**Port Scanning using TCP SYN Half Open**

1. **Attack name: Half open Scanning**

**Description:** A TCP Half Open Scan determines if a port is open by performing the first half of a three-way handshake. It is also referred as the SYN scanning. In SYN scanning, the hostile client or attacker attempts to set up a TCP/IP connection with a server at every possible port. This is done by sending a SYN (synchronization) packet, as if to initiate a three-way handshake, to every port on the server.

When two hosts want to communicate with each other, a connection must be established between them. In TCP, three-way handshake takes place before any communication begins.

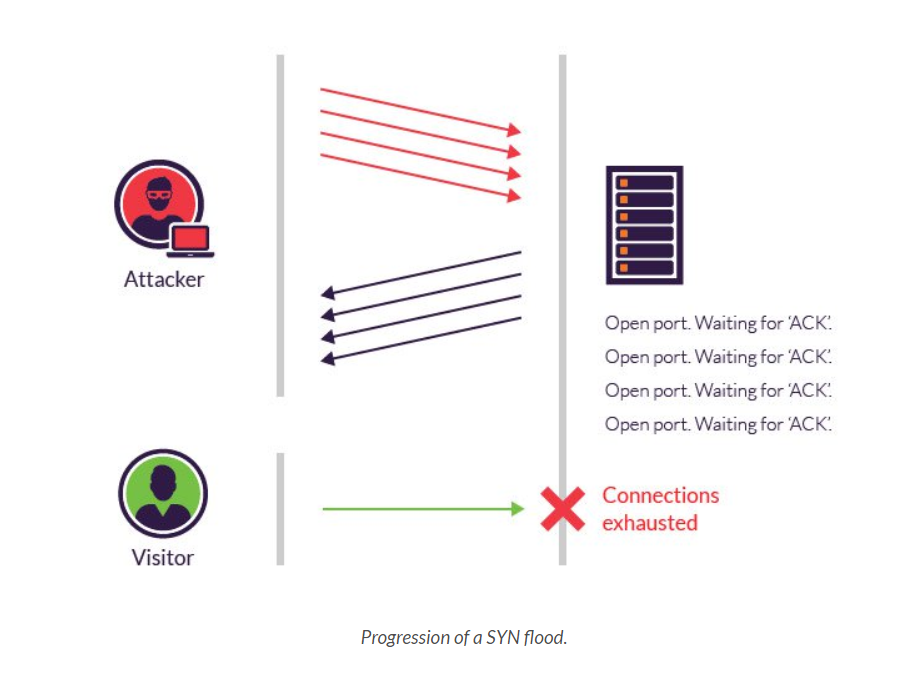
TCP Half Open Scanning follows the below steps:

* Host A sends the SYN packet (TCP packet with SYN flag set) to host B
* If the port is open, B responds by sending SYN+ACK packet else RST+ACK packet is sent.

Since Host A does not send any additional ACK packet, it is called half open connection.

**2. Attack flow diagram**

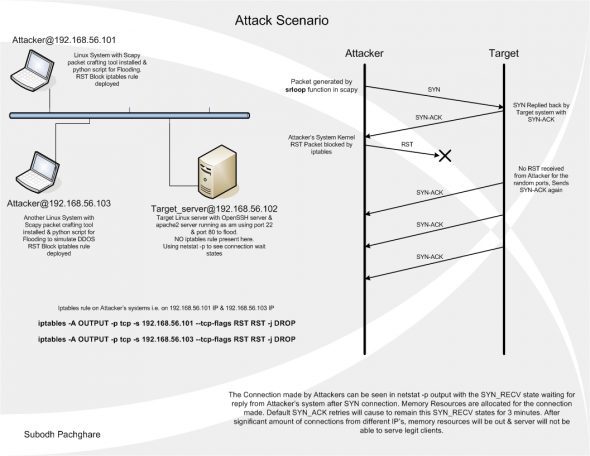
**Half open connection establishment flow**



* If no response is received after several retransmissions, the port is marked as filtered.
* The port is also considered open if a SYN packet (without the ACK flag) is received in response.

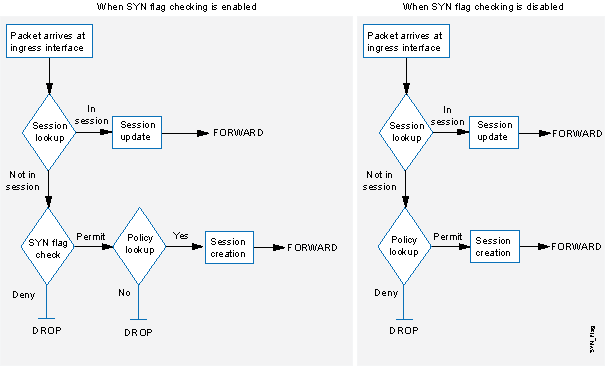
**3. Project setup and environment:**

**Attack Scenario**

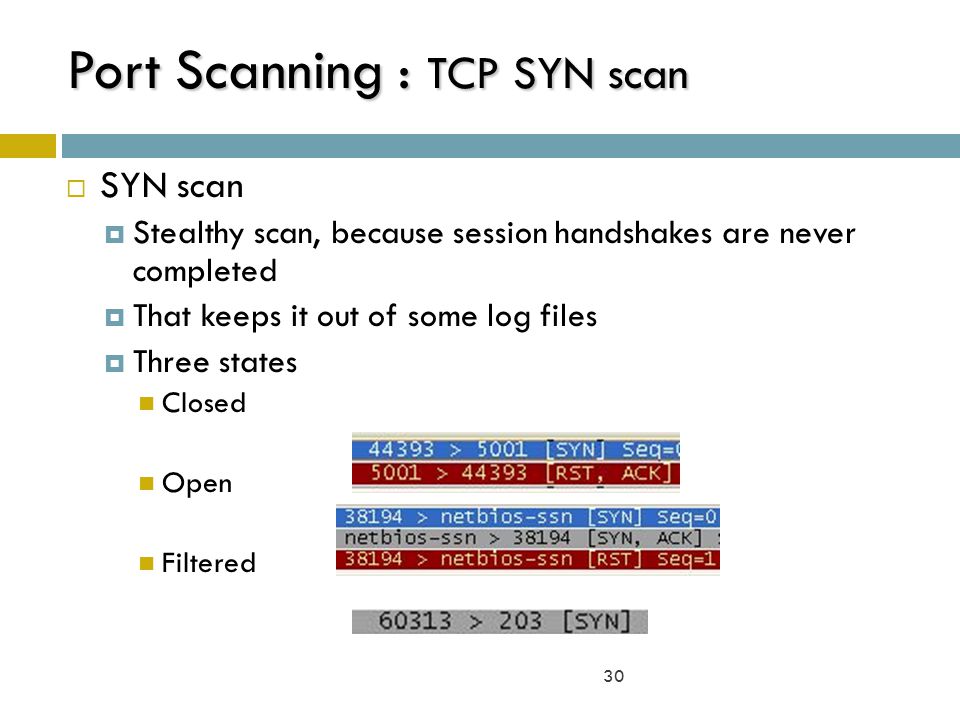


**Operating systems used, attack specific configurations:**

* Three Ubuntu Server VMs are connected through the VirtualBox “Hostonly” network adapter.
* In the diagram, the target server is 192.168.56.102; 192.168.56.101 and 192.168.56.103 are the attackers.
* To simulate a DDoS, we will generate SYN flood packets with Scapy (which has functions to manually craft abnormal packets with the desired field values), and use iptables, in multiple Oracle VirtualBox virtual machines running Ubuntu 10.04 Server.
* Two “attacker” VMs send packets to a “target server” VM.
* In a real-life scenario, attackers target a server on ports that are in the LISTEN state, to bring down the service



***Attack flow diagram of TCP SYN scan***



**A TCP Connect scan has the following characteristics:**

1. Speed: TCP SYN scanning is fast compared to other types of scans.
2. Stealth: TCP SYN scanning is stealthy and SYN scan detection is fraught with false positives.
3. Open Port: Detects that a port is open via a successful SYN/ACK to the SYN.
4. Closed Port: Detects that a port is closed via a successful RST to the SYN
5. Filtered Port: No response, or ICMP messages, indicates the presence of a filter.
6. Unfiltered Port: Cannot distinguish between a state-fully filtered port and an unfiltered port.

**Attack Prerequisites**

* This scan type is not possible with some operating systems (Windows XP SP 2). On Linux and Unix systems it requires root privileges to use raw sockets.

**4. Instructions to execute the code:**

Three Ubuntu Server VMs are connected through the VirtualBox “Hostonly” network adapter.

For ex, In the diagram, the target server is 192.168.56.102; 192.168.56.101 and 192.168.56.103 are the attackers.

Use scapy 2.2.0.

Extract the Scapy source, and as the root, run python setup.py install.

Run Scapy with the command scapy.

Run the Python script (below, SYN\_Flood\_Scapy.py) in the attacker VMs to send malformed SYN connections to the target.

**Source Code**

**! /usr/bin/env python**

**import sys**

**from scapy.all import \***

**#conf.verb=0**

**print "Field Values of packet sent"**

**p=IP(dst=sys.argv[1],id=1111,ttl=99)/TCP(sport=RandShort(),dport=[22,80],seq=12345,ack=1000,window=1000,flags="S")/"HaX0r SVP"**

**ls(p)**

**print "Sending Packets in 0.3 second intervals for timeout of 4 sec"**

**ans,unans=srloop(p,inter=0.3,retry=2,timeout=4)**

**print "Summary of answered & unanswered packets"**

**ans.summary()**

**unans.summary()**

**print "source port flags in response"**

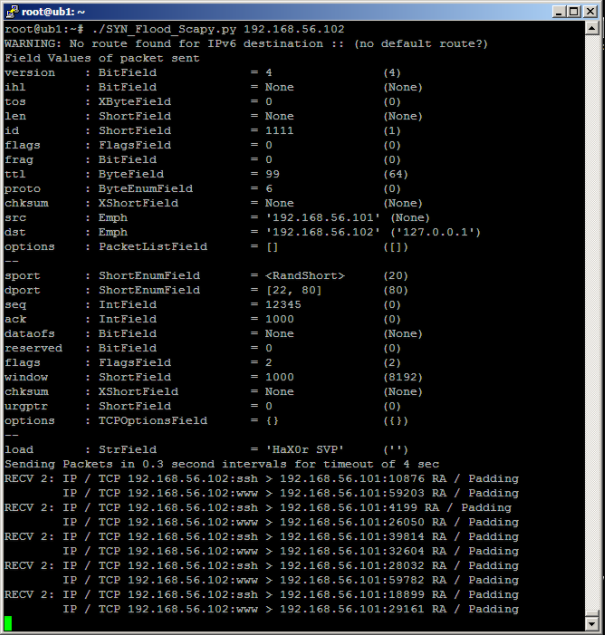
**#for s,r in ans:**

**# print r.sprintf("%TCP.sport% \t %TCP.flags%")**

**ans.make\_table(lambda(s,r): (s.dst, s.dport, r.sprintf("%IP.id% \t %IP.ttl% \t %TCP.flags%")))**

* This script will take the destination IP as input, and will create connections from different ports.
* The randshort() function is used to generate random port numbers for the sport (source port) of the TCP packet
* The destination port (dport) is set to port 22 (SSH) and 80 (Apache Web server). The TCP connect flag is set to SYN using the flags option.
* The srloop function sends p crafted packets at intervals of 0.3 seconds. The results of srloop are collected in ans (for answered packets) and unans (for unanswered packets). The gathered results are displayed in a table format for the reply flags and TTL values.
* Finally, the script reports SA (SYN-ACK) responses, and gives the results as answered/unanswered packets.

**5.Output screens to show successful attack**



**Prevention measures on target server**

The below solution will reject all suspicious TCP connections with a TCP RST packet, to prevent potential DDoS. All connections in the SYN\_RECV state will be closed forcibly with RST packets. Allow certain number of attempts from a single IP address, to take care of packet loss, which sometimes does happen, due to network errors. Thus, legitimate clients have a chance to reconnect.

**# Description : SYN Flood Prevention using iptables against Scapy SYN packets generated**

**> /var/log/DDOS\_IP.log**

**> /tmp/test1.txt**

**> /tmp/test2.txt**

**trap "echo ;echo Caught EXIT signal;iptables -F;echo Iptables entries cleared;echo HaX0R SVP" EXIT**

**while true;**

**do**

**date >> /var/log/DDOS\_IP.log**

**netstat | grep -E "ssh|www" | grep -iv ESTABLISHED | awk '{print $5}' | cut -d : -f 1 | sort | uniq -c >> /var/log/DDOS\_IP.log**

**for pip in `netstat | grep -E "ssh|www" | grep -iv ESTABLISHED | awk '{print $5}' | cut -d : -f 1 | sort | uniq`**

**do**

**conntrack=`netstat | grep -E "ssh|www" | grep -iv ESTABLISHED | awk '{print $5}' | cut -d : -f 1 | grep $pip | wc -l`;**

**while read line**

**do**

**if [ "$line" = "$pip" ]**

**then**

**continue 2**

**fi**

**done < /tmp/test2.txt**

**if [ "$conntrack" -gt "25" ]**

**then**

**iptables -I INPUT -s $pip -p tcp -j REJECT --reject-with tcp-reset**

**echo "$pip" >> /tmp/test1.txt**

**fi**

**done**

**cat /tmp/test1.txt | sort | uniq > /tmp/test2.txt**

**sleep $1**

**done**

**Drawback**

SYN scanning requires the ability to access "raw sockets" to create the packets. As a result, it is not possible to perform a SYN scan from some systems (Windows XP SP 2). On other systems (BSD, Linux) administrative privileges are required to write to the raw socket.

**6. References**:

<https://www.incapsula.com/ddos/attack-glossary/syn-flood.html>

<https://www.google.com/search?q=Attack+flow+diagram+of+tcp+syn+scan&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjMqJbKuoPaAhXs54MKHdDGBWUQ_AUICigB&biw=1692&bih=815&dpr=1.13#imgrc=r_xfXdIrmbG6OM>:

<http://opensourceforu.com/2011/10/syn-flooding-using-scapy-and-prevention-using-iptables/>