Appreared in the proceedings of the Second International Innovations and Real-time Applications of Distributed Sensor Networks Symposium, Washington DC, September 2006.

A Novel Sliding-window Algorithm for Fast Detection of SYN Flooding Attacks on Distributed Sensor Networks

Kiran S. Balagani and Vir V. Phoha, *Senior Member, IEEE*

Abstract—**We present 'SYNFloodAlert,' a novel sliding-window algorithm to detect SYN flooding attacks on TCP based distributed sensor networks. In a SYN flooding attack, the number of SYN packets entering the TCP's incomplete connection queue is excessively greater than the number of packets leaving the incomplete connection queue. SYNFloodAlert detects SYN flooding attacks on a victim server by monitoring the discrepancy in the number of SYN packets entering and leaving the incomplete connection queue. SYNFloodAlert uses polynomial function approximation to generate two functions, one function to estimate the number of packets entering the incomplete connection queue and another function to estimate the number of packets leaving the incomplete connection queue. The function associated with packets entering the queue is used to predict number of packets leaving the queue with 'α' time lead, where 'α' is the service time of the incomplete connection queue. Similarly, the function associated with the packets leaving the queue is used to predict number of packets entering the queue with 'α' time lag. A SYN flooding attack is signaled when the difference between prediction errors of the two functions predicting the number of packets entering and leaving the queue crosses a predefined threshold.**

Experiments were conducted in an isolated environment with network traffic generated by SURGE Web traffic simulator, which simulated background TCP traffic of up to 400 Web clients. Multiple SYN flooding attacks were launched using scripts written in C language. Results of the experiments showed that the SYNFloodAlert algorithm, at 100% attack detection accuracy, has a false alarm rate as low as 0.021 and has an average detection delay of 116 seconds.

Index Terms— **Transmission control protocol (TCP), SYN flooding attack, polynomial function approximation, detection accuracy**

Manuscript received on September 1, 2006. This work was supported in part by the Army Research Office under Grant No. DAAD 19 – 01 – 0646.

Kiran S. Balagani is a PhD student in Computational Analysis and Modeling program at Louisiana Tech University, Ruston, LA 71272 USA (email: ksb011@latech.edu, phone: 318 – 514 – 9745).

Vir V. Phoha is a Professor of Computer Science at Louisiana Tech University, Ruston, LA 71272 (e-mail: phoha@latech.edu, phone: 318 – 257 – 2298).

I. Introduction

SYN flooding attacks are one of the most popular forms of denial-of-service attacks that exploit the TCP's connection mechanism. To initiate a TCP connection, the client sends a SYN packet to the server. The server creates a new entry for the SYN packet in the incomplete connection queue and responds by sending a SYN+ACK packet. The client acknowledges the server by sending an ACK packet, completing the three-way handshake. Most Berkeley-derived TCP implementations maintain an incomplete connection queue for each listening socket. An entry remains on the incomplete connection queue until the SYN+ACK packet is acknowledged by the client or until the entry times out. The connection timeout for each entry is typically 75 seconds [1]. The objective of a SYN flooding attack is to overwhelm the incomplete connection queue of a victim server by sending a large volume of SYN packets with spoofed IP sources. Because of a filled incomplete connection queue waiting for ACK packets from clients, the victim server rejects all incoming connections causing denial of service for new connections originating from legitimate sources.

There has been considerable debate (see papers [2, 3, 4, 5, 6, 7]) on using traditional TCP for reliable communication within wireless distributed sensor networks. The major drawback in using TCP based end-to-end connections is that the TCP adds an additional overhead with acknowledgements and retransmissions on the low powered and low memory micro-sensors used in sensor networks. While using or not using TCP within a sensor network presents a separate concern of its own, it is becoming increasingly important for wireless sensor networks that are destined for deployment in remote and harsh environments to connect to the Internet or other TCP based external networks for the purpose of data collection, data analysis, and most importantly, for event monitoring. In such scenarios, if an adversary

launches SYN flooding attacks on the device(s) connecting the sensor network to another external TCP network, all communications between the sensor network and the event monitoring systems are disrupted, making the sensor network virtually useless over the duration of the attack.

In this paper we present 'SYNFloodAlert', a sliding-window algorithm to detect SYN flooding attacks. The SYNFloodAlert algorithm detects SYN flooding attacks by monitoring the discrepancy between the number of TCP packets entering and leaving the incomplete connection queue when SYN flooding is in progress. The SYNFloodAlert algorithm first finds a polynomial function to approximate the number of SYN packets entering the incomplete connection queue. This function is called 'arrival function' and is used to predict the number of packets leaving the incomplete connection queue with α lead time, where 'α' is the service time of the incomplete connection queue. Next, the SYNFloodAlert algorithm finds another polynomial function to approximate the number of ACK (or RST) packets that release the SYN packets from the incomplete connection queue. This function is called the 'departure function' and is used to predict the number of packets entering the incomplete connection queue with α time lag. Prediction errors generated by the 'arrival function' and the 'departure function' when predicting the number of packets leaving and entering the incomplete connection queue are monitored. Large prediction errors of the 'arrival function' and the 'departure function' indicate that the rate at which packets enter the incomplete connection is greater than the rate at which packets leave the incomplete connection queue. The SYNFloodAlert algorithm signals a SYN flooding attack if the prediction errors are greater than a predefined threshold.

The SYNFloodAlert algorithm is a desirable tool for monitoring SYN flooding attacks on distributed sensor networks because: (1) SYNFloodAlert is *stateless*, i.e., it does not allocate resources on a per-connection basis to monitor SYN flooding attacks. The statelessness of the algorithm makes it resilient to SYN flooding; (2) SYNFloodAlert is *reliable* because it uses the discrepancy between the number of packets entering and leaving the incomplete connection queue to detect SYN flooding. This discrepancy is a foolproof signature of an effective SYN flooding attack in progress; and (3) SYNFloodAlert does not require separate signature-learning sessions to identify SYN flooding attacks and is therefore *immediately operational* when mounted on a remotely deployed sensor network.

The SYNFloodAlert algorithm was tested on datasets generated from seven simulation runs conducted in an isolated laboratory environment. Each simulation run involved launching several SYN flooding attacks on an Apache Web server handling HTTP requests from up to 400 simulated clients. In three of the seven simulation runs, a sudden increase in background TCP activity was generated by activating 100 additional clients accessing the server. On a single threshold parameter set for all the datasets, the SYNFloodAlert algorithm, at 100% detection accuracy, has a false alarm rate of 0.021 with an average detection delay of 116 seconds.

The rest of the paper is organized as follows. In Section 2 we present research related to SYN flooding attack detection and mitigation. In Section 3 we present data generation. In Section 4 we present the SYNFloodAlert algorithm. In Section 5 we present results of SYNFloodAlert. We conclude our work and give future directions in Section 6.

II. Related Research

In this section we briefly discuss the research related to detection and mitigation of SYN flooding attacks. We classify SYN flooding attack detection methods into two categories (1) 'state-based' methods, which maintain a per-connection state and (2) 'stateless' methods, which do not maintain any per-connection state. SYN Cache [8], SYN Cookies [9], SYN Kill [10], and SYN Proxying [11] are some examples of state-based methods to detect and mitigate SYN flooding attacks. SYN Cache replaces the per-socket linear incomplete connection queue with a global hash table. The 'cachelimit' parameter imposes an upper bound on the memory that the SYN Cache uses and the 'bucket size' parameter limits number of entries per hash bucket, bounding the time required for searching the entries. An entry overflow is handled by performing a first-in-first-out (FIFO) drop of entries on the hash list. SYN Cookies replace the SYN Cache's overflow handling mechanism by sending a SYN cookie instead of dropping an entry from the hash list. A SYN cookie contains the initial sequence number, which is returned in the final phase of the TCP's three way handshake. As connection establishment is performed by the returning ACK, a secret is used to validate the connection. The SYN Kill mechanism classifies addresses of all incoming packets into 'good' or 'evil' classes based on observed network traffic and the input supplied by the administrator. A decision process based on a

finite state machine determines the correct state membership of each incoming packet and sends RST packets in response to deter connection establishment attempts from 'evil' IP sources. SYN Proxying sets a threshold on the number of SYN packets passing through a firewall per second. When the number of incoming SYN packets reaches the threshold, the firewall proxies all incoming SYN packets by storing the incomplete connections in a queue. The incomplete connections remain in the firewall's queue until the connection is completed or until the request times out.

The major drawback of state-based methods to detect and mitigate SYN flooding attacks is that they allocate resources to monitor TCP connections. Resource allocations make the state-based methods inherently vulnerable to rapid exhaustion of the allocated resources when encountering flooding.

Wang *et al.* [12] and Blazek *et al.* [13] present stateless methods based on sequential and batch sequential change point detection theory to detect SYN flooding attacks. Wang *et al.* use the discrepancy between the SYN-FIN (RST) pairs to detect SYN flooding. The drawback of using the discrepancy between SYN-FIN pairs as a signature for detecting SYN flooding attacks is that an attacker can paralyze the detection mechanism by flooding a mixture of SYN and FIN packets. Moreover, change detection algorithms may be sensitive to daytime variations in TCP activity, causing the number of false alarms to increase when the attack triggering threshold is inappropriately selected.

III. Data Generation

Experimental setup for data generation consisted of seven Pentium IV PCs. One PC was configured as an Apache Web server that hosted 2000 files. Five PCs were used to run the Scalable URL Reference Generator (SURGE) program [14] to generate synthetic Web traffic workloads of up to 400 clients. One PC was used to launch SYN flooding attacks. The size of the incomplete connection queue, defined by the SOMAXCONN parameter of the Apache server, was set to 1024. SYN flooding attacks were launched using SENDIP tool. SENDIP [15] is a command line tool that uses Libpcap library to forge an arbitrary number of IP, TCP, UDP, and RIP packets. The SENDIP program was configured to flood the Apache server with 3500 SYN packets per minute. TCPDUMP program [16] was configured to collect all inbound and outbound TCP packets at port 80 of the Apache server.

Table 1. Details of datasets generated in seven simulation runs conducted in an isolated laboratory environment.

Dataset	Number of clients simulated	Number of clients added	TCP traffic duration in minutes	SYN flooding duration in minutes
1	100	--	51.25	16.00
2	200	--	54.00	23.00
3	300	--	59.00	22.00
4	400	--	58.25	22.50
5	100	100	66.25	22.75
6	200	100	74.25	21.75
7	300	100	78.00	22.50

Table 1 gives details of the datasets generated in seven simulation runs. Each simulation run involved generating normal TCP activity at the Apache server using SURGE clients and launching SYN flooding attacks on the Apache server at some point during its normal operation. Seven such simulation runs were performed, each simulation run having different levels of TCP traffic activity. The column 'Number of clients simulated' in Table 1 indicates the level of TCP activity generated by a pool of clients simultaneously accessing the Apache server. The 'Number of clients added' column gives the increase in number of clients accessing the Apache server when the server is already handling certain number of clients. For example, 'Dataset 5' initially contains TCP traffic generated by 100 clients. At some point during the same simulation run, an additional 100 clients were activated to generate sudden increase in normal TCP traffic activity. The reason for increasing the number of clients within the same simulation run is to test the robustness of the SYNFloodAlert algorithm over varying levels of TCP traffic activity. 'TCP traffic duration' in Table 1 gives the duration of a simulation run. 'SYN flooding duration' gives the duration for which the Apache server was SYN flooded in a simulation run.

IV. SynFloodAlert Algorithm

A. Notation

$a(t)$	Number of packets entering the incomplete connection queue every T seconds.
$d(t)$	Number of packet leaving the incomplete connection queue every T seconds.
D	Detection window.
E_a	Estimation window for packets entering the incomplete connection queue.
E_d	Estimation window for packets leaving the incomplete connection queue.
f_a	Polynomial function approximating $a(t)$ in the estimation window E_a . Also called 'arrival function'.
f_d	Polynomial function approximating $d(t)$ in E_d. Also called 'departure function'.

$P_{a\text{-}d}$	Prediction window for which *d(t)* is predicted using the polynomial function 'f_a'.
$P_{d\text{-}a}$	Prediction window for which *a(t) is* predicted using the polynomial function 'f_d'.
$Err_{a\text{-}d}$	Prediction error incurred when f_a is used to predict the points of *d(t)* in the prediction window $P_{a\text{-}d}$.
$Err_{d\text{-}a}$	Prediction error incurred when f_d is used to predict the points of *a(t)* in the prediction window $P_{d\text{-}a}$.

B. Dataset Preprocessing and Sampling

Datasets obtained from simulation runs were preprocessed to contain only the packets participating in the TCP connection establishment process, i.e., SYN, SYN+ACK, ACK, and RST packets. An incomplete connection queue was implemented to determine the number of packets entering and leaving the queue every T seconds. The queue was implemented as a linear list of size 1024, which corresponds to the SOMAXCONN parameter of the Apache server.

For each of the seven datasets, we obtain two time series (1) the time series '*a(t)*' gives the number of packets entering the incomplete connection queue every T seconds and (2) the time series '*d(t)*' gives the number of packets leaving the incomplete connection queue every T seconds. Here, T is the sampling time, set to 15 seconds in our experiments.

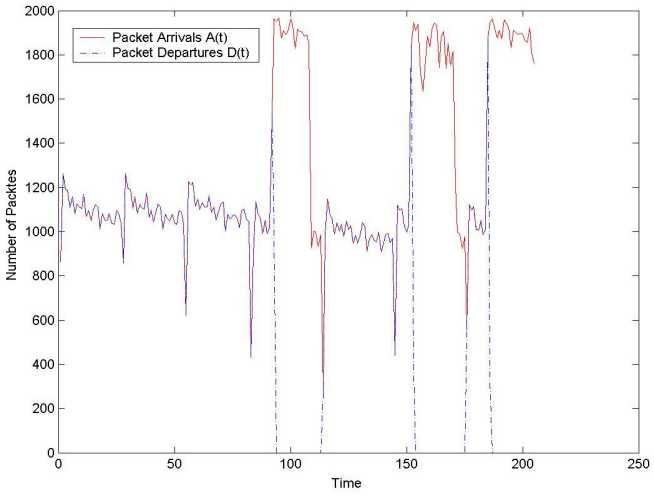

Figure 1. Number of SYN packet arrivals *a(t)* (solid line) and the number of packet departures *d(t)* (dotted line) in Dataset 1. Three SYN flooding attacks were launched in Dataset 1 at Time equal to 100, 150, and 200.

As an example, Figure 1 illustrates the results of preprocessing and sampling Dataset 1. The plots in Figure 1 show the number of packets entering the incomplete connection queue i.e., *a(t)* and the number of packets leaving the incomplete connection i.e., *d(t)*. Packets were sampled at T = 15 seconds. SYN flooding attacks were launched three times in Dataset 1 at Time equal to 100, 150, and 200. During these periods we observe (from Figure 1) that the number of packets entering the incomplete connection queue is excessively greater than the number of packets leaving the incomplete connection queue. The rest of the datasets were similarly preprocessed and sampled before running the SYNFloodAlert algorithm.

C. The SYNFloodAlert Algorithm

Here we give the steps in SYNFloodAlert algorithm for detecting SYN flooding attacks.

Input Preprocessed dataset with *a(t)* and *d(t)*.

Step 1 Set detection window counter 'i' as *1*. Initialize the sizes of the detection window D^i, estimation windows E_a^i and E_d^i, and prediction windows P_{a-d}^i and P_{d-a}^i.

Step 2 Set errors Err_{a-d}^i and Err_{d-a}^i to 0.

Step 3 Find a polynomial function f_a^i (arrival function) to approximate the number of packets entering the incomplete connection queue, i.e., *a(t)* within the estimation window E_a^i.

Step 4 Predict the number of packets leaving the incomplete connection queue, i.e., *d(t)*, in the prediction window P_{a-d}^i using f_a^i .

Step 5 Calculate the prediction error Err_{a-d}^i .

Step 6 Find a polynomial function f_d^i (departure function) to approximate the number of packets leaving the incomplete connection queue, i.e., *d(t)*, within the estimation window E_d^i.

Step 7 Predict the number of packets entering the incomplete connection queue, i.e., *a(t)*, in the prediction window P_{d-a}^i using f_d^i.

Step 8 Calculate the prediction error Err_{d-a}^i .

Step 9 If $\left|Err_{a-d}^i - Err_{d-a}^i\right| > \tau$, signal a SYN flooding attack.

Step 10 Increment i and repeat steps 2 - 9 until the end of the input dataset.

D. Polynomial Function Approximation

We use polynomial interpolation [17] to approximate the arrival and departure functions in Step 3 and Step 6 of the SYNFloodAlert algorithm. Given a set of points $(t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n)$ over the interval $[t_1, t_n]$, polynomial function interpolation finds an m^{th} degree polynomial of the form

$$f(t) = a + bt + ct^2 + dt^3 + \cdots + lt^{m-1}$$

so that the sum of squared error (*SSE*) given by

$$SSE = \sum_{j=1}^{n} \left(y_j - f(t_j)\right)^2$$

is minimal over the interval $[t_1, t_n]$. To find an m^{th} degree polynomial, it is required that n is at least equal to *m-1*. In the SYNFloodAlert algorithm, the arrival function f_a interpolates the number of packets entering the incomplete connection queue within the estimation window E_a and departure function f_d interpolates the number of packets leaving the queue within the estimation window E_d.

E. Prediction Errors

The prediction errors Err_{a-d} and Err_{d-a} are calculated in Step 4 and Step 7 of the SYNFloodAlert algorithm. The prediction errors of the functions f_a and f_d are given as:

$$Err_{a-d} = \max_{j=s+1}^{s+k} \left(\left| f_a(t_j) - d(t_j) \right| \right), \text{ and}$$

$$Err_{d-a} = \max_{j=s-1}^{s-k} \left(\left| f_d(t_j) - a(t_j) \right| \right)$$

where $a(t_j)$ is the number of packets arriving in the incomplete connection queue, $d(t_j)$ is the number of packets leaving the queue, '*s*' is the size of the estimation window E_a (or E_d), and '*k*' is the size of the prediction window P^i_{a-d} (or P^i_{d-a}).

F. Working of SYNFloodAlert Algorithm

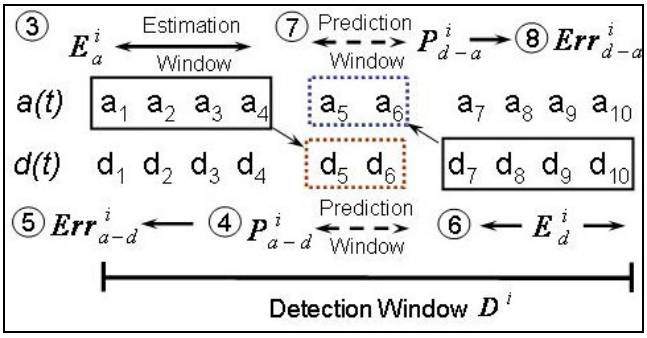

Figure 2. An iteration in the SYNFloodAlert SYN flooding attack detection algorithm.

Figure 2 illustrates steps 3-8 of the SYNFloodAlert algorithm. In Figure 2, the detection window counter *i* is initialized (to 1), the size of the detection window D^i is set to 10, the sizes of the estimation windows E_a^i and E_d^i are set to 4, the sizes of the prediction windows P_{a-d}^i and P_{d-a}^i are set to 2, and the prediction errors Err_{a-d}^i and Err_{d-a}^i of the SYNFloodAlert algorithm are set to 0. In the third step of SYNFloodAlert, we find the arrival function f_a^i that approximates function *a(t)* over the estimation window E_a^i. In the fourth step, we predict *d(t)* over the prediction window P_{a-d}^i using f_a^i. In the fifth step, we calculate the prediction error Err_{a-d}^i between the values predicted by f_a^i and the original values of *d(t)* in the prediction window P_{a-d}^i. In the sixth step, we find the departure function f_d^i to approximate the function *d(t)* over the estimation window E_d^i. In the seventh step, we predict *a(t)* in the prediction window P_{d-a}^i using f_d^i. In the eight step, we calculate the error Err_{d-a}^i between the values predicted by f_d^i and the original values of *d(t)* in the window P_{d-a}^i. High values of Err_{a-d}^i and Err_{d-a}^i indicate the failure of the functions f_a^i and f_d^i in predicting the number of packets entering and leaving the incomplete connection queue. In the ninth step of the SYNFloodAlert algorithm, a SYN flooding attack is signaled if the absolute difference between the errors Err_{a-d}^i and Err_{d-a}^i is greater than a predefined flooding threshold τ. Finally, the detection window D^i is moved forward and steps 3-9 are repeated.

V. RESULTS

In this section we present the results of the 'SYNFloodAlert' algorithm on the seven datasets discussed in Section 3. To gauge the performance of SYNFloodAlert, we consider three performance criteria (1) detection accuracy, (2) detection delay, and (3) false alarm rate. Detection accuracy is the ratio of the number of attacks detected to the total number of attacks in the dataset. Detection delay is the time elapsed between the launch time of an attack and the time when the SYNFloodAlert algorithm detects an attack. False alarm rate is the number of normal TCP connections identified as attacks.

For the results presented in this section, the size of the detection window D was set to 9. The sizes of estimation windows E_a and E_d were set to 4. The

prediction window sizes P_{a-d} and P_{d-a} were set to 1. The number of packets entering and leaving the incomplete connection queue was sampled at 15 second intervals. The attack detection threshold τ was set to 3000 for all the datasets.

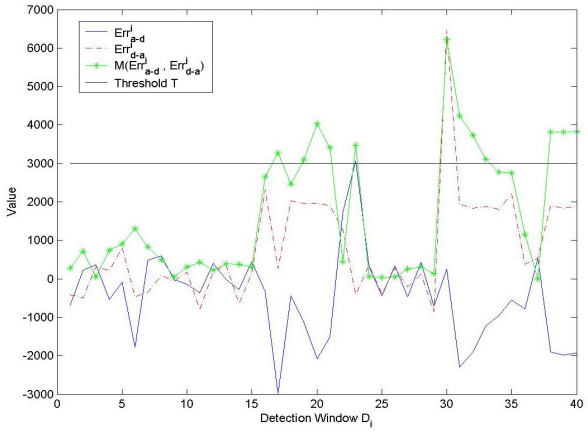

Figure 3. Prediction errors Err^i_{a-d} and Err^i_{d-a} when SYNFloodAlert was run on Dataset 1. The plot in solid line is Err^i_{a-d}. The plot in dotted line is Err^i_{d-a}. The '*' marked plot is the difference between the prediction errors.

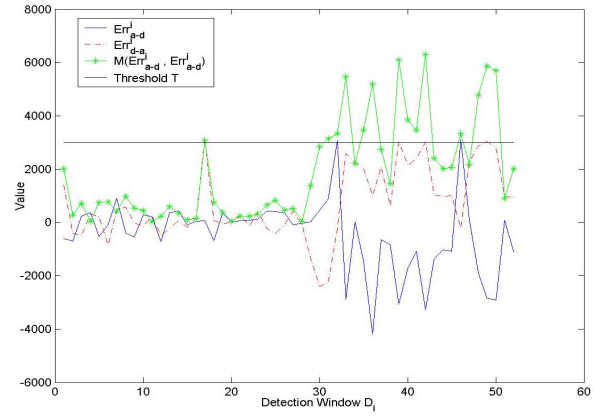

Figure 4. Prediction errors Err^i_{a-d} and Err^i_{d-a} when SYNFloodAlert was run on Dataset 5. The plot in solid line is Err^i_{a-d}. The plot in dotted line is Err^i_{d-a}. The '*' marked plot is the difference between the prediction errors.

The plots in Figure 3 and Figure 4 illustrate the variation in the prediction errors Err^i_{a-d} and Err^i_{d-a} when SYNFloodAlert was run on Dataset 1 and Dataset 5 respectively. The horizontal (solid) line in the figures indicate that the threshold value for signaling a SYN flooding attack was set to 3000. SYNFloodAlert signals a SYN flooding attack whenever the difference between the prediction errors is greater than the threshold**.** In Figure 3, the SYNFloodAlert algorithm signaled an attack three times (1) between the 15th and 20th detection window, (2) between the 20th and 25th detection window, and (3) between the 35th and 40th detection window. Similarly, in Figure 4, the SYNFloodAlert algorithm signaled an attack three times (1) between the 25th and 30th detection window, (2) between the 30th and 35th detection window, and (3) between the 35th and 40th detection window. Similar plots were obtained for the remaining datasets. Further insights on the performance of the SYNFloodAlert algorithm are obtained from Table 2.

Table 2. Results of SYNFloodAlert on the seven datasets. Attack Start Time is the time at which SYN flooding was launched on the Apache server. Detection time is time at which SYNFloodAlert detected an attack. Detection delay is the time difference between the Attack Start Time and Detection Time.

Dataset	Attack Start Time	SYNFloodAlert Detection Time	Detection Delay (min)	% False Alarms
1	23.75	26.25	2.25	2.50
	38.25	38.75	0.50	
	46.25	48.75	1.50	
2	28.00	28.75	0.75	2.37
	36.25	37.50	1.25	
	46.25	47.50	1.25	
3	12.75	13.75	1.00	0
	23.00	23.75	0.75	
	32.25	33.75	1.50	
4	22.00	23.75	1.75	0
	30.75	31.25	0.50	
	41.00	42.50	1.50	
5	40.00	40.00	0.00	1.92
	48.50	50.00	1.50	
	58.50	58.75	0.25	
6	47.25	48.75	1.50	3.42
	57.50	60.00	2.50	
	66.75	67.50	0.75	
7	51.50	53.75	2.25	4.88
	60.25	62.50	2.25	
	70.50	71.25	0.75	

Table 2 gives the attack start times, attack detection times, attack detection delays, and the percentage of false alarms obtained when the SYNFloodAlert algorithm was run on the seven datasets. The false alarm rate and detection delay for the datasets was recorded at 100% detection accuracy i.e., when SYNFloodAlert detected all occurrences of SYN flooding attack in the datasets. From the values in Table 2 we calculated the average number of false positives and the average detection latency of the SYNFloodAlert algorithm. The average number of false alarms recorded over seven hours of normal TCP activity was 0.021 and the average detection latency over 21 attack instances was 116 seconds.

VI. CONCLUSIONS

In this paper we presented a new sliding-window algorithm called 'SYNFloodAlert' for fast detection

of SYN flooding attacks on TCP based distributed sensor networks. We showed that the SYNFloodAlert algorithm is reliable because it uses the discrepancy between SYN packets entering and leaving the incomplete connection queue, which is a foolproof signature of an effective SYN flooding attack. Further, the SYNFloodAlert algorithm was tested on seven datasets generated in an isolated laboratory. The datasets contained SYN flooding attacks launched on Apache Web server handling varying workloads of TCP traffic generated by up to 400 simulated clients. The SYNFloodAlert algorithm was effective in detecting all 21 attacks launched on the Apache server with an average detection delay of 116 seconds. Future work in this research will combine the SYNFloodAlert detection algorithm with an intelligent data structure to effectively mitigate the denial of service caused by a SYN flooding attack.

REFERENCES

[1] J. Postel, *"RFC 793: Transmission control protocol."* September, 1981. Status: STANDARD.

[2] H. Balakrishnan, S. Seshan, E. Amir and R. H. Katz. *"Improving TCP/IP Performance over Wireless Networks."* In proceedings of the 1st ACM Conference on Mobile Computing and Networking, Berkeley, CA, USA, November 1995.

[3] X. Luo, K. Zheng, Y. Pan and Z. Wu. *"A TCP/IP implementation for wireless sensor networks."* In proceedings of the 2004 International Conference on Systems, Man, and Cybernetics, Hague, Netherlands, October, 2004.

[4] A. Dunkels. *"Full TCP/IP for 8-hit architectures."* In proceedings of the 1st International Conference on Mobile Systems, Applications, and Services (MOBISYS'03), San Francisco, CA, USA, May 2003.

[5] A. Dunkels, J. Alonso, T. Voigt and H. Ritter. *"Distributed TCP Caching for Wireless Sensor Networks."* In proceedings of the Third Mediterranean Workshop on Ad Hoc Networks(Med-Hoc Net 2004), Bodrum, Turkey, June 2004.

[6] A. Dunkels, T. Voigt and J. Alonso. *"Making TCP/IP Viable for Wireless Sensor Networks."* In proceedings of the 1st European Workshop on Wireless Sensor Networks, Berlin, Germany, January 2004.

[7] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler and J. Anderson. *"Wireless sensor networks for habitat monitoring."* In proceedings of the 1st ACM Workshop on Wireless Sensor Networks and Applications, Atlanta, GA, USA, September 2002.

[8] J. Lemon. *"Resisting SYN flood DoS attacks with a SYN cache."* In proceedings of the 2002 USENIX BSD Conference, San Francisco, California, February 2002.

[9] D. J. Bernstein and E. Schenk. *"Linux Kernel SYN Cookies Firewall Project."* Online resource at cr.yp.to/syncookies.html. Last accessed on August, 2006.

[10] Netscreen 100 Firewall Appliance. Online resource at www.netscreen.com. Last accessed on August, 2006.

[11] C.L. Schuba, I.V. Krsul and M.G. Kuhn. *"Analysis of a Denial of Service Attacks on TCP."* In proceedings of 1997 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, May 1997.

[12] H. Wang, D. Zhang and K.G. Shin. *"Detecting SYN Flooding Attacks."* In proceedings of IEEE Infocom'2002, June 2002.

[13] R.B. Blazek, 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."* In proceedings of the 2001 IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point, NY, June 2001.

[14] P. Barford and M. Crovella. *"Generating representative Web workloads for network and server performance evaluation."* In proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, WI, USA, November 1998.

[15] Project purple. Online resource at: www.earth.li/projectpurple/progs/sendip.html. Last accessed on August, 2006.

[16] TCPDUMP Public Repository. Online resource at www.tcpdump.org. Last accessed on August, 2006.

[17] R. L. Burden and J. D. Faires. *Numerical Analysis*, Edition 7, Brooks/Cole, Pacific Grove, CA, USA

Kiran S. Balagani received the M. S. degree in computer science from Louisiana Tech University, Ruston. His research interests are anomaly detection, network security, pattern recognition, and Web mining. Mr. Balagani is currently pursuing PhD degree in computational analysis and modeling at Louisiana Tech University.

Vir V. Phoha (M'96–SM'03) received the M.S. and Ph.D. degrees in computer science from Texas Tech University, Lubbock. He is a Professor of Computer Science at Louisiana Tech University, Ruston. His research interests include anomaly detection, network and Internet security, Web mining, control of software systems, intelligent networks and nonlinear systems. Dr. Phoha is a member of the ACM.