Anno Accademico 2021/2022

# Laboratorio di Internet


Politecnico di Torino

# Report Individuale

*Nmap*

Gruppo 14

Alessandro Ciullo (s269589)

# 1 Configurazione di rete

Nell'effettuare i test é stata usata la seguente configurazione di rete:

| | |
|---|---|
| Indirizzo di rete | 192.168.146.211 |
| Default gateway | 192.168.146.29 |
| NetMask | 255.255.255.0 |

| | |
|---|---|
| Livello fisico scheda di rete | usb0 (Ethernet II) |
| Velocità | 100Mbps |
| Stato Duplex | Full |
| Stato del collegamento | BROADCAST,MULTICAST,UP,LOWER UP |

Tabella 1.1: Configurazione di rete

# 2 Introduzione

L'obiettivo di questo laboratorio è l'apprendimento dell'utilizzo del software "Nmap" che in combinazione con "Wireshark", software con cui abbiamo già maturato una discreta esperienza, ci permette di effettuare la scansione delle porte di un dato host. Tramite lo strumento è infatti possibile conoscere lo stato di un host e delle sue porte sfruttando i più popolari protocolli internet.

# 3 Scansione porte 7,22,53 di un host Android

In questa sezione ho connesso il mio computer, con sistema operativo "Arch Linux", al mio smartphone "Android" tramite thetering-USB

## 3.1 Scansione senza privilegi di Root

Ho eseguito la scansione delle porte 7, 22 ,53 lanciando il comando:

```
1 nmap 192.168.146.29 -p 7,22,53
```

Lanciando il comando senza privilegi di amministratore la scansione di default che viene effettuata è la "TCP Connect Scan", che fa uso della system call "connect" per avviare dei tentativi di handshake con l'host in esame. Nmap prima di cominciare la scansione vera e propria, valuta la presenza di un host attivo all'indirizzo specificato eseguendo delle connect verso le porte più popolari: 80 (http) e 443 (https). Nel momento in cui ottiene una risposta, anche negativa (RST), procede ad effettuare una "reverse DNS lookup" per ricavare l'hostname associato all'ip nel caso dovesse esistere.

| Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|
| 0.000000000 | 192.168.146.211 | 192.168.146.29 | TCP | 74 | 39086 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3408111082 TSecr=0 WS=128 |
| 0.000022397 | 192.168.146.211 | 192.168.146.29 | TCP | 74 | 56824 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3408111082 TSecr=0 WS=128 |
| 0.006900208 | 192.168.146.29 | 192.168.146.211 | TCP | 54 | 80 → 39086 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 0.007116260 | 192.168.146.211 | 192.168.146.29 | DNS | 87 | Standard query 0x1c94 PTR 29.146.168.192.in-addr.arpa |
| 0.007916675 | 192.168.146.29 | 192.168.146.211 | TCP | 54 | 443 → 56824 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 0.008954058 | 192.168.146.29 | 192.168.146.211 | DNS | 87 | Standard query response 0x1c94 No such name PTR 29.146.168.192.in-addr.arpa |
| 0.009074193 | 192.168.146.211 | 192.168.146.29 | TCP | 74 | 33090 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3408111091 TSecr=0 WS=128 |
| 0.009089212 | 192.168.146.211 | 192.168.146.29 | TCP | 74 | 36512 → 53 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3408111091 TSecr=0 WS=128 |
| 0.009097391 | 192.168.146.211 | 192.168.146.29 | TCP | 74 | 35692 → 7 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3408111091 TSecr=0 WS=128 |
| 0.011240833 | 192.168.146.29 | 192.168.146.211 | TCP | 54 | 22 → 33090 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 0.011241445 | 192.168.146.29 | 192.168.146.211 | TCP | 74 | 53 → 36512 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=67833562 TSecr=3408111091 WS=256 |
| 0.011271580 | 192.168.146.211 | 192.168.146.29 | TCP | 66 | 36512 → 53 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3408111093 TSecr=67833562 |
| 0.011241706 | 192.168.146.29 | 192.168.146.211 | TCP | 54 | 7 → 35692 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 0.011327888 | 192.168.146.211 | 192.168.146.29 | TCP | 66 | 36512 → 53 [RST, ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3408111093 TSecr=67833562 |

Figura 3.1: Wireshark

Come mostrato in figura 3.1 seguono una serie di "connect" verso le porte specificate nel comando. Se si ottiene una risposta positiva (SYNACK) la porta viene etichettata come "aperta", e dopo che il kernel ha terminato il three way handshake la connessione viene chiusa attraverso la system call "close" che inoltra un segmento TCP con flag RST attivo; talvolta se il SYNACK arriva in tempi "lunghi" Nmap potrebbe aver già indicato al SO di concludere il tentativo di connessione e perciò viene generato direttamente un segmento di RST. Se si dovesse ricevere una risposta negativa (RSTACK), la porta viene etichettata come "chiusa" e la connessione non viene istanziata. Se non si ottiene nessuna risposta, invece, Nmap inoltra una seconda richiesta di connessione in modo da assicurarsi che il pacchetto precedente non sia andato perso, e se nuovamente non riceve risposta procede ad etichettare la porta come "filtrata".

Nel caso in questione possiamo visualizzare i risultati ottenuti:

```
[alessandro@alessandro ~]$ nmap 192.168.146.29 -p 7,22,53
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-24 01:10 CEST
Nmap scan report for 192.168.146.29
Host is up (0.0054s latency).

PORT   STATE  SERVICE
7/tcp  closed echo
22/tcp closed ssh
53/tcp open   domain

Nmap done: 1 IP address (1 host up) scanned in 0.06 seconds
```

## 3.2 Scansione con privilegi di amministratore

In questo secondo caso abbiamo invece eseguito la scansione con privilegi di amministratore lanciando il comando:

```
[alessandro@alessandro ~]$ sudo nmap 192.168.146.29 -p 7,22,53
[sudo] password di alessandro:
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-24 02:28 CEST
Nmap scan report for 192.168.146.29
Host is up (0.0048s latency).

PORT   STATE  SERVICE
7/tcp  closed echo
22/tcp closed ssh
53/tcp open   domain
MAC Address: 1A:8D:D5:6D:C7:33 (Unknown)
```

Come si può evincere dai risultati ottenuti e dai pacchetti mostrati in figura 3.2 l'esecuzione della scansione viene fatta seguendo un diverso algoritmo chiamato "TCP SYN (Stealth) Scan".

| Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|
| 0.000000000 | 7e:42:da:ae:0b:80 | Broadcast | ARP | 42 | Who has 192.168.146.29? Tell 192.168.146.211 |
| 0.006180540 | 1a:8d:d5:6d:c7:33 | 7e:42:da:ae:0b:80 | ARP | 42 | 192.168.146.29 is at 1a:8d:d5:6d:c7:33 |
| 0.056349046 | 192.168.146.211 | 192.168.146.29 | DNS | 87 | Standard query 0x6bfb PTR 29.146.168.192.in-addr.arpa |
| 0.296497770 | 192.168.146.29 | 192.168.146.211 | DNS | 164 | Standard query response 0x6bfb No such name PTR 29.146.168. |
| 0.314115834 | 192.168.146.211 | 192.168.146.29 | TCP | 58 | 38723 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 0.314164129 | 192.168.146.211 | 192.168.146.29 | TCP | 58 | 38723 → 53 [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 0.314181254 | 192.168.146.211 | 192.168.146.29 | TCP | 58 | 38723 → 7 [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 0.316189610 | 192.168.146.29 | 192.168.146.211 | TCP | 54 | 22 → 38723 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 0.316190766 | 192.168.146.29 | 192.168.146.211 | TCP | 58 | 53 → 38723 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 |
| 0.316291406 | 192.168.146.211 | 192.168.146.29 | TCP | 54 | 38723 → 53 [RST] Seq=1 Win=0 Len=0 |
| 0.316324863 | 192.168.146.29 | 192.168.146.211 | TCP | 54 | 7 → 38723 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |

Figura 3.2: Wireshark

Sfruttando i privilegi di amministratore Nmap può elaborare in maniera diretta i pacchetti che inoltra, e fare uso di protocolli che necessitano di tali permessi. In questo caso infatti la presenza di un host attivo all'indirizzo dato viene verificata sfruttando il protocollo ARP a cui si è solitamente obbligati a rispondere

(tranne in caso di violazione di semantica) quando la richiesta proviene dall'interno della propria LAN. Una volta accertata la presenza dell'host si tenta nuovamente di ricavarne l'hostname.

Come nel caso precedente si prova a stabilire una connessione con le porte indicate nel comando, ma poichè i pacchetti sono costruiti in maniera diretta da Nmap, ci sono alcune differenze. Rispetto ai pacchetti generati senza permessi di Root infatti hanno le seguenti caratteristiche: sono più brevi poichè non vengono inserite opzioni aggiuntive dal SO all'header TCP; la porta sorgente e il numero di sequenza sono gli stessi per ogni pacchetto poichè non vengono selezionati secondo i paradigmi del kernel; la grandezza della "recive windows" è diversa e il checksum non viene verificato prima dell'invio del pacchetto.

Nel caso di una risposta positiva non viene instaurata la connessione come deciso dalle norme del protocollo in quanto il sistema operativo, non a conoscenza del pacchetto inoltrato da nmap, interpreta il segmento SYNACK ricevuto come una "conversazione incompleta" e perciò risponde con un RST terminando l'handshake senza nemmeno scomodare Nmap per la risposta. Tutti questi accorgimenti permettono un'interazione minima con l'host mappato e ne deriva il nome "Stealth" della popolare scansione.

# 4 Scansione porte 7,22,53 di un host fuori dalla propria LAN

Avendo ottenuto comportamenti praticamente coincidenti analizzando un altro dispositivo sulla mia LAN ho deciso di fare un'analisi delle porte verso il sito Scanme.nmap.org, che per fini didattici autorizza la scansione del suo server.

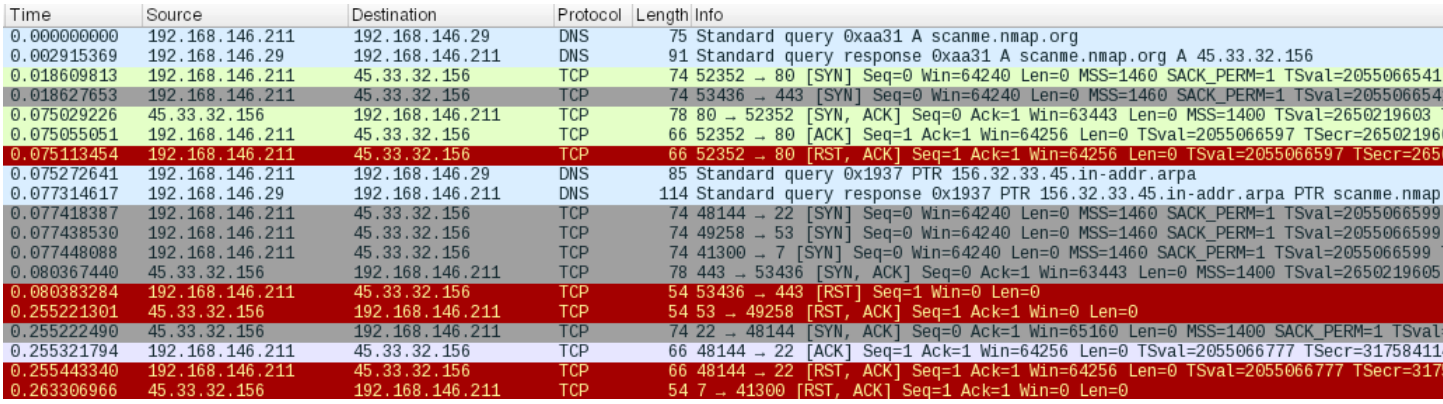
## 4.1 Scansione senza privilegi di Root

In questa sezione eseguo la scansione senza permessi di amministratore lanciando il comando:

```
[alessandro@alessandro ~]$ nmap scanme.nmap.org -p 7,22,53
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-24 04:29 CEST
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.098s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f

PORT   STATE  SERVICE
7/tcp  closed echo
22/tcp open   ssh
53/tcp closed domain
```

Questa volta invece che l'indirizzo ip ho usato l'host name e perciò la prima operazione svolta è una DNS query come mostrato in figura 4.1. In maniera perfettamente identica al caso 3.1 vengono svolte delle operazioni prima di verifica e poi di scansione sull host indicato, separate da una reverse DNS lookup, che questa volta è riuscita a ricavare l'hostname (seppur già ne fossimo a conoscenza). Osservando i pacchetti catturati è importante far caso alla porta 443, con cui non si riesce a stabilire una connessione nei tempi necessari, e quindi non viene portato a termine l'handshake come spiegato precedentemente.



| Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|
| 0.000000000 | 192.168.146.211 | 192.168.146.29 | DNS | 75 | Standard query 0xaa31 A scanme.nmap.org |
| 0.002915369 | 192.168.146.29 | 192.168.146.211 | DNS | 91 | Standard query response 0xaa31 A scanme.nmap.org A 45.33.32.156 |
| 0.018609813 | 192.168.146.211 | 45.33.32.156 | TCP | 74 | 52352 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2055066541 |
| 0.018627653 | 192.168.146.211 | 45.33.32.156 | TCP | 74 | 53436 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=205506654 |
| 0.075029226 | 45.33.32.156 | 192.168.146.211 | TCP | 78 | 80 → 52352 [SYN, ACK] Seq=0 Ack=1 Win=63443 Len=0 MSS=1400 TSval=2650219603 |
| 0.075055051 | 192.168.146.211 | 45.33.32.156 | TCP | 66 | 52352 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2055066597 TSecr=26502196 |
| 0.075113454 | 192.168.146.211 | 45.33.32.156 | TCP | 66 | 52352 → 80 [RST, ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2055066597 TSecr=265 |
| 0.075272641 | 192.168.146.211 | 192.168.146.29 | DNS | 85 | Standard query 0x1937 PTR 156.32.33.45.in-addr.arpa |
| 0.077314617 | 192.168.146.29 | 192.168.146.211 | DNS | 114 | Standard query response 0x1937 PTR 156.32.33.45.in-addr.arpa PTR scanme.nmap |
| 0.077418387 | 192.168.146.211 | 45.33.32.156 | TCP | 74 | 48144 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2055066599 |
| 0.077438530 | 192.168.146.211 | 45.33.32.156 | TCP | 74 | 49258 → 53 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2055066599 |
| 0.077448088 | 192.168.146.211 | 45.33.32.156 | TCP | 74 | 41300 → 7 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2055066599 |
| 0.080367440 | 45.33.32.156 | 192.168.146.211 | TCP | 78 | 443 → 53436 [SYN, ACK] Seq=0 Ack=1 Win=63443 Len=0 MSS=1400 TSval=2650219605 |
| 0.080383284 | 192.168.146.211 | 45.33.32.156 | TCP | 54 | 53436 → 443 [RST] Seq=1 Win=0 Len=0 |
| 0.255221301 | 45.33.32.156 | 192.168.146.211 | TCP | 54 | 53 → 49258 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 0.255222490 | 45.33.32.156 | 192.168.146.211 | TCP | 74 | 22 → 48144 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1400 SACK_PERM=1 TSval |
| 0.255321794 | 192.168.146.211 | 45.33.32.156 | TCP | 66 | 48144 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2055066777 TSecr=31758411 |
| 0.255443340 | 192.168.146.211 | 45.33.32.156 | TCP | 66 | 48144 → 22 [RST, ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2055066777 TSecr=317 |
| 0.263306966 | 45.33.32.156 | 192.168.146.211 | TCP | 54 | 7 → 41300 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |

Figura 4.1: Wireshark

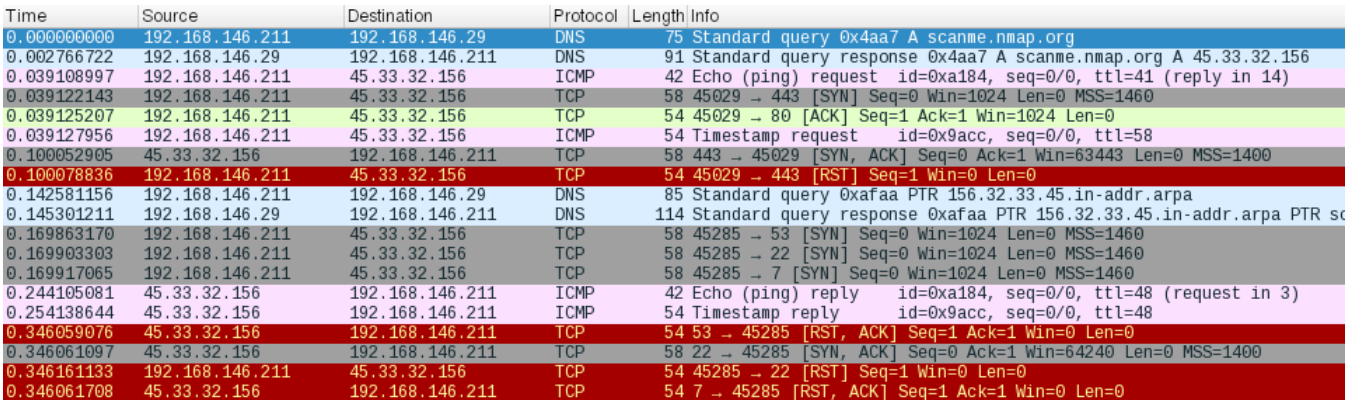
## 4.2 Scansione con privilegi di amministratore

In questa quarta esperienza abbiamo invece invocato il comando con i permessi di root verso un host fuori dalla nostra LAN:

```
[alessandro@alessandro ~]$ sudo nmap scanme.nmap.org -p 7,22,53
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-24 04:42 CEST
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.099s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f

PORT   STATE  SERVICE
7/tcp  closed echo
22/tcp open   ssh
53/tcp closed domain
```

Come mostrato i risultati ottenuti sono gli stessi, ma l'algoritmo utilizzato è il "TCP Stealth". A differenza del caso in cui il dispositivo scansionato è sulla propria LAN, la certifica della presenza di un dispositivo all'indirizzo dato viene svolta in maniera diversa poichè non può abusare del protocollo ARP. In questo caso fa uso di 4 diversi pacchetti: TCP SYN (porta 80/433), TCP ACK (porta 80/433), ICMP Echo ed ICMP Timestamp. Il segmento TCP ACK viene inviato per ottenere un messaggio di RST dall'host scansionato come reazione ad un pacchetto inaspettato, aumentando quindi le possibilità di ricevere una risposta, mentre i messaggi ICMP sono inoltrati per ottenere informazioni aggiuntive come la data in uso sul sistema.



| Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|
| 0.000000000 | 192.168.146.211 | 192.168.146.29 | DNS | 75 | Standard query 0x4aa7 A scanme.nmap.org |
| 0.002766722 | 192.168.146.29 | 192.168.146.211 | DNS | 91 | Standard query response 0x4aa7 A scanme.nmap.org A 45.33.32.156 |
| 0.039108997 | 192.168.146.211 | 45.33.32.156 | ICMP | 42 | Echo (ping) request id=0xa184, seq=0/0, ttl=41 (reply in 14) |
| 0.039122143 | 192.168.146.211 | 45.33.32.156 | TCP | 58 | 45029 → 443 [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 0.039125207 | 192.168.146.211 | 45.33.32.156 | TCP | 54 | 45029 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0 |
| 0.039127956 | 192.168.146.211 | 45.33.32.156 | ICMP | 54 | Timestamp request id=0x9acc, seq=0/0, ttl=58 |
| 0.100052905 | 45.33.32.156 | 192.168.146.211 | TCP | 58 | 443 → 45029 [SYN, ACK] Seq=0 Ack=1 Win=63443 Len=0 MSS=1400 |
| 0.100078836 | 192.168.146.211 | 45.33.32.156 | TCP | 54 | 45029 → 443 [RST] Seq=1 Win=0 Len=0 |
| 0.142581156 | 192.168.146.211 | 192.168.146.29 | DNS | 85 | Standard query 0xafaa PTR 156.32.33.45.in-addr.arpa |
| 0.145301211 | 192.168.146.29 | 192.168.146.211 | DNS | 114 | Standard query response 0xafaa PTR 156.32.33.45.in-addr.arpa PTR sc |
| 0.169863170 | 192.168.146.211 | 45.33.32.156 | TCP | 58 | 45285 → 53 [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 0.169903303 | 192.168.146.211 | 45.33.32.156 | TCP | 58 | 45285 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 0.169917065 | 192.168.146.211 | 45.33.32.156 | TCP | 58 | 45285 → 7 [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 0.244105081 | 45.33.32.156 | 192.168.146.211 | ICMP | 42 | Echo (ping) reply id=0xa184, seq=0/0, ttl=48 (request in 3) |
| 0.254138644 | 45.33.32.156 | 192.168.146.211 | ICMP | 54 | Timestamp reply id=0x9acc, seq=0/0, ttl=48 |
| 0.346059076 | 45.33.32.156 | 192.168.146.211 | TCP | 54 | 53 → 45285 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 0.346061097 | 45.33.32.156 | 192.168.146.211 | TCP | 58 | 22 → 45285 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1400 |
| 0.346161133 | 192.168.146.211 | 45.33.32.156 | TCP | 54 | 45285 → 22 [RST] Seq=1 Win=0 Len=0 |
| 0.346061708 | 45.33.32.156 | 192.168.146.211 | TCP | 54 | 7 → 45285 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |

Figura 4.2: Wireshark

# 5 Scansione porte 1-150 di un host Android

In questa sezione portiamo a termine una scansione su un numero consistente di porte e ne analizziamo il comportamento, utilizzando lo script bash 5.9 per raccogliere i dati necessari dal file di cattura esportato da Wireshark e usando uno script Gnuplot 5.10 per ottenerne dei grafici.

## 5.1 TCP SYN scan

Lanciamo la scansione di default con i permessi di root utilizzando il comando:

```
[alessandro@alessandro ~]$ sudo nmap 192.168.146.29 -p 1-150
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-24 05:42 CEST
Nmap scan report for 192.168.146.29
Host is up (0.0037s latency).
Not shown: 149 closed tcp ports (reset)
PORT   STATE SERVICE
53/tcp open  domain
MAC Address: 1A:8D:D5:6D:C7:33 (Unknown)
```

Dopo pochi istanti otteniamo i risultati sul terminale e su Wireshark e ne ricaviamo il grafico 5.1 che mette in relazione le porte scansionate con l'istante di tempo in cui è avvenuta la scansione. Come mostrato in figura i messaggi di SYN vengono inoltrati in "burst". Nel primo burst vengono analizzate le porte più comunemente usate (HTTP–80, FTP–21, SSH–22, POP3–110 IMAP–143) per motivi di ottimizzazione, dopodiché le altre porte vengono scansionate in maniera casuale, a meno che non sia specificato diversamente (-r), in modo da destare meno sospetti.

Non essendo presente un firewall tra me e il mio telefono nessuna porta viene identificata come "filtered" e l'unica porta aperta è la 53 (DNS) in quanto svolgendo la funzione di hotspot si occupa anche della risoluzione dei domini.

Per via del funzionamento del protocollo TCP, qualsiasi porta, alla ricezione di un TCP SYN è tenuta solitamente a rispondere con un SYNACK o un RSTACK, perciò i messaggi di "probe" hanno una percentuale di successo elevatissima e non è quasi mai necessario inviarne due per una stessa porta se non in caso di filtraggio. I segmenti SYN mandati sono infatti esattamente 150 come si può evincere con maggior facilità dal grafico 5.2 che è stato ottenuto riordinando le porte scansionate.

Ho inoltre ripetuto la scansione anche verso il server "scanme.nmap.org" ottenendo risultati coincidenti a meno della porta 443 che viene scansionata per verificare la presenza di attività all'indirizzo poichè Nmap non può far uso di ARP al di fuori della propria LAN. Sono stati infatti scansionati esattamente 151 indirizzi e la scala delle y del grafico 5.5 e del grafico 5.6 è stata dilatata per accogliere il protocollo HTTPS.

## 5.2 UDP scan

In quest'ultima esperienza ho invece eseguito una scansione di tipo UDP. Questo algoritmo di mappatura necessità obbligatoriamente dei permessi di amministratore e se non gli vengono forniti si ottiene il seguente messaggio:

```
[alessandro@alessandro ~]$ nmap 192.168.146.29 -p 1-150 -sU
You requested a scan type which requires root privileges.
QUITTING!
```

Ho quindi lanciato il comando correttamente:

```
Nmap done: 1 IP address (1 host up) scanned in 0.23 seconds
[alessandro@alessandro ~]$ sudo nmap 192.168.146.29 -p 1-150 -sU
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-24 05:42 CEST
Nmap scan report for 192.168.146.29
Host is up (0.0024s latency).
Not shown: 148 closed udp ports (port-unreach)
PORT   STATE         SERVICE
53/udp open          domain
67/udp open|filtered dhcps
MAC Address: 1A:8D:D5:6D:C7:33 (Unknown)
```

I privilegi di Root sono necessari affinchè si possano i leggere messaggi ICMP, fondamentali per l'analisi delle porte UDP. L'unico modo per avere la certezza che una porta sia chiusa è infatti ricevere un messaggio ICMP "Destination Unreachable (type 3) Port Unreachable (code3)" dall'host scansionato, contenente in esso i primi 64 bit del pacchetto UDP indirizzato ad una porta chiusa. Questa volta non si ha, però, una distinzione netta tra gli stati di una porta, e si può ottenere come risultato lo stato doppio "open—filtered". Questo è causato dal protocollo UDP, che usando un paradigma "best effort", se riceve dei pacchetti formattati in maniera errata su una porta aperta non è tenuto a rispondere in nessuno modo e scarta semplicemente il pacchetto. É infatti un risultato molto comune poichè Nmap invia dei pacchetti UDP vuoti. Per ovviare a questo problema Nmap genera anche un payload coerente con i protocolli più popolari, come avviene per DNS e NTP. Quando si ottiene una risposta UDP da una porta, essa viene etichettata come "aperta", e riceve dal sistema operativo, all'oscuro delle nostre azioni, un messaggio ICMP di "Port Unreachable".

Caratteristica fondamentale di questa scansione è il tempo necessario al suo completamento: Le porte chiuse infatti rispondono soltanto attraverso il protocollo ICMP che è solitamente limitato a non più di un messaggio per secondo; inoltre per via delle molto comuni "non risposte" nmap sonda spesso delle porte più di una volta, fenomeno causato anche dalle lunghe attese di ICMP che lasciano pensare ad un pacchetto perso.

Osservando il grafico 5.3 è infatti semplice notare come il tempo necessario alla scansione sia di circa 150 secondi (1 pacchetto ICMP per secondo), mentre il grafico riordinato 5.4 mostra in maniera chiara come alcune porte siano scansionate molteplici volte.

Ho infine ripetuto la scansione UDP verso l'host "scanme.nmap.org" ottenendo risultati analoghi come mostrato in figura 5.7 e in figura 5.8.

# Appendice

## Andrid Host

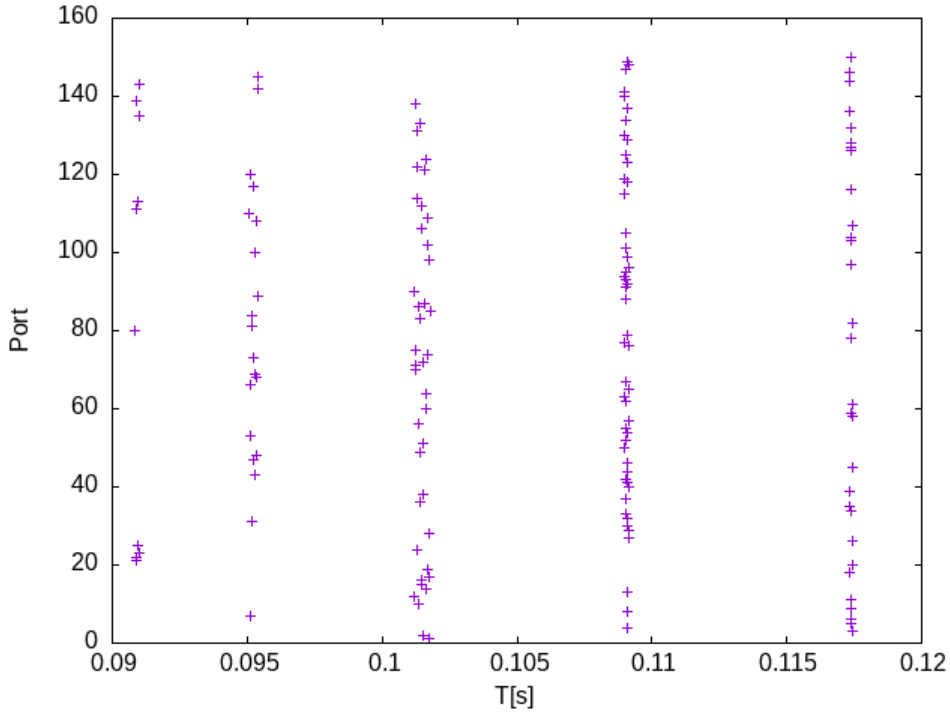



Figura 5.1: Grafico scansione porta 1-150 tcp su rete locale

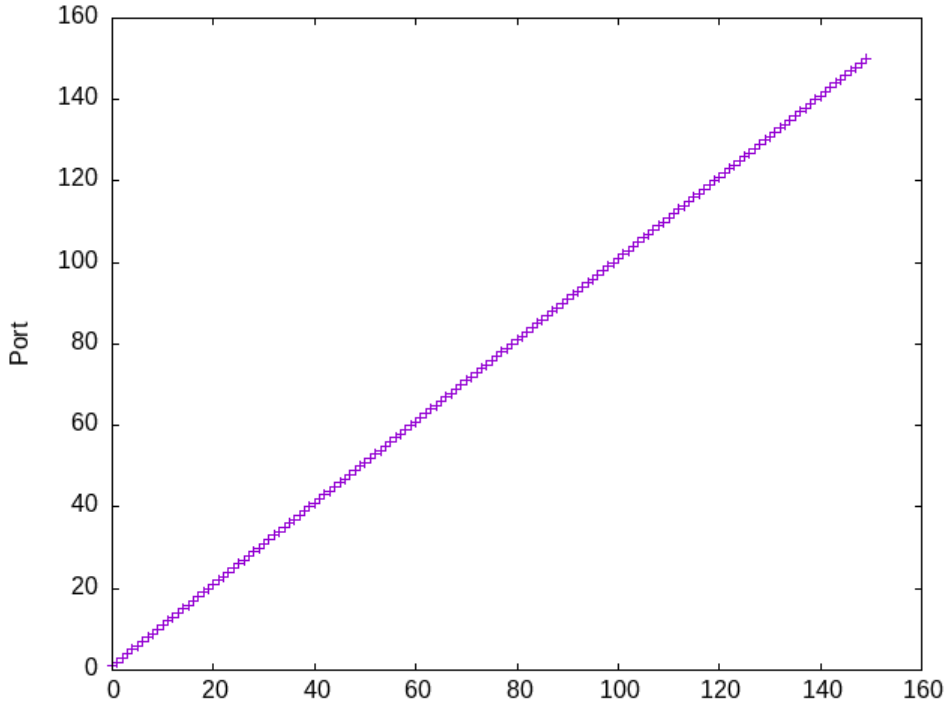



Figura 5.2: Grafico con porte riordinate (tcp)

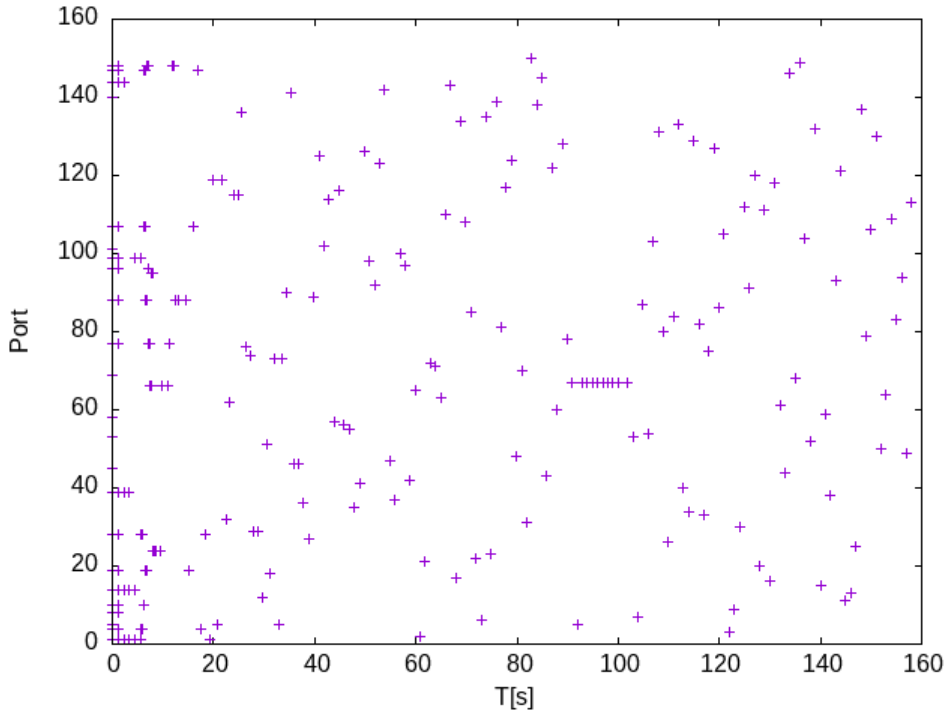



Figura 5.3: Grafico scansione porta 1-150 udp su rete locale

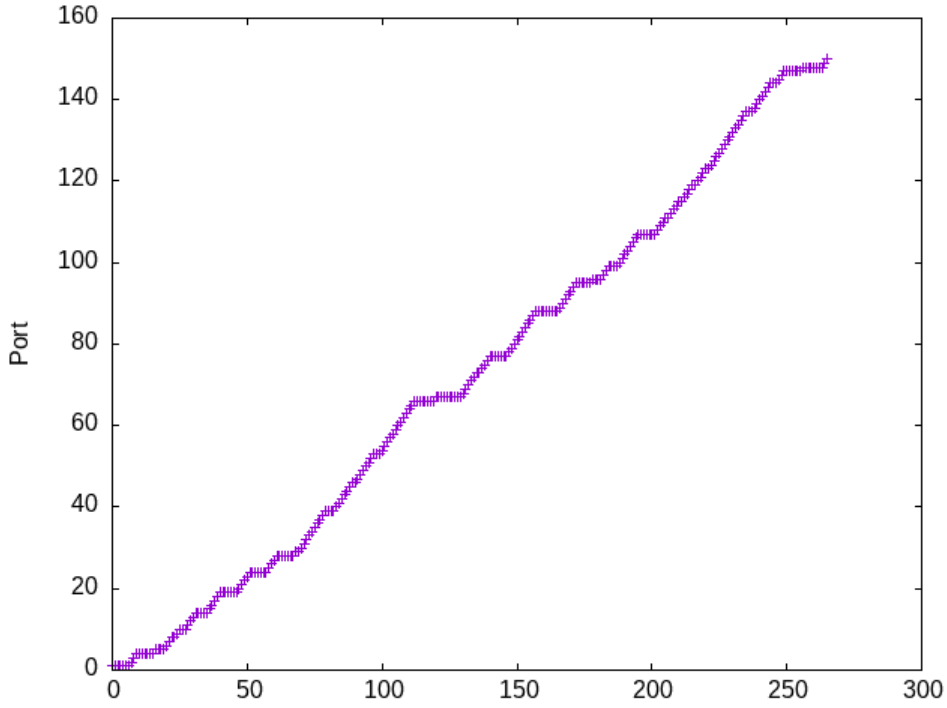



Figura 5.4: Grafico con porte riordinate (udp)

**scanme.nmap.org**

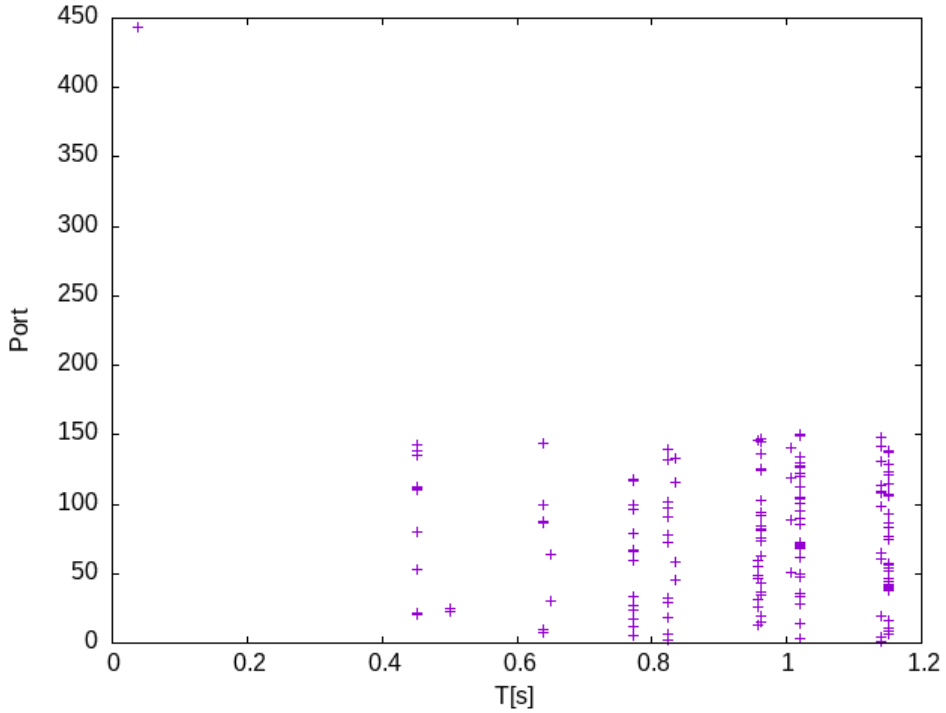



Figura 5.5: Grafico scansione porta 1-150 tcp su rete esterna

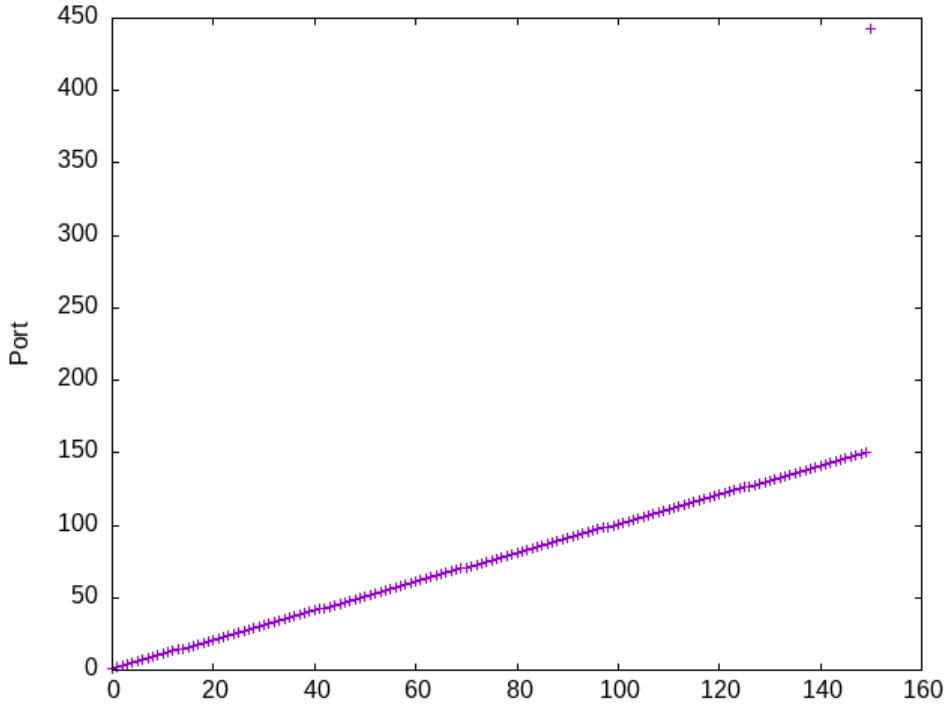



Figura 5.6: Grafico con porte riordinate (tcp)

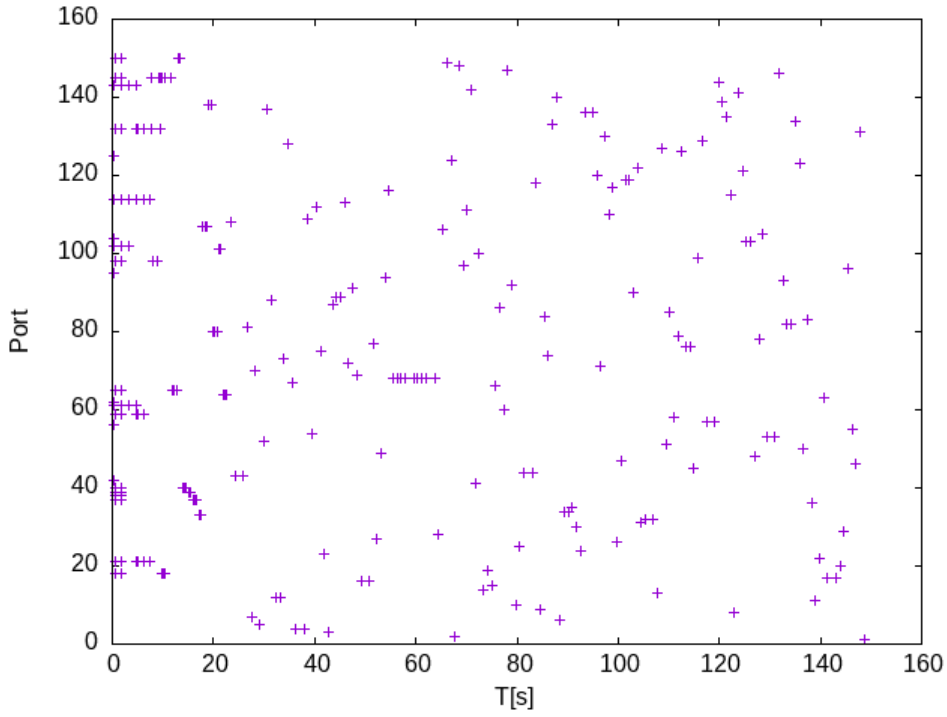



Figura 5.7: Grafico scansione porta 1-150 udp su rete esterna

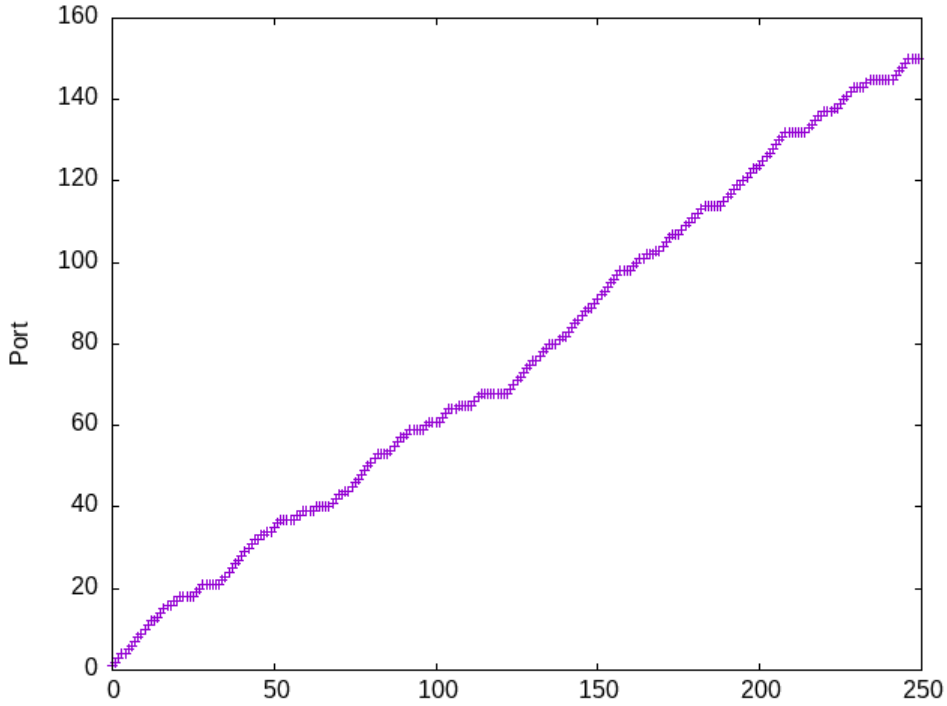



Figura 5.8: Grafico con porte riordinate (udp)

```
tail client.dat -n +2 > tmp.dat

cat tmp.dat | tr -s ' ' | cut -d ' ' -f3 > time.dat
cat tmp.dat | tr -s ' ' | cut -d ' ' -f10 > port.dat
paste time.dat port.dat > time-port.dat

rm time.dat
rm port.dat
rm tmp.dat
```

Figura 5.9: Script per la raccolta dei dati

```
set term png
set out "scan_plot.png"
set xlabel " T[s] "
set ylabel " Port "

plot "time-port.dat" using 1:2 title "" with point

set xlabel ""
set out "ordered_port.png"
plot "< sort -n -k 2 time-port.dat" using :2 title "" with point
```

Figura 5.10: Script per la generazione dei grafici con GNUPLOT