www.revistageintec.net
ISSN: 2237-0722
revista GEINTEC Gestão, Inovação e Tecnologias

An Indigenous Solution for SYN Flooding

Muhammad Junaid[1]; Fahad Ali Khan[2]; Ali Imran Jehangiri[3]; Yousaf Saeed[4]; Mehmood Ahmed[5]; Luqman Shah[6]; Muhammad Naeem[7]

[1]Department of IT, The University of Haripur, Pakistan.

[1]mjunaid@uoh.edu.pk

[2]Department of IT, The University of Haripur, Pakistan.

[3]Department of IT, Hazara University, Mansehra, Pakistan.

[4]Department of IT, The University of Haripur, Pakistan.

[5]Department of IT, The University of Haripur, Pakistan.

[6]Department of IT, The University of Haripur, Pakistan.

[7]Department of IT, Abbottabad University of Sciences and Technology, Abbottabad, Pakistan.

Abstract

SYN flooding is one of the most challenging problems that many networks applications face, particularly those that are security-related. Disrupting a server's daily function and assigning it to other tasks leaves it a constantly busy server that processes little usable data. In this research, a comprehensive INDIGSOL approach is demonstrated that not only detects SYN flooding but also prevents the attacker(s) from making such attempts in the future. The designed approach has four modules such as node registration and validation, packet capturing, dynamic check system, and hook activation. The approach is further checked and compared with some state-of-the-art baselines on various parameters like detection time, response/processing time, and number of malicious packets detection. It is observed that INDIGSOL performed better than other baselines with an average accuracy of 99% malicious packet detection in six scenarios along with 13.4% faster detection time and 11.2% faster response/processing time. Overall, the provided solution is scalable, robust, and highly accurate that prevents SYN flooding in a timely manner.

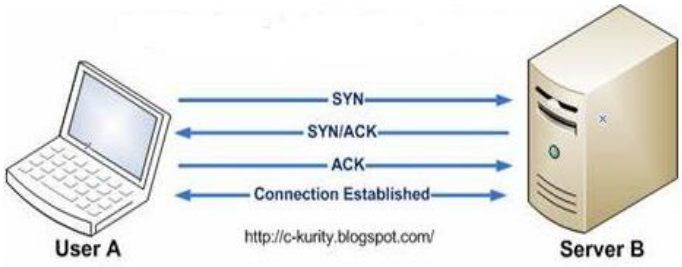
Keywords: Network, SYN Flooding, Protocol, Performance.

1. Introduction

TCP is a stable, connection-oriented transport protocol. The TCP's job is to keep the network's intricacies hidden from the upper layers. Until data can be shared, the two hosts involved in a conversation must first create a link, according to the connection-based protocol. The three-way

handshake is used in TCP to do this. Data sequencing and acknowledgment are the only two issues that we are dealing about when it comes to reliability [1]. Per byte in every segment is assigned a sequence number, and TCP accepts all data bytes obtained from other end. ACKs ingest a series number but are not accepted themselves, which is absurd [1]-[3]. TCP requires hosts to create a link in order to share data. The 3-way handshake is a three-step mechanism used by TCP to create a bond. The method is seen in Fig.1 in which machine A is running a client program and it wants to bind to a server program on machine B. At (1), the client requests a connection from the server. This is the sole objective of the SYN flag. The client informs the server that the sequence number field is correct and should be double-checked. The client can use its ISN (Initial Sequence Number) to configure the sequence number field in the TCP header. The server will answer with its own ISN (thus the SYN flag is set) and an acknowledgement of the client's first segment (which is the client's ISN+1) after receiving this segment (2).

Figure 1 - TCP Three-way Handshake



After that, the client acknowledges the server's ISN (3) and now data transmission is possible [4]. There are situations in which a packet is sent from client to the server and before taking acknowledgement, the connection is broken making it half open TCP connection. This connection remains in the buffer until proper acknowledgement is received. However, in SYN flooding, the attacker/attacking host intentionally not complete the TCP connection and tries to make more and more TCP half open connections. The time comes when servers' backlog gets many numbers of such incomplete connections that makes it busy and unavailable for legitimate users for connection. This situation is called SYN flooding and has been studied by number of researchers with variety of solutions [5]-[7].

An intruder uses a DoS attack to open a huge number of half-open TCP/IP (Transmission Control Protocol/Internet Protocol) connections by intentionally breaking the three-way handshake [8]. It is similar to many DoS attacks in that it does not exploit a program error, but rather a flaw in the protocol's implementation. Most SMTP agents are unwise and will welcome whatever is sent their

way, so mail-bombing DoS attacks succeed. The ICMP ECHO command takes advantage of the fact that most kernels can actually respond to ICMP ECHO request packets one by one. Because of the current implementation of TCP's link establishment protocol, the flooding TCP SYN DoS attack's function prevails [9]. A flooding attack is initiated by sending a huge number of packets to the victim in order to overwhelm one or more of the victim's resources and reduce network capacity, often to the extent that the networks can no longer be used [10]. The so-called attacks are usually DDoS attacks that employ amplifiers to boost their strength. Flooding attacks come in a variety of forms, each based on a particular protocol:

- TCP/IP protocols: ICMP, UDP, TCP, IP
- Application protocols: HTTP, FTP, DNS

However, these attacks can be carried out using a variety of protocols; the most common are those that use ICMP and UDP packets, which do not require the setting up of a connection between the communicating entities. The primary aim of these attacks is to consume the victim's bandwidth before causing any of the victim's systems to go down. The victim may be a single host or a whole IP network, which is critical. To detect this type of attack, metrics parameters such as throughput, number of packets, and number of bytes are used [11]. The proposed solution named as INDIGSOL (Indigenous Solution) is designed and implemented to prevent the SYN flooding in real environment. This solution has four modules namely:

1. Node registration and validation
2. Packet capturing
3. Dynamic checksum
4. Hook activator

INDIGSOL has the capability to both detect and prevent SYN flooding in a dynamic environment as the system has been thoroughly tested with number of scenarios. In this solution, every node needs to get register in the database. The provided defense system keeps a check on the network nodes dynamically and looks for live clients after fix intervals of time. Now once the network is on its routine work it will keep a check on all the incoming traffic towards that specific Server and thus will always be alert for any sort of anomaly. The packet is initially monitored to see whether it is arriving from the legitimate node or not, in case of legitimate it is forwarded whereas illegitimate packet is discarded and informed to the user. Secondly, it also checks for number of SYN from every user and if goes beyond limit, it will start discarding the packets. Overall, INDIGSOL provides very comprehensive and efficient solution in preventing the SYN attack as compared to number of state of art baselines.

SYN Flooding has a serious and long-term effects on the operating systems, since it takes advantage of the TCP/IP protocol by opening a huge number of half-open connection sockets [12]-[13]. Each and every device that is linked to the Internet has the possible vulnerability of the SYN attack based on TCP-based network services. In comparison to the threats on the specific servers, these attacks may also be directed at individual hosts. If these hosts are on the routers or other network server, other TCP services should be enabled. The attack significance may be overstated. Depending on the method, the attack can differ; however, the attack itself is the same. The TCP protocol is being used by all applications is fundamental [14]. Following are the main contributions of this research:

- To identify the potential vulnerable host/hosts facing SYN flooding.
- To implement a strong monitoring system that would prevent host/hosts/devices in getting attacked by the SYN attack.
- To keep track of the illegitimate traffic so that prevention mechanism be assured in future communication.

The rest of the research is organized as section two provides literature review, methodology is in section three, section four details about results and discussions and lastly conclusions are in section five.

2. Literature Review

This section is providing brief overview of different TCP header components, properties and state of the art techniques used to prevent SYN flooding. Tab. 1 shows various acronyms used in this research as below:

Table 1 - Acronyms used in the Research and their Meaning

Acronym	**Meaning**	**Acronym**	**Meaning**
SYN	Synchronize	TCP	Transmission Control Protocol
IP	Internet Protocol	ISN	Initial Sequence Number
DoS	Denial of Service	SMTP	Simple Mail Transfer Protocol
ICMP	Internet Control Message Protocol	UDP	Universal Datagram Protocol
HTTP	Hypertext Transfer Protocol	FTP	File Transfer Protocol
DNS	Domain Name System	SSN	Synchronize Sequence Numbers
RCVD	Received	ACK	Acknowledgement
RST	Reset	URG	Urgent
FIN	Finish	Sock	Socket
BSD	Berkeley Software Distribution	3WHS	Three Way Hand Shake
ARP	Address Resolution Protocol	DLL	Dynamic Link Layer

2.1. TCP Control Flags

There are six TCP control flags in TCP header. However, SYN flooding exploits only three them. Details of the flags are as follows:

1. Synchronize Sequence Numbers (SSN): The field synchronize sequence numbers is right. The 3-way handshake [15] is the only time this flag is accurate. It instructs the receiving TCP to examine the sequence number field and record the value as the connection (typically initiator's the client's) initial sequence number. TCP sequence numbers are nothing more than 32-bit counters. They range between 0 and 4,294,967,295. Per byte of data (along with some flags) sent over a TCP link is sequenced. The sequence number of the first byte of data in the TCP section will be stored in the sequence number field in the TCP header.
2. Acknowledgement (ACK): The field for the acknowledgment number is right. Almost always, this flag is raised. The acknowledgment number field in the TCP header stores the value of the next predicted sequence number (from the other side), as well as acknowledging all data up to this ACK number minus one.
3. Reset (RST): It is used to destroy the referenced connection and hence all memory structures are torn down.
4. Urgent (URG): The urgent pointer is valid. This is TCP's way of implementing out of band (OOB) data. For instance, in a telnet connection a `ctrl-c` on the client side is considered urgent and will cause this flag to be set.
5. Push (PUSH): The receiving TCP should not queue this data, but rather pass it to the application as soon as possible. This flag should always be set in interactive connections, such as telnet and rlogin.
6. Finish (FIN) The sending TCP is finished transmitting data but is still open to accepting data.

2.2. Ports

TCP offers a user interface called a port to grant simultaneous access to the TCP module. The kernel uses ports to distinguish network operations. They only exist at the transport layer. A TCP port serves as an endpoint for network communications when combined with an IP address. In reality, all Internet connections are defined by four numbers at any given time: the source IP address and source port, as well as the destination IP address and destination port. Servers are connected to 'well-known'

ports so that they can be seen on any device with a common port. TCP port 23 is used for the telnet daemon.

2.3. TCP Memory Structures and the Backlog

It's important to look at the memory structures that TCP produces when a client SYN arrives and the link is pending (that is, when TCP is in the SYN SENT or SYN RVCD state and TCP is in the SYN SENT or SYN RVCD state and TCP is in the SYN SENT or SYN RVCD state and TCP is in the SYN SENT or SYN [16]. There are three memory structures reserved for every pending TCP link under BSD type network code (we do not mention the method (proc) structure and file structure) [17]-[18].

1. Socket Structure (socket ()): Holds data about the communications link's local end, including the protocol that is used, information about the state, information about addressing, reference queues, buffers detail, and flags information.
2. Structure of the Internet Protocol Control Block (inpcb ()): TCP (and UDP) use PCBs at the transport layer to store different pieces of information required by TCP. They store TCP state, information about IP address, port numbers information, IP header prototype along with options, and a pointer to the destination address's routing table entry. When a TCP-based service calls listen () then PCBs are generated for that server.
3. TCP Control Block Structure (tcpcb ()): The TCP control block holds information unique to TCP, such as timer's information, details about sequence numbers, flow control status information, and details of OOB data. To store network records, Linux employs a separate memory allocation scheme. Instead of the pcb() and tcpcb() functions, the socket structure is included.
4. Sock Structure (sock()): It includes protocol specific information because most of the data structures are using TCP.
5. SK Structure (sk_buff()): It contains protocol specific information comprising of packet header information and a sock().

2.4. Backlog Queue

There are a lot of memory structures here. They must be allocated any time a client SYN arrives on a legitimate port (a port where a TCP server is listening ()). A busy host could quickly

waste all of its memory just attempting to process TCP connections if there was no cap. (This will be an even more straightforward DoS attack) [19]. There is, however, a restriction to the number of concurrent link requests a given TCP may have for a given socket. This cap is known as the backlog, and it refers to the length of the queue where incoming (but unfinished) connections are stored. This queue cap extends to both the number of missing connections (the 3-way handshake hasn't been completed) and the number of completed connections that the application hasn't taken from the queue through the approve () call. If this backlog cap is exceeded, TCP will quietly reject all incoming link requests before the pending connections are resolved [20]. The backlog is not a significant amount of work. TCP is normally very quick when it comes to establishing connections. If a link arrives before the queue is complete, the receiving TCP would more likely have space in its queue when the client retransmits the connection request section. The queue sizes of various TCP implementations vary. There is also a 3/2 'grace' margin in BSD style networking code. TCP can support up to backlog3/2+1 connection and if it calls listen with a backlog of 0, this will cause a socket to make one link [21]. There are few common backlog values as mentioned in the following Tab.2:

Table 2 - Common Backlog Values

Operating Systems used	Backlog values	SBL + Graece Notes
Sun OS 4.x.x	5	8
IRIX 5.2	5	8
Linux 1.2.x	10	10
Win NTs 3.5.1	6	6
Win NTw 4.0	6	6

It's necessary to watch as the receiving TCP processes an incoming section to see just how the attack operates. For BSD-style networking, the following is valid, and only processes applicable to this paper are discussed [22]. When a packet arrives, it is demultiplexed and sent up the protocol stack to TCP, which is in the LISTEN state.

1. Get header information: The TCP and IP headers are retrieved and stored in memory by TCP.
2. Verify the TCP checksum: The section is verified using the regular Internet checksum. If it fails, there is no ACK received, and the section is dropped, with the assumption that the client can retransmit it. The PCB feature associated with the relation is located by TCP. TCP drops the section and sends a RST if it is not identified. (As an aside, TCP manages connections that come on ports where the server isn't listening ().) The server has not called connect () or listen () if the PCB () persists, but the state is CLOSED (). About the fact that

the section is dropped, no RST is submitted. It is expected that the client resends its link order [23].

3. Create new socket: A slave socket is generated when a section arrives for a listening ()ing socket. A socket (), tcpcb(), and another pcb() are all generated here. TCP is not yet dedicated to the link, but if an error occurs, a flag is set to allow TCP to drop the socket (and break the memory structures). TCP finds beyond the backlog cap to be a mistake, and the relation is rejected. This is precisely why the attack succeeds, as we can see. Otherwise, the TCP status of the new socket is set to LISTEN, and the passive open is attempted. If the section includes a RST, an ACK, or no SYN, it is dropped. If it includes an ACK, it is discarded, a RST is submitted, and the memory constructs are destroyed (the ACK is called a mistake at this stage because it makes no sense for the connection). The section is dropped if it does not have the SYN bit set. Processing occurs if the section includes a SYN [24].
4. Address processing: TCP then saves the client's addresses in a buffer, connects to the client with pcb(), processes all TCP options, and sets the initial send sequence (ISS) number, ACK and the SYN [25]. The client receives a SYN, ISS, and ACK from TCP. At this stage, the link institution timer is set to 75 seconds. SYN RCVD is the new condition. Now at this time the socket has been committed to TCP. Since the intended client response is never sent, we can see that this is the condition in which the target TCP will be when under attack. The situation remained SYN RCVD until the link establishment timer ends, at which point all connection-related memory structures are lost and the socket returns to the LISTEN state [26].

To go along with the ever-present network security threat of SYN flood, the Microsoft has devised its own approach, such as avoiding RAW sockets, it is not believed that this is the best solution [27]. Anyone from everywhere, regardless of privileges or access rights, will make YAHOO Out of Bounds to all of us in the vast dark world of the Internet. Imagine that the PING command is regarded by all of us as a fast search of operation, but no one is aware of the havoc it causes on the receipting end. TCP SYN Flood is the same concept; it was and continues to be a source of concern for most network tycoons.

An attacker first attempts to create a link with the server, after which he attempts to send any bogus TCP packets to the server in such a manner that all of these packets request the creation of a connection with that server. However, any clever server will never allow a client to request more than a certain number of connections, necessitating the attacker's implementation of the spoofing principle, and therefore someone spoofs itself any time he requests a connection, thus filling up the server's

Back log and rendering him an all-time busy server [28]. The underlying premise behind this issue is that it has to be dealt with on a local basis, and as the network grows and widens, it will become more difficult to deal with it at a higher level [29]. If we establish a monitoring scheme that verifies the validity of all outgoing traffic by inspecting all outgoing packets in suspicious mode and inquiring the sender about packets' authenticity and origin.

Table 3 - State of the Art SYN Flooding Techniques

Proposed By	Year	Technique	Pros	Cons
Shin et al [30]	2013	AVANT-GUARD (Connection migration mechanism)	Detection and prevention of SYN flood	Delay in establishing new connections Buffer saturation
Ambrosin et al [31]	2015	Line Switch (SYN proxy technique)	Removes buffer saturation	Reduces memory size for storing the connections
Chin et al [32]	2015	Collaborative technique (Monitors and Correlators)	Mitigates DDOS attacks Detects only Spoofed packets	Connecting monitor to every switch makes it costly
Fichera et al [33]	2015	OPERETTA	Suitable for general traffic	Delay in benign TCP connection
Mohammadi et al [34]	2017	SLICOTS	Low overhead Less response time	Costly in implementing rules
Dhawan et al [35]	2015	SPHINX	Detecting rate of packets in general traffic Efficient in detection	Static solution generates False alarm Does not differentiate between legitimate and illegitimate packets
Evmorfos et al [36]	2020	Neural Network Architecture (Deep learning predictive model)	High accuracy and low false positive in attack detection	False negatives are excluded Less scalability
Putu et al [37]	2020	SPI using CSF	Minimum SYN violations are observed and server is safer	Scalability problem
Cheng et al [38]	2018	ADADM (M-SMKL+S-SMKL)	Early DDoS attack detection Accuracy in detection	Slow convergence speed Lack of multidimensions
Jin et al [39]	2018	SVM	Low false alarm rate	Low accuracy in ICMP attack detection
Gautam et al [40]	2020	Proposed method (SVM)	Detection of DDoS attack	Very low attack detection accuracy Less parameters
Duwairi et al [41]	2020	ISDSDN	Efficient and lightweight Less processing load	Low accuracy
Nugraha et al [42]	2014	sFlows	Periodic detection of SYN attack improves efficiency	Less scalable Complexity
Kumar et al [43]	2018	SAFETY	Less processing delay More efficient in early detection	Single victim destination node

3. Research Methodology

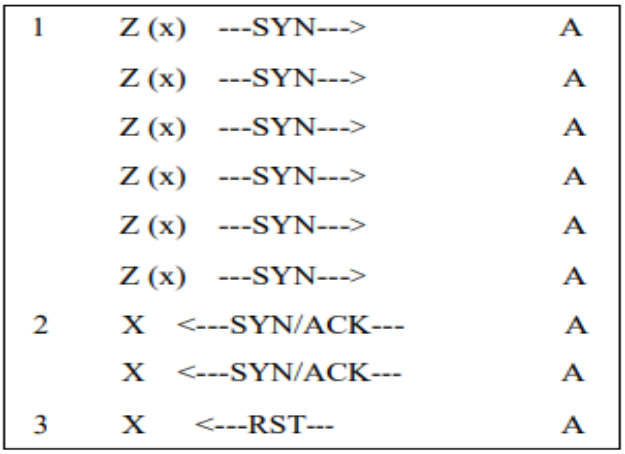
3.1. Problem Description

The TCP SYN Flood always remains a matter of great concern in the Internet world. Generally speaking, there has never been provided a full-fledged and multidimensional solution to this problem. Most of the solutions available deal specific issues or aspects of the flooding process. So there always remains a dire need for a comprehensive and compact solution to TCP SYN Flood [44]. It's important to look at the memory structures that TCP produces when a client SYN arrives and the link is pending (that is, when TCP is in the SYN SENT or SYN RVCD state and TCP is in the SYN SENT or SYN RVCD state and TCP is in the SYN SENT or SYN RVCD state and TCP is in the SYN SENT or SYN_ Wide memory systems such as backlog queues [45]. They must be allocated any time a client SYN arrives on a legitimate port (a port where a TCP server is listening (). There is, however, a restriction to the number of concurrent link requests a given TCP may have for a given socket. This queue cap extends to both the number of missing connections (the 3-way handshake hasn't been completed) and the number of completed connections that the application hasn't taken from the queue through the approve () call. When this backlog cap is reached, TCP will quietly reject all incoming link requests before the pending connections are resolved. The backlog is not a significant amount of work. It does not have to be that way [46].

The attack's signature is straightforward. A SYN flood occurs when a significant number of SYN packets arrive on a network without the corresponding reply packets which means flooding has occurred. The perpetrator will use IP to conceal his identity. the act of spoofing When an intruder inserts a bogus IP address, this is known as IP spoofing. source address, implying that everyone believes the packet originated from somewhere else than the original sender. In this situation, the attacker sends a TCP/IP packet to the victim's computer with the following information: the root address was spoofing to a computer that was not online at the time on the internet. The SYN bit is set since this is the first packet in a new link. The packet is received by the victim's server, which then sends a packet back with the SYN and ACK bits collection. The victim's computer sits and waits for a response at this stage, but it never gets one because the spoofed IP address of the machine that started the link isn't online. The attacker repeats this procedure until the buffer is full [47]. A TCP communication begins when a client sends a message to a server that includes the SYN flag in the TCP header. Normally, the server would send a SYN/ACK to the client specified by the IP header's 32-bit source address. The client would then submit an acknowledgement to the server, allowing data

transmission to begin. The intended TCP, however, cannot complete the 3-way handshake when the client IP address is spoofed to be that of an unreachable host and will continue trying until it clocks out. That is the attack's foundation. The attacking host sends a few SYN requests to the target TCP port (we found that as few as six are sufficient) (for example, the telnet daemon). The attacking host must also spoof the source IP address to that of another, currently unreachable host (the target TCP would send its response to this address). TCP will be notified that the host is unreachable by IP (via ICMP), but TCP finds these errors to be temporary and leaves the resolution to IP (reroute the packets, etc.) essentially ignoring them [48]. Since the intruder would not allow *any* host to receive the SYN/ACKs that would be coming from the target TCP, which will evoke a RST from that host, the IP-address must be unreachable. The threat will be thwarted. The steps are as follows:

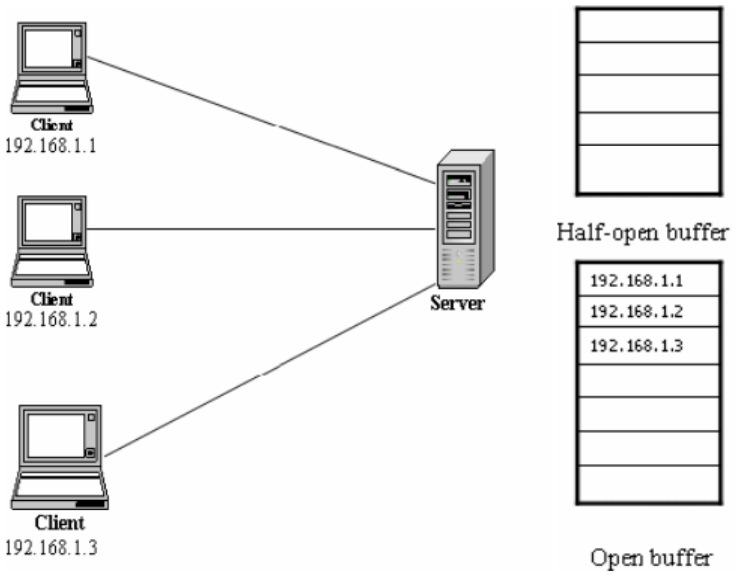
Figure 2 - Targeted Port is being Flooded



The invading host sends a slew of SYN requests at (1). to the aim of filling the target's backlog queue with pending requests interconnections (2) The goal reacts to what you say with SYN/ACKs. It thinks it knows where the SYNs are coming from. At this time, any subsequent requests to this TCP port will be rejected at this time. The targeted port has been overwhelmed [49]. The attack's signature is fairly straightforward [50]. A SYN flood attack occurs when a significant number of SYN packets arrive on a network without the corresponding reply packets. The intruder may use IP spoofing to conceal his identity. IP spoofing is where an attacker inserts a false source address into a packet such that the recipient believes it originated from somewhere other than the actual sender. The attacker sends a TCP/IP packet to the victim's computer with the source address spoofed to a machine that is not actually connected to the network. The SYN bit is set [51] since this is the first packet in a new link. The packet is received by the victim's server, which then sends a

packet back with the SYN and ACK bits collection. The victim's computer sits and waits for a response at this stage, but it never gets one because the spoofed IP address of the machine that started the link isn't online. The attacker repeats this step until the buffer is fully full [52]-[53].

Figure 3 - Normal LAN Scenario



We can observe from the fig. 3 that how a normal LAN setup goes. In this case all the normal or more specifically the legal clients go through three-way handshake Process. As long as they haven't completed the three-way handshake (3WHS) so their entry will remain in the TCP incomplete buffer queue [54]. After completing 3WHS process completes so the entry is moved up to Full connection queue or Open buffer. In this case we can see that as all the three legal clients have completed their 3WHS process so their entry is made up in the full open queue. And they can continue their routine business with the server. In fact, the server contains a list of all the legal clients with itself and we can see the details of each and every client through C:\ > arp –a (IP Address). Below is shown an ARP request generated for IP address 192.168.1.2:

Figure 4 - Execution of an ARP

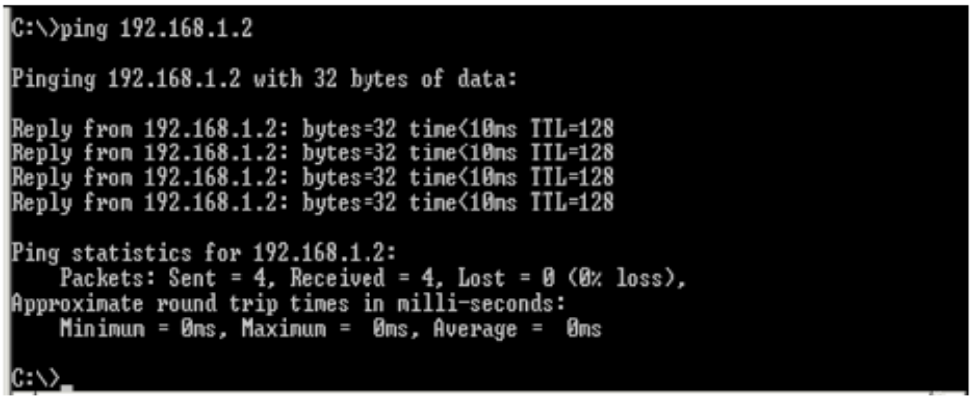
```
C:\>arp -a 192.168.1.2

Interface: 192.168.1.1 on Interface 0x1000003
  Internet Address      Physical Address      Type
  192.168.1.2           00-08-c7-85-63-47     dynamic

C:\>_
```

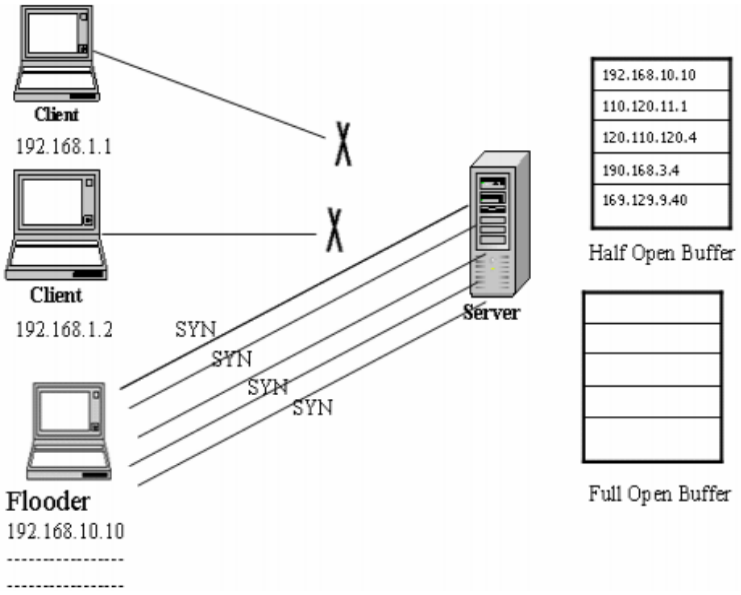
In fact, we can found out whether a client is alive or dead by sending an ICMP echo request. Below are details provided by an ICMP echo request:

Figure 5 - Ping Request Output



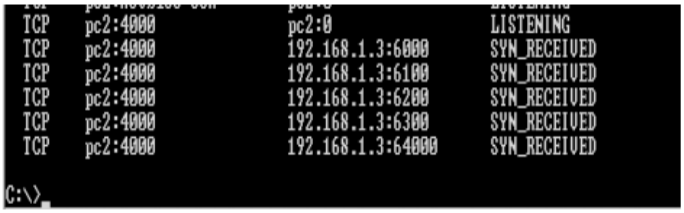
While talking of a TCP SYN flood, we can look at the following scenario which is totally opposite to the previous scenario [55]. Here what happens that a hacker who intends to make a DoS attack uses the IP address of a legal or trusted client and start the TCP SYN flood by sending multiple (more than 1000) SYN requests to the server while changing his IP address and port by doing IP Spoofing and Port Spoofing. Thus, the server cannot find whether the connection requests are genuine or fake and thus responds to each request by a SYN/ACK statement (as considering it a normal 3WHS), at the same time it makes entry of all these requests in the half open connection queue. Figure 6 depicts the situation more clearly.

Figure 6 - SYN Flooding Scenarios



Once the flood is on its way we can see the merciful condition of the server by executing netstat command like this. C: \ > netstat –a –p tcp

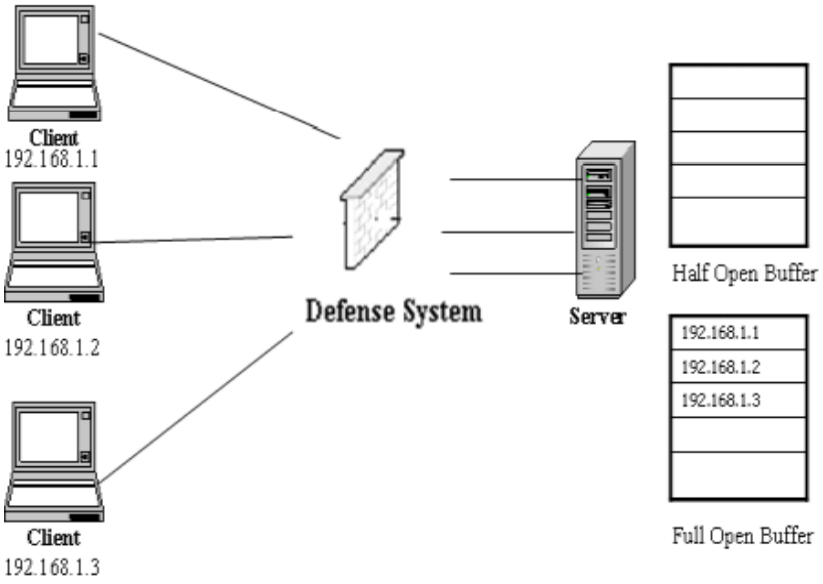
Figure 7 - Output of a Netstat after Successful Flooding Attack



3.2. The Solution

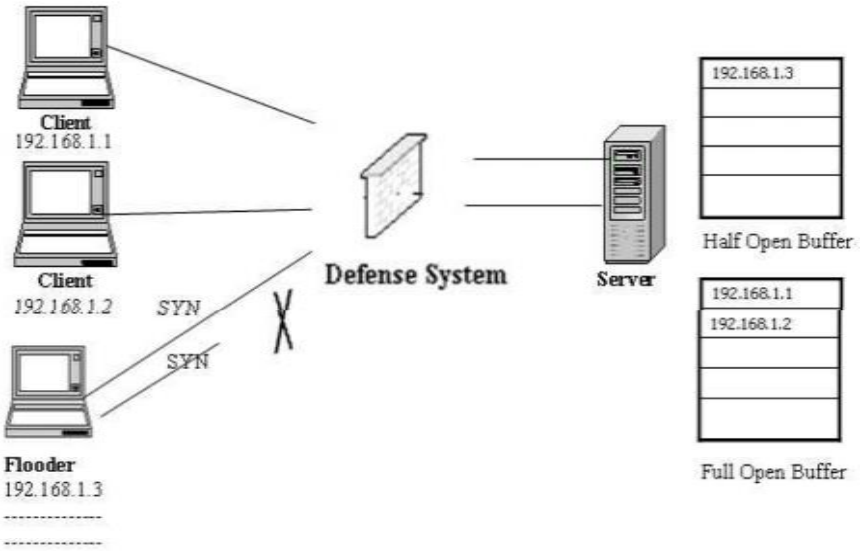
The provided defense system keeps a check on the network nodes dynamically and looks for live clients after fix intervals of time. Now once the network is on its routine work it will keep a check on all the incoming traffic towards that specific Server and thus will always be alert for any sort of anomaly. Basically, the sole purpose of this system will be to look for the SYN packets. Once a SYN packet arrives at the server so before it ids handed over to the destined application it is analyzed whether is coming from the valid node or not. If it is from a valid node so it is forwarded to the desired application without any delay, otherwise it is immediately discarded and the user is informed that it has discarded an illegal packet which was most probably destined to look for a valid server and then impose an attack [56].

Figure 8 - LAN Guarded by our Defense System



The second way or policy which it implements is that if suppose the packet is coming from a legal client so it is let to go but it then checks for number of SYN packets coming from that specific application per time and thus if the ratio goes above a fixed limit so it is considered as a suspicious act and the system gets more tight. Now if even a single SYN packet above the specific number is received so the client is immediately paralyzed by its extreme defense system and he is unable to send even a single SYN packet [57]. At the same time the user is informed that it has stopped that specific client from doing so and has made him an illegal or suspicious user.

Figure 9 - Defense System Detecting Flooder



The basic purpose of this defense system is only to provide an effective and in time solution to TCP SYN Flood attack. Thus, we worked out on different modules independently in order to make it one of the most efficient and up-to-date solution. Different modules are organized and arranged independently, and this is the key to its being and efficient and responsive. it was our utmost wish that we should come up with a module and efficient response system and it was only possible when we worked out thoroughly on its design process by identifying its major modules and their interrelationship.

Implementation Modules: The whole defense system consists of the following major modules which are inter related in such a way that the provide the maximum functionality:

1. Node Registration and Validation Module.
2. Dynamic Packet capturer
3. Dynamic Check System Module
4. Hook Activator Module

System would dynamically search for the live nodes and would make their entry in the database then on the basis of this entry it will decide the fate of all the incoming packets.

3.3.1. Node Registration and Validation Module

This is the 1st and foremost operation performed by the system whenever executed by administrator. As discussed earlier the system goes through the available database and also let the user if he wants to introduce a new node into the database. Now whenever the user makes an entry of a node IP address so it tries to validate its authenticity by executing a ping and ARP request on that specific IP address. Further, the user gives an IP address which generate Message "Node Not Found" which means that an error has been occurred in the validation process.

Error: Node Not Found

Possible Causes: Invalid IP address.

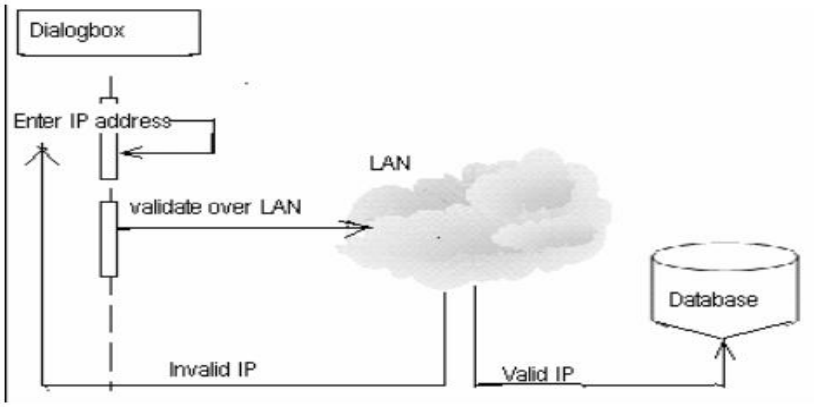
Invalid IP Address Format.

Client Powered Off.

Network too Busy (No response to Ping Request)

If all the above cases do not exist, then it means that user has entered a valid IP address. So, the next steps this module will perform is to trace out the MAC address of this IP address. When MAC address is traced out, its entry is entered in the back end database.

Figure 10 - Sequence Diagram for Validation Module

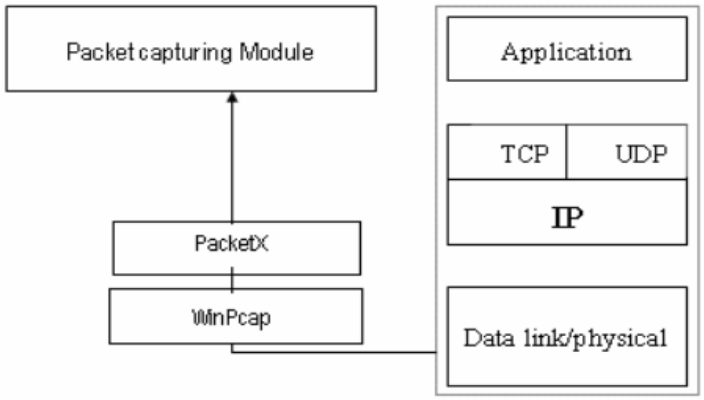


3.2.2. Packet Capturing Module

After registering the new nodes (if there are) the dynamic packet capturing module starts its backend processing by analyzing all the incoming packets. This module dynamically captures all the

incoming packets at Layer 2, i-e Data Link Layer. Windows never allows an application to directly access the data link layer instead it provides certain DLLs which facilitates the user by giving access to the Data Link Layer. This module is using the WinPcap library which enables to capture data packets from the LAN. In fact WinPcap is a library built in Microsoft Visual C++ and to use it in the .NET framework a wrapper name PacketX is provided which enables one to sort out WinPcap library.

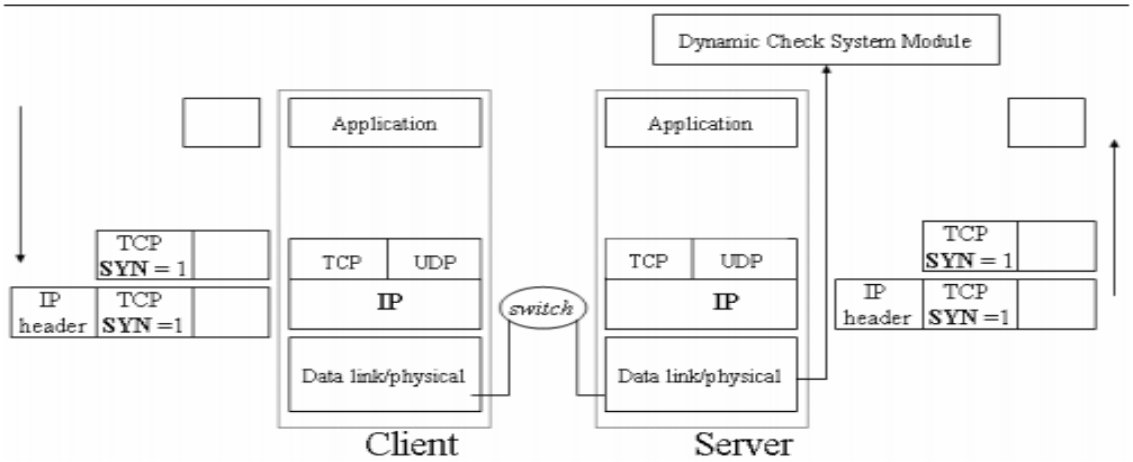
Figure 11 - Working Level of Packet Capture



3.3.3. Dynamic Check System Module

The dynamic check system module gets activated once the packet is captured. This is the main area where the SYN flood Detection policy has been implemented and also the required defense system is defined. This makes it core area which goes through many inner modules including activating other modules on the server as well as synchronizing with that specific client which is either flooder or acting as a flooder (hijacked by an attacker).

Figure 12 - Working Level of Dynamic Check System

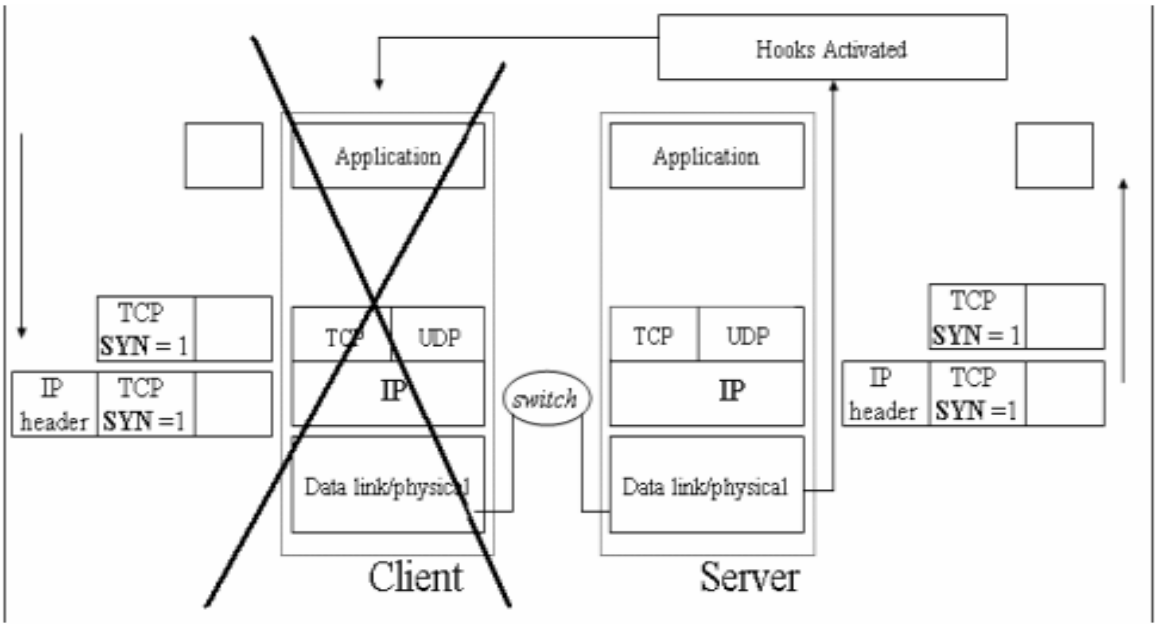


Received: 08.06.2021 – Accepted: 09.07.2021

3.3.4. Hook Activator

Once a packet has been identified for being a member of flood, then before the next packet may reach the destination, the active hooks are activated. They are responsible for discarding all the incoming traffic from that specific client. One of the main features implemented by our security system is that it not only stops all the traffic coming from that node but also totally paralyzes that client and would no more be able to generate any flood.

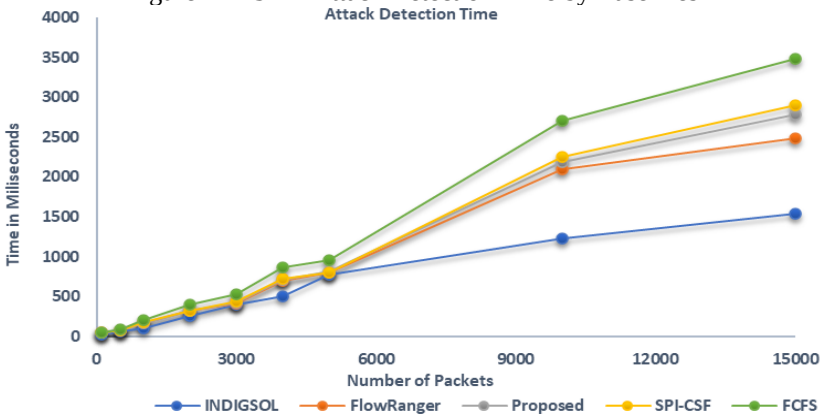
Figure 13 - Working Level of Hooks (Activated on Both Client and Server)



4. Results and Discussion

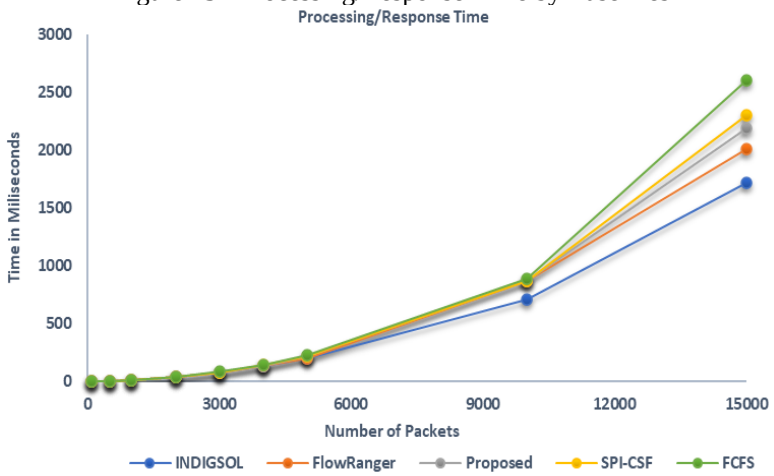
The System is tested with respect to various factors like Detection time, Processing time, and Diagnosis. Some of the factors were responding slowly in start but with some changes they get better and were efficient enough to tackle the situation [58]. There are various factors which help to determine the performance in the real scenario. These are SYN attack detection time, response/processing time for the packet and detecting attacking packets. The comparative performance of the three factors is reflected in Figures 14, 15 and 16. The baselines used in this research consists of INDIGSOL, FlowRanger [59], Proposed [60], SPI-CSF [61] and FCFS [59] algorithms.

Figure 14 - SYN Attack Detection Time by Baselines



It is shown in the Figure 14 that the INDIGSOL algorithm is taking least time to detect attack whenever it is performed. The algorithms like FlowRanger, proposed, SPI-CSF and FCFS are taking more time respectively. This shows that the performance of INDIGSOL is not affected by the number of packets which are gradually increased in six different scenarios. These scenarios comprised of 1000, 3000, 6000, 9000, 12000 and 15000 packets, respectively. At the start, it is shown that all baselines are performing well till 6000 (scenario two) packets but after further increased in packets transmission, other baselines suffer in performance but INDIGSOL remained better. This shows that INDIGSOL has better scalability in determining the attacked packets with minimum time as compared to other baselines.

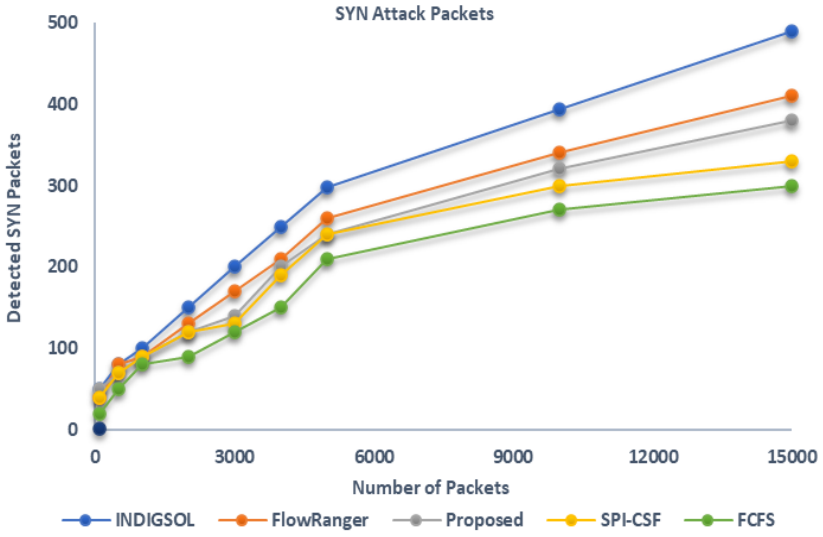
Figure 15 - Processing/Response Time by Baselines



Similarly, Figure 15 is showing the processing/response time when SYN flood is detected. It is once again clear that INDIGSOL due to its ability to detect the malicious packet early can response to attack immediately as compared to other baselines. In this case the attacker is stopped to send the flooded packets and further cannot send more.

In the Figure 15, attaching packets detected by the defense system is shown. There are six scenarios in which 50, 100, 200, 300, 400 and 500 malicious packets are injected and checked with all baselines in order to validate their performance. It is observed that in the first case with 50 malicious packets in a total of 100 packets, the baselines FlowRanger, SPI-CFS and FCFS could not detect 10, 10 and 20 malicious packets in form of SYN flood. The algorithms like INDIGSOL and proposed detected all malicious packets with complete accuracy. In the second scenario, baselines such as Proposed, FlowRanger, SPI-CSF and FCFS could not detect 10, 10, 10 and 20 malicious packets whereas, INDIGSOL detected all 100 malicious packets in a total of 1000 packets. In the third scenario with 200 malicious packets in a total of 3000 packets, the baselines such as FlowRanger, Proposed, SPI-CSF and FCFS could not detect 30, 60, 70 and 80 packets respectively. However, the INDIGSOL was able to accurately detect all 200 malicious packets with least response time. In the fourth scenario, 300 malicious packets are injected in a total of 5000 packets. The baselines FlowRanger, Proposed, SPI-CSF and FCFS could not detect 40, 60, 60 and 90 malicious packets whereas, INDIGSOL is left with 2 packets undetected. In the fifth scenario, 400 malicious packets are injected in which INDIGSOL, FlowRanger, Proposed, SPI-CSF and FCFS could not detect 7, 60, 80, 100 and 130 malicious packets. In the last scenario, 500 malicious packets are inducted leaving 13, 90, 120, 170 and 200 malicious packets undetected for INDIGSOL, FlowRanger, Proposed, SPI-CSF and FCFS respectively. Overall, the accuracy in detection of malicious packets for INDIGSOL, FlowRanger, Proposed, SPI-CSF and FCFS are 99%, 85%, 79%, 74% and 63% respectively.

Figure 16 - Attacking Packets Detected by Baselines



Received: 08.06.2021 – Accepted: 09.07.2021

4.1. Statistical Analysis

In order to perform statistical analysis, a parametric test with 2 variables is needed because comparison of one baseline is performed with INDIGSOL at a time. For this purpose, ANOVA test is performed to determine the significance of our study given in Tab. 4. Similarly, the values such as mean, standard deviation (SD), p-value, and t-value in the same Table are also given. The level of significance, meanwhile, is set to $p< 0.05$. Furthermore, the hypothesis is defined as:

H0: INDIGSOL and other baselines have no difference.

H1: A significant difference exists between INDIGSOL and other baselines.

Table 4 - Statistical Comparison of INDIGSOL with Baselines

ANOVA test (SYN Detection time)						
Source of Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	3253.81	6	932.30	3.1953	0.01134	2.146408
Within Groups	14566.88	60	300.587			
Total	17820.69	66				
ANOVA test (Processing/Response time)						
Source of Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	20710.01	6	3318.144	2.102806	0.027472	2.36407
Within Groups	1024003	60	16841.24			
Total	1044713.01	66				
ANOVA test (Number of malicious packets)						
Source of Variation	*SS*	*Df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1.13E+08	6	16830331	4.558748	0.00558	2.23654
Within Groups	2.55E+08	60	3941930			
Total	3.68E+08	66				

It is visible that the p-values in all three cases are less than the significance level of <0.05, which states that their exist a significant difference between IDIGSOL and baselines. The null hypothesis H0 is rejected and alternate hypothesis H1 is there accepted.

5. Conclusion

The availability of computing resources is important to perform useful operations. SYN flooding over the years is one of the major securities problems that opens large number of half open TCP connections and making the server all time busy. This will result in unavailability of computing resources to the legitimate users. In this research, INDIGSOL approach is proposed as a defense system that monitors and stop SYN flood packets at the origin. The adopted approach is an efficient

in such situations, as most of the applications built for this purpose are designed in Low Level Languages. Four modules of the INDIGSOL are developed and implemented such as node registration and validation, packet capturing, dynamic check system, and hook activation which perform extensive monitoring and filtering on the incoming packets and forwards only legitimate ones. By comparing this approach with some baselines in terms of malicious packets detection time and response time in six real-time scenarios helps to efficiently run the network with 99% average accuracy. To perform Exclusive filtering, WINDOWS HOOKS are used which then needs to be configured according to one's requirements, and this is considered to be a difficult task. Because of security concerns the TCP/IP protocol stack of WINDOWS operating systems is dynamically reconfigured often.

Acknowledgment: Thanks to our families & colleagues who supported us morally.

Funding Statement: The authors extend their appreciation to King Saud University for funding this work through Researchers Supporting Project number (RSP-2021/387), King Saud University, Riyadh, Saudi Arabia.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, pp. 219-232, 2004.

J. Xu and W. Lee, "Sustaining availability of web services under distributed denial of service attacks," *IEEE Transactions on Computers*, vol. 52, no. 2, pp. 195-208, 2003.

R.K.C. Chang, "Defending against flooding-based distributed denial-of-service attacks: a tutorial," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 42-51, 2002.

Y. Gilad and A. Herzberg, "Fragmentation considered vulnerable: blindly intercepting and discarding fragments," *in Proc. WOOT*, San Francisco, USA, pp. 1-2, 2011.

A. Degirmencioglu, H.T. Erdogan, M. A. Mizani and O. Yılmaz, "A classification approach for adaptive mitigation of SYN flood attacks: Preventing performance loss due to SYN flood attacks," *in Proc. NOMS,* Istanbul, Turkey, pp. 1109-1112.

T.M. Thang, C. Q. Nguyen and K. V. Nguyen, "Synflood spoofed source DDoS attack defense based on packet ID anomaly detection with bloom filter", in Proc. *ACDT,* Hanoi, Vietnam, pp. 75-80, 2018.

M. Junaid, M. Hussain, A. Masood, A. Noreen and M. A. Whala, "Evaluation of framework for the comparative analysis of symmetric block ciphers," *in Proc. ICCSA*, Jeju, Korea, pp. 1-4, 2009.

G. John, "Analysis Techniques for detecting coordinated attacks and probes," *in Proc. of USENIX Workshop on Intrusion Detection and Network Monitoring,* California, USA, pp. 329-339, 1999.

W.R. Stevens, “TCP/IP Illustrated, The protocols”, in *The Addison-Wesley Professional Computing Series, 2nd* ed, vol.1. Karnataka, India: Pearson, pp. 1-111, 2011.

I. Fadia, “Network security: a hacker's perspective”, 2nd ed, India, Macmillan India Ltd, 2006.

J. Postel, “Transmission control protocol, STD 7, RFC 793”, in *The Stormshield Network Security*, vol. 2, version 2.5. France: Stormshield press, pp. 1-34, 2016.

E. Alomari, “Botnet-based distributed denial of service (DDoS) attacks on web servers: classification and art,” *International Journal of Computer Applications*, vol. 49, no. 7, pp. 24-32, 2012.

A. Sharma and A. Bhasin, "Critical investigation of denial of service and distributed denial of service models and tools," *in Proc. ICACCCN*, Noida, India, pp. 546-550, 2018.

A. Anitha, J. Jaya Kumari and G.V. Mini, "A survey of P2P overlays in various networks," *in Proc. ICSCCN*, Thuckalay, India, 277-281, 2011.

S. Khak Abi, "Preventing SYN flood dos attacks: an improvement to SYN cookies," *in Proc. ICISI*, TX, USA, 235-235, 2009.

M. Chan, H. Litz, and D. R. Cheriton, “Rethinking network stack design with memory snapshots,” *in Proceedings of Hot OS*, SAP, New Mexico, 1-22, 2013.

R. Braden, *Request for comments (RFC1122)*, California, USA: Internet Engineering Task Force, 1989. https://www.rfc-editor.org/rfc/pdfrfc/rfc1122.txt.pdf

L. DeNardis, “The internet design tension between surveillance and security,” *IEEE Annals of the History of Computing*, vol. 37, no. 2, pp. 72-83, 2015.

T. Nakashima and S. Oshima, "A detective method for SYN flood attacks," *in Proc. ICICIC*, Beijing, China, pp. 48-51, 2006.

C. Cristian, P. Druschel, and D.S. Wallach, "Performance analysis of TLS web servers," *in Proc. NDSSS*, California, USA, 2002.

S. Farraposo, K. Boudaoud, L. Gallon and P. Owezarski, “Some issues raised by DoS attacks and the TCP / IP Suite,” *in Proc. SAR*, Batz surmer, France, 2005.

P. Owezarski, “Internet network metrology and attack analysis,” *in Proc. SAR*, Nancy, France, 2003.

K. Kang, “Anomaly detection of hostile traffic based on network traffic distributions,” *in Proc. ICOIN*. Estoril, Portugal, pp. 781-790, 2007.

T. Baba and S. Matsuda, "Tracing network attacks to their sources," *IEEE Internet Computing*, vol. 6, no. 2, pp. 20-26, 2002.

J. Liu, Z. Lee and Y. Chung, “Dynamic probabilistic packet marking for efficient IP traceback,” *Computer Networks*, vol. 51, no. 3, pp. 866-882, 2007.

W.J. Drake, V.G. Cerf and W. Kleinwächter, “Internet fragmentation: an overview,” *in Proc. WEF*, 1-67, 2016. Krynica, Poland.

K. Ingols, M. Chu, R. Lippmann, S. Webster and S. Boyer, "Modeling modern network attacks and countermeasures using attack graphs," *in Proc. CSA*, HI, USA. pp. 117-126, 2009.

White Paper, *CERT Advisory CA-1996-21: TCP syn flooding and IP spoofing attacks*. Carnegie Mellon University, USA, 2006: http://www.cert.org/advisories/CA-1996-21.html

K. Hafner, “Cyberpunk: outlaws and hackers on the computer frontier,” 1st ed., vol. 1. NY, USA: Simon & Schuster Press, pp. 1-400, 1995.

Received: 08.06.2021 – Accepted: 09.07.2021

S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: scalable and vigilant switch flow management in software-defined networks," *in Proc. SIGSAC conference on Computer & communications security*, Berlin, Germany, pp. 413–424, 2013.

M. Ambrosin, M. Conti, F. De Gaspari, and R. Poovendran, "Line switch: efficiently managing switch flow in software-defined networking while effectively tackling dos attacks," in Proc. *The 10th ACM Symposium on Information, Computer and Communications Security*, Singapore, pp. 639–644, 2015.

T. Chin, X. Mountrouidou, X. Li, and K. Xiong, "Selective packet inspection to detect dos flooding using software defined networking (SDN)," *in IEEE 35th International Conference on Distributed Computing Systems Workshops*, Columbus, USA, pp. 95–99, 2015.

S. Fichera, L. Galluccio, S.C. Grancagnolo, G. Morabito, and S. Palazzo, "Operetta: An open flow-based remedy to mitigate tcp synflood attacks against web servers," *Computer Networks*, vol. 92, no. 1, pp. 89–100, 2015.

R. Mohammadi, R. Javidan and M. Conti, "SLICOTS: An SDN-based lightweight countermeasure for TCP SYN flooding attacks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 487-497, 2017.

M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "Sphinx: detecting security attacks in software-defined networks." *in Proc. NDSS*, CA, USA, pp. 1-15, 2015.

S. Evmorfos, G. Vlachodimitropoulos, N. Bakalos, and Erol Gelenbe, "Neural network architectures for the detection of SYN flood attacks in IoT systems," *In Proc. PETRA*. Karfu, Greece, Article 69, pp.1–4, 2020.

E. Pratama and I.P. Agus, "Tcp syn flood attack prevention using SPI method on CSF a PoC," *Bulletin of Computer Science and Electrical Engineering*, vol. 1. no. 2, pp. 63-72, 2020.

J. Cheng, C. Zhang, X. Tang, V.S. Sheng, Z. Dong et al., "Adaptive DDoS attack detection method based on multiple-kernel learning," *Security and Communication Networks*, vol. 1, Article ID 5198685, 19 pages, 2018.

J.I. Ye, X. Cheng, J. Zhu, L. Feng and L. Song, "A DDoS attack detection method based on SVM in software defined network," *Security and Communication Networks*, vol. 2018, Article ID 9804061, 8 pages, 2018.

D. Gautam and V. Tokekar, "A novel approach for detecting DDoS attack in MANET," *Materials today: Proceedings*, vol. 29, no. 2, pp. 674-677, 2020.

A. Duwairi, A. Quraan, and A. Qader, "ISDSDN: mitigating SYN flood attacks in software defined networks," *J Netw Syst Manage*, vol. 28, pp. 1366–1390, 2020.

M. Nugraha, I. Paramita, A. Musa, D. Choi and B. Cho, "Utilizing OpenFlow and S Flow to detect and mitigate SYN flooding attack," *J. Kor. Multimed. Soc*, vol. 17, no. 8, pp. 988–994, 2014.

P. Kumar, M. Tripathi, A. Nehra, M. Conti and C. Lal, "SAFETY: Early detection and mitigation of TCP SYN food utilizing entropy in SDN," *IEEE Trans. Netw. Serv. Manag*, vol. 15, no.4, pp. 1545–1559, 2018.

S. Kumar, *"Classification and detection of computer Intrusions,"* PhD. dissertation, Purdue University, USA, 1995.

C.E. Landwehr, A.R. Bull, J.P. McDermott and W.S. Choi, "A taxonomy of computer program security flaws," *ACM Computing Surveys*, vol. 26, no. 3, pp. 211- 254, 1994.

U. Lindqvist and E. Jonsson, “How to systematically classify computer security intrusions,” *In Proc. IEEE Symposium on Security and Privacy*, CA, USA, pp. 154-163, 1997.

J.W. Haines, L.M. Rossey, R.P. Lippmann and R.K. Cunningham, "Extending the DARPA off-line intrusion detection evaluations," *in Proc. DISCEX*, CA, USA, pp. 35-45, 2001.

K. Kendall, “A database of computer attacks for the evaluation of intrusion detection systems,” Masters. Dissertation, MIT, USA, 1999.

R.P. Lippmann, “Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion evaluation,” *in Proc of the DISCEX*, USA, pp 12-26, 2000.

J. Undercoffer, J. Pinkston, A. Joshi and T. Finin, “A target-centric ontology for intrusion detection,” *in proc. of the IJCAI*, Acapulco, Mexico, pp. 47-58, 2003.

S. Specht and R. Lee, “Taxonomies of distributed denial of service networks, attacks, tools, and countermeasures,” *in Proc. ICPADS*, USA, pp. 543-550, 2004.

J. Mirkovic and P. Reiher, “A taxonomy of DDoS attack and DDoS defense mechanisms,” *SIGCOMM Computer Communication Review*, vol. 34, no. 2 pp. 39–53, 2004.

A. Hussain, J. Heidemann, and C. Papadopoulos, “A framework for classifying denial of service attacks,” *in Proc. SIGCOMM*, New York, NY, USA, pp. 99–110, 2003.

N. Beheshti, Y. Ganjali, M. Ghobadi, N. McKeown and G. Salmon, “Experimental study of router buffer sizing,” *in Proc. SIGCOMM*, New York, NY, USA, 1pp. 97–210. 2008.

M.A. Saleh and A. Manaf, "Optimal specifications for a protective framework against HTTP-based DoS and DDoS attacks," *in Proc. ISBAST*, KL, Malaysia, pp. 263-267, 2014.

P. Owezarski and N. Larrieu, “Internet traffic characterization – an analysis of traffic oscillations,” *In Proc. HSNMC*, Toulouse, France, pp. 370-374 2004.

S. S. Manvi and P. Venkataram, "Adaptive bandwidth reservation scheme for multimedia traffic using mobile agents," *in Proc. ICHSNMC (Cat. No.02EX612)*, Jeju, South Korea, pp. 370-374, 2002.

S. Dietrich, N. Long, and D. Dittrich, “Analyzing distributed denial of service tools: the shaft case,” *in Proc. USENIX conference on System administration*, USA, pp. 329–340, 2000.

L. Wei and C. Fung, "Flow Ranger: a request prioritizing algorithm for controller DoS attacks in software defined networks," *in Proc. ICC*, London, UK, pp. 5254-5259, 2015.

C. Deokjai, "Utilizing openflow and sflow to detect and mitigate SYN flooding assault," *Journal of the Korean Multimedia Society*, vol. 17, no. 8, pp. 988–994, 2014.

I. Pratama, "TCP syn flood attack prevention using SPI method on CSF: a PoC", *Bulletin of Computer Science and Electrical engineering*, vol. 1, no. 2, pp. 63–72, 2020.