

ENHANCING DATA SECURITY TO HANDLE DOS AND SYN FLOOD ATTACK WITH HARDWARE IMPLEMENTATION

¹ANSILLA J.D., ²Dr.N.VASUDEVAN, ³Dr.S.RAVI

¹Research Scholar, Department of Computer Science and Engineering & Information Technology,
Dr. MGR Educational and Research Institute University, INDIA

²Dean, Faculty of Engineering and Technology, SRM University, Vadapalani part, INDIA

³Head of Department and Professor, Department of Electronics & Communication Engineering,
Dr. MGR Educational and Research Institute University, INDIA

E-mail: ¹ancee007@gmail.com, ²vasu_adth@yahoo.com, ³ravi_ml@yahoo.com

ABSTRACT

Network security has become an important issue in organization, business and public data handling and has grown exponentially in recent times. The threats to the valuable information on the network are becoming more widespread and more sophisticated. The 'SYN' field in the TCP segment reflects the status of the socket setup. This 'SYN' bit if set by unauthorized process can prevent genuine users from getting connected. This work focuses on secure communication between end-points terminal and protect information during transmission. In this paper, DES algorithm is implemented on ARM processor using embedded C and provides better efficiency with hardware implementation is demonstrated.

Keywords: *Trusted Nodes, Data Security, Data Encryption Standard (DES), Socket Shell, SYNC FLOOD*

1. INTRODUCTION

In the network security landscape, there are various types of threatening attacks against which defending techniques are required. As the entire world is no more in the absence of Internet, Security under the Thirsty-For-Research criteria. As the usage of Network for the majority of our applications increases, the risks against security also get increased. Amongst the threatening attacks, the Denial-of-Service (Dos) rocks up. This is clear from a statement from Incapsula that "In the 2013-2014 DDoS Threat Landscape Report, the security firm found that DDoS attacks that request network connections using larger-than-normal SYN packets have become very popular and now account for more than 51 percent of large-bandwidth attacks" and also a web security firm in Incapsula added that 'Large-packet and reflection attacks ratchet up the bandwidth of denial-of-service in the first three months of 2014'. As such, the Distributed Denial of Service is taken into consideration in this work. And also, there exists various techniques to attain DDoS. Amongst SYN flooding is a type of measure to attain DDoS and hence is used in this work

In this work, the technique of "SYN Flood" is used to achieve DOS. A SYN flood tries to exhaust states in the TCP/IP stack. Since TCP maintains

reliable connections, each connection needs to be tracked somewhere. The TCP/IP stack in the kernel handles this, but it has a finite table that can only track so many incoming connections. A SYN flood uses spoofing to take advantage of this limitation. The attacker floods the victim's system with many SYN packets, using a spoofed nonexistent source address. Since a SYN packet is used to initiate a TCP connection, the victim's machine will send a SYN/ACK packet to the spoofed address in response and wait for the expected ACK response. Each of these waiting, half-open connections goes into a backlog queue that has limited space. Since the spoofed source addresses don't actually exist, the ACK responses needed to remove these entries from the queue and complete the connections never come. Instead, each half-open connection must time out, which takes a relatively long time.

As long as the attacker continues to flood the victim's system with spoofed SYN packets, the victim's backlog queue will remain full, making it nearly impossible for real SYN packets to get to the system and initiate valid TCP/IP connections (refer figure1).

Security measures are to be taken against the attack

2. DOS

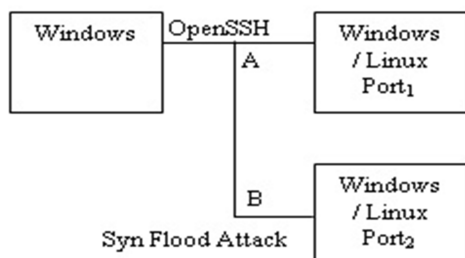


Figure 1: Block diagram illustrating SYN flood attack. Link 'A' will not be serviced due to flooding of packets from port₂

A Denial-of-Service (DoS) attack is characterized in the cloud network by explicitly making an attempt to prevent the legitimate users of a service from accessing it and disable the computer from the network. As shown in fig2, in DoS, when a computer and its network connection or the computers and the network of the websites it access are targeted, the attacker could prevent the access to the email, websites, online accounts, banking etc. A heavy traffic is created by flooding with large quantity of external communication requests or traffic. Increased traffic causes unavailability of resources and hence the denial of service happens to occur.

A Distributed Denial-of-Service (DDoS) attack is a DoS in which a multitude of compromised systems attack a single target. Examples of DoS include flooding a network creating network traffic and avoiding legitimate traffic, to disrupt connections between two machines, preventing a particular host to access service, illegal uses of resources like using anonymous ftp area for the storing of illegal copies of commercial software, consuming disk space etc.

Amongst the variety of forms of DoS attacks, the basic types of attacks include the consumption of scarce and limited resources, destruction and alteration of configuration information and physical destruction or alteration of network components. SYN flood attack, using forged User Datagram Protocol (UDP) packets to connect the echo service on one machine to the Character Generation Protocol (CHARGEN) service on another machine, resulting in consuming the entire bandwidth between them, generating a large number of packets and directed to the network as basically ICMP echo

packets but in principle nothing, consuming the resources like using the limited data structures available to hold process information as similar to process identifiers, process table entries, process slots etc., generating excessive number of mail messages, generating to be logged errors, intentionally placing files in anonymous areas or network shares, causing the system to crash or become unstable by sending unexpected data over the network, attacks in printers, tape devices, network connections etc. and similar forms the former category of the type of attack. The next phase of the types of attack overwhelms with the changing the routing information in the routers and making the network disabled, modifying the windows registry on a Windows NT machine and making certain functions unavailable etc. and hence dealing the improperly configured computer issues. The later part of the categorization put forth the security issues in physical with unauthorized access to computers, routers, network wiring closets, network backbone segments, power and cooling stations and the similar critical components.

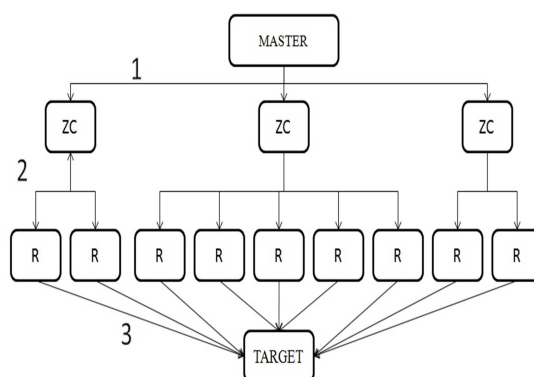


Figure 2 Hierarchy Of Dos Attack

The hacker begins the assailant by exploiting vulnerability in one computer system and it is made as the DDoS Master. The attack master, also known as the bot master recurrently identifies and infects other vulnerable systems with malware. Eventually, the hacker instructs the controlled machines to launch an attack against a specified target. The outpouring of packets to the target causes DoS. A computer under the control of an intruder is said to be a Zombie or bot. Group of co-opted computers is a Zombie army. The users of the computers in the Zombie army are unaware that they are affected. They perform the work of the user in simultaneous with the process of the hacked activity. The Kaspersky labs and Symantec have identified botnets as the largest threat to internet security rather than the spam, virus, worms, etc. Thus, "Denial of

Service Attack” (DOS Attack) is a major type of attack, the cloud network faces today, in which, interruption or suspension of service is experienced.

3. SYN FLOODING

Amongst the various types of DoS attacks, SYN flooding forms the major part. This attack became well known in 1966 and variations of it are seen still. Any service that binds to and listens on a TCP socket is potentially vulnerable to TCP SYN flooding attacks. In addition to attacks launched at specific hosts, this attack could also be launched against the router or other network systems if the hosts enable the TCP services like echo.

For any TCP connection to begin with for communication, the TCP provides a good feature of authorization. When the client requests the server for any data, the server authenticates it. An initial authorization is usually made for the connection to begin with, i.e., the 3-way Handshake protocol is made use of.

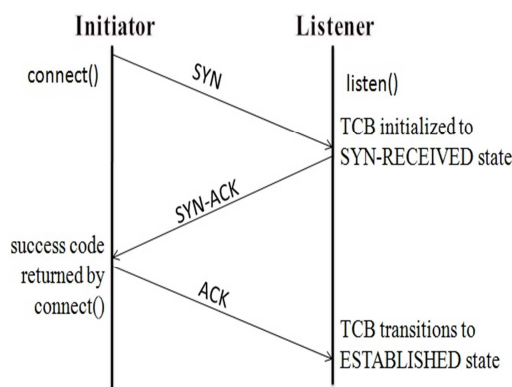


Figure 3: 3Way Handshake Protocol For TCP/IP Connection Establishment

In the 3-way handshake protocol, SYN & ACK messages are indicated by either the SYN bit or the ACK bit inside the TCP header and the SYN-ACK message has both the SYN and ACK bits turned on, i.e., set to 1, in the TCP header. TCP knows whether the network TCP socket connection is opened, synchronized or established with the help of the SYN and ACK messages during the establishment of a new socket connection. Again when the communication ends, another 3-way communication is performed to tear down the TCP socket connection.

The Transmission Control Block (TCB) is a transport protocol data structure that holds all information about a connection. The information includes maintaining information about the local

and remote socket numbers, send and receive buffers, security and priority values, and the current segment in the queue. TCB also manages send and receive sequence numbers.

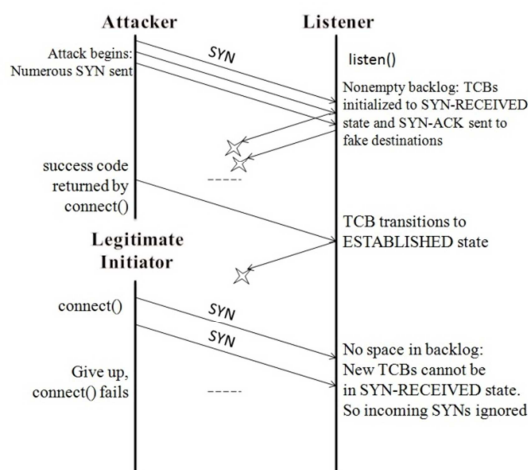


Figure 4: SYN Flooding Represented As Sequence Diagram

The TCP SYN-RECEIVED state is used to indicate that the connection is only half open and the legitimacy of the request is still in question. The significant characteristic to make a note of is that the TCB is allocated based on reception of the SYN packet before the connection is fully established before the entire authentication of the initiator. This situation leads to a clear potential DoS attack where incoming SYNs cause the allocation of so many TCBs that a host kernel memory is exhausted. In order to avoid this memory exhaustion OS generally associate a backlog parameter with a listening socket that sets a limit to the number of TCBs simultaneously in the SYN-RECEIVED state. But, this too has given rise to the next criteria which itself is vulnerable to an attack. When the backlog gets filled up, new service connections couldn't be entertained until and otherwise any TCB gets rid of the SYN-RECEIVED state.

Hence, the ultimate intention of the TCP SYN flooding attack is to fill the entire backlog by sending enough SYN packets. The attacker uses fake IP addresses, so that no response could be triggered to update the TCB from getting rid of the SYN-RECEIVED state. In the meantime TCP waits for a long time before reaping the half opened connections, service is denied to the application process on the listener for legitimate new TCP connection requests (refer figure 5)

4. IMPLEMENTATION OF SYN FLOODING

4.1. SSH

Socket Secure Shell (SSH) is UNIX based command interface. It provides an open protocol for securing network communications with authentication, encryption and data integrity. Secure shell client/server solutions provide command shell, file transfer and data tunneling services for TCP/IP applications. Combat of Password theft and other security threats can be achieved as the network connection between the client and server is encrypted. In precise, SSH is an insecure network.

The protocols include SSH transport layer protocol, SSH user authentication protocol and SSH Connection protocol.

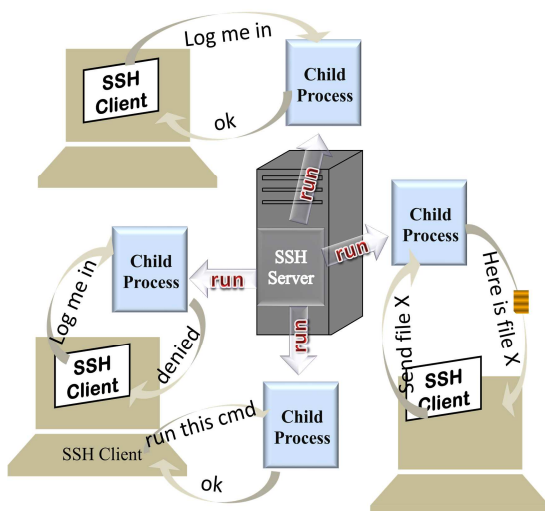


Figure 5: SSH ARCHITECTURE

SSH transport layer protocol provides server authentication, confidentiality and integration services like compression etc. SSH user authentication protocol provides client side user authentication and runs on top of the SSH transport layer protocol. SSH connection protocol multiplexes the secure tunnel provided by SSH transport layer protocol and user authentication protocol. These logical channels can be used for secure interactive shell sessions, TCP port forwarding, carrying X11 connections etc.

The functionality of secure shell includes secure command shell, secure file transfer and port forwarding.

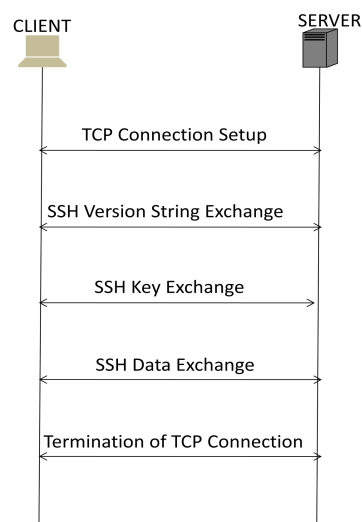


Figure 6: SSH_TLP

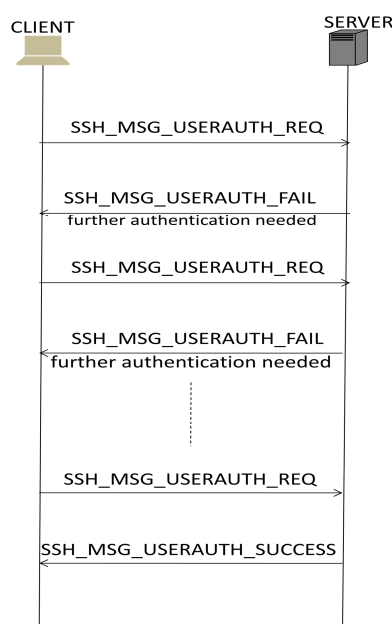


Figure 7: SSH USER AUTHENTICATION

In this powerful software based approach to network security, it itself automatically encrypts and decrypts the data, resulting in transparent encryption, i.e., users unaware of their data gets encrypted and decrypted.

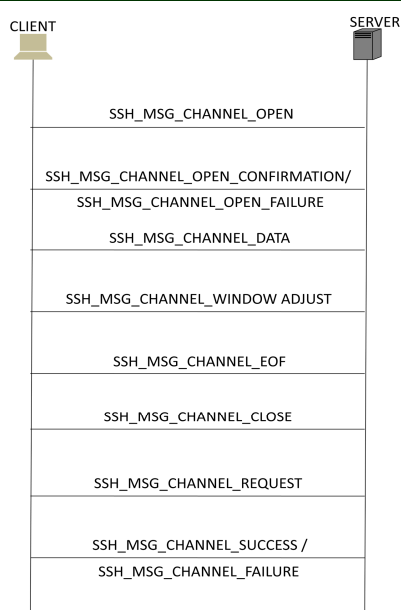


Figure 8: SSH CONNECTION PROTOCOL

Secure command shell or remote logon allows editing files, viewing the contents of directories and accessing the custom db applications. It helps to remotely start batch jobs, start, and view or stop services and processes, create user account, change permissions to files and directories etc.

Tunneling or Port forwarding is a tool to provide security to TCP/IP connections. After port forwarding is setup, SSH reroutes traffic from a program(client) and sends it across the encrypted tunnel, then delivers it to a program on other side(server). Multiple applications can transmit data over a single multiplexed channel, eliminating the need to open additional vulnerable ports on firewall or router.

Secure File Transfer Protocol is a separate protocol layered over SSH protocol. It encrypts the username / password and data being transferred. As well as it uses the same port as the SSH server. Network Address Translation could be avoided using this.

There are many security benefits on using SSH. The Authentication of the user and that of the Host is also facilitated with SSH. Data Encryption and Data Integrity could be well performed using SSH.

However, the threats addressed by SSH include Eavesdropping or Packet Sniffing, Man-in-the-middle attack and Insertion & Replay attacks.

4.2. OPENSSSH

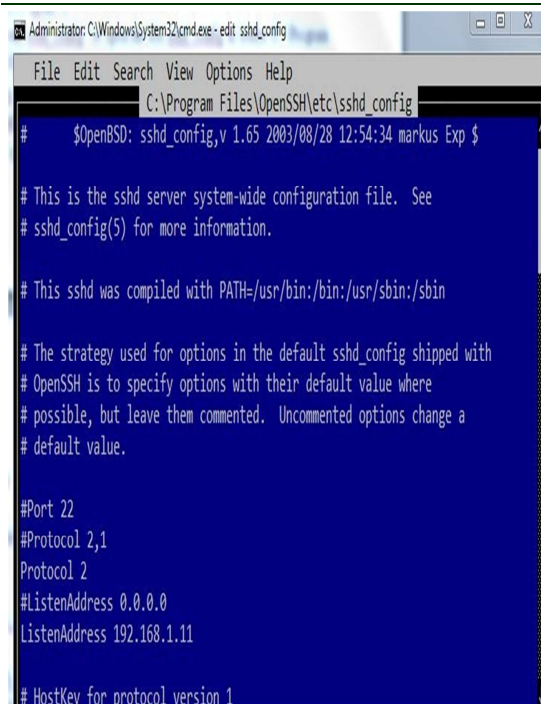
OpenSSH[OpenBerkeley Software Distribution Secure Shell] is a free version of the SSH connectivity tools. It provides secure tunneling capabilities and several authentication methods, and supports all SSH protocol versions. It could also be elucidated as a set of computer programs providing encrypted communication sessions over a computer network using the SSH protocol. It is developed by the Open BSD project as an open source alternative to the SSH. OpenSSH replaces rlogin and telnet with ssh, rcp with scp and ftp with sftp. The included is sshd(server side) and utilities like ssh_add, ssh_agent, ssh_keysign, ssh_keyscan, ssh_keygen and sftp_server.

The features of OpenSSH include various factors. This open source project has free licensing. It provides strong encryption to be concatenated and some of the techniques include 3DES (Data Encryption Standard), Blowfish, AES(Advanced Encryption Standard) and Arcfour. With the supporting feature X11 forwarding, encryption of X Window system Traffic is possible. Encrypting the channels for the legacy protocols is supported with the Port Forwarding. There are strong Authentication techniques like the Public key method, One Time Passwords method and the Kerberos authentication.

The agent forwarding is possible with the Single_Sign_On process. It is a good Interoperable tool compliance within the SSH 1.3, 1.5 and 2.0 protocol standards. The SFTP Client and Server support is available in both SSH1 and SSH 2 protocols. It is featured with Kerberos and AFS Ticket Passing. Data Compression is also essentially available with OpenSSH.

4.2.1. Steps to Configure OpenSSH:

1. Install openssh.exe software by downloading from this link Open command prompt as administrator.
2. Go to the OpenSSH directory (C:\Program Files (x86)\OpenSSH) in command prompt.
3. Go to etc Directory ("cd etc").
4. Type command "edit sshd_config" or open the sshd_config file using WordPad in folder (C:\Program Files (x86)\OpenSSH\bin\).
5. In file sshd_config default ListenAddress is 0.0.0.0



```

Administrator: C:\Windows\System32\cmd.exe - edit sshd_config
File Edit Search View Options Help
C:\Program Files\OpenSSH\etc\sshd_config
# $OpenBSD: sshd_config,v 1.65 2003/08/28 12:54:34 markus Exp $

# This is the sshd server system-wide configuration file. See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented. Uncommented options change a
# default value.

#Port 22
#Protocol 2,1
Protocol 2
#ListenAddress 0.0.0.0
ListenAddress 192.168.1.11

# HostKey for protocol version 1

```

Figure 9: Configuring Openssh

6. To change the ipaddress of the windows system to ListenAddress and remove the symbol (#) in the sshd_config file and save it.

ListesnAddress < ip_windows_machine>

Note: <ip_windows_machine> - ip address of the windows system.

4.2.2. Steps to get the server running:

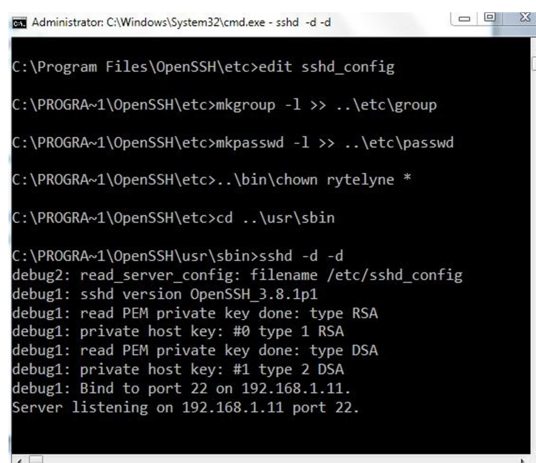
Below is a step by step process to get the server running and the screenshot is shown below.

Note: Enter the below commands in windows command prompt

```

>mkgroup -l >> ..\etc\group
>mkpasswd -l >> ..\etc\passwd
>..\bin\chown <username> *
>cd ..\usr\sbin
>sshd -d -d

```



```

Administrator: C:\Windows\System32\cmd.exe - sshd -d -d
C:\Program Files\OpenSSH\etc>edit sshd_config
C:\PROGRA~1\OpenSSH\etc>mkgroup -l >> ..\etc\group
C:\PROGRA~1\OpenSSH\etc>mkpasswd -l >> ..\etc\passwd
C:\PROGRA~1\OpenSSH\etc>..\bin\chown rytelyne *
C:\PROGRA~1\OpenSSH\etc>cd ..\usr\sbin
C:\PROGRA~1\OpenSSH\usr\sbin>sshd -d -d
debug2: read_server_config: filename /etc/sshd_config
debug1: sshd version OpenSSH_3.8.1p1
debug1: read PEM private key done: type RSA
debug1: private host key: #0 type 1 RSA
debug1: read PEM private key done: type DSA
debug1: private host key: #1 type 2 DSA
debug1: Bind to port 22 on 192.168.1.11.
Server listening on 192.168.1.11 port 22.

```

Figure 10: Server Running

Note:

<username> - windows system account name.

```
>net start opensshd
```



```

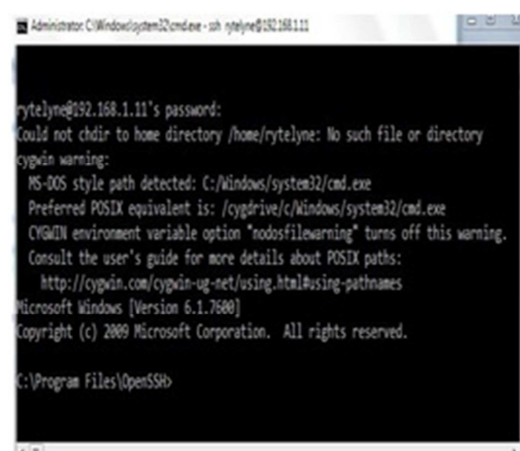
Administrator: C:\Windows\System32\cmd.exe
debug1: read PEM private key done: type RSA
debug1: private host key: #0 type 1 RSA
debug1: read PEM private key done: type DSA
debug1: private host key: #1 type 2 DSA
debug1: Bind to port 22 on 192.168.1.11.
Server listening on 192.168.1.11 port 22.

C:\PROGRA~1\OpenSSH\usr\sbin>net start opensshd
The OpenSSH Server service is starting.
The OpenSSH Server service was started successfully.

C:\PROGRA~1\OpenSSH\usr\sbin>

```

Figure 11: Server Started



```

Administrator: C:\Windows\System32\cmd.exe - ssh -i
rytelyne@192.168.1.11's password:
Could not chdir to home directory /home/rytelyne: No such file or directory
cygwin warning:
MS-DOS style path detected: C:\Windows\system32\cmd.exe
Preferred POSIX equivalent is: /cygdrive/c/Windows/system32/cmd.exe
CYGWIN environment variable option "nodosfilewarning" turns off this warning.
Consult the user's guide for more details about POSIX paths:
http://cygwin.com/cygwin-ug-net/using.html#using-pathnames
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Program Files\OpenSSH

```

Figure 12: Server

4.3. LIBNET

Libnet is a generic networking API that provides access to several protocols. The installation includes the following steps:

```
# ./configure
# make
# make install
```

4.3.1. Libnet Library Functions:

The following libnet library functions are used.

✓ **libnet_init_packet();**

This function is used to allocate memory for packets.

✓ **libnet_get_prand();**

The function is used to obtain pseudo-random numbers for various IP fields.

✓ **libnet_seed_prand();**

The function is used to seed the randomizer. In main program while loop the function libnet_seed_prand() will generate random ip address.

✓ **libnet_build_ip();**

The function is used to prepare the ip headers.

✓ **libnet_write_ip();**

The function is used to send the injected packet to the host ip address.

4.4. EXPERIMENTAL STUDY

- Connect on Ubuntu system (by below given command) to port 22 openssh on windows system.

```
$ssh -v <target host>
```

- The connection is established and the target file system (C:\Program Files\OpenSSH) files can be accessed

5. PERUSAL ON PRESENT PREVENTIVE MEASURES AGAINST SYN FLOODING

Now, the implemented SYN flood attack is to be solved, so that an attack free data transmission in the cloud network is to be achieved. Amongst, there are various techniques that could find a solution to this attack. On the overlook of this attack, a beginner could analyze on an idea about increasing the TCB backlog size. But the fact to be noted behind is that, an attacker is ready to flood, that extent too. Thus, there are various factors analyzed and a few of them include by reducing the time-out time of the half opened connections. The usage of SYN cookies and SYN caches are also been analyzed. Apart, ingress filters, firewalls and proxies, active monitoring of packets, allocation of a TCB only after a successful connection establishment is made, etc. are also been analyzed.

However, there are enough difficulties faced with this technique, in a sense to not satisfying the exact requirement, i.e., for each type of remedy, the hacker dominates with its advanced version. For the above said preventive measures, each faces its own demerits like implementation costs, heavy traffic, improper functioning, altering the design goals of the protocol. Some of the techniques are in the shoddy part as they would lack the basic functionalities and qualities of TCP.

Moreover, that software trying to get rid of SYN Flooding too possess some other demerits such as the user unfriendliness, uneasy to adopt, implementation risks etc. and prevent from being recognized as a standard.

6. PROPOSED MEASURE FOR SYN FLOODING ATTACK

To vandalize the SYN Flooding attack, the proposed technique could serve to achieve the goal and get rid of the DDoS. On an assay at this attack, it is proven that there must be some mechanism implemented around the TCB. Hence, a script to restrict the packet entry is interspersed. The source IP address of each packet is analysed. When the full backlog consists of more number of packets than the threshold level, then that particular IP address is tagged as malware and future packets from the corresponding IP address is not entertained. Thus, those packets from such type of IP address don't get into, leading to the avoidance of SYN RECEIVED state. On implementing this, SYN packets from the malware IP address are not accepted or in otherwise said, no response is made

to those SYN packets, i.e., as a go-getter, TCB doesn't enter into SYN RECEIVED state.

Moreover, the corresponding IP address is stored as a malware address. When the nonempty backlog state is reached, the script gets activated and reaps the packets from the corresponding source IP address and in the future, the packets from that fastidious IP address is prevented. Apart from this, after a particular time interval, the SYNACK packet is retransmitted and if no response is obtained again, the corresponding connection is reaped away. The time interval to be set is directly proportional to the network traffic, i.e., in case of high or less traffic, a high or low interval is set. Thus, a reset is made in the TCB backlog after the stipulated time interval

7. OUTPUT

7.1. WITHOUT USING SYNFLOOD.C



Figure 13 Without Using Synflood

7.2. USING SYNFLOOD

```
$gcc $(libnet-config --defines) -o
synflood synflood.c -lnet
$sudo ./synflood <target host><target
port> //This program must run as root.
```

In the above program, the host <target host> is a Windows machine running an openssh server on port 22.

7.2.1. To Test the SYNfloods flow using tcpdump

1. Open the new terminal and type the following command

```
$sudo tcpdump -D
```

OUTPUT:

1. eth0

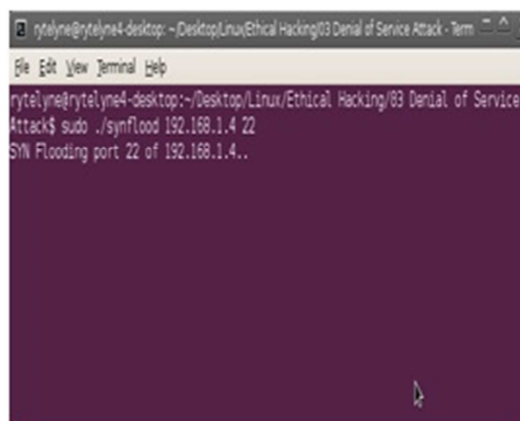


Figure 14 Synflooding

2. wlan0
3. usbmon1 (USB bus number 1)
4. usbmon2 (USB bus number 2)
5. usbmon3 (USB bus number 3)
6. usbmon4 (USB bus number 4)
7. usbmon5 (USB bus number 5)
8. usbmon6 (USB bus number 6)
9. usbmon7 (USB bus number 7)
10. any (Pseudo-device that captures on all interfaces)
11. lo

2. If we are using LAN internet connection select eth0 option (from the above output)

```
$sudo tcpdump -i eth0
```

3. If we are using Wifi internet connection select wlan0 option (from the above output)

```
$sudo tcpdump -i wlan0
```

4. The tcpdump output below shows the spoofed SYN packets flooding the host from apparently random IPs

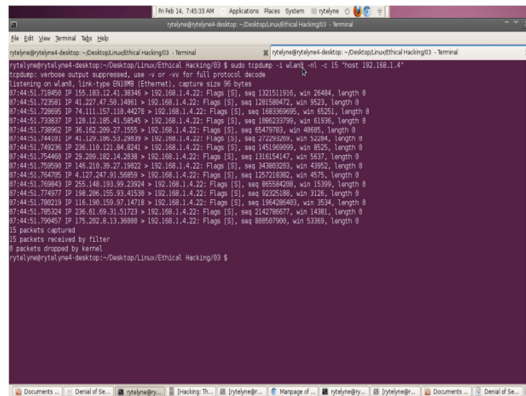
```
$sudo tcpdump -i wlan0 -
nl -c 15 "host <target
host>"
```

Note:

<target host> - ip address of the server.

-c : it specify the number of packets to capture.
-i : it only capture from desire interface.

6.2 Output using tcpdump tool



7. TO TEST THE SYN FLOODS ATTACK

➤ While the program is running in attackers system (Linux), legitimate connections cannot be made to port 22 on windows system.

➤ Try to connect on Ubuntu system (by below given command) to port 22 openssh on windows system and notice how it will be unable to connect (Connection refused) because of sync flooding.

```
$ssh -v <target host>
```

7.1. Output for synflood attack

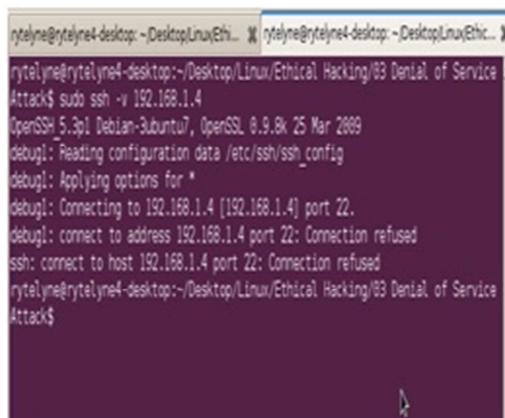


Figure 13 SYN Flood attack

8. TO TEST SYN FLOODING ON HOST (WINDOWS) SYSTEM USING WIRESHARK

The wireshark output below shows the spoofed SYN packets flooding the host from apparently random IPs. While the program is running, legitimate connections cannot be made for victims (windows) system. As seen in the below snapshot, the sync packets are flooded onto the windows system

8.1. Output for synflooding on host(windows) system using wireshark

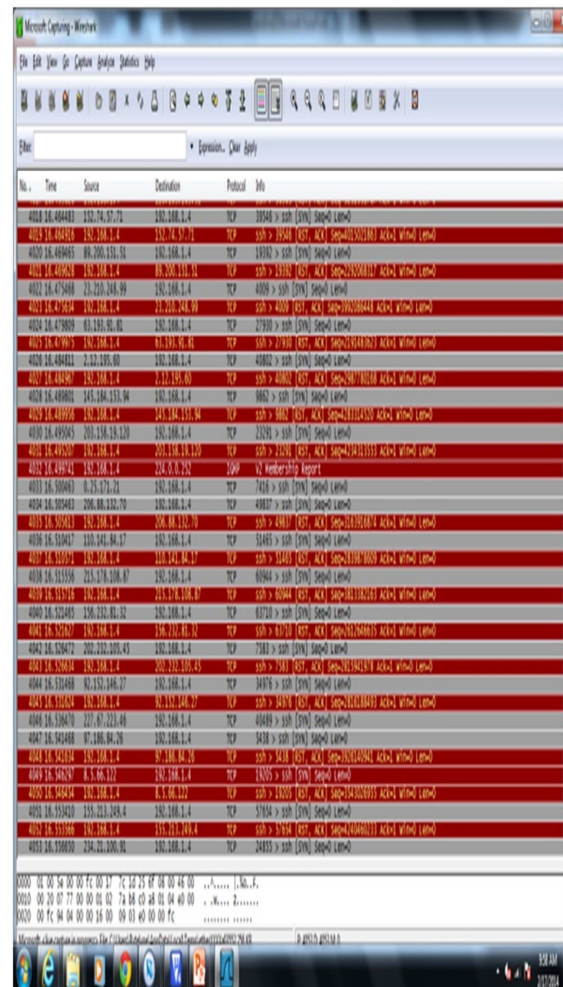


Figure 14 Testing SYN Flooding using Wreshark

9. TO AVOID SYN FLOOD ATTACK

Two broad classes of solutions to SYN flooding attacks have evolved, corresponding to where the defenses are implemented. The first class of solutions involves hardening the end-host TCP implementation itself, including altering the algorithms and data structures used for connection lookup and establishment, as well as some solutions that diverge from the TCP state machine behavior during connection establishment, as described in RFC 793. The second class involves hardening the network, either to lessen the likelihood of the attack preconditions (an army of controlled hosts or the propagation of IP packets with spoofed source addresses), or to insert middle boxes that can isolate servers on the networks behind them from illegitimate SYNs.

10. CONCLUSION

The work can prevent link failure, node failure in mesh, star and inner ring networks. The conventional fast transmit and recovery process is retained to avoid congestion issues. In this work, a secure webserver algorithm is embedded with the hardware core (ARM processor) to prevent code crash. Additionally, node can be protected in conjunction with security control (such as physical access, authentication, authorization or network controls) to adequately ensure the confidentiality, integrity and availability of the node link.

REFERENCES

- [1] Aman Mahajan, Haresh Joshi, Sahil Khajuria, Anil K Verma, "ICMP, SNMP: Collaborative Approach to Network Discovery and Monitoring", International Journal of Smart Sensors and Ad Hoc Networks (IJSSAN) ISSN No. 2248-9738 Volume-1, Issue-3, 2012, pp8-12
- [2] HE Li-juan, WANG Dong-hong, "Topology discovery algorithm of the SNMP-based network-layer protocol", Journal of Shijiazhuang Vocational Technology Institute, VOL.21 NO.6, 2009
- [3] Li, Yanbing; Ma, Yue; Wang, Wei; Wan, Xiaoqiang, "A link layer topology discovery algorithm based on STP", Jisuanji Gongcheng/Computer Engineering, v32, n18, p109-110+113, Sep 20, 2006
- [4] Yang Qiuxiang, "Algorithm Research of Topology Discovery on SNMP", International Conference on Computer Application and System Modeling (ICCSM 2010), v12, p 496-497.
- [5] J. Wei-hua, Du jun "The application of ICMP protocol in network scanning", Proceedings of International Conference on PDCAT, pp.904-906 Aug 2003.
- [6] Madalina Baltatu, Antonio Lioy, Fabio Maino, Daniele Mazzocchi. "Security Issues in Control, Management and Routing Protocols". 22-25 May 2000. URL : <http://www.terena.nl/tnc2000/proceedings/3A/3a2.pdf> (10 Dec 2001).
- [7] Alex Peeters. "ICMP Header Format". 4 October, 1999. URL : <http://citap.freesevers.com/publications/tcp-ip/tcpip012.htm> (10 Dec 2001)
- [8] Lindsay van Eden. "The Truth About ICMP". 17 May 2001 URL : <http://www.sans.org/infosecFAQ/threats/ICMP.htm> (10 Dec 2001)
- [9] Abdullah H. Alqahtani, Mohsin Ifikhar, "TCP/IP Attacks, Defenses and Security Tools", International Journal of Science and Modern Engineering (IJISME) ISSN: 2319-6386, Volume-1, Issue-10, September 2013 pp:42-47
- [10] "Wireshark", online, www.wireshark.org. (last accessed on 25 May 2013)
- [11] Bellovin, Steven M. "A look back at." Computer Security Applications Conference, 2004. 20th Annual. IEEE, 2004
- [12] Trabelsi, Zouheir, and Khaled Shuaib, "NIS04-4: Man in the Middle Intrusion Detection." Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE. IEEE, 2006
- [13] Yan, Boru, et al, "Detection and defence of DNS spoofing attack," Jisuanji Gongcheng/Computer Engineering 32.21 (2006): 130-132
- [14] Zaraska, Krzysztof, "Prelude IDS: current state and development perspectives," URL <http://www.prelude-ids.org/download/misc/pingwinaria/2003/paper.pdf> (2003)
- [15] Yao, Xiaoyu, and Chen ZHAO, "Research on Implementation and Application of Linux Kernel Firewall Netfilter [J]," Computer Engineering 8 (2003): 042