Detecting Distributed Denial-of-Service Attacks by analyzing TCP SYN packets statistically

Yuichi Ohsita
Graduate School of
Information Science and Technology,
Osaka University
1-5 Yamadaoka, Suita,
Osaka 565-0871, Japan
Phone: +81-6-6879-4542
E-mail: y-ohsita@ist.osaka-u.ac.jp

Shingo Ata
Graduate School of Engineering,
Osaka City University
3-3-138 Sugimoto, Sumiyoshi-ku,
Osaka 558-8585, Japan
Phone: +81-6-6605-2191
E-mail: ata@info.eng.osaka-cu.ac.jp

Masayuki Murata
Graduate School of
Information Science and Technology,
Osaka University
1-5 Yamadaoka, Suita,
Osaka 565-0871, Japan
Phone: +81-6-6879-4540
E-mail: murata@ist.osaka-u.ac.jp

Abstract— **Distributed denial-of-service attacks on public servers have recently become more serious. More are SYN Flood attacks, since the malicious attackers can easily exploit the TCP specification to generate traffic making public servers unavailable. To assure that network services will not be interrupted, we need faster and more accurate defense mechanisms against malicious traffic, especially SYN floods. One of the problems in detecting SYN Flood traffic is that server nodes or firewalls cannot distinguish the SYN packets of normal TCP connections from those of SYN Flood attack. Moreover, since the rate of normal network traffic may vary, we cannot use an explicit threshold of SYN arrival rates to detect SYN Flood traffic. In this paper we introduce a mechanism for detecting SYN Flood traffic more accurately by taking into consideration the the time variation of arrival traffic. We first investigate the statistics of the arrival rates of both normal TCP SYN packets and SYN Flood attack packets. We then describe our new detection mechanism based on the statistics of SYN arrival rates. Our analytical results show that the arrival rate of normal TCP SYN packets can be modeled by a normal distribution and that our proposed mechanism can detect SYN Flood traffic quickly and accurately regardless of time variance of the traffic.**

Index Terms— **Distributed Denial of Service (DDoS), SYN Flood, Statistical Analysis, Normal Distribution, Traffic Monitoring**

I. Introduction

The recent rapid growth and increasing utility of the Internet are making Internet security issues increasingly important. Denial-of-service (DoS) attacks are one of the most serious problems and must be resolved as soon as possible. These attacks prevent users from communicating with service providers and have damaged many major web sites all over the world.

The number of attacks is increasing, and the techniques used to attack servers are more complex. In the distributed denial-of-service (DDoS) attack often seen recently, multiple distributed nodes attack a single server concurrently. A malicious user tries to hack remote nodes by exploiting the vulnerabilities of software running on them, installs an attacking program on hijacked nodes, and keeps them waiting for an order to attack a victim server. When the malicious user sends a signal to them, they begin to attack to the same server. Even if the rate of attack for each node is small, the attack traffic can cause serious damage at the victim server when the number of hijacked nodes is large.

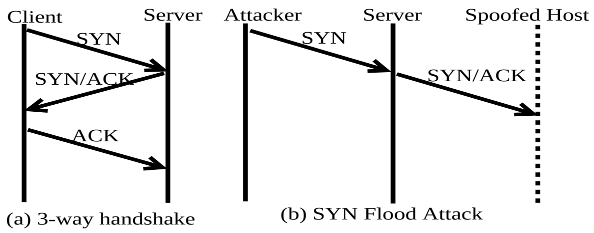

Fig. 1. Overview of a 3-way handshake and a SYN Flood attack

There are many kinds of DDoS attacks such as Smurf attacks [1], UDP floods [2], and SYN flood attacks [3]. In Smurf and UDP attacks, attackers generate many ICMP or DUP packets to exhaust the capacity of the victim's network link. In SYN Flood attacks, attackers send so many connection requests to one server that users cannot connect to that server. Because attackers can easily put servers into a denial-of-service state this way, about 90% of all DoS attacks are SYN Flood attacks [4].

SYN Flood attacks explit the TCP (Transmission Control Protocol) specification. In the TCP, a local node communicates with a remote node by way of a virtual connection established by a process called a 3-way handshake. As shown in Figure 1(a), a client first sends a server a SYN requesting to establish a connection. Then the server sends the client a SYN/ACK packet acknowledging receipt of the SYN packet. When the client receives the SYN/ACK packet, the client sends the server an ACK packet acknowledging receipt of the SYN/ACK packet and begins to transfer data.

In the 3-way handshake, the state in the server waits for the ACK packet from the client is called the *half-open* state. The server in the half-open state prepares for communication with the client, for example, by allocating a buffer. Since a server in the half-open state is using some of its resources for the client, the number of half-open states should be limited. The number of connections it can maintain while it is in the half-open state is controlled in a backlog queue. SYN packets in excess of the number that can be held in the backlog queue are discarded, and the server sends RST packets to notify clients whose SYN packets are discarded.

Figure 1(b) shows an overview of a SYN Flood attack. Attackers send SYN packets whose source address fields are spoofed. The server receiving these SYN packets sends SYN/ACK packets to spoofed addresses. If the node having the spoofed address actually exists, it sends a RST packet for the SYN/ACK packet because it didn't send the SYN packet. If there is no host having the spoofed address, however, the SYN/ACK packet is discarded by the network and the server waits in vain for an ACK packet acknowledging it. For losses of SYN/ACK packets, the server has a timer in the backlog queue, and half-open states exceeding the timer are removed. When the backlog queue is filled with spoofed SYN packets, however, the server cannot accept SYN packets from users trying to connect to the server.

Because the packets used in SYN Flood attacks do not differ from normal TCP SYN packets except in the spoofing of the source addresses, it is difficult to distinguish them from normal TCP SYN packets at the victim server. This is why SYN Flood attacks are hard to detect.

In this paper, we propose a method that detects attacks more quickly and more accurately by taking the time-of-day variance of traffic into consideration. In Section II we explain related works. In Seccion III we explain how we gathered data on router traffic and investigated the characteristics of normal traffic statistically. We then describe a new detection algorithm based on the results of our statistical analysis. In Section IV, We describe the definition of the attack traffic used in this paper and we show through trace-driven simulations that our method can detect all of attack traffics we defined. In Section V we conclude by briefly summarizing the paper and mentioning some of the future work we intend to do.

II. Related Work

Many methods to defend servers from these attacks have been proposed.

In the ingress filtering [5], the internal router is configured to block packets that have source addresses from outside the internal network. This method cannot, however, remove all attack packets because attack packets with addresses of internal network cannot be blocked.

SYN cache [6] and SYN cookie [7] are mechanisms in server nodes. In the SYN cache mechanism, the server node has a global hash table to keep *half-open* states of all applications, while in the original TCP these are stored in the backlog queue provided for each application. As a result, the node can have more number of *half-open* states and the impact of SYN Flood attack can be reduced. However, this mechanism does not resolve the problem of SYN Flood attacks fundamentally. On the other hand, the SYN cookie mechanism can remove the backlog queue by using a *cookie* approach. In the original TCP the server node first allocates the server's resources and sends SYN/ACK packet. It is because there is no method to validate whether the received ACK packet after sending the SYN/ACK packet is really the acknowledgment of the SYN/ACK packet (i.e., the final packet of 3-way handshake). The SYN cookie embeds a *magic number* encrypted by the header of the SYN packet (e.g., IP addresses, port numbers) into the sequence number of the SYN/ACK packet. The server node then verify the ACK packet of the SYN/ACK packet by decrypting the sequence number of the ACK packet. The server node then allocate the server's resource only when the ACK packet is valid. This mechanism can remove the backlog queue, however, the process of encryption may become another weakness against the high-rated SYN packets. Moreover the SYN cookie cannot retransmit the SYN/ACK packet when it is lost.

If routers can handle the state of TCP handshakes at the servers, for example, they can detect SYN Flood attacks more easily. Some firewalls therefore mitigate the damage of attacks by sending a SYN/ACK packet on behalf of the server and letting SYN packets to the server through only when the router receives the ACK packet of the delegated SYN/ACK packet from the client [8]. The server thus does not need to hold many half-open states for spoofed SYN requests. The cost of this mechanism, however, is high because firewalls need to handle the states of TCP connections. Also, while this method can avoid the damage of momentary attacks, it is vulnerable to long-term attacks that which overwhelm the firewalls. In such case, routers must detect attacks as quickly as possible and setup a filter to remove attack packets or to limit the rates of attack traffic.

Several methods for detecting attacks have been proposed, one of them is found in [9] which utilizes the normalized difference between the number of SYN or SYN/ACK packets and the number of FIN or RST packets. If the rate of SYN packets is much higher than that of FIN or RST packets, the router recognizes that the attacking traffic is mixed into the current traffic.

Another method is to detect the mismatch between bidirectional packets [10]. When a server is not under attack, packet arrival rates for both directions are almost the same or at least of the same order, because the TCP needs an ACK packet for each packet that is sent. If the packet arrival rate for one direction is much higher than that for the other direction, the traffic in the high-rate direction might include some attack packets. In this mechanism, however, the router cannot detect the attack until the server has a vacancy in the backlog queue the server uses to store can reply SYN/ACK packets for spoofed SYN packets. MULTOPS [11] is one of similar version which checks the asymmetries of traffics for both directions with the granularity of subnet. [12] detects attacks by the number of source addresses. If the number of source addresses increase rapidly, the current traffic might include attack packets.

These methods have several problems, however, one of which is that they cannot detect attacks until servers are seriously damaged or until most of the connections are closed. Another is that they may mistake high-rate normal traffic for attack traffic because they do not take into consideration the normal time-of-day variation of network traffic. Attack traffic should be identified more accurately by considering the variance of normal traffic.

III. Statistical analysis of traffic and attack detection method

In this section, we first describe how we gathered the data we used to model normal traffic and how we analyzed that data. We then describe the algorithm we use to detect the attack traffic.

A. *Monitoring and classification of real traffic*

We deployed a traffic monitor at the gateway of Osaka University. We used an optical-splitter to split the 1000 Base-SX fiber-optic cable and recorded the headers of all of packets transferred on this link. That is, we monitored all the packets in both the inbound and outbound directions at Osaka University.

We use `tcpdump` [13] to read the headers of packets. Although `tcpdump` cannot necessarily read the headers of all packets at wire-speed, we confirmed that the headers of less than 0.01% of the packets were not recorded and these losses did not affect the results of our statistical analysis.

We first classified monitored packets into *flows*. We defined a series of packets which have the same (src IP, src port, dest IP, dest port, protocol) fields as a single *flow* and we classify these *flows* into the following five groups.

Group N	Flows that completed the 3-way handshake and were closed normally by an FIN or RST packet at the end of connections.
Group Rs	Flows terminated by a RST packet before a SYN/ACK packet was received from the destination host. These flows were terminated this way because the destination host was not available for the service specified in the SYN request.
Group Ra	Flows terminated by a RST packet before an ACK packet for the SYN/ACK packet was received. These flows were terminated this way because the SYN/ACK packets were sent to a host that was not in the Internet.
Group Ts	Flows containing only SYN packets. These flows are not terminated explicitly (i.e., by RST/FIN packets) but by the timeout of flows. There were three reasons that flows could be classified into this group. One was that, the destination node did not respond the SYN packet. A second was that the source address of the SYN packet was spoofed and the destination sent the SYN/ACK packet to the spoofed address. The third was that all of the SYN/ACK packets were discarded by the network (e.g., because of due to e.g., network congestion).
Group Ta	Flows containing only SYN and its SYN/ACK packets. Like Group Ts flows, these flows were terminated by the timeout of flows. In this case, however, it was because all the ACK packets were dropped.

TABLE I
CLASSIFICATION OF FLOWS

Group	number of flows	percentage
N	18,147,469	85.1
Rs	622,976	2.9
Ra	75,432	0.3
Ts	2,435,228	11.4
Ta	2,009	0.0

To identify the traffic of normal flows, we focused on the Group N flows. Hereafter, we refer these flows as *normal traffic* and to Groups Rs, Rs, Ts and Ta flows as *incomplete traffic*.

B. *Time-dependent variation of normal traffic and its statistical modeling*

In the work reported in this paper, we used the traffic data for 5 days: from 17:55 on March 20, 2003 to 19:45 on March 24, 2003. The average rate of incoming traffic (from the Internet to the campus network) was about 12.0 Mbps and the average rate of outgoing traffic was about 22.4 Mbps. During busy hours (09:00 to 17:00) the average incoming and outgoing rates were respectively 37 Mbps and 55 Mbps. A total of 1,983,116,637 TCP packets were monitored, 21,615,220 of which were SYN packets. The total number of flows that were monitored, however, was only 21,283,114. The difference between the number of SYN packets and the number of flows is due to the retransmission of SYN packets.

The numbers of flows classified into each of the five groups are listed in Table I. These values were obtained using 180 seconds as the timeout. That is, if there are more than 180 seconds after the last packet in of the flow, we considered the flow to be terminated.

The time-dependent variations of SYN arrival rates of all flows, the flows in *normal traffic* and the flows in *incomplete traffic* are shown in Figures 2, 3, and 4. Points where the arrival rate rises sharply (e.g., 28,000 sec, 42,000 sec, and 57,000 sec) seem to be due to *incomplete traffic*. These results also show that we would mistakenly identify many points as attacks if we set a single threshold for the SYN arrival rates because the arrival rates of the normal traffic change over time. We can also see that the distribution of SYN arrival rates seems to be proportionally more bimodal in *incomplete traffic* than in the *normal traffic*

To confirm this impression, we fitted the SYN arrival rates of normal traffic to a normal distribution. We periodically measured the arrival rates of SYN packets classified in Group N and calculated the average and the variance of those rates. The equation for the normal distribution $F(x)$ with the mean ζ and the variance σ^2 of measured SYN arrival rates is

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\sigma} exp[\frac{-(y-\zeta)^2}{2\sigma^2}]dy. \quad (1)$$

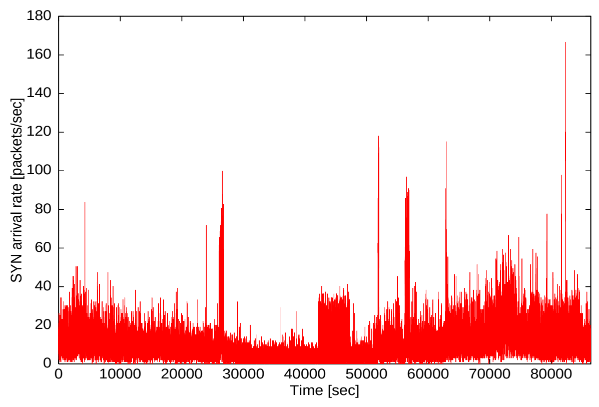

Fig. 2. Time-dependent variation of SYN arrival rates (all flows)

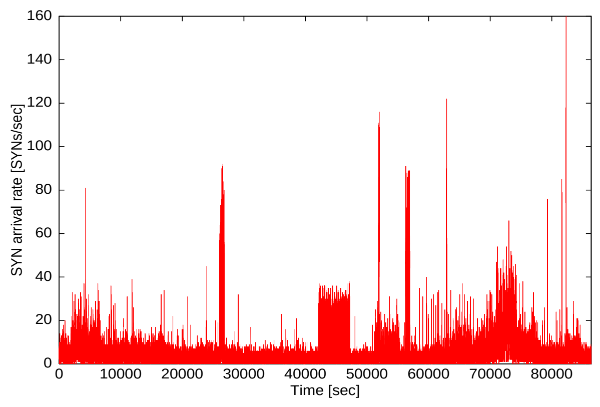

Fig. 4. Time-dependent variation of SYN arrival rates (incomplete traffic)

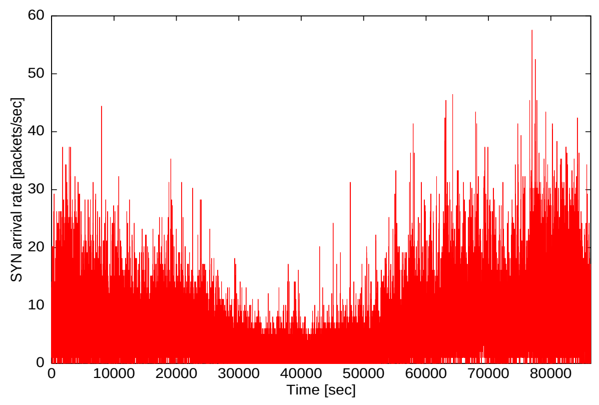

Fig. 3. Time-dependent variation of SYN arrival rates (normal traffic)

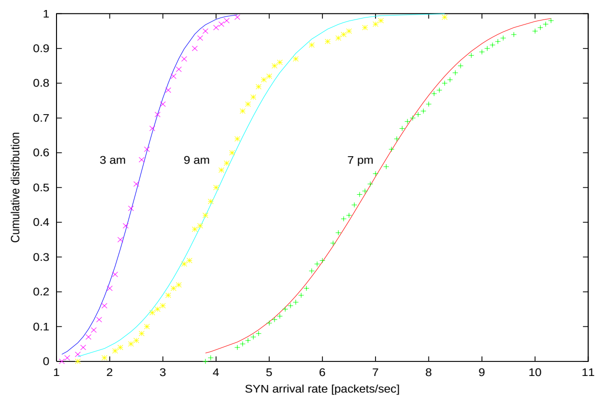

Fig. 5. Comparisons between the distributions of SYN rates and the normal distributions having the same averages and variances (*normal traffic)*

Because arrival rates are positive values, however, we use only the nonnegative part of this equation. That is, because arrival rates change with the range of $[0, \infty)$ we use

$$G(x) = \frac{F(x) - F(0)}{1 - F(0)}. \tag{2}$$

Figure 5 shows the result of fitting the normal traffic to Eq. (2). This figure compares the cumulative distributions of SYN packet arrival rates with the cumulative normal distributions having the same averages and variances. The three sets of curves are for the data obtained in 10-second intervals at 3:00, 9:00, and 19:00. We used 100 samples to obtain the SYN rate distributions. From this figure we can see that most parts of the SYN rate distributions of the *normal traffics* can be modeled by the normal distribution. We also verified that above result can be applied other times in which results are not shown due to the space limitation. That is, we can conclude that the distribution of SYN arrival rate of the *normal traffic* could be modeled by a normal distribution regardless the time-of-day variations.

The distribution of SYN arrival rates of all flows, on the other hand, was sometimes quite different from a normal distribution. This is because the distribution of SYN arrival rates of incomplete flows was far from a normal distribution. Figure 6 compares the distribution of SYN arrival rates of all flows -which includes attack traffic- with the normal distribution having the same average and variance. Because of many high-rate SYN packets caused by attack traffic, the distribution of SYN rates is no longer a normal distribution.

C. *Attack detection method based on statistics of SYN arrival rates*

As described above, the SYN arrival rates of the *normal traffic* were normally distributed, while the distribution of the SYN arrival rates of network traffic including attack traffic was far from a normal distribution when it includes the attack traffic. We should thus be able to identify attack traffic by observing the difference between the distribution of SYN arrival rates and a normal distribution. This can be done as follows. For each measurement of SYN arrival rate, calculate the average of squared differences from the normal distribution. Denote the number of sampled SYN rates as n. Sort them in ascending order and label them $r_i (1 \leq i \leq n)$. Denote as D the average of squared differences from normal distribution:

$$D = \frac{\sum_{i=1}^{n} (G(r_i) - i/n)^2}{n}. \tag{3}$$

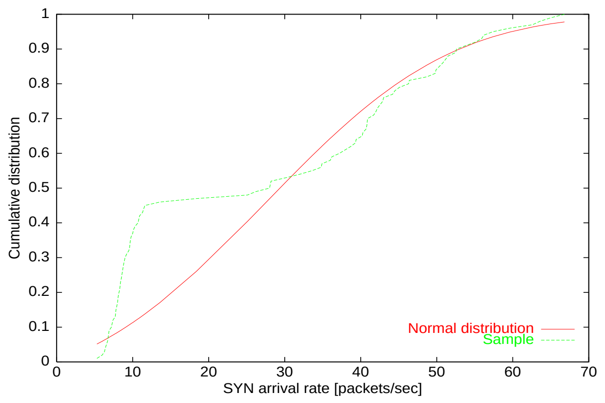

Fig. 6. Distribution of SYN packet arrival rate when attacks started.

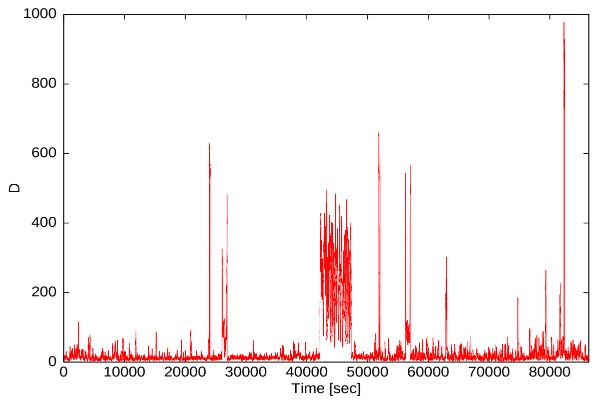

Fig. 7. Variation of average of squared differences between the sampled SYN rates and the normal distribution (all flows)

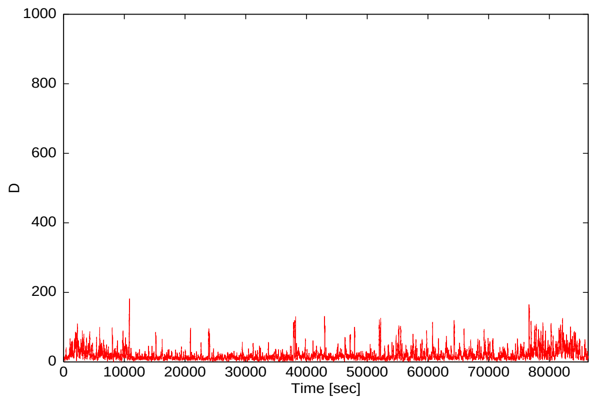

Fig. 8. Variation of average of squared differences between the sampled SYN rates and the normal distribution (*normal traffic*)

TABLE II

DEFAULT CONFIGURATION OF BACKLOG QUEUE

OS	max length	timeout (sec)
Linux	1,024	180
Solaris	1,024	240
Windows 2000 server	200	40

We calculated the value of D for each of our measurements of the SYN arrival rate. Figure 7 shows the variation of D for all flows and Figure 8 shows the variation of D for the *normal traffic*. According to these results, the averages of the squared differences for the *normal traffic* are small regardless of time. The averages of the squared differences for all flows, on the other hand, rise rapidly at several points (we call them *spikes* throughout this paper). Comparing Figures 7 and 8 suggests that these *spikes* are caused by the *incomplete traffic* including attack traffic. Therefore, we can detect attack traffic easily by modeling the arrival rates of SYN packets as a normal distribution, calculating the average of squared differences, and setting a threshold for D as the boundary between normal traffic and attack traffic. Note here that using the normal distribution for calculating D requires some computational overheads. However, the overhead can be reduced by having a kind of normal distribution table.

IV. Performance Evaluation

A. Definition of attack traffic

In this paper, we define the attack traffic that must be detected as traffic that can put a server into a denial-of-service state. This state occurs when the backlog queue is full and new SYN packets arrive at the server. The length of the backlog queue is configured by a setting in the operating system, and the backlog queue default parameters for some widely used operating systems are listed in Table II. The timeout values in this table are the times at which the half-open connections in the backlog queue are removed. That is, the half-open connections persisting longer than the timeout are closed by the server. To put a server into a denial-of-service state, the attacker has to supply a number of SYN packets exceeding the maximum length of backlog queue within the timeout period. In this paper, we suppose target servers are running Linux and we define attacks as cases when more than 1024 SYN packets that do not complete the 3-way handshake are sent within 180 seconds. Scanning our 5-day data, we found total 10 points satisfying this definition of attack traffic.

B. Accuracy of proposed detection method

We evaluated our detection algorithm by using a trace-driven simulation based on the traffic data we measured. We set sampling period to 10 second and used 100 samples to calcuate D.

We define the probability (P_n) of not detecting the attack traffic and the probability (P_f) of erroneously detecting an attack as the followins:

$$P_n = \frac{\text{number of attacks not detected}}{\text{number of attacks satisfying the definition}} \quad (4)$$

$$P_f = \frac{\text{number of points erroneously detected as attacks}}{\text{number of points detected as attacks}}$$

Probabilities of P_n and P_f are shown in Figure 9 as a function of the threshold for D. This figure shows that all attacks could be detected when we set the threshold to less than 90. Though probability of detecting erroneously was 5 % (only

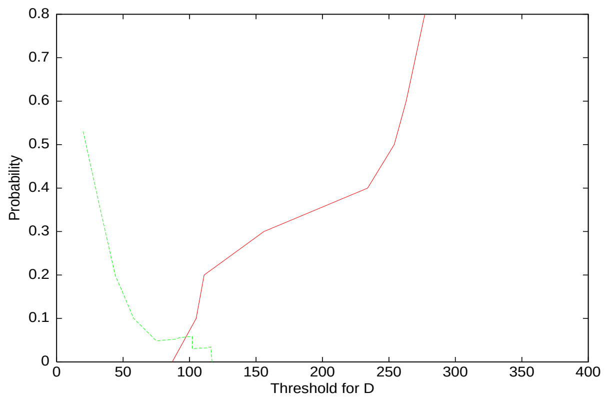

Fig. 9. Relation between threshold for D and the probabilities of not detecting an attack (——) and of erroneously detecting an attack(- - -).

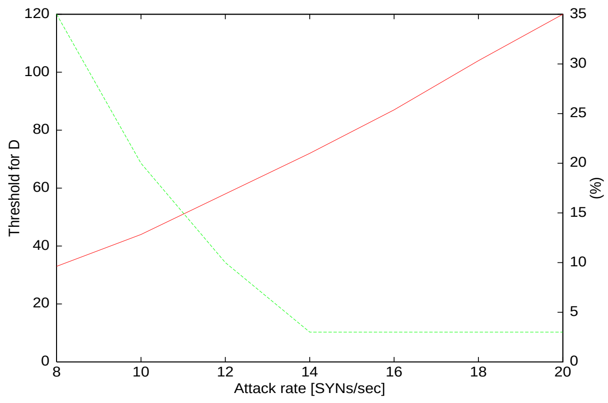

Fig. 10. Relation between the SYN rate of attack traffic and the threshold for D needed to detect attacks (——) and the probability of erroneously detecting an attack (- - -).

2 points in 5-days traffic) when the threshold was 90, these erroneous detections were caused by a single client sending about 20 SYNs/sec. In these points, a client tried to create many HTTP sessions. From the viewpoints of fairness and resource managements, this relatively high-rate traffic should be limited. It can, after all, be regarded such traffic as “attack traffic” directed at the Internet itself rather than a specific server.

C. Detectable SYN rate of attack traffic

We also examine the SYN rates of attacks that can be detected. Because low-rate attack traffic was not found in our data, we simulated such traffic by injecting low-rate attack traffic into the traced traffic at the inbound interface.

The threshold for D needed to detect all attacks and the probability of detecting an attack erroneously are shown in Figure 10 as a function of the SYN rate of the attack traffic. We can see that a lower threshold is needed to detect the lower-rate attack traffic but that a lower threshold also causes more erroneous detections. In this case, Figure 10 shows that in attacks whose SYN rates were more than 14 SYNs/sec could be detected without detecting any attacks erroneously. The probability of erroneous detection cannot be 0 because of the same observation described in the previous subsection.

Figure 11 shows the dynamics of D from the beginning of the attacks. In this figure, the SYN rates of the attacks are 12 SYNs/sec, 16 SYNs/sec and 18 SYNs/sec. This figure shows that the averages of squared differences increase gradually after the beginning of attacks. When the threshold is set to 70, attacks with SYN rates higher than 16 SYNs/sec can be detected within 60 seconds. In this case, the number of half-open states caused by attack is 960, which is smaller than the length of backlog queue in Linux.

To show that our mechanism can detect attacks faster, we compare the time needed to detect attacks on our method with the time on the method proposed in [9]. Throughout this paper, we refer it as SYN-FIN method.

We first note here a brief description of the SYN-FIN method. First, we calculate Δ_i which is the difference between

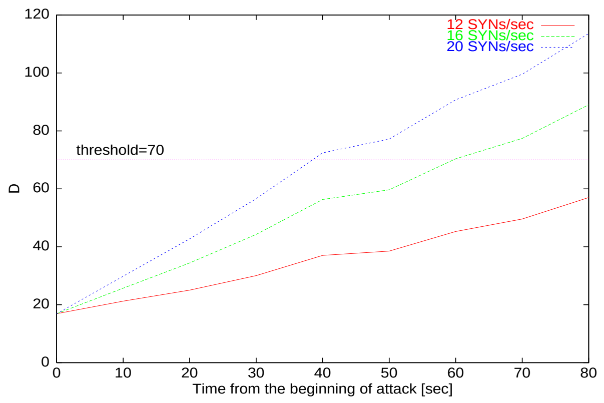

Fig. 11. Average of squared differences versus time after the beginning of attacks with various SYN rates.

the number of SYN or SYN/ACK packets and the number of RST or FIN packets. We then obtain the normalized value of Δ_i by dividing the average number of RST or FIN packets F, which is given by $x_i = \Delta_i/F$. We then calculate y_i from

$$y_i = \begin{cases} 0 & (y_{i-1} + x_{i-1} - \alpha \leq 0) \\ y_{i-1} + x_{i-1} - \alpha & (otherwise) \end{cases} \quad (5)$$

Finally, we determine the traffic has some attacks by detecting the value of y_i exceeds the threshold T.

In the simulation, we set the values of α and T to be 0.15 and 0.37 respectively, which are the optimized parameters to detect attacks as fast as possible. We set the threshold of D in our method to be 70, which can detect attacks without detecting any attacks erroneously.

Figure 12 compares the time to detect attacks between our method and SYN-FIN method. We varied the rate of attacking traffic and measure the time needed to detect the attacking traffic. From this figure, we can observe that our method is much faster to detect attacks than SYN-FIN method. One of the reasons is because SYN-FIN method uses a non-parametric approach to estimate the difference the characteristic of normal from the one of attacking traffics, while our method adopts a

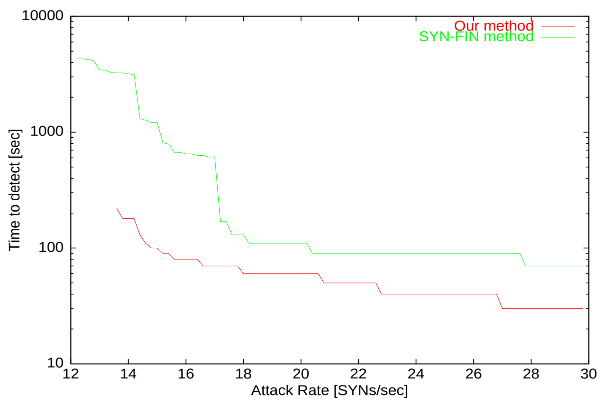

Fig. 12. Time to detect attacks.

parametric approach (i.e., we model that the SYN rate of the normal traffic follows the normal distribution) to estimate it. The parametric approach can detect faster and more accurate than the non-parametric approach in the cases if the model is appropriate. However, SYN-FIN method has an advantage that it can also detect attacks with lower rate (e.g., less than 13 SYNs/sec). Our method cannot detect them because the traffic having the low rate attacks still follows the normal distribution.

D. *Resource needed by detection method*

From above results, our method can work with only 100 samples of SYN rates. If we monitor D for each destination address, we need 100 samples for each address. The captured traffic has 1,000 destination addresses in 1,000 seconds of inbound traffic, and 10,000 destination addresses in 1,000 seconds of outbound traffic. According to Figure 3, arrival rates are not so large and we can then assume a small range of integer value (i.e., 16 bits) is enough for counting SYN rates. Then we need 200 KByte for incoming traffic and 2 Mbyte for outgoing traffic.

V. Conclusion and Future Work

We analyzed the traffic at an Internet gateway and the results showed that we can model the arrival rates of normal TCP SYN packets as a normal distribution. Using this result, we described a new attack detection method taking the time variance of arrival traffic into consideration. Simulation results show that our method can detect attacks quickly and accurately regardless of the time variance of the traffic. Our future works are to set the threshold dynamically, to optimizesome configurable parameters (e.g., sampling intervals, number of samples to obtain the distribution) and to model other types of traffic (e.g UDP Flood , ICMP Flood).

References

[1] "CERT advisory CA-1998-01 smurf IP Denial-of-Service attacks." available at `http://www.cert.org/advisories/CA-1998-01.html`, January 1998.
[2] "CERT advisory CA-1996-01 UDP port Denial-of-Service attack." available at `http://www.cert.org/advisories/CA-1996-01.html`.
[3] "CERT advisory CA-1996-21 TCP SYN flooding and IP spoofing attacks." available at `http://www.cert.org/advisories/CA-1996-21.html`, September 1996.
[4] D. Moore, G.M. Voelker, and S. Savage, "Inferring internet Denial-of-Service activity," Proceedings of the 2001 USENIX Security Symposium, pp.9–22, August 2001.
[5] P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing." RFC 2267, January 1998.
[6] J. Lemon, "Resisting SYN flooding DoS attacks with a SYN cache," Proceedings of USENIX BSDCon'2002, pp.89–98, February 2002.
[7] A. Zuquete, "Improving the functionality of SYN cookies," Proceedings of 6th IFIP Communications and Multimedia Security Conference, pp.57–77, September 2002.
[8] T. Darmohray and R. Oliver, "Hot spares for DoS attacks," The Magazine of USENIX and SAGE, vol.25, no.4, p.3, July 2000.
[9] H. Wang, D. Zhang, and K.G. Shin, "Detecting SYN flooding attacks," Proceedings of IEEE INFOCOM 2002, pp.1530–1539, June 2002.
[10] J. Mirkovic, D-WARD: DDoS network attack recognition and defence, Ph.D. thesis, Computer Science Department, University of California, Los Angels, June 2003.
[11] T.M. Gil and M. Poletto, "MULTOPS: A data-structure for bandwidth attack detecrion," Proceedings of USENIX Security Symposium' 2001,, pp.23–38, August 2001.
[12] C.L. Tao Peng and K. Ramamohanarao, "Detecting distributed denial of service attacks using source IP address monitoring." available at `http://www.ee.mu.oz.au/pgrad/taop/research/detection.pdf`, November 2002.
[13] "Tcpdump public repository." available at `http://www.tcpdump.org/`.