A Measurement Based Detective Method for SYN Flood Attacks

Takuo Nakashima†, Shunsuke Oshima‡ and Kazuki Miura†
† Department of Information Science, Kyushu Tokai University
9-1-1 Toroku, Kumamoto, Japan
{taku@ktmail,30iis147@stmail}.ktokai-u.ac.jp
‡ Yatsushiro National College of Technology
Information and Electronic Engineering
2627, Hirayama-Shinmachi, Yatsushiro, Kumamoto, Japan
oshima@as.yatsushiro-nct.ac.jp

Abstract

DoS(Denial of Service) attacks are easily performed by utilizing the weakness on TCP at the connection establishing phase. If an attacker sends SYN packets with a spoofed source IP address, the server should remain the half-open state for each connection. This attack called the SYN flood attack is hardly filtered by the router in such a case that the IP address is spoofed. Early detection of this SYN flood attacks as well as identification of the mechanism of escaping from the half-open state on TCP is required. In this paper, we present a measurement based detective method for SYN flood attacks at an early stage. We implemented a program to send the SYN packet and collected the SYN+ACK and RESET response packet from servers on different OS's. Our detective procedure takes the following steps. Firstly, Our method builds a standard model generated by observations for the specified metrics of the server activated by our attacking program. We capture the response packet to determine the metrics of standard model and find the threshold value of each metric to identify whether the server is attacked or not. Secondly, the packet response rate and the average response delay are adopted as the metric on FreeBSD platform, while the packet loss rate is adopted on Linux platform. Finally, we detect the slight variations of response packet, if the value exceeds the pre-determined threshold value, then the detective host executes to send the RST packet releasing the half-open state on TCP.

1 Introduction

DoS attacks are easily performed by utilizing the weakness of the network protocol and by iterating requests of service for the application. Most organization have opened their Web sites and other ports on TCP to maintain their sites. So, these attacks aim at directly the application of Web server or TCP protocol to suspend their Internet services. In a DDoS (Distributed Denial of Service) attack, the assault is coordinated across many hijacked systems by a single attacker[1]. SYN flood attacks [2][3] disturb the establishment of the TCP connection. An attacker does not respond to the SYN+ACK packet from the server following a huge amount of SYN packets sent to the server from the attacker with spoofed source IP addresses. As a result, TCP on the server should keep the huge number of the half-open state for each connection and exhaust the memory resource to be followed by the cease of server function to be down eventually.

To prevent DDoS attacks, three different types of method are classified: prevention, detection and counterattack. Prevention method tries to prevent attacks based on the preemptive measurement to built the tolerant system. Detection method focuses on early detection for intrusions or attacks and focuses on notification by the alarm as soon as possible. In this method, accuracy and quickness are important factors. Counterattack method tries to some actions after detecting the attack. Types of these actions are included the filtering[4], pushback[5] and traceback[6][7], and mitigate the influence of the DDoS attacks and finally tries to identify the attack source if possible. The method we propose is mainly categorized into the detective method, and in addition, we support mitigation to exhaust resources of the server.

In this paper, we present a measurement based detective method for SYN flood attacks in early stage.

We implemented a program to send the SYN packet and collected the SYN+ACK response packet from the server. Our detective procedure takes the following steps. Firstly, Our method builds a standard model generated by observations for the specified metrics of the server activated by our attacking program. We capture the response packet to determine the metrics of standard model and find the threshold value of each metric to identify whether the server is attacked or not. Secondly, the packet response rate and the average response delay are adopted as the metric on FreeBSD platform, while the packet loss rate is adopted on Linux platform. Finally, we detect the slight variations of response packet, if the value exceeds the pre-determined threshold value, then the detective host executes to send the RST packet releasing the half-open state on TCP.

This paper is organized as follows. First, the experimental setup and the packet sequence pattern is described in Section 2. Section 3 is the results of observations followed by the detective method in Section 4. Section 5 is the summary and discussion for future work.

2 Experimental Setup and Packet Sequence

In this section, we introduce explanation of the experimental setup followed by packet sequence patterns.

2.1 Experimental Setup

Our experiments configure the attacking machine as a client, and observe receiving and sending packets from and to the server to infer the performance dynamics on the server. The client operates on the FreeBSD 5.3 platform as the attacker, while the server operates one of the major different OS's, i.e. FreeBSD 5.3, Linux FedoraCore 5 (Kernel version 2.6.16) and Windows Server 2003 SE SP1. We implemented packet generating program by C and set up this program on the client. The client sends packets sequentially to the specific port on TCP on the server. In this experiment, we examined 50, 100, 500, 1000 and 5000 sequence SYN request packets with setting one or multi spoofed source IP addresses. Avoiding the unnecessary load caused by probing packets, we prepared another measuring host to probe all packet sequences between the client and the server with timing data. This measuring host captures all SYN, SYN+ACK and RESET packets and reserves receiving times on the server.

Table 1. Classification of response patterns.

Spoofed Source IP address	FreeBSD	Linux	Windows
one	Pattern II	Pattern I	Pattern I
multi	Pattern II	Pattern II	Pattern II

As the results, we observed the following two packet sequence patterns, and shows the classification of response packet sequence patterns in Table 1. To spoof different multi source IP addresses, we incremented the IP address from the lease significant bit and sequence number on the SYN packet to capture the corresponding SYN+ACK packet. These experiments were executed on our private network and response packets for SYN packet with spoofed source IP address were destroyed on this network.

Different source IP addresses are used in the case of the multi spoofed source IP addresses.

2.2 Packet Sequence Pattern I

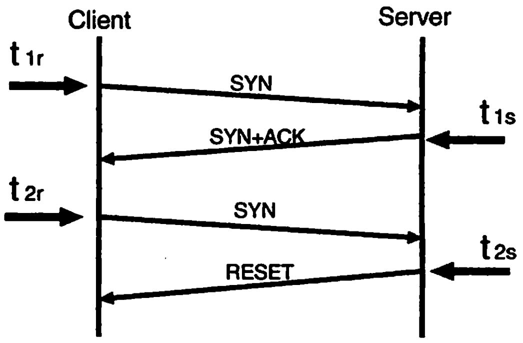

Figure 1. Packet sequence of Pattern I.

Figure 1 illustrates one sequence of response packets for two sequential SYN packets from the client. Firstly, the client sends one SYN packet at the time(t_{1r}), then the server responses the SYN+ACK packet at the time(t_{1s}). If the client sends the next SYN packet sequentially in a short time with the same spoofed source IP address at the time(t_{2r}), the server does not respond to the spoofed packet with the SYN+ACK packet but does so with the RST packet at the time(t_{2s}). These response patterns are repeated one after another for the sequential SYN requests with the same spoofed source IP address.

2.3 Packet Sequence Pattern II

Figure 2 illustrates the other sequence of response packet for one SYN packet. First the client sends one SYN packet at the time (t_{1r}), then the server firstly responses the SYN+ACK packet at the time (t_{1s}). If this client replies back to the server with ACK packet, then the connection is established. If the client, however, is an attacking host, and spoofs source IP address on the first SYN packet, then SYN+ACK packet is abandoned on the network. As the server remains half-open state, it repeats response to SYN packets with SYN+ACK packets with different time intervals. The iterating number of the SYN+ACK packet depends on the server's OS type and interval times approximately increases on the manner of exponential order.

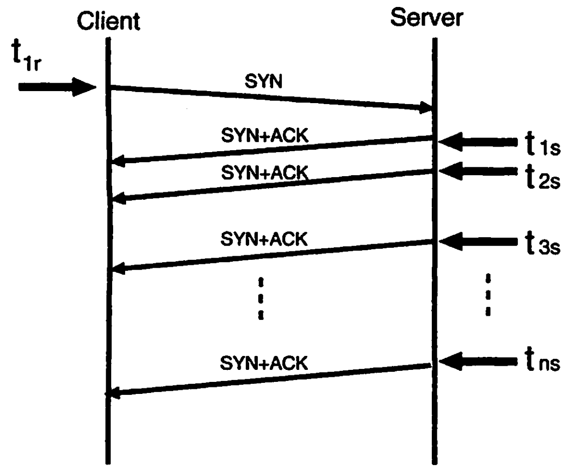

Figure 2. Packet sequence of Pattern II.

3 Results of Observations

In this section, we show the observed raw data collected in our experiments. We evaluated the elapsed time of response packets from the request time of the first packet. As each sequential SYN packet is marked within the sequence number field in the IP header using the sequential number, the corresponding elapsed time is estimated. Due to the variations of response patterns of each operating system, we discuss the following three typical classes in detail.

- The server is on FreeBSD platform and receives sequential SYN packets with one spoofed source IP address.

- The server is on Linux and receives sequential SYN packets with multi spoofed source IP addresses.

- The server is on Windows and receives sequential SYN packets with multi spoofed source IP addresses.

3.1 Measured Packet for FreeBSD

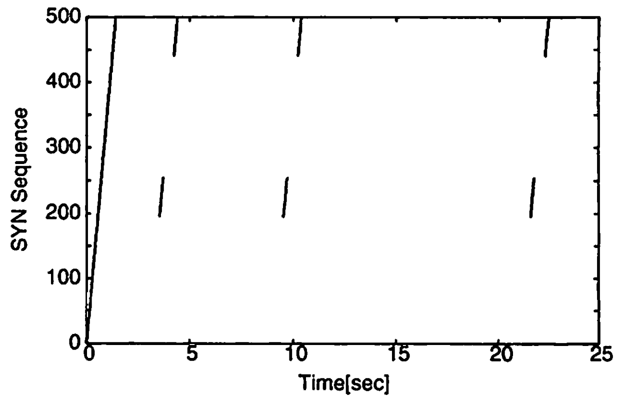

Figure 3. sequential response for FreeBSD.

Figure 3 shows the elapsed time of all SYN+ACK packets responding from the FreeBSD server as the results of 500 sequential SYN packets with one spoofed source IP address. The FreeBSD server performs the responding packet of SYN+ACK to iterate four times. We observed the 50-SYN case, of which result is not shown, to find that if the number of sequential SYN packets is smaller than 50 packets, every SYN+ACK response packet is sent. On the other hand, 500 sequential SYN packets lead to intermittent response packet at second, third and fourth response. Two responding patterns are illustrated in Figure 3. Firstly, the elapsed time for the response packet increases with a linear manner. Secondly, one part of sequential packet response is omitted similarly in the case of second, third and fourth response case.

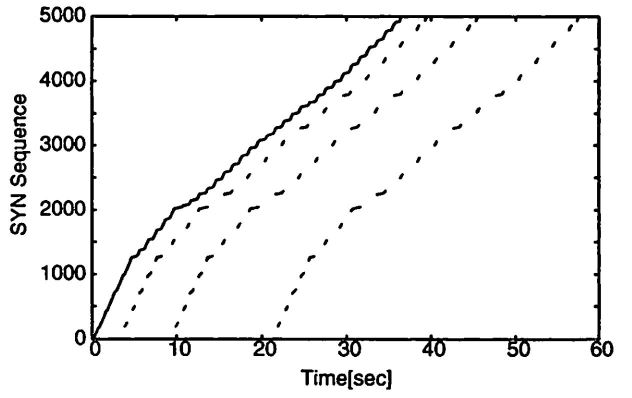

Figure 4. 5000 sequential response for FreeBSD.

Figure 4 illustrates all response SYN+ACK packets activated by 5000 sequential SYN packets. Two identical characters, the intermittent response and the linear increase, are observed in Figure 4 as seen in Figure 3. Firstly, the intermittent response appears to the second, third and fourth response case. Secondly, elapsed time of each response increases with a nearly linear manner. Figure 4, however, indicates that overloads on the server generate fluctuations and delays for response packets. The quantitative discussion will appear in the next subsection.

3.2 Measured Packet for Linux

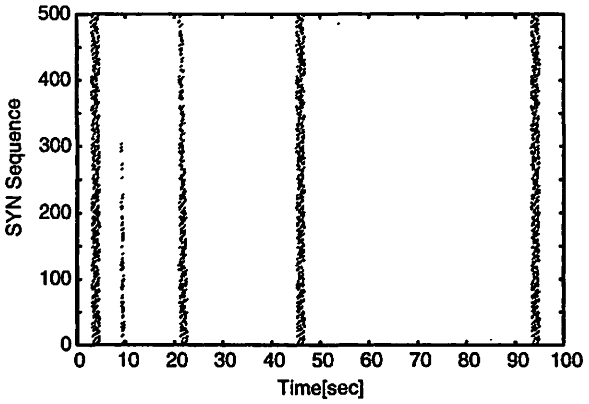

Figure 5. 500 sequential response for Linux.

Figure 5 illustrates the elapsed time of all SYN+ACK packets responding from the Linux server as the results of 500 sequential SYN packets with multi spoofed IP addresses. As all first SYN+ACK packets simultaneously response to SYN packets, all plots are on the vertical axis in the Figure 5. The total packet number of SYN+ACK response is six in terms of this Linux operating system.

Compared to the FreeBSD case, the feature of sequential responding has disappeared in this Linux case. The responding order is likely to be of random manner, and some response packets are lost at the last part of response sequence in especially the third response case. In spite of loss of the third response, we find that the fourth and other responses are generated. It means that the Linux retains the state information, but cannot control the responding process.

3.3 Measured Packet for Windows

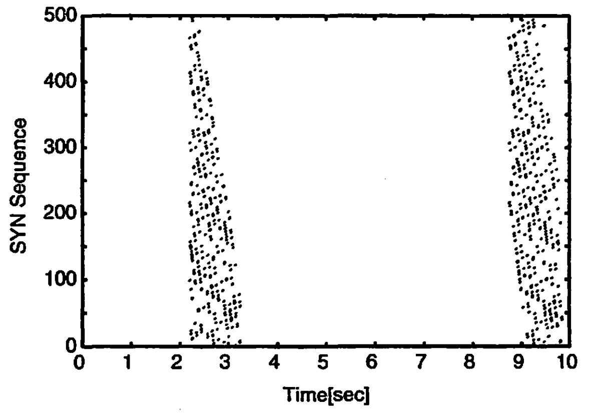

Figure 6. 500 sequential response for Windows.

Figure 6 shows the elapsed time of all SYN+ACK packets responding from the Windows server as the results of 500 sequential SYN packets with multi spoofed IP addresses. First SYN+ACK responses are plotted on the vertical axis similar to the Linux cases. In addition, three packets are totally responded on the Windows case. The packet loss has appeared similarly at the second SYN+ACK response.

4 Detective Method

In this section, we propose the SYN flood detective method to compare the standard model and actual system performance.

4.1 Metrics for FreeBSD

From these observations described previously, we select response rate and average response delay in metrics to identify whether the server is attacked or not.

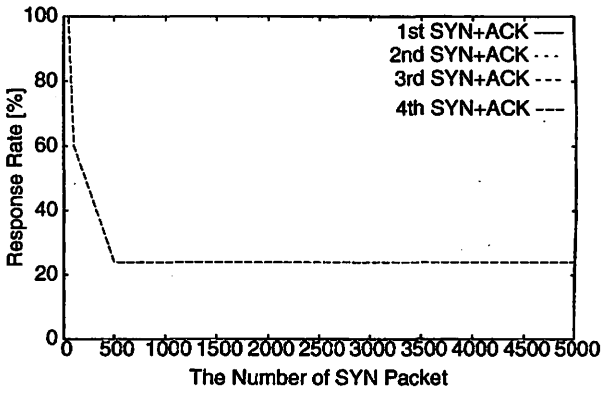

Figure 7. Packet response rate.

First metric is the packet response rate [%] which defined by

$$\frac{\textit{the number of actual response packets}}{\textit{the number of expected response packets}} \times 100[\%].$$

Figure 7 shows the packet response rate of each SYN+ACK response. First SYN+ACK response maintains 100 % for each sampled SYN request. In other SYN+ACK cases, packet response rates vary similarly as to first decrease to 60 % for 100 sequence SYN packets, then decrease to 24 % for more than 500 sequence of SYN requests and the rate keeps unchanged for larger SYN sequential requests. This observation means that TCP response completely for the first SYN request, but reduces quickly for additional, i.e. from 2nd to 4th, SYN+ACK packets. As the result, we can set packet response rate as one threshold metric. If this response rate decays under 50 % for the additional SYN+ACK packets, we could close the half-open state on TCP.

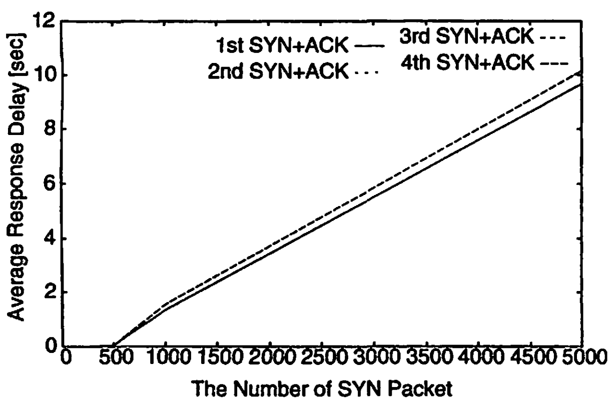

Figure 8. Average response delay.

Second metric is the average response delay calculated as follows: First we measure the standard response delay, i.e. $t_{1s} - t_{1r}$, $t_{2s} - t_{1r}$, $t_{3s} - t_{1r}$ and $t_{4s} - t_{1r}$, from the response time for the sequential 30 SYN requests as the normal stable state. Then we define that the average response delay is the average of actual response delay minus the expected response delay predicted by the standard response delay. Figure 8 illustrates the average response delay of each SYN+ACK response. Every response remains under 100 milliseconds in the case of smaller then 500 SYN request packets. SYN request packets increase more then 500, average response delays raise linearly and finally reach about 10 seconds. Additional SYN+ACK response, i.e. 2nd, 3rd and 4th response, delay similarly and delay more than 1st SYN+ACK response. We can set average response delay as second threshold metric. If the average response delay exceeds 1 second, we could close the half-open state on TCP.

4.2 Metrics for Linux

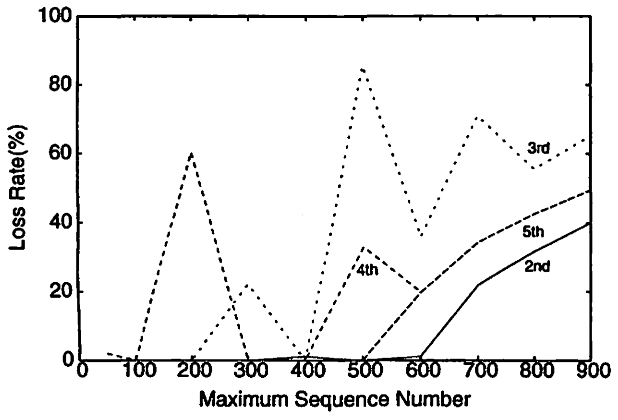

Figure 9. Loss rate.

Different from FreeBSD platform, it is inappropriate to adopt the delay as a metric to detect attacks for non-sequential responses corresponding to sequential SYN requests in Linux platform. As we discussed in subsection 3.2, a number of fourth and fifth response packets are sent though second or third response packets with the same sequence number are lost. It means Linux platform have kept the half-open state for each connection regarding the legitimate connection. Therefore, we adopt the packet loss to identify whether the server is attacked or not.

Figure 9 illustrates the packet loss rate for each response packet in Linux platform. We examine a new experiment to observe response packets for sequential maximum sequence numbers from 100 to 900 with 100 steps, and calculate the packet loss rate of each response packet. All first response packet to each SYN request packets responds quickly showing 0 % packet loss rate.

We can capture the following features from the shape of loss rate in Figure 9. The third response packet locating in the middle of responding sequence has the largest loss rate compared to other responses and the following responses can not decrease the some boundary value of packet loss rate especially in the area with maximum sequence number exceeding 500. If the sequence number exceeds 500, the packet loss rate of each response overly increases, and every loss rate exceeds 20 % after reaching 700. If the source IP addresses are spoofed a same address, the loss rate is calculated from the total response packets. But if the source IP addresses are spoofed in diversified addresses, it is difficult to capture only the attacker's packets. Therefore, we adopt the occurrence of the packet loss as the detective metrics.

4.3 Detective Method for FreeBSD

We defined two thresholds, i.e. the packet response rate and the average response delay. The proposed detective method works as follows.

- We set the detective system in front of the observed server. This system can bc a packet forwarding machine or a packet observing machine.
- We capture packet flows on TCP establishing phase, then keep flows causing the half-open state on the server.
- We observe packet flows whether a packet flow decay under the threshold of the packet response rate and/or exceed over the threshold of the average response delay.
- If the packet flows exceed through these thresholds, then detective system sends the RST packet to the server.

As the result, the server can escape the half-open state on TCP quickly.

4.4 Detective Method for Linux

Based on the discussion in subsection 4.2, we propose the following detective procedure adopting the packet loss as a metric.

- We estimate the response delay for each response and it's fluctuating periods for variously spoofed IP addresses. These estimated values consist of the threshold values to determine whether the response packet is lost or not.
- If the real response packet exceeds the threshold values, then the detective host identifies that the server falls in overload condition by a number of attacks and sends the RST packet using the observed connections.

5 Conclusion

In this paper, we presented a measurement based detective method for SYN flood attacks at an early stage. We implemented the SYN packet-sending program and collected the SYN+ACK and RST response packets from the server. After observing the activity of the server, we classified response patterns into two, and examined the experiments to capture the threshold of metrics to identify whether the server is attacked or not. We analyzed the response rate and the average response delay for FreeBSD platform and packet loss rate for Linux platform. From the observation for response packets of two OS's, the response rate decays 24 % or the average response delay exceed 1 second will make the server enter the heavy load condition activated by the attackers on FreeBSD platform. On the other hand, if the response packet is not reached on the detective host until the pre-measured response threshold time, then the detective host identifies the heavy load condition of the server on Linux platform.

In the future, the packet-forwarding machine is to be conducted with the detective method herein proposed to be managed to work in the real network.

References

[1] Rocky K. Chang: Defending against flooding-based distributed denial-of-service attacks: a tutorial: *Communications Magazine, IEEE*, 40(10), pp.42-51, Oct (2002).

[2] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram, and D. Zamboni: Analysis of a denial of service attack on TCP. *In Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pp.208-223 (1997).

[3] H. Wang, D. Zhang, and K. G. Shin: Detecting SYN flooding attacks, *In Proceedings of IEEE INFOCOM*, Vol.3,pp.1530-1539 (2002).

[4] M. Sung and J. Xu.: IP traceback-based intelligent packet filtering: A novel technique for defending against internet DDoS attacks, *IEEE Transactions on Parallel and Distributed Systems*, 14(9),pp.861-872 (2003).

[5] J. Ioannidis and S. M. Bellovin: Implementing pushback: Router-based defense against DDoS attacks, *In Proceedings of Network and Distributed System Security Symposium*, The Internet Society, February (2002).

[6] S. Savage, D. Wetherall, A. Karlin, and T. Anderson: Practical network support for IP traceback, *In Proceedings of the ACM SIGCOMM Conference*,pp.295-306, (2000).

[7] A. C. Snoeren: Hash-based IP traceback, *In Proceedings of the ACM SIGCOMM Conference*, pp.3-14 (2001).