

1. The source host initiated the communication by sending a TCP packet to the destination host the SYN flag (SYNchronize sequence numbers) set to 1. In this packet reside the source IP address and port number as well as the destination IP address and port numbers (in addition to several other fields which are inconsequential for this discussion).

2. The destination host responds by sending a TCP packet to the source host with the flags SYN and ACK (ACKnowledge) set to 1. The response is sent to the source IP address and port of the initial packet in step 1.

3. The source host sends the destination host another TCP packet with the ACK flag set to 1. This completes the 3-way handshake and normal data communication can start.

In a TCP SYN Flood attack, the source (attacker) host simply fails to complete step 3 leaving the destination (victim) host with an unfinished communication session. When the victim's TCP socket receives the message in step one, it allocates buffers, increments counters, initiates timers, and increases communication stacks in preparation for the communication that is to follow. In addition, processor time is spent building the reply packet (step 2) and sending it back.The attacker can overwhelm the victim's computer resources by sending a "flood" of packets with the SYN flag set to 1 (step 1) and never bothering returning any response (step 3).The TCP SYN Flood attack
Implemented is the Neptune algorithm and implementation. In this algorithm, not only is step 3 of the TCP handshake ignored, the source address in the SYN packet of step 1 is set to an unreachable destination (for example a non-routable IP address). IP spoofing is used in this implementation therefore; it is virtually impossible to track the origin of the packet since the return address is fake. The victim's computer now expends time to try to deliver a packet to an inexistent destination. IP spoofing is used in this implementation therefore; it is virtually impossible to track the origin of the packet since the return address is fake. The victim's computer now expends time to try to deliver a packet to an inexistent destination. The Neptune implementation also allows the attacker to specify a specific service to deny. In a classical TCP SYN Flood attack, the attacker generally tries to prevent the victim's computer from servicing any legitimate requests. The Neptune implementation however, allows the attacker to choose a specific TCP service port to overwhelm. In other words, the attacker can choose to bring down only a web server for example (port 80). A simulation for an attack on a FTP server running Windows 2000 IIS FTP has been tested. small TCP SYN Flood attack against an FTP server (IP address 148.166.161.115). In this particular attack, only three SYN flood packets were sent (Nos. 1, 3, 5) against an FTP server (port 21 destination). For each of the packets, the server replies with and ACK-SYN packet which in turn ends up nowhere (Nos. 2, 4, 6). The server then retries to send replies a further two times before giving up (Nos. 7-12). When looking closer at the actual packets we can see the spoofed packet clearly with the SYN flag set to 1 and the spoofed source IP address of 10.10.1.1. Similarly, the return packet is destined for nowhere and has the ACK and SYN flags set to 1. The application also takes care of using different source port numbers and sequence numbers. This prevents the victim's computer from assuming that packets all come from the same client in the same host. By changing the return port and sequence numbers, a single computer can force another host to allocate several connection resources.

B1. Neptune Algorithm (Extracted from source code)

```
                * IP address information */
struct sockaddr_in sin;
register int i=0,j=0;
int floodcontrol=0;
unsigned short sport=161+getpid();
* Build TCP header */
packet.tcp.source=sport;   /* 16-bit Source port number
*/
packet.tcp.dest=htons(dport);    /* 16-bit Destination
port */
packet.tcp.seq=49358353+getpid();  /* 32-bit Sequence
Number */
```

```
packet.tcp.ack_seq=0;   /* 32-bit Acknowledgement
Number */
 packet.tcp.doff=5;    /* Data offset */
 packet.tcp.res1=0;    /* reserved */
 packet.tcp.urg=0;    /* Urgent offset valid flag */
 packet.tcp.ack=0;    /* ACK flag */
 packet.tcp.psh=0;    /* Push flag */
 packet.tcp.rst=0;    /* Reset flag */
 packet.tcp.syn=1;    /* SYN flag */   packet.tcp.fin=0;
/* Finish sending flag */
 packet.tcp.window=htons(242);   /* 16-bit Window
size */
 packet.tcp.check=0;   /* 16-bit checksum (to be filled
in below) */
 packet.tcp.urg_ptr=0;   /* 16-bit urgent offset */
 /* Build IP header */
 packet.ip.version=4;   /* 4-bit Version */
 packet.ip.ihl=5;    /* 4-bit Header Length */
 packet.ip.tos=0;    /* 8-bit Type of service */
 packet.ip.tot_len=htons(40);   /* 16-bit Total length */
 packet.ip.id=getpid();   /* 16-bit ID field */
 packet.ip.frag_off=0;   /* 13-bit Fragment offset */
 packet.ip.ttl=255;    /* 8-bit Time To Live */
 packet.ip.protocol=IPPROTO_TCP;   /* 8-bit
Protocol */
 packet.ip.check=0;    /* 16-bit Header checksum (filled
in below) */
 packet.ip.saddr=sadd;   /* 32-bit Source Address */
 packet.ip.daddr=dadd;   /* 32-bit Destination Address
                          */
```

III. FLOODING PATTERN OF DDOS ATTACKS

A DDoS attack deploys multiple attacking entities to deny legitimate application from obtaining a service. The DDoS attacks overwhelm the target host and associated network links with extraordinary huge amount of packets that the victims are incapable to handle. Legitimate traffic is simply blocked. Such brute force attacks do not rely on particular network protocols or system weakness.

As shown in Fig. 1, the attacker simply exploits the huge resource asymmetry between the Internet and the victim. The magnitude of the increased traffic is large enough to crash the victim machine by resource exhaustion, or jam its Internet connection by bandwidth exhaustion, or both Therefore, DDoS attacks can effectively take the victim off the Internet. To avoid being caught by trace back techniques, attackers launch attacks using spoofed IP addresses form innocent victims.

To overwhelm the victim, DDoS flows converge toward the victim host. Therefore, we can observe abnormal traffic volume changes on routers along the paths of aggregation. The spatio-temporal traffic pattern tends to form a tree rooted the last- hop router to the edge network where the victim resides. By recognizing such tree-like attack patterns at each end router, we can detect the DDoS attacks.

At the early stage of DDoS attack, the abnormal changes are not obvious at each router due to the huge data rate in the core network. Meanwhile, routers cannot afford to monitor traffic on flow or packet level. We define a traffic flow by a set of packets satisfying a 5-tuple qualifier: {source IP address, destination IP address, source port, destination port, protocol applied} during a given observation window. Thus, such a flow is observable by the router.

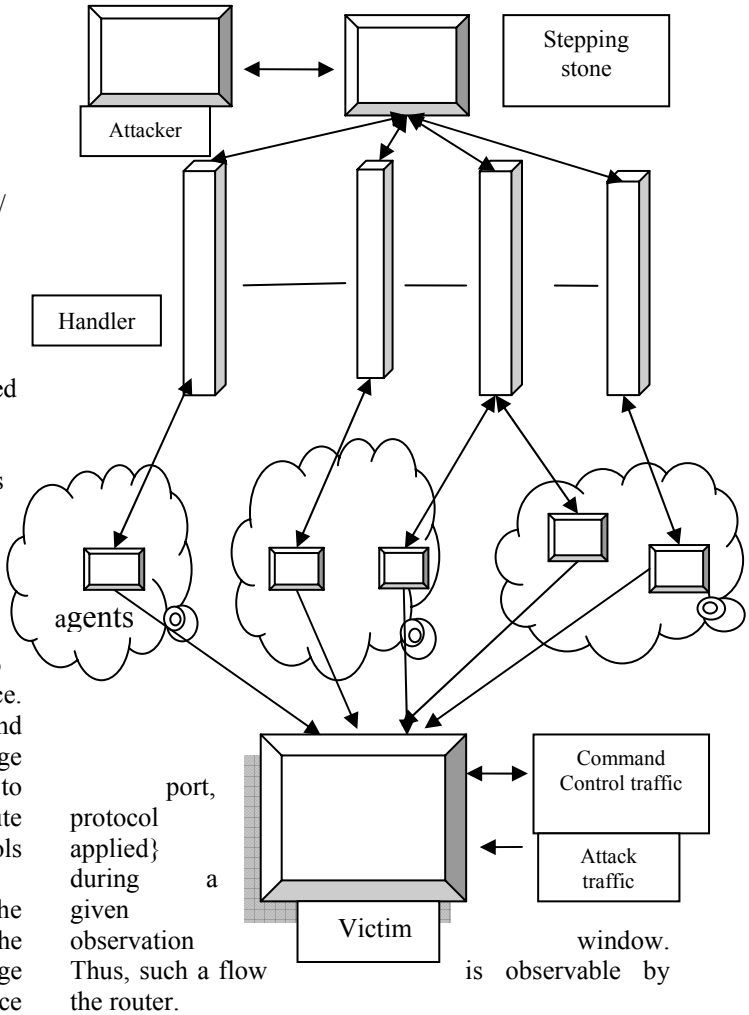

FIG 1. Traffic pattern of a typical DDoS attack.

III.ATTACK PATTERN RECOGNITION ALGORITHM

Algorithm 1: Attack Pattern Recognition

Input: x(t,i): Incoming packet in time slot t at port i

y(t,i): Outgoing packet in time slot t at port i

$\bar{x}(t-1,i)$ Average of packet arrivals up to time t-1 at port i)

$\bar{y}(t-1,i)$: Average of outgoing packets up to time t-1 at port i

Output: Alert packets sent to central CAT server.

Procedure:

01: Update historical average of I/O packets in a flow
02: Calculate DFAin and DFAout using Eqs. (1) and (3)
03: If DFAin > threshold Then
04: Calculate DR and OR using Eqs. (5) and (6)
05: If DR ≈ 1 Then
06: If OR ≈ 1 Then
07: Suspicious pattern detected, alert packet sent;
08: Else if OR > 1 Then
09: Suspicious pattern detected, alert packet sent;
10: End If
11: Else if DR < 1 AND OR ≈ 1 Then
12: Suspicious pattern detected, alert packet sent;
13: End If
14: End If

a. DR ≈ 1 and OR ≈ 1: The flow cuts through the router.
The router essentially forwards all increased traffic.

b. DR < 1 and OR ≈ 1: The outgoing flow merges multiple incoming flows, but not all incoming flows
contain abnormally increased packets. As all of them
are forwarded out through port iout, this is a partial
aggregation pattern .

c. DR ≈ 1 and OR > 1: The outgoing flow merges multiple incoming flows, each incoming flow contains abnormal increases with same deviation rate and they aim at the same destination. The router is a merge point on the attacking path and it is a full aggregation pattern .

d. DR < 1 and OR < 1: The changes are scattered, so it is not part of a DDoS attack.

IV. MULTI-DOMAIN DDOS ATTACK DETECTION

We need to extend the scale of Server-based DDoS detection to multiple network domains. Inter-domain communication is thus needed in the alert aggregation process. The Attack Pattern recognization algorithm is to perform wide-area network anomaly detection. We must reach agreement to resolve conflicts between security policies applied in different domains. The routers at various domains exchange alert packets under agreed terms. The idea of cross-domain DDoS defense is illustrated in Fig.2.
Multiple servers at different AS domains must be protected by dedicated VPN channels among them. The alert packets generated by ISP routers from different domains may follow different policies and data formats. Multiple CAT servers must work together to resolve the conflicts. We will reveal the performance attributes tied to policy fusion methodology applied. It was suggested to approach the policy conflict problem through trust negation[5].

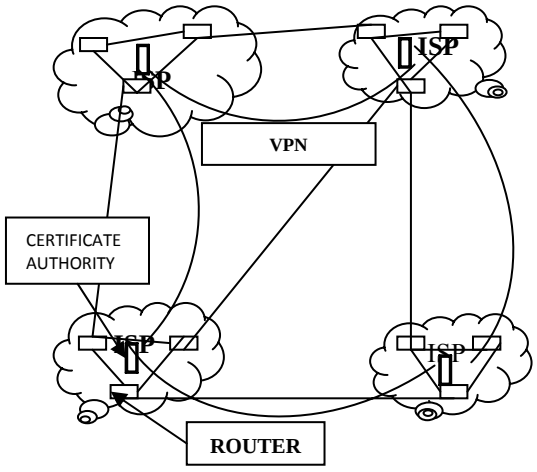

Figure 2- Multiple servers in several ISP domains communicating with each other to resolve the conflicts in security policies

V. CONCLUSIONS AND FURTHER WORK

The complexity of DDoS attack patterns grows fast, as new network vulnerability is identified and more sophisticated attack tools are available. There is no magic that can handle all types of DDoS attacks. The shared sources in collaboration Grids and community networks are especially prone to such attacks. One solution works well in a given

network environment but may fail in other networks.

All the implementations done in these simulations consist of very simple and light loaded attacks, which can cause severe amounts of damage. DOS attacks can be stealthy covert and easily delivered. The Neptune implementation for example, is only 10Kbytes in size and can cause devastation to a service. When combined with the power of a DDOS attack, Denial of Service is a truly powerful attack. Although our implementations are not sophisticated, they serve as examples of what such programs can do and the damage they can cause.

VI. References

1]. Anderson, T., R. Mahajan, N. Spring, and D. Wetherall, "Rocketfuel: An ISP Topology Mapping Engine," http://www.cs.washington.edu/research/ networking/rocketfuel/, Feb. 2006

[2] Blazek, R., H. Kim, B. Rozovskii, and A. Tartakovsky, "A Novel Approach to Detection of Denial-of-Service Attacks via Adaptive Sequential and Batch-sequential Change-Point Detection Methods," Proc. of the 2001 IEEE Workshop on Information Assurance and Security, June 2001.

[3] Berman, F., G. Fox, and A. Hey (editors), Grid Computing, John Wiley, England, 2003.

[4] Monk, T. and K. Claffy, "Cooperation in Internet Data Acquisition and Analysis," Coordination and Administration of the Internet Workshop, Cambridge, MA., Sept. 8-10, 1996. (CAIDA Project, http://www.caida.org/)

[5] Cai, M., K. Hwang and Y. Chen, "Hybrid Intrusion and Anomaly Detection with Weighted Signature Generation", IEEE Trans. On Dependable and Secure Computing, revised Sept. 2005.

[6] Frank Kargl, Joern Maier, Michael Weber, "Protectingweb servers from distributed denial of service attacks," Proceedings of the tenth international conference on World Wide Web, April 2001, pp. 514.

[7] Errin Fulp et. al. "Preventing Denial of Service Attacks on Quality of Service," DARPA Information Survivability Conference and Exposition (DISCEX II'01)Volume II-Volume 2 , June 2001, pp. 1155.

[8] Gresty, Q. Shi, M. Merabti, "Requirements for a General Framework for Response to Distributed Denial-of- Service," 17th Annual Computer Security Applications Conference (ACSAC'01), December 2001, pp. 422.