DETECTION OF SYN FLOODING ATTACKS USING LINEAR PREDICTION ANALYSIS

Dinil Mon Divakaran, Hema A. Murthy and Timothy A. Gonsalves
Department of Computer Science and Engineering
Indian Institute of Technology, Madras
Email: {dinil,hema,tag}@TeNeT.res.in

***Abstract*— This paper presents a simple but fast and effective method to detect TCP SYN flooding attacks. Linear prediction analysis is proposed as a new paradigm for DoS attack detection. The proposed SYN flooding detection mechanism makes use of the exponential backoff property of TCP used during timeouts. By modeling the difference of SYN and SYN+ACK packets, we are successfully able to detect an attack within short delays. We use this method at leaf routers and firewalls to detect the attack without the need of maintaining any state.**

Keywords— Linear prediction analysis, DoS attack, TCP SYN flooding, Exponential Backoff.

I. INTRODUCTION

Any act to deny legitimate use of a service can be classed as a Denial of Service (DoS) attack [1]. A Denial of Service attack is a major security threat to the services provided through the Internet resulting in large scale revenue losses. The 2004 CSI/FBI Computer Crime and Security Survey reports that denial of service was the top source of financial loss due to cybercrime in 2004 [2]. Analysis shows that more than 90% of the attacks use TCP and TCP SYN flooding is the most prevalent among them [3].

A TCP SYN flooding attack consists of a stream of TCP SYN packets directed to a listening TCP port at the victim [4]. It exploits the vulnerability in TCP's three-way handshake mechanism and its limitations in maintaining half-open connections. A pictorial representation of TCP's three-way handshake is shown in Fig. 1.

A server goes to the SYN-RECEIVED state when it receives a SYN packet from a client with its Initial Sequence Number (ISN) [5]. Responding to the connection request, the server sends a SYN+ACK packet that contains its ISN and an acknowledgment for the SYN packet received. In normal conditions, the client will send an ACK as soon as it receives the SYN+ACK packet from the server, and this causes the server to go to the ESTABLISHED state. If the final ACK does not reach the server, the connection will remain *half-open* until the associated timer expires. The timeout period is usually 75 seconds. Since the memory allocated for maintaining half-open connections is finite, there is a limit to the maximum number of half-open connections. Once this maximum count is exceeded, the server starts dropping requests. Exploiting this limitation, an attacker floods a victim (server) with spoofed IP addresses. The spoofed IP addresses are chosen such that the server will neither receive an ACK packet to complete the three-way handshake, nor a RST packet to tear down the illegitimate connection request and free the resources. Hence the SYN backlog queue overflows resulting in dropping of legitimate connection requests. If the attack lasts for a prolonged period, legitimate users will be deprived of the services provided by the victims.

A consequence of the SYN flooding attack is that a service can be brought down by sending a few tens of SYN requests per second. Early detection of TCP SYN flooding attack is essential to bring the disrupted services back to normal. The fact that e-commerce mostly depends on TCP based applications makes this problem all the more important.

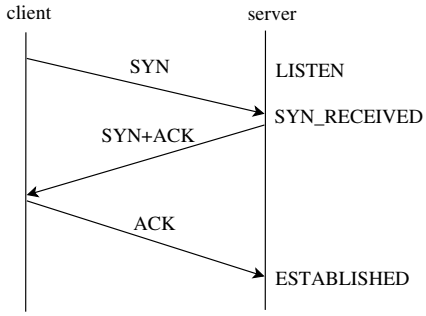

Fig. 1. TCP three-way handshake

In the past, LP analysis has been successfully used to detect faults in a network [6]. This paper proposes LP analysis of traffic as a new approach to detect TCP SYN flooding attacks. We take advantage of the exponential backoff property of TCP (used during timeouts) [5] to detect a unique pattern at the time of attack based on the difference of number of SYN and SYN+ACK packets with respect to a network. The proposed system is fast and stateless, and can be easily implemented. Equally important is the fact that the system can detect low intensity attacks (attacks with low flooding rate, but which is still able to bring down a service). Though we focus only on the detection of SYN flooding attack in this paper, it should be noted that LP analysis can be easily extended to detect any high intensity DoS attack. In fact, any change in traffic patterns can be detected using LP analysis, provided we choose an appropriate parameter to capture the pattern and use the parameter to model the traffic.

The rest of the paper is organized as follows. Section II discusses related work. Parameters used for modeling and the issues of detection near to the source of attack are discussed in Section III. Section IV details detection using LP analysis. Performance evaluation is discussed in Section V. Finally, concluding remarks are drawn in Section VI.

II. RELATED WORK

There exist mechanisms to defend a victim or network from SYN flooding attack. SYN cookies [7] removes the need for a backlog queue by encrypting necessary information into a cookie. The cookie, which is a function of the source address, source port, destination address, destination port, and a random secret seed, is sent as the sequence number to the client in the SYN+ACK packet and returned to the server in the final portion of the three-way handshake. The drawback of this approach is that it can not encode all the TCP options from the initial SYN into the cookie, thus breaking the TCP semantics. SYN cache [8] still maintains states, but using much less

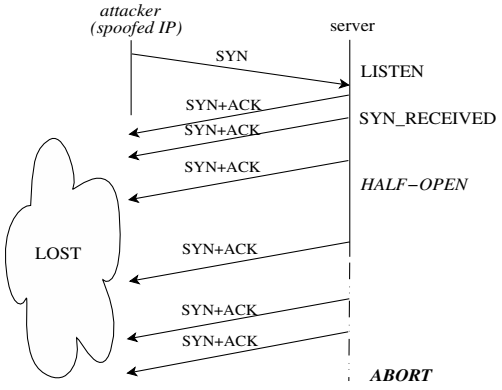

Fig. 2. TCP SYN attack scenario

resource on the server. While the above two are limited to the victim server, defense mechanisms like SYNDefender [9] and SYNkill [10] are installed at firewalls. They continuously monitor the TCP traffic, and inject ad-hoc TCP packets in the network to mitigate the attack and hence slow down the connection even in the absence of attack. L. Ricciulli *et al* [11] proposed defense mechanism based on random drops. As legitimate connection requests may still be dropped, this approach is not a complete solution to the problem. Apart from the limitations described above, these approaches are stateful degrading, the end-to-end TCP performance. A stateless approach is a better alternative.

Past works have also focused on stateless detection methods. In [12], the authors use CUSUM-type algorithm to test if the number of SYN packets over a given interval exceeds a particular threshold. The disadvantage of using the number of SYN packets is the difficulty in detecting an attack to a server in a large network. For example, for a network having many servers, an increase of SYN counts by 30 per second is normal when considering the entire network; where as, it might as well be an attack to a victim server. Number of SYN packets is not the best representation for half-open connections.

The authors of [13] use a better parameter, namely the difference in the number of SYN and FIN packets. The flaw here is that the attacker can easily defeat the system by sending equal number of FIN packets when flooding the victim with SYN packets. Moreover, the authors state that for a SYN flooding attack with a rate of 33 SYNs per second, the detection probability was only 70%. Whereas, a server with SYN backlog queue of length 1024, and the half-open connection expiry time of 75 seconds , can be overwhelmed with an attack rate of not more than 20 SYNs per second. The detection delay [1] is also not promising as it varies from 20 seconds to 8 minutes.

In [14], an active probing scheme was proposed to detect SYN flooding attacks. A method similar to traceroute [5] is used to obtain the path delay between server and client by sending packets with Time-to-Live set at the IP headers. Using this method, a half-open connection is classified as either normal or abnormal. A very long delay is regarded as a network failure, and the corresponding half-open connection is considered normal. Otherwise, the half-open connection is considered as abnormal, one caused due to SYN attack. Such a method essentially fails when the spoofed source IP address happens to meet an actually congested router. In such cases, the half-open connection will be considered as normal and the attack will go undetected.

[1]The time from the start of attack to the time the attack is detected is defined as the detection delay.

III. Parameter for modeling

The detection probability is largely dictated by the parameter we consider for modeling. The number of SYN packets during a time interval can be one such parameter. But, since the number of SYN packets do not provide accurate information on number of half-open connections, it can not be considered as a potential candidate. Past work has used the difference between number of SYN and FIN packets as a parameter for modeling [13]. As the authors pointed out, the weakness of the SYN-FIN pairs scheme lie in its vulnerability to simple counter measures. An attacker, who has knowledge of how the detection mechanism works, can paralyze the SYN-FIN detection mechanism by flooding equal number of FIN packets as SYN packets. Moreover, the SYN-FIN mechanism is dependent on duration of TCP session. Hence there are possibilities of false alarms when clients connect from a less reliable low-bandwidth link.

A better parameter is the difference of SYN and SYN+ACK packets. The advantage here is two-fold. First, the time interval between SYN and SYN+ACK is no more bound by the TCP session. In fact, it depends on how fast a server can generate the SYN+ACK once it receives the SYN, which is normally too small. The second advantage is that we can make use of the retransmission of SYN+ACK packets (as illustrated in Fig. 2) during attacks.

The estimation of the RTO (retransmission timeout) is given in RFC 2988 [15]: 'Until a round-trip time (RTT) measurement has been made for a segment sent between the sender and receiver, the sender SHOULD set RTO as 3 seconds.' A round-trip time of a TCP segment is defined as the time it takes for the segment to reach the receiver and for a segment carrying the generated acknowledgment to return to the sender. When the retransmission timer expires, the RFC requires the following to be done:

- Retransmit the earliest segment that has not been acknowledged by the TCP receiver.
- The host MUST set RTO as 2 * RTO. A maximum value MAY be placed on RTO provided it is at least 60 seconds.
- Start the retransmission timer, such that it expires after RTO seconds (for the value of RTO after the doubling operation in the previous step).

The doubling of the RTO is called the exponential backoff [5]. Upon receiving the connection request, the server allocates resources to handle and track the new connection, then responds with a SYN+ACK packet. During a SYN flooding attack, the server won't receive the final ACK. On time-outs, the victim server will keep on retransmitting the SYN+ACK packets until either the maximum value of RTO is reached or a maximum number of retransmission trials have exceeded. The default value for the maximum number of retransmissions is usually 5. As the RTO is initialized to 3 seconds, retries are attempted at 3, 6, 12, 24, and 48 seconds, doubling the time-out value after each retransmission. This means that once the server receives a SYN packet from a spoofed unreachable IP, it will be sending at least 3 SYN+ACK (one SYN+ACK + 2 retransmissions of SYN+ACK) during the next 10 seconds with the spoofed IP as destination. Assume the server is attacked at a rate of 100 SYN packets per second starting at time $t = 1$. The server will be sending 100 SYN+ACK packets during each of the first three seconds. At $t = 4$, the RTO timer expires. Therefore, the server will not only send acknowledgments for all the 100 SYN packets it received at that time, but will also retransmit all the 100 SYN+ACK packets that were sent at $t = 1$ and didn't get acknowledged. In total, it

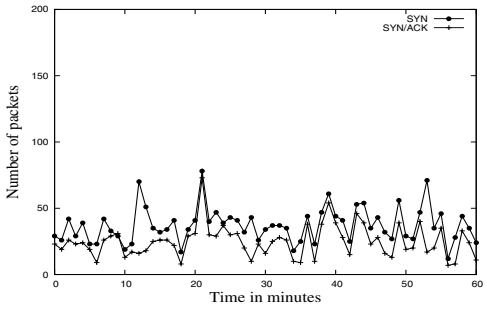

Fig. 3. Flow of incoming SYN and outgoing SYN+ACK packets

will generate 200 SYN+ACK packets at $t = 4$ and each of the next five seconds ($t = 4, 5, 6, 7, 8, 9$). Similarly, at $t = 10$, the server will retransmit the SYN+ACK packets that were sent at $t = 7$. It will also retransmit, for the second time, all the packets that were retransmitted at $t = 4$ and whose associated RTO value of 6 seconds expired. Apart from these 200 retransmissions of SYN+ACK packets, it will send 100 SYN+ACK packets in response to the 100 SYN packets received at $t = 10$. Hence, starting at $t = 10$ for the next 12 seconds (this is the next RTO value) the server will send 300 SYN+ACK packets. In short, with a SYN attack rate of 100 packets per second, the victim will be sending out not less than 1800 SYN+ACK packets for the first 10 seconds interval, 3000 SYN+ACK packets during the next 10 seconds interval, 3900 packets during the third 10 seconds interval and so on.

It should also be noted that under normal conditions, the difference between the number of SYNs and SYN+ACKs is very small, as compared to the total number of SYNs (TCP connection requests). Fig. 3 shows the count of inbound SYN packets and outbound SYN+ACK packets for the TeNeT network [16] during a normal scenario. From the exponential backoff algorithm and retransmission property of TCP, it is clear that the difference between number of inbound SYN and outbound SYN+ACK packets (we refer to this difference as Δ_v) is a very good parameter that can be used for detecting SYN flooding attack.

A. Issues

The retransmissions of SYN+ACK packets will be generated from the victim side. Therefore, the approach based on the difference of the number of incoming SYN packets and outgoing SYN+ACK packets can be used to detect an attack at the victim side [2]. The detection mechanism can be deployed on either the firewall or the router that connects the local network to the Internet. In fact, the detection system should be installed at both the firewall and the router, with different sets of thresholds (thresholds are explained later). In this way, the firewall will be able to detect and take remedial actions against attacks to any hosts in the network. The router with higher values of thresholds can detect any attack aimed at overwhelming the firewall. Whenever an attack is detected, an existing defense mechanism such as SYNDefender [9] or SYNkill [10] can be invoked. Even though these defense systems will slow down the TCP connections, it will only be functioning during the duration of attack. It has been found that 80% of the attack last for less than 30 minutes, and 90% of the attacks last for less than an hour [3].

[2]We use the term outgoing to refer to packets leaving from the local network to the Internet.

The dynamics of flow of SYN packets and SYN+ACK packets will be reversed at the source of attack. Near to the source of attack (at the firewall or router), the difference in the number of outgoing SYN packets and the incoming SYN+ACK packets (we call this Δ_s) will be higher during the time of attack as compared to normal scenario. While Δ_s can still be modeled using LP, the detection delay will be longer when considering other issues. Most importantly, when the sources of attacks are distributed (as in distributed DoS), the amount of traffic generated at each of these source will be considerably less. If the sources are scattered in different networks (autonomous systems), the detection delay will increase as the time required to sense a considerable increase in Δ_s will be more. Besides, the idea of exploiting the TCP retransmissions won't work out here, as the SYN+ACK packets are sent to non-existing spoofed IPs. Therefore, to obtain a visible increase in Δ_s, the polling interval has to be increased, thereby increasing the detection delay. Moreover, an attacker who knows the mechanism can foil the detection mechanism if he/she has access to different sources scattered in different networks. Consider an attack scenario at the rate of r SYN packets per second at a victim v. If there are five sources used for the attack $\{s_1, s_2, s_3, s_4, s_5\}$, then the attack rate from each of the sources is $\frac{r}{5}$. Even though the value of Δ_s will increase beyond the normal values, the attacker can coordinate the sources to keep the value of Δ_s in the normal range. This can be achieved by s_1 sending $\frac{r}{5}$ SYN+ACK packets to s_2, s_2 sending $\frac{r}{5}$ SYN+ACK packets to s_3 and so on, and finally s_5 sending $\frac{r}{5}$ SYN+ACK packets to s_1. This counter attack is feasible only if the attacker has control over sources located in different networks.

IV. Detection using LP analysis

A. LP Analysis

Linear prediction is a mathematical operation where future values of a discrete-time signal are estimated as a linear function of past outputs and inputs[17]. That is, a signal s_n is considered to be the output of a system with some unknown input u_n such that

$$s_n = -\sum_{k=1}^{p} a_k s_{n-k} + G\sum_{l=0}^{q} b_l u_{n-l}, \qquad b_0 = 1 \tag{1}$$

where $a_k, 1 \leq k \leq p$, $b_l, 1 \leq l \leq q$, and the gain G are the parameters of the system. For applications where the input is totally unknown, the system can be modeled as an all-pole model, where the signal is assumed to be a linear combination of past values and some input u_n:

$$s_n = -\sum_{k=1}^{p} a_k s_{n-k} + G u_n \tag{2}$$

In the frequency domain, we rewrite Eq. 2 taking the z transform on both sides. The all-pole transfer function of the system, $H(z)$ is given by

$$H(z) = \frac{S(z)}{U(z)} = \frac{G}{1 + \sum_{k=1}^{p} a_k z^{-k}} \tag{3}$$

Since the input u_n is totally unknown during traffic analysis, the signal s_n can be predicted only approximately from a linearly weighted summation of past samples. Let $\tilde{s}_n$ denote the approximation of s_n, where

$$\tilde{s}_n = -\sum_{k=1}^{p} a_k s_{n-k} \tag{4}$$

Now, the error between the actual signal value s_n and the predicted signal value $\tilde{s}_n$ is given by

$$e_n = \frac{s_n - \tilde{s}_n}{s_n} \tag{5}$$

The parameters a_k are obtained as a result of the minimization of the mean or total squared error with respect to each of the parameters. The algorithm to compute the coefficients is explained in [17].

Instead of computing the LP coefficients just once for the entire signal, they are computed for each of the (possibly) overlapping frames in the signal. Initially, a signal is blocked into frames of N samples or signal values. Adjacent frames are separated by M samples, and the frames will overlap depending on the value of M. So, the first frame will consist of first N samples, the second frame begins M samples after the first sample, overlapping the first frame by $N - M$ samples. Similarly, the third frame begins 2M samples after the first sample and overlaps the second frame by $N - M$ samples and first frame by $N - 2M$ frames. The frames thus formed are processed separately and LP coefficients are computed for each frame. Once the LP coefficients are found, we can predict the next signal value.

A sample or signal value is a measurement of the traffic using a parameter, which is dependent on the particular attack that is being analysed. The value of such a parameter for a time interval is represented by the signal value. We use Δ_v as the signal value in our experiments.

B. Spectrum and Entropy

The anomaly in traffic can also be detected using spectral analysis and entropy estimation. In this paper, *anomalous signal* is a value calculated at the time of an attack, or else we refer to it as *normal*. Similarly, a frame that consists of only normal signal values is called *normal frame*, and one with one or more anomalous signal values is called an *anomalous frame*.

Once we have the LP coefficients, we can compute the power spectrum $P(\omega)$ of the corresponding frame:

$$P(\omega) = |H(z)|^2 = \frac{G^2}{|1 + \sum_{k=1}^{p} a_k z^{-k}|^2}, \quad z = e^{j\omega} \tag{6}$$

The magnitude spectrum of Eq. 6 is obtained by dividing G^2 by the magnitude square of the FFT of the sequence: $1, a1, a2, ..., a_p$. If the number of points used to represent the spectrum is denoted by m, then the probability of a point x_k

$$p(x_k) = \frac{x_k^2}{\sum_{i=1}^{m} x_i^2} \tag{7}$$

Entropy of a given spectrum of points $X = \{x_1, x_2, ..., x_m\}$ is computed as

$$\Theta(X) = -\sum_{i=1}^{m} p(x_i) log(p(x_i)) \tag{8}$$

Entropy is a good measure of DoS attack detection as entropy corresponding to normal frames will be different from entropy corresponding to anomalous frames.

C. Detection Algorithm

The detection system basically does time series analysis of traffic to detect an attack. A signal value s_i is the difference of number of incoming SYN and outgoing SYN+ACK packets during the i^{th} interval. The signal value $\tilde{s}_n$ for the next time slot n, is predicted using the coefficients. We also define thresholds, α, for the error and β for deviation from normal values of entropy. The algorithm begins by computing the LP coefficients of initial normal frames offline. The entropies of the frames are also calculated. The following steps are then used for online detection:

- Compute the signal value from the packet headers at the end of each time slot.
- Compute the error using Eq. 5.
- Compute the entropy of the new frame formed by advancing the current frame by one and including the new signal value.
- Alarm is set if the error is greater than α and entropy is greater than β times the previous value. A defense mechanism is triggered.
- Recompute the LP coefficients and entropy periodically using normal frames.

The last step in the above algorithm is important. It makes sure that frames that deviate from the normal are not selected for recomputing of LP coefficients and entropy. This is ensured by selecting only those frames that are very close to the normal (which in turn is guaranteed using the error value and entropy computed for signals in the frame).

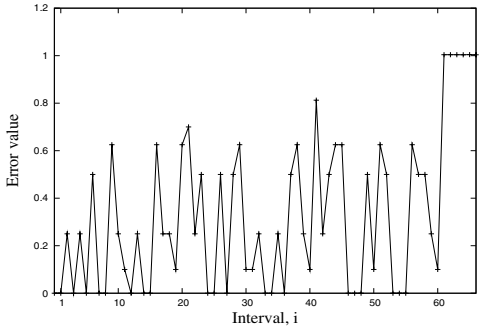

Fig. 4. Prediction error for signal values

V. Evaluation

Traces of normal traffic were collected from the TeNeT network [16] for evaluating the proposed detection method. The traces were obtained using the tcpdump tool [5]. Only those packets which have their SYN bits set (SYN or SYN+ACK packet) were collected for experiments. The abnormal traffic was generated with different attack rates, starting from 20 SYNs per second.

The signal values were calculated for intervals of 10 seconds. A frame consisted of 6 signal values. The testing was performed using 60 consecutive normal signal values, followed by 6 anomalous signal values. Frame was advanced by one signal value ($M = 1$). Since signal value at $t = 61$ is the first anomalous signal value, the frame number 56 which has signal values at $t = \{56, 57, 58, 59, 60, 61\}$ is the first anomalous frame. Similarly, frames 57, 58, 59, 60 and 61 have 2, 3, 4, 5 and 6 anomalous signal values, respectively. Figures 4, 5 and 6 were obtained for SYN attack at a rate of 20 SYNs per second.

A. Results

Fig. 4 shows the error for each signal value as per equation 5. As the first 60 signal values were calculated from normal traffic, the error value is not greater than 0.7 for all of them, except for one signal value. The maximum error among these is 0.81. But, as clear from the figure all the six anomalous signal values toward the right showed high error greater than 1.0, starting from $t = 61$.

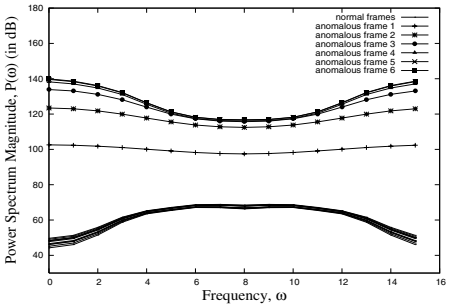

Fig. 5. Magnitude spectrum of transfer function

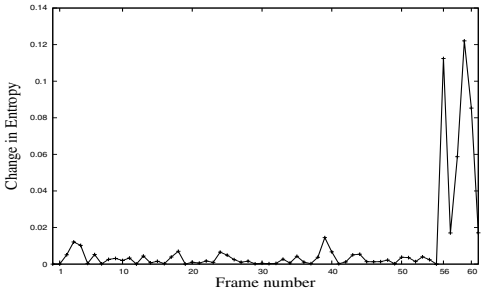

Fig. 6. Entropy difference between adjacent frames

Fig. 5 shows the magnitude spectrum corresponding to some of the normal frames (frame number 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55) and all the anomalous frames using 16 FFT points ($m = 16$). As explained above, frame 56 is the anomalous frame 1, frame 57 is the anomalous frame 2 and so on. From the graph it can be observed that as the error in the frame (number of anomalous signal values) increases, the amplitude of the spectrum increases. This happens as the gain factor of the transfer function is proportional to the error.

Fig. 6 is a plot of the absolute difference between entropy of current frame and previous frame. Change of entropy between normal frames is usually of the order of 10^{-3}; whereas, from normal to abnormal frame the change in entropy is of the order of 10^{-1}. From the graph, it is clear that the entropies of normal frames are concentrated at a value different from the entropies of anomalous frames. Entropies of anomalous frames varied as the number of anomalous signals in the frame increased.

Table I shows the best-case and worst-case detection delays for different attack rates. The attacks were generated at different times of a 10 seconds interval. The algorithm described in IV-C was used to detect the SYN attack. We see that the detection delay decreases as the intensity of the attack increases. Even for low intensity attacks, the worst-case delay is a modest 17 seconds.

B. Observations

As seen from the graphs, both error and entropy can be used to detect an attack. The advantage of using entropy as a parameter for detection is that there arises no false detection alarms. For the experiments conducted, it was observed that an error value > 0.9 ($\alpha = 0.9$) and entropy value that deviated at least by an order of 50 ($\beta = 50$) from that of the normal traffic, could detect all the attacks simulated without any false alarm. With the increase in attack rate (and hence Δ_v), the entropy kept on increasing, even when the error remained very close to 1. Since the time interval (polling interval) was taken as 10 seconds, the detection delay depended on the time the attack began during this interval. As seen in table I, the detection system is able to detect low intensity attacks which has rates as low as 20 SYNs per second. The number of SYN+ACK packets generated during a 10 seconds interval in such an attack scenario is too high when compared to the normal scenario. As the attack rate increases, the victim server will send out more SYN+ACK packets, which will reduce the detection delay. Therefore, a SYN attack at the rate of 500 SYNs per second can be detected within a delay of 13 seconds, in the worst case (this happens when the attack starts 3 seconds before the polling interval, and the detection system has to wait until the end of the two such intervals, assuming the polling interval is set as 10 seconds).

VI. Conclusion

In this work, we have used LP analysis to detect TCP SYN flooding attack within very short detection delay of the order of a few seconds. This work highlights two important results. First, linear prediction can be used to analyse network traffic. Even though the paper has focused on TCP SYN flooding attack, it is pretty evident that the same method can be used to detect other DoS attacks such as UDP flooding, ICMP flooding etc. Choosing appropriate parameters, this approach should be able to detect other TCP based DoS attacks.

Second, we have substantiated that the difference in the number of incoming SYN and outgoing SYN+ACK packets is a potentially useful parameter. It can be used to detect both low intensity and high intensity SYN flooding attacks by deploying the LP based detection system at the firewall connecting the local network to the Internet. The firewall can then defend against the attack by triggering a defense mechanism. Though such a specialized firewall can by disabled by a flood of 14,000 packets per second [3], we argue that by increasing the flooding rate to such a high value the attacker is increasing the probability of being tracked down. Also, as mentioned earlier, the LP based approach can easily be tuned using appropriate parameters to detect such high flooding rates and therefore can be deployed at the routers not only to detect the attack, but also to detect the source of high intensity attacks.

To detect the source of a SYN flooding attack during a low intensity attack, we can still make use of the difference in outgoing SYN and incoming SYN+ACK packets, though the detection delay will be longer. This happens as the the increase in Δ_s near the source will not be as evident as the increase in Δ_v near the victim, for small polling intervals. In addition, considering the fact that the sources of attack can be distributed in different networks, we need to analyse the traffic near the sources and find out a way to achieve detection of the source of SYN flooding attack. We also have to make analysis on how this approach can be used to detect other TCP based low intensity attacks.

TABLE I
DETECTION DELAY IN SECONDS FOR DIFFERENT ATTACK RATES

SYNs per second	Best delay	Worst delay
20	8	17
30	7	16
40	6	15
50-90	5	14
$\geq$ 100	4	13

REFERENCES

[1] "Computer Emergency Response Team, CERT Coordination Center, Denial of Service Attacks," http://www.cert.org/tech_tips/denial_of_service.html, Oct. 1997.

[2] L. Gordon et al., "CSI/FBI Computer Crime and Security Survey," http://i.cmpnet.com/gocsi/db_area/pdfs/fbi/FBI2004.pdf, 2004.

[3] David Moore, Geoffrey M. Voelker, and Stefan Savage, "Inferring Internet Denial-of-Service Activity," in *Proceedings of the 10th USENIX Security Symposium*, Aug. 2001, pp. 9–22.

[4] "Computer Emergency Response Team, CERT Advisory CA-1996-21, TCP SYN Flooding and IP Spoofing Attacks," www.cert.org/advisories/CA-1996-21.html, Sept. 1996.

[5] W. Richard Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, 1994.

[6] A. Ramasamy, Hema A. Murthy, and Timothy A. Gonsalves, "Linear Prediction For Traffic Management And Fault Detection," in *Proceedings of the International Conference on Information Technology, ICIT 2000*, Dec. 2000.

[7] D. J. Bernstein, "Syn cookies," http://cr.yp.to/syncookies.html.

[8] Jonathan Lemon, "Resisting syn flood dos attacks with a syn cache.," in *Proceedings of BSDCon 2002*, Feb. 2002, pp. 89–97.

[9] Checkpoint Inc, "TCP Syn Flooding Attack and the Firewall-1 Synde-fender," www.checkpoint.com/products/firewall-1, 1997.

[10] Christoph L. Schuba and Ivan V. Krsul and Markus G. Kuhn and Eugene H. Spafford and Aurobindo Sundaram and Diego Zamboni, "Analysis of a Denial of Service Attack on TCP," in *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, May 1997, pp. 208–223.

[11] L. Ricciulli and P. Lincoln and P. Kakkar, "TCP SYN Flooding Defense," in *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation (CNDS '99)*, 1999.

[12] Siris V. A and Papagalou. F, "Application of anomaly detection algorithms for detecting SYN flooding attacks," in *Proceedings of the IEEE GLOBECOM '04*, 2004, vol. 4, pp. 2050–2054.

[13] H. Wang, D. Zhang, and K. Shin, "Detecting SYN flooding attacks," in *Proceedings of IEEE INFOCOM 2002*, June 2002, pp. 1530–1539.

[14] Bin Xiao, Wei Chen, Yanxiang He, and Edwin Hsing-Mean Sha, "An Active Detecting Method Against SYN Flooding Attack," in *Proceedings of 11th International Conference on Parallel and Distributed Systems, ICPADS 2005*, July 2005, pp. 709–715.

[15] V. Paxson and M. Allman, "RFC 2988 - Computing TCP's Retransmission Timer," www.ietf.org/rfc/rfc2988.txt, Nov. 2000.

[16] "The Telecommunications and Computer Networking Group, Indian Institute of Technology, Madras," www.TeNeT.res.in.

[17] J. Makhoul, "Linear Prediction: A Tutorial Review," *Proceedings of the IEEE*, vol. 63(4), pp. 561–580, 1975.