NEMESI:Using a TCP Finite State Machine against TCP SYN Flooding Attacks

A. Gemona, I. Duncan, A. Miller
University of St Andrews
School of Computer Science
{ag,ishbel,alanr}@dcs.st-and.ac.uk

Abstract - Over the last few years the Internet has seen a continuous rise of malicious traffic. These include the Denial of Service (DoS) attacks, viruses, Trojans, spam mails and worm attacks. In this paper we focus on experiments with TCP SYN flooding attacks. We introduce a new approach to prevent such attacks based on passive monitoring of the frequency of TCP SYN packets and peak intervals, with respect to other packets, in combination with a dynamically adapted connection drop time.

I. INTRODUCTION

A Denial of Service (DoS) attack is a type of network attack that is designed to make the services unavailable to users. The effects of DoS [1] and the effects range from slow-down or unavailability of a network service to financial losses[2]. Typically a business will measure the financial cost of an attack based on the period that the services are down[3].

In Denial of Service (DoS) the source of the attack is a single host while in the Distributed Denial of Service (DDoS) multiple sources target the same destination. Both forms of attack have the same result but not always the same network traffic behaviour.

In this research we experiment with flood attacks based on TCP SYN packets. We focus on TCP flooding attacks because this is the most common type of attack according to the CAIDA [4] project which shows that. The 90% of the DoS traffic is TCP-based attacks. It is commonly believed that the existence of TCP SYN flooding attacks is based on a protocol design weakness [5] in combination with limited operating system resources (such as queue length and buffer space).

In this paper we propose the NEMESI system for defending against TCP SYN flooding attacks. We describe the system architecture and present experimental results based in defending against attacks from the TFN (Tribe Flood Network) flooding tool. We propose a detection algorithm based on TCP Finite State Machine modelling and monitoring of the peak intervals of TCP SYN packets.

The remainder f the paper Is structured as follows. Section 2 described the problem area and presents related work. Sections 3 to 8 describe the layers architecture of the system. Section 9 presents experimental results for the validation of the system and section 10 presents conclusions and further work.

II. PROBLEM DEFINITION AND RELATED WORK

The goal of any solution to SYN flooding attacks is to continue to accept connection requests while under attack. There are several ways to achieve this goal, but none of them is highly effective.

In a standard TCP implementation each packet received is temporarily buffered in a receive queue for processing [6]. As connections are initialised by the TCP stack, queue space is freed for new connections. During a SYN flood attack a large amount of SYN packets are received and the queue becomes full of pending requests. At this point any further SYN packets received are dropped by the server.

The maximum length of time a packet can spend in the queue is determined by a timeout value. A connection request will be dropped by the server if the three-way handshake has not completed within the timeout period.

Increasing the length of the queue and reducing the timeout value will reduce the effects of such attacks. However making the queue longer requires more memory and since the rate of packet arrival will still exceed the rate at which connections can be processed, the queue will eventually become full again. Shortening the timeout period adversely affects the establishment of real connections by enforcing a shorter window for the establishment handshake to take place.

Both of these can be classified as brute-force solutions that may waste of kernel memory and computation resource (slowing down the server's response time) but will be effective for some network configurations and attack types.

One solution to reduce the threat of SYN flood attacks is to reduce the amount of information stored for each in-progress connection. In turn this means some modification to the TCP communication protocol. This involves checking how valid the information is, as there may be spoofed packets. To apply the above solution, process power and memory space are essential. By monitoring traffic at the network gateway we can check the validity of the information that the target host is receiving and detect the DoS without using resources of the t host.

We categorize the solutions against such attacks based on the approach used. These will include the algorithm, the network installation place where the defence is applied and the level of defence provided. We choose Bro[7], Snort [8] and D-WARD [9] from intrusion detection systems (IDS)

Bro is a stand-alone intrusion detection system (IDS) that observes network traffic and that characteristic makes

it a network intrusion detection system (NIDS). Snort is a lightweight anomaly based NIDS that can be installed anywhere in the network with the consideration of what to protect. Both use the *libpcap* packet capture library to sniff packets. *libpcap* is extensively used by Bro, Snort and SHADOW [10] and appears to be a valid, well-known and reliable capturing library.

Bro uses an event engine to look up the connection state. The structure of the system is layered and that gives compatibility to the design and algorithm. The detection is based on policies that are defined by the user to recognize general malicious traffic. Snort uses rules and does pattern matching to detect attacks and probes in real time and issues alerts.

D-WARD (**D**DoS Net**w**ork **A**ttack **R**ecognition and **D**efence) is a system that is located at the source network router. It observes outgoing and incoming traffic and uses a statistical approach to define legitimate traffic patterns. The system attempts to separate attacking flows from legitimate flows and therefore identify the attacking machines.

III. STRUCTURE OF THE PROPOSED SYSTEM

NEMESI[1] is located in the gateway. The point of detection and the point of action have essential importance for DoS[11]. Therefore applying monitoring at the edge of the network and specifically at the gateway can provide defence against SYN flood. This section is dedicated to describing the levels of the system architecture, the algorithms and experiments that show the performance.

The system architecture is separated in levels (Figure 1) based on the functionality. In detail the lower level is the **monitoring**, where packets are captured from the network. Next is the second level, where the connections are identified as new or old through the **connection modelling**. Here the connections pass through the Finite State Machine (FSM) and data is gathered, for instance the frequencies of SYN (synchronization), SYN/ACK (acknowledgement of the synchronization) and ACK (acknowledgement) packets. Data is checked from the **detection** algorithm which is in the third level. The fourth level is the **action.** After an attack is detected, a reset packet is sent to the host to reallocate the service. An extensive description of each of the levels is given below.

A. The Monitoring level

In this section we underline two main characteristics of the functionality of this system level. One is the way the packets are captured and the second is which traffic are captured, as we are aware of incoming and outgoing traffic, internal and external traffic.

We use libpcap in order to sniff packets form the network. We attempt to make the system compatible and fast. We found libpcap a reliable library because we gain portability for NEMESI to different UNIX based OS.

There are two types of traffic that we see at the gateway, incoming and outgoing. Firewalls for instance can provide security by blocking the malicious packets from the incoming traffic; those that come from the Internet. In our approach we attempt to monitor both types of traffic. With the use of ebtables we can bridge the interfaces on the gateway. One interface, eth1, is for the traffic inside the protected network and the eth2 is the interface that listens to the traffic that comes from the outside of the protected network. By bridging the two interfaces, both traffics can be monitored.

B. The Connection modelling level

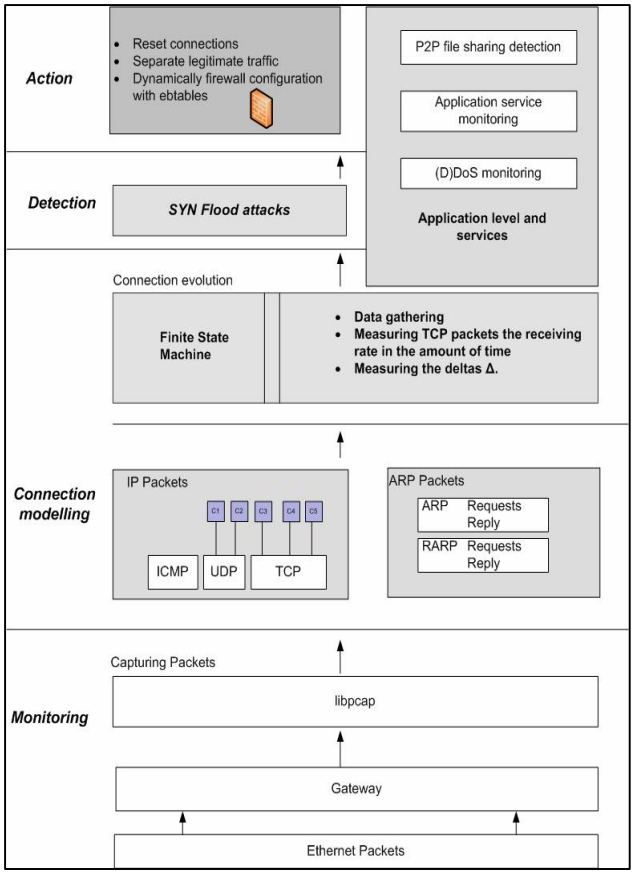

Figure 1 Level architecture of the system

On this section we examine the receiving frequency of SYN packets, the intervals of TCP packets and the type of connections (new or exciting connections). There are two options for a packet when it arrives. One is that the packet belongs to a connection that is already known to the NEMESI (that means it is in the connection's list) and the other is that the packet belongs to unassigned connection in the NEMESI's list. Below there is a representation of the connection modelling algorithm in pseudo code form.

[1] Nemesi was the Greek goddess of punishment.

Connection modelling pseudo code is :

1. listElement=search_List(ConnectionList,saddr,th_sport,daddr,th_dport,th_seq);
2. theConn=listElement==NULL?NULL:listElement->data;
3. if(theConn==NULL){
   Conn newConn=newConnection (saddr, daddr, th_sport, th_dport, th_flags, th_ack, th_seq);
   insertResult=insertToList(ConnectionList,newConn);
   if(insertResult==NULL){
   error;
4. FSM(theConn,ip,tcp);

The number of SYNs and other types of TCP packets received are countered. The deltas of the SYNs and total TCP traffic are also measured. These measurements will give data to use for the later level of the system where the detection is taking place. The pseudo code is presented below.

Data gathering pseudo code is:

1) For every TCP packet
   If SYN
   Num_currentSYNs ++;
2) Current_ΔSYNs= Num_currentSYNs - Num_previousSYNs;
3) Current_ΔTCPpackets=Num_currentTCPpackets – Num_previous_TCPpackets;
4) Δ (deltas) are saved per unit time

This pseudo code demonstrates the 1st step in the counter for the SYNs while there is a counter for every type of packet. The differences, Δ, are saved periodically every 2ms. The FSM output returns the number of connections in various states to determine the traffic load.

C. The TCP Finite State Machine (FSM)

As the main characteristic during the SYN flood attack is the multiple initializations for connections requests, a finite state machine has been build to give states. The connection state is SYN_RCVD when a synchronization request arrives. A large number on the SYN_RCVD state could determine the SYN flood attack. The importance of checking the connection's state is vital. Based on the RFC 793 (TCP state transition diagram), a new Finite State Machine (FSM) is designed for the NEMESI,[12]. The states describe the sender, the receiver and the NEMESI state. The TCP state transition diagram is based on the two end applications, sender side and receiver side. Because NEMESI is in the middle of the sender and receiver, new states are created to specify the status of the connection but can also be used to extract information for the general network status.

We use C++ to implement the algorithm. The FSM function has a defined type `Conn`, where `*Conn` is the structure containing the packets fields (for instance source ports, source IPs). The FSM( ) takes as arguments the standard headers structures `const struct iphdr *ip, const struct tcphdr *tcp` and the return value from the function that detects if there is any history of that connection or it is new. Then the TCP header flag is examined. A SYN packet for instance will be checked as in: `if (tcp->th_flags & TH_SYN)`. Then it is checked for other flags to be on for example if it is an SYN/ACK packet, the SYN and ACK flag is on. This check is accuracy can be used to detect spoofed packets if the flags that are on have no meaning. An attacker who is building packets may use combination of flags that is not valid and that could be detected. The attacker's tool will send a number of packets in order to reserve resources. This will be observed from the monitor at this point with this check. After the flag interpretation the algorithm checks if the connection is new or has history by at the argument passed. When the connection exists is checks the previous state, for accuracy, and then gives a new state.

D. The Detection level

In this section we discuss the design principles of the detection level, the algorithm and the experimental setup where attacks are launched.

NEMESI is separated in to layers. Each layer has a specific applicability. Despite the layered system the defence algorithm is designed in a way that is strongly bounded with the previous levels. The FSM output and the counting results from the connection modelling level is needed on the detection layer. By measuring the FSM output we get the number of connections in various states and in this case the number of connections in SYN_RCVD state. This is compared to the limit of connections on SYN_RCVD state. The rate of SYN packets per 2ms is calculated at the monitoring level and the values are examined on that level for any anomaly.

The pseudo code for the detection algorithm is:

A. CompareΔSYN=compare(Current_ΔSYNs, Previous_ΔSYNs);
B. CompareΔTCP=compare(Current_ΔTCPpackets, Previous_ΔTCPpackets);
C. If (CompareΔSYN AND CompareΔTCP increasing)
D. If (SYN_ACK packets are not increasing with the same rate)
E. Check the number of connection on SYN_RCVD state
F. Check the number of connection on ESTABLISHED state

The above presentation of the detection algorithm is an overview of it. When the attack is detected then we need to know from where the attack is coming from (internal or external) in order to act against it. The ebtables are configured to bridge the external and internal interfaces to find the direction of the attack. If an attack is detected then the action algorithm makes the decision for sending reset packets to reallocate the service.

E. Action level

The action level is the last level of the system. NEMESI aims to block the malicious traffic and continue the functionality of the network without bringing it down. During the screening process, between the detection and action part of the system; alerts are triggered. These are mainly:

The system receives a high traffic load that contains connection requests with the same source or multiple IPs. This defines an increase of the SYN rate.

There is an anomaly between the numbers of connections in states.

When a SYN flood is detected then, in order to block these requests, reset (RST) packets are sent to the IPs that request connectivity. The real requests would try to reconnect. If there is no reply, then the malicious packets have been stopped in the gateway. The system does not use or execute any of the TCP implementation data structures. It uses the information for the received packets and its own system data structures which keep the information of the received packets and extracted data. This saves heavy kernel procedures.

IV. ANALYSIS RESULT

In order to determine how fast NEMESI detects the attacks or if there are any false positives or characteristics of real flooding traffic traces, attacks have to be launched. We configure an experimental network to launch real attacks with prototype tools. Tribe flood network (TFN), stacheldraht and Trin00 are attacking tools that could cover most of the attack cases. Those tools have complicated structure and a study for Tribe Flood Network (TFN) and stacheldraht has been done from Dittrich[13], [14]and for Distributed Denial of Service tools from Dietrich[15]. The TFN has been used in our research work (figure 2a) to launch attacks in the network configured in Figure 2b.

All the machines have FEDORA Core installed. The attack is initialized at the first level. In the second level Zombie machines send the bogus requests. Zombies are "clients" that are infected with a client version of the attacking tool. These machines run a client application; in our case we run TFN the client application td. The Zombie machine does not send bogus packets until the master/control machine sends an acknowledgement to start the attack. These applications may have different versions of building packets.

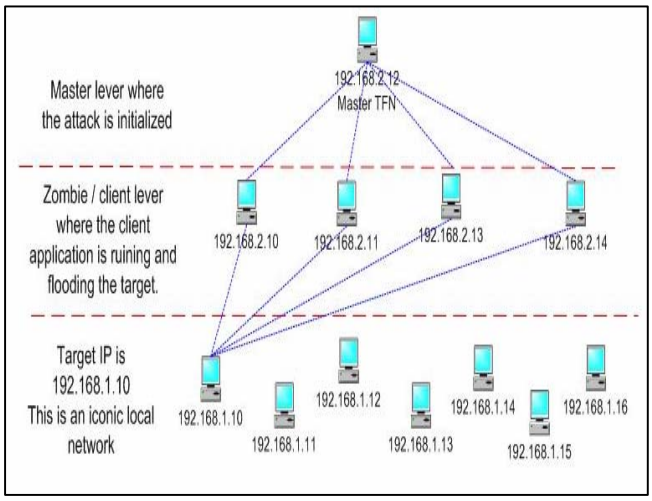

Figure 2a. TFN scheme

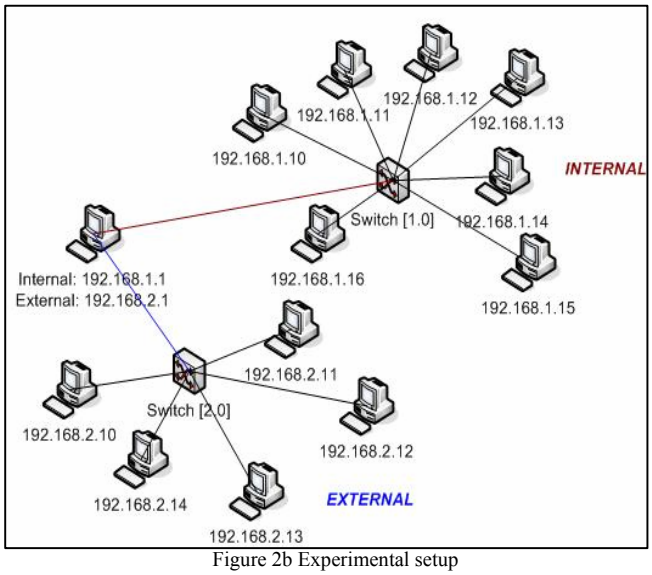

Figure 2b Experimental setup

Running experiments with TFN we succeed to monitor malicious traffic. NEMESI's results are shown below in Figure 3 and Figure 4.

Figure 3a: A sample of departmental web server traffic during term time. This traffic is characterised as normal. This figure shows the evolution for opening and closing connections. The numbers for FIN_ACK, SYN and SYN_ACK packets follow the same patterns.

Figure 3b: The ratio of the number of connections in the SYN_RCVD state to the number of connection in all states. A variable ratio is observed. This is expected, because the number of SYN packets has to be close or the same to the number of connection on SYN_RCVD state. These measurements are for the same sample of traffic (figure 3a, b).

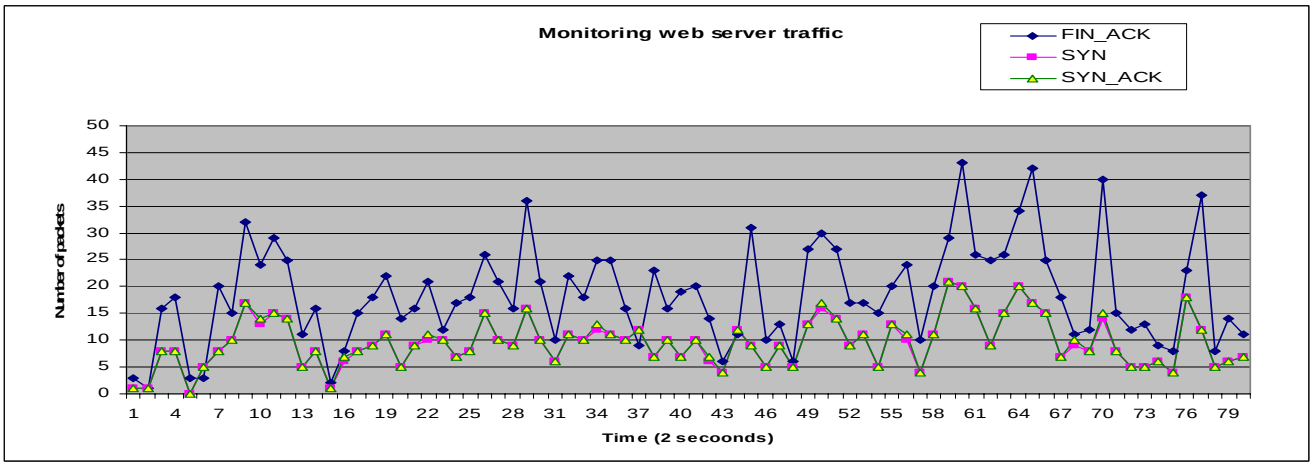

Figure 3a A sample of the departmental web server traffic

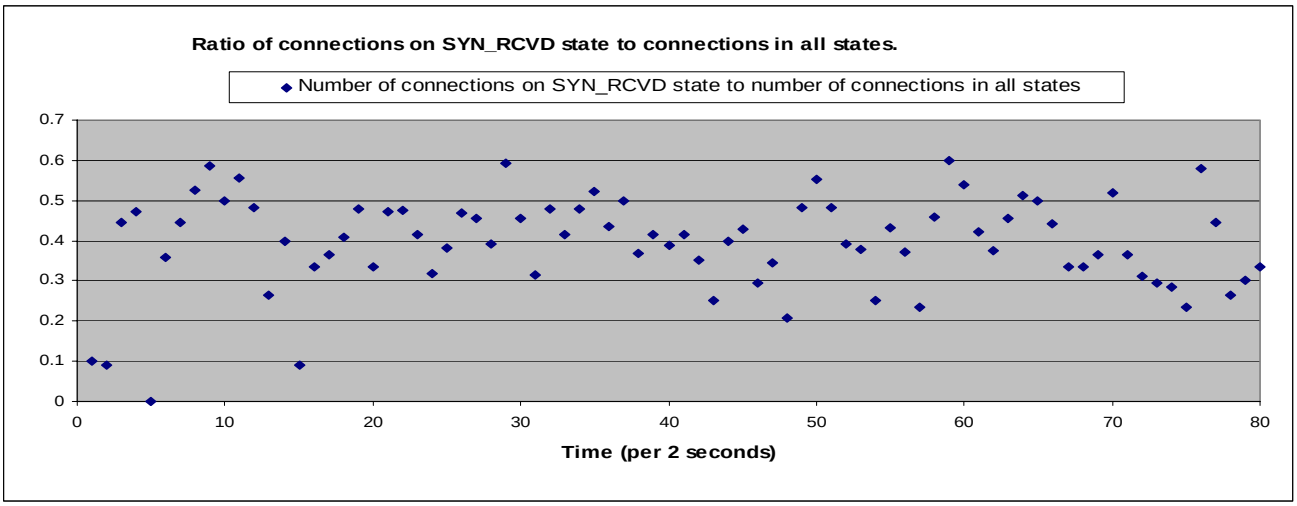

Figure 3b The ratio of connections in the SYN_RCVD state to number of connections in all states

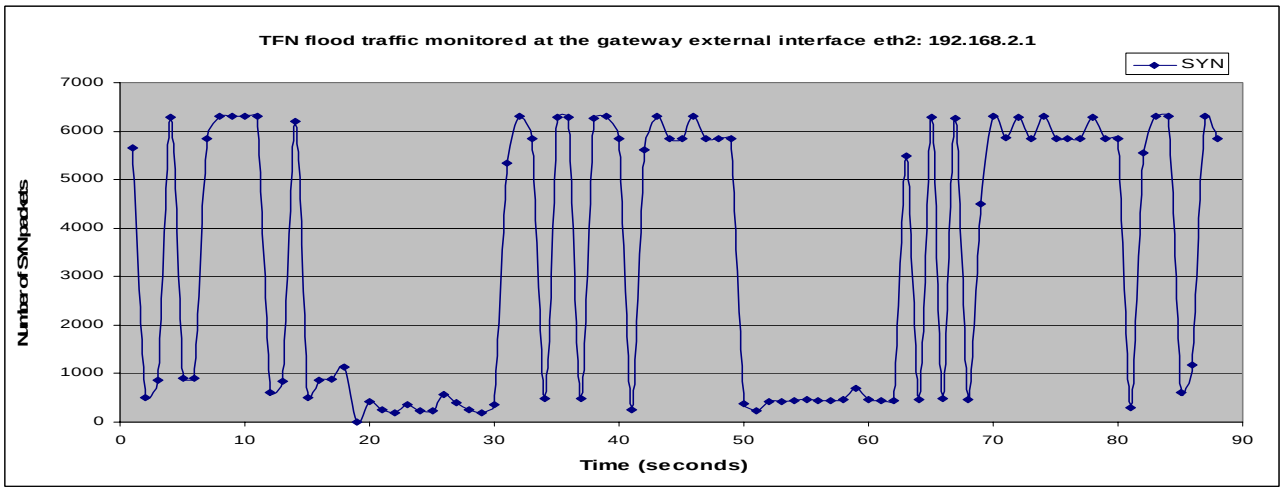

Figure 4a SYN flood attack traffic

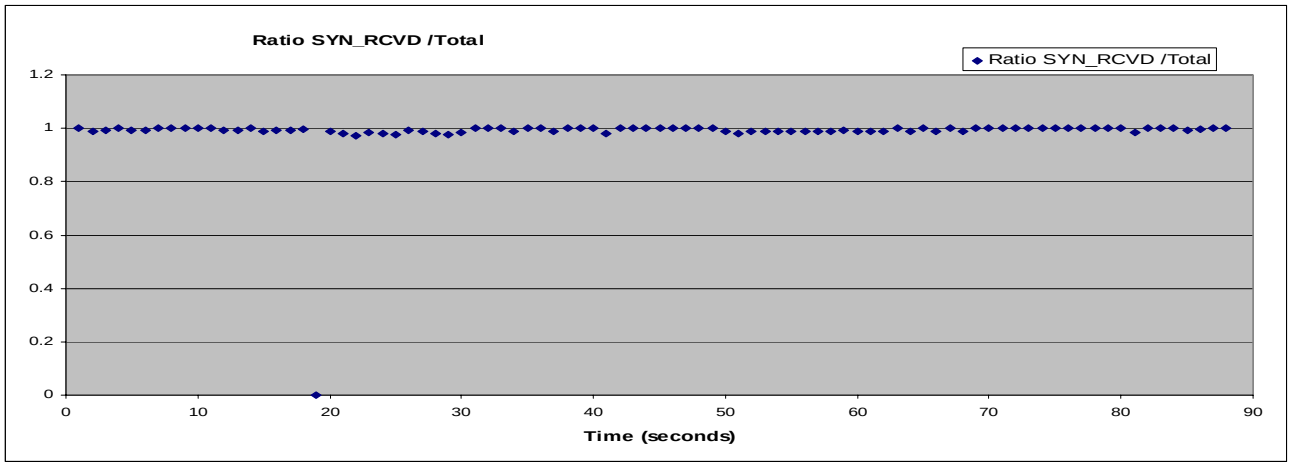

Figure 4b The ratio of connections in the SYN_RCVD state to number of connections in all states.

Figure 4a: This snapshot of traffic is captured from the gateway point, during SYN flood attack. This type of attack is characterized as a pulse attack [9] because it sends floods periodically till the services are overwhelmed. The service that is attacked is the http (web service).

Figure 4b: Shows the ratio of number of connections in the SYN_RCVD state to the number of connection in all states. A large amount of connections is saved under the SYN_RCVD state at the beginning of flood and remain until the end of data capture.

V. CONCLUSIONS

NEMESI is designed to use a combination of network and host approaches against TCP SYN flood attacks. The system uses passive monitoring at the gateway and a TCP Finite State Machine to generate states for connections. It is a small, flexible, and compatible system with the capability of detecting other types of flooding attacks as an extension of this work. It does not require any modification in either hardware or software. The system can distinguish SYN Flood attacks from Flash Crowd (FC). In order to test the detection algorithm we launch real attacks with prototypes tools to get traffic patterns. We use these patterns as attack conditions for the algorithm. Launching real attacks we can measure the performance of the system and examine for any false positives.

Characteristics of the functionality:

1. Sniffing gateway traffic using libpcap. The implementation of the software is in C++.
2. Saving the host and network resources by applying a defense at the gateway
3. The use of TCP Finite State Machine (FSM) is essential for NEMESI for flooding detection. In extension the detection algorithm uses the SYN rates, arriving intervals and does anomaly traffic pattern recognition.
4. Intrusion detection systems can detect attacks without necessarily blocking the attack traffic. We aim to distinguish the malicious traffic from the legitimate traffic and stop it at the gateway.

REFERENCES

[1] OUT-LAW.COM, P.M. MP pitches Denial of Service law to Parliament, Journal article, 2005, www.channelregister.co.uk/2005/03/10/mp_pitches_denial_of_service_law_to_parliament/

[2] Kleinbard, D. eBay, Buy.com, CNN.com and Amazon come under attack; FBI probes Yahoo! incident, 2000 http://money.cnn.com/2000/02/08/technology/yahoo/

[3] Leyden, J. Blaster copycat author jailed for 18 months, http://www.theregister.co.uk/2005/01/31/blaster_kiddo_sentencing/

[4] Moore, D. Inferring Internet Denial-of-Service Activity CAIDA. in Proceedings of the 10th USENIX Security Symposium Washington, D.C., USA. 2001

[5] Glenn Carl, G.K., Richard R. Brooks, and Suresh Rai. Denial-of-Service Attack-Detection Techniques. 2006

[6] Wright, G.R. and W.R. Stevens, TCP/IP Illustrated , Volume 2: Addison - Welsey. 1171.

[7] Paxson, V., Bro : A system for detecting network intruders in real-time. 1998, Lawrence Berkeley National Laboratory: Berkley

[8] Roesch, M., SNORT - Lightweight intrusion detection for networks. 1999, Proceedings of LISA '99: 13th Systems Administration Conference.

[9] Mirkovic, J., D-WARD: Source-End Defence Against Distributed Denial-of-Sevice Attacks. 2003, University of California: Los Angeles.

[10] Northcutt, S. About the SHADOW Intrusion Detection System,http://www.nswc.navy.mil/ISSEC/CID/, Naval Surface Warfare Center, 1998

[11] Adam Greenhalgh, M.H., filipe Huici, Using Routing and Tunneling to combat DoS attacks. 2005, University Colleg London.

[12] Gemona Anastasia, A.M.R., Ishbel Duncan, Colin Allison, END TO END DEFENCE AGAINST DDOS ATTACKS. 2004, University of St Andrews, School of Computer Science: IADIS in Spain

[13] Dittrich, D., The "Tribe Flood Network" distributed denial of service attack tool. 1999, University of Washington.

[14] David, D., The "stacheldraht" distributed denial of service attack tool. 1999, University of Washington: Washington

[15] Detrich S., L.N., Dittrich, D. Analyzing Distributed Denial Of Service Tools: The Shaft Case. 2000: Proceedings of the 14th Systems Administration Conference (LISA 2000) New Orleans, Louisiana, USA December 3– 8, 2000