**Dynamic Port Forwarding with SSH and SOCKS Tunneling**

**SSH Local Port Forwarding**

**Scanning the Pivot Target**

nmap -sT -p22,3306 10.129.202.64

Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-24 12:12 EST

Nmap scan report for 10.129.202.64

Host is up (0.12s latency).

PORT STATE SERVICE

22/tcp open ssh

3306/tcp closed mysql

Nmap done: 1 IP address (1 host up) scanned in 0.68 seconds

The Nmap output shows that the SSH port is open. To access the MySQL service, we can either SSH into the server and access MySQL from inside the Ubuntu server, or we can port forward it to our localhost on port 1234 and access it locally. A benefit of accessing it locally is if we want to execute a remote exploit on the MySQL service, we won't be able to do it without port forwarding. This is due to MySQL being hosted locally on the Ubuntu server on port 3306. So, we will use the below command to forward our local port (1234) over SSH to the Ubuntu server.

**Executing the Local Port Forward**

ssh -L 1234:localhost:3306 [ubuntu@10.129.202.64](mailto:ubuntu@10.129.202.64)

run this command on local host means kali machine

The -L command tells the SSH client to request the SSH server to forward all the data we send via the port 1234 to localhost:3306 on the Ubuntu server. By doing this, we should be able to access the MySQL service locally on port 1234. We can use Netstat or Nmap to query our local host on 1234 port to verify whether the MySQL service was forwarded.

**Confirming Port Forward with Netstat**

Noor123@htb[/htb]$ netstat -antp | grep 1234

(Not all processes could be identified, non-owned process info

will not be shown, you would have to be root to see it all.)

tcp 0 0 127.0.0.1:1234 0.0.0.0:\* LISTEN 4034/ssh

tcp6 0 0 ::1:1234 :::\* LISTEN 4034/ssh

**Confirming Port Forward with Nmap**

Dynamic Port Forwarding with SSH and SOCKS Tunneling

Noor123@htb[/htb]$ nmap -v -sV -p1234 localhost

**Forwarding Multiple Ports**

Noor123@htb[/htb]$ ssh -L 1234:localhost:3306 -L 8080:localhost:80 ubuntu@10.129.202.64

**Enabling Dynamic Port Forwarding with SSH**

Noor123@htb[/htb]$ ssh -D 9050 ubuntu@10.129.202.64

The -D argument requests the SSH server to enable dynamic port forwarding. Once we have this enabled, we will require a tool that can route any tool's packets over the port 9050. We can do this using the tool proxychains, which is capable of redirecting TCP connections through TOR, SOCKS, and HTTP/HTTPS proxy servers and also allows us to chain multiple proxy servers together. Using proxychains, we can hide the IP address of the requesting host as well since the receiving host will only see the IP of the pivot host. Proxychains is often used to force an application's TCP traffic to go through hosted proxies like SOCKS4/SOCKS5, TOR, or HTTP/HTTPS proxies.

To inform proxychains that we must use port 9050, we must modify the proxychains configuration file located at /etc/proxychains.conf. We can add socks4 127.0.0.1 9050 to the last line if it is not already there.

**Checking /etc/proxychains.conf**

Noor123@htb[/htb]$ tail -4 /etc/proxychains.conf

# meanwile

# defaults set to "tor"

socks4 127.0.0.1 9050

**Using Nmap with Proxychains**

proxychains nmap -v -sn 172.16.5.1-200

e more important note to remember here is that we can only perform a full TCP connect scan over proxychains. The reason for this is that proxychains cannot understand partial packets. If you send partial packets like half connect scans, it will return incorrect results. We also need to make sure we are aware of the fact that host-alive checks may not work against Windows targets because the Windows Defender firewall blocks ICMP requests (traditional pings) by default.

[A full TCP connect scan](https://nmap.org/book/scan-methods-connect-scan.html) without ping on an entire network range will take a long time. So, for this module, we will primarily focus on scanning individual hosts, or smaller ranges of hosts we know are alive, which in this case will be a Windows host at 172.16.5.19.

We will perform a remote system scan using the below command.

**Enumerating the Windows Target through Proxychains**

Noor123@htb[/htb]$ proxychains nmap -v -Pn -sT 172.16.5.19

**Using Metasploit with Proxychains**

proxychains msfconsole

**Using rdp\_scanner Module**

msf6 > search rdp\_scanner

msf6 > search rdp\_scanner

Matching Modules

================

# Name Disclosure Date Rank Check Description

- ---- --------------- ---- ----- -----------

0 auxiliary/scanner/rdp/rdp\_scanner normal No Identify endpoints speaking the Remote Desktop Protocol (RDP)

Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/rdp/rdp\_scanner

msf6 > use 0

msf6 auxiliary(scanner/rdp/rdp\_scanner) > set rhosts 172.16.5.19

rhosts => 172.16.5.19

msf6 auxiliary(scanner/rdp/rdp\_scanner) > run

|S-chain|-<>-127.0.0.1:9050-<><>-172.16.5.19:3389-<><>-OK

|S-chain|-<>-127.0.0.1:9050-<><>-172.16.5.19:3389-<><>-OK

|S-chain|-<>-127.0.0.1:9050-<><>-172.16.5.19:3389-<><>-OK

[\*] 172.16.5.19:3389 - Detected RDP on 172.16.5.19:3389 (name:DC01) (domain:DC01) (domain\_fqdn:DC01) (server\_fqdn:DC01) (os\_version:10.0.17763) (Requires NLA: No)

[\*] 172.16.5.19:3389 - Scanned 1 of 1 ho

**Using xfreerdp with Proxychains**

proxychains xfreerdp /v:172.16.5.19 /u:victor /p:pass@123

**By Noor ur Rehman MS (Computer Science)**

**Remote/Reverse Port Forwarding with SSH**