Relazione individuale

Analisi del comportamento di NMAP

Michele Ferrero 268542

Gruppo 22 Anno 2021/2022

Introduzione

L'esperienza consiste nell'analizzare il comportamento dello strumento "nmap", in particolare le funzioni di scan delle porte TCP e UDP sia come superuser che come user.
La rete può essere di due tipi in base al tipo di analisi e di comportamento analizzato:

- Due host (A e B) collegati mediante cavo ethernet ed aventi entrambi il sistema linux "Xubuntu".
- Due host (A ed il server "scanme.nmap.org") collegati mediante rete wireless e diversi router, aventi entrambi sistema operativo linux (per trovare il sistema operativo del server si è utilizzata l'opzione -O della nmap, la quale in base alle risposte ai pacchetti inviati dall'host A riesce a stimare quale sistema operativo sia in uso con una buona % di accuratezza). Per limitare il traffico indesiderato si è utilizzato il router del telefono, così da avere unicamente un host connesso.

La prima configurazione è stata utilizzata sia in laboratorio che a casa mentre la seconda è stata utilizzata unicamente a casa.
Si riportano le configurazioni dell'host A in entrambe le configurazioni:

Configurazione 1 host A:	
physical layer advertised	10baseT/Half 10baseT/Full 100baseT/Half 100baseT/Full 1000baseT/Full
Current speed of eth0	1000Mb/s
Current duplex status	Full
Link status	Active
IP address	172.16.22.2
netmask	255.255.255.192
Ip address default gateway	0.0.0.0 (not assigned)

Configurazione 2 host A:	
Current speed of wlan0	130Mb/s
Current duplex status	Half
Link status	Active
IP address	192.168.136.64
netmask	255.255.255.0
Ip address default gateway	192.168.136.223

Scansione delle porte TCP (config 1)

La TCP port scan varia in base ai permessi che vengono dati alla nmap, infatti, una qualsiasi funzione in linux per utilizzare un raw socket, e quindi avere la possibilità di inviare in rete dei raw packets, necessita di essere eseguita dal superuser.

Nmap utilizza il raw socket per avere pieno controllo sulla comunicazione dal layer 3 in sù. Infatti, quando una applicazione layer 7 invia un pacchetto, i layers sottostanti gestiscono l'invio e la ricezione della risposta in maniera totalmente trasparente: l'utilizzo di un raw socket permette ad una applicazione di scrivere e ricevere direttamente payload livello 2 così da gestire in maniera autonoma e controllata lo scambio di pacchetti.
Quando si utilizza il comando

nmap IP_ADDRESS -sT

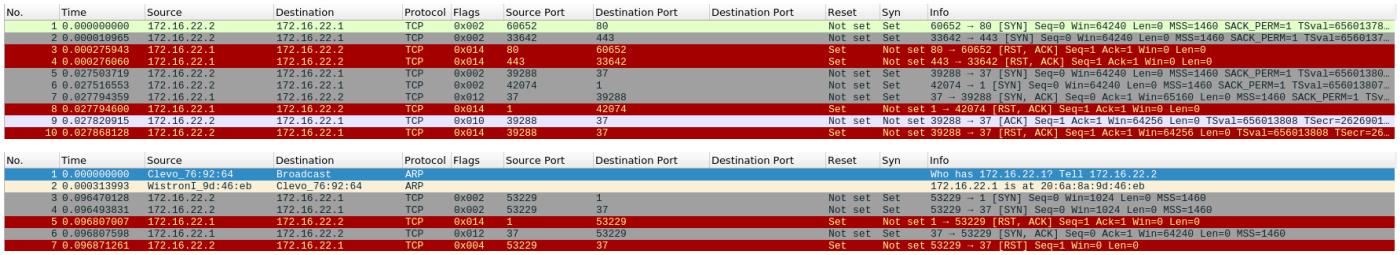
avendo nmap solamente i permessi base non è in grado di utilizzare raw sockets e quindi si limita ad effettuare una "TCP connect scan", la quale consiste nell'instaurazione di una connessione TCP per verificare lo stato della porta: se questa è aperta la 3-way handshake terminerà con successo altrimenti l'host B restituirà un pacchetto TCP con flag RST attivo, il quale indica il rifiuto di connessione (Reset).
Se invece si vogliono utilizzare i raw sockets il comando da utilizzare è il seguente

sudo nmap IP_ADDRESS -sS

il quale esegue l'applicazione "nmap" come superuser, permettendole di effettuare una scansione migliore in termini di efficienza e possibilità di non essere scoperti rispetto alla "TCP connect scan". La nmap, eseguita dall'host A, sfrutta i raw socket per la "TCP stealth scan" ovvero eseguire per ciascuna porta una "half 3-way handshake" con l'host B. Per "half 3-way handshake" si intende che, dopo il pacchetto SYN inviato da A e la risposta da parte di B con un SYN/ACK ad indicare lo stato aperto della porta, A risponde utilizzando un pacchetto avente flag RST attivo in quanto non interessata alla effettiva apertura della connessione.
Le principali differenze tra queste due modalità sono le seguenti:

- La TCP connect scan è più lenta, essa deve terminare la 3-way handshake aprendo la connessione per poi chiuderla, invece la TCP stealth rifiuta la connessione appena scopre che la porta è aperta.
- La TCP stealth scan è più difficile da individuare, essendo che non apre effettivamente una connessione essa non risulta nei log di sistema del server.
- La TCP stealth scan utilizzando un raw socket può gestire autonomamente il payload dei pacchetti inviati, per questo limita al minimo il numero di bytes da trasmettere e riduce la complessità del protocollo. Infatti, non solo invia dei pacchetti SYN con unica opzione la negoziazione della MSS (perché strettamente necessaria), ma non cambia nemmeno la source port ed il sequence number, essendo questi inutili per questo scopo.

Es: [sudo] nmap 172.16.22.1 [-sS/-sT] -p 1,37 (1 chiusa 37 aperta)



No.	Time	Source	Destination	Protocol	Flags	Source Port	Destination Port	Destination Port	Reset	Syn	Info
1	0.000000000	172.16.22.2	172.16.22.1	TCP	0x002	60652	80		Not set	Set	60652 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=65601378…
2	0.000010965	172.16.22.2	172.16.22.1	TCP	0x002	33642	443		Not set	Set	33642 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=6560137…
3	0.000275943	172.16.22.1	172.16.22.2	TCP	0x014	80	60652		Set	Not set	80 → 60652 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4	0.000276066	172.16.22.1	172.16.22.2	TCP	0x014	443	33642		Set	Not set	443 → 33642 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
5	0.027503719	172.16.22.2	172.16.22.1	TCP	0x002	39288	37		Not set	Set	39288 → 37 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=65601380…
6	0.027516553	172.16.22.2	172.16.22.1	TCP	0x002	42074	1		Not set	Set	42074 → 1 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=656013807…
7	0.027794359	172.16.22.1	172.16.22.2	TCP	0x012	37	39288		Not set	Set	37 → 39288 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSv…
8	0.027794600	172.16.22.1	172.16.22.2	TCP	0x014	1	42074		Set	Not set	1 → 42074 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
9	0.027820915	172.16.22.2	172.16.22.1	TCP	0x010	39288	37		Not set	Not set	39288 → 37 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=656013808 TSecr=2626901…
10	0.027868128	172.16.22.2	172.16.22.1	TCP	0x014	39288	37		Set	Not set	39288 → 37 [RST, ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=656013808 TSecr=26…

No.	Time	Source	Destination	Protocol	Flags	Source Port	Destination Port	Destination Port	Reset	Syn	Info
1	0.000000000	Clevo_76:92:64	Broadcast	ARP							Who has 172.16.22.1? Tell 172.16.22.2
2	0.000313993	WistronI_9d:46:eb	Clevo_76:92:64	ARP							172.16.22.1 is at 20:6a:8a:9d:46:eb
3	0.096470128	172.16.22.2	172.16.22.1	TCP	0x002	53229	1		Not set	Set	53229 → 1 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
4	0.096493831	172.16.22.2	172.16.22.1	TCP	0x002	53229	37		Not set	Set	53229 → 37 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
5	0.096807007	172.16.22.1	172.16.22.2	TCP	0x014	1	53229		Set	Not set	1 → 53229 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
6	0.096807598	172.16.22.1	172.16.22.2	TCP	0x012	37	53229		Not set	Set	37 → 53229 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
7	0.096871261	172.16.22.2	172.16.22.1	TCP	0x004	53229	37		Set	Not set	53229 → 37 [RST] Seq=1 Win=0 Len=0

Individuazione dell'host B (config 1 e 2)

L'applicazione nmap deve essere in grado di determinare qualora l'host B sia effettivamente attivo o meno prima di effettuare scansioni, questo per differenziare lo stato della porta "filtered" dall'host non presente in rete, in quanto entrambi non restituiscono pacchetti. Se nmap viene eseguito come user (indipendentemente da dove si trovi l'host B) allora prova a fare due tcp connect, uno alla porta 80 ed uno alla porta 443 e, se l'host B risponde (anche un pacchetto RST indica comunque che l'host è attivo in rete), allora inizia l'effettiva scansione delle porte. Questo metodo è però soggetto ad errori, per esempio se si avesse un firewall sulle porte 80 e 443 dell'host B la nmap restituirebbe "Host down" quando in realtà l'host è up ma filtrato. Se nmap viene invece eseguito dal superuser allora può utilizzare metodi più accurati per l'individuazione del secondo host, in particolare se questo è situato nella stessa rete dell'host A allora si può usare il protocollo ARP, metodo che prima nmap non poteva usare in quanto senza raw socket non è in grado di sapere l'esito della ARP request essendo la reply nota solo al layer 3, mentre se l'host B è situato in una rete esterna nmap invia 4 pacchetti differenti per avere una maggior possibilità di eludere il firewall del secondo host. Questi pacchetti sono: una echo request, una timestamp request, un TCP SYN alla porta 443 ed un TCP ACK alla porta 80.
Il primo dei quattro pacchetti che ritorna una risposta viene utilizzato durante la connessione per verificare che l'host B sia ancora attivo.

Esempi con il comando [sudo] nmap ip_interno/esterno_alla_LAN -p 1 :

No.	Time	Source	Destination	Protocol	Flags	Source Port	Destination Port	Destination Port	Reset	Syn	Info
3	11.504227992	Clevo_76:92:64	Broadcast	ARP							Who has 172.16.22.1? Tell 172.16.22.2
4	11.504501361	WistronI_9d:46:eb	Clevo_76:92:64	ARP							172.16.22.1 is at 20:6a:8a:9d:46:eb
5	11.504506432	172.16.22.2	172.16.22.1	TCP	0x002	47238	80		Not set	Set	47238 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=34449526…
6	11.504507780	172.16.22.2	172.16.22.1	TCP	0x002	34812	443		Not set	Set	34812 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3444952…
7	11.504779213	172.16.22.1	172.16.22.2	TCP	0x014	80	47238		Set	Not set	80 → 47238 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
8	11.504779369	172.16.22.1	172.16.22.2	TCP	0x014	443	34812		Set	Not set	443 → 34812 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
9	11.810143775	172.16.22.2	172.16.22.1	TCP	0x002	53384	1		Not set	Set	53384 → 1 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=344495293…
10	11.810469023	172.16.22.1	172.16.22.2	TCP	0x014	1	53384		Set	Not set	1 → 53384 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
11	16.735547502	WistronI_9d:46:eb	Clevo_76:92:64	ARP							Who has 172.16.22.2? Tell 172.16.22.1
12	16.735557022	Clevo_76:92:64	WistronI_9d:46:eb	ARP							172.16.22.2 is at 80:fa:5b:76:92:64
15	76.288503332	Clevo_76:92:64	Broadcast	ARP							Who has 172.16.22.1? Tell 172.16.22.2
16	76.288812375	WistronI_9d:46:eb	Clevo_76:92:64	ARP							172.16.22.1 is at 20:6a:8a:9d:46:eb
17	76.396537954	172.16.22.2	172.16.22.1	TCP	0x002	57605	1		Not set	Set	57605 → 1 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
18	76.396872368	172.16.22.1	172.16.22.2	TCP	0x014	1	57605		Set	Not set	1 → 57605 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
19	81.504619673	WistronI_9d:46:eb	Clevo_76:92:64	ARP							Who has 172.16.22.2? Tell 172.16.22.1
20	81.504641636	Clevo_76:92:64	WistronI_9d:46:eb	ARP							172.16.22.2 is at 80:fa:5b:76:92:64

TCP Connect scan prima e TCP Stealth scan dopo (LAN)

No.	Time	Source	Destination	Protocol	Flags	Source Port	Destination Port	Destination Port	Reset	Syn	Info
1	0.000000000	12:18:16:15:99:3c	IntelCor_14:f0:aa	ARP							Who has 192.168.136.64? Tell 192.168.136.223
2	0.000020690	IntelCor_14:f0:aa	12:18:16:15:99:3c	ARP							192.168.136.64 is at 5c:80:b6:14:f0:aa
9	17.094201793	192.168.136.64	45.33.32.156	ICMP							Echo (ping) request id=0xc32b, seq=0/0, ttl=56 (reply in 15)
10	17.094226812	192.168.136.64	45.33.32.156	TCP	0x002	45111	443		Not set	Set	45111 → 443 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
11	17.094233636	192.168.136.64	45.33.32.156	TCP	0x010	45111	80		Not set	Not set	45111 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
12	17.094239109	192.168.136.64	45.33.32.156	ICMP							Timestamp request id=0xe4af, seq=0/0, ttl=47
13	17.403641865	45.33.32.156	192.168.136.64	TCP	0x004	80	45111		Set	Not set	80 → 45111 [RST] Seq=1 Win=0 Len=0
14	17.403641874	45.33.32.156	192.168.136.64	TCP	0x014	443	45111		Set	Not set	443 → 45111 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15	17.403643789	45.33.32.156	192.168.136.64	ICMP							Echo (ping) reply id=0xc32b, seq=0/0, ttl=49 (request in 9)
16	17.403644096	45.33.32.156	192.168.136.64	ICMP							Timestamp reply id=0xe4af, seq=0/0, ttl=49
19	17.474519830	192.168.136.64	45.33.32.156	TCP	0x002	45367	1		Not set	Set	45367 → 1 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
20	17.922420146	45.33.32.156	192.168.136.64	TCP	0x014	1	45367		Set	Not set	1 → 45367 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

TCP Stealth Scan (Fuori LAN)

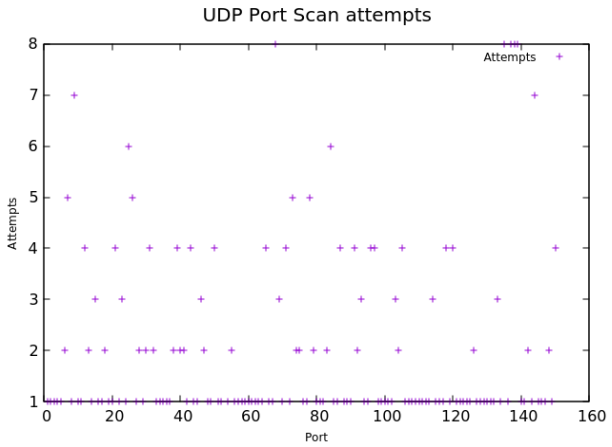
Scansione delle porte UDP (config 2)

La nmap offre, tra i vari servizi, anche lo scan di porte UDP. Questo è possibile solo tramite superuser e viene effettuata inviando pacchetti UDP all'host B e verificando la risposta di quest'ultimo. Se questo risponde con ICMP type 3 code 3 (destination unreachable) indica che la porta selezionata è chiusa, se risponde con un ICMP con code diversi dal 3 indica che la porta UDP è filtrata, se risponde con pacchetto di risposta significa che la porta è aperta (caso molto raro, l'host B risponde soltanto se il payload del pacchetto UDP è corretto ma utilizzando per molte porte, eccetto le più utilizzate, un pacchetto vuoto difficilmente si otterrà una reply) oppure non risponde proprio e questo ultimo caso è simbolico del problema dello scan UDP. Infatti, non si può sapere se una porta sia effettivamente aperta o filtrata senza inviare un UDP con payload specifico per il servizio, opzione presente in nmap come "-sV" (service and version probes). La libreria della opzione di service probes è molto limitata per due principali ragioni: perché i protocolli sono talmente tanti che difficilmente si può avere un payload corretto per ognuno di essi e perché alcuni protocolli richiedono degli identifier specifici (che nmap non conosce a priori) per generare una risposta.
La scansione UDP è molto più lenta rispetto a quella TCP sia perché molti sistemi operativi limitano le ICMP type 3 e quindi nmap deve rallentare per evitare di avere risultati poco attendibili sia perché UDP offre un servizio "best effort", con conseguente perdita di pacchetti durante lo scanning e quindi rallentamenti. Nmap tenta al più 8 volte con pacchetti UDP prima di dichiarare una porta "open|filtered".
Es: sudo nmap scanme.nmap.org -sU -p 122-123 (122 chiusa 123 aperta, caso raro di risposta senza service probes)

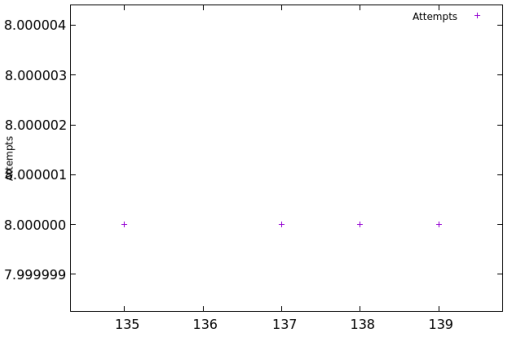
120	69.897508355	192.168.136.64	45.33.32.156	ICMP							Echo (ping) request id=0xf54d, seq=0/0, ttl=45 (reply in 124)
121	69.897537507	192.168.136.64	45.33.32.156	TCP	0x002	55959	443		Not set	Set	55959 → 443 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
122	69.897545511	192.168.136.64	45.33.32.156	TCP	0x010	55959	80		Not set	Not set	55959 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
123	69.897552546	192.168.136.64	45.33.32.156	ICMP							Timestamp request id=0x7ab1, seq=0/0, ttl=40
124	70.261870789	45.33.32.156	192.168.136.64	ICMP							Echo (ping) reply id=0xf54d, seq=0/0, ttl=49 (request in 120)
125	70.261882833	45.33.32.156	192.168.136.64	TCP	0x004	80	55959		Set	Not set	80 → 55959 [RST] Seq=1 Win=0 Len=0
126	70.261886448	45.33.32.156	192.168.136.64	TCP	0x014	443	55959		Set	Not set	443 → 55959 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
127	70.261871129	45.33.32.156	192.168.136.64	ICMP							Timestamp reply id=0x7ab1, seq=0/0, ttl=49
130	70.373131592	192.168.136.64	45.33.32.156	NTP				123			NTP Version 4, client
131	70.373156064	192.168.136.64	45.33.32.156	UDP				122			56215 → 122 Len=0
164	70.581469672	45.33.32.156	192.168.136.64	NTP				56215			NTP Version 4, server
165	70.581527772	192.168.136.64	45.33.32.156	ICMP				56215			Destination unreachable (Port unreachable)
166	70.584388338	45.33.32.156	192.168.136.64	ICMP				122			Destination unreachable (Port unreachable)

Generazione ed elaborazione dati per i grafici per verificare il numero di ritrasmissioni (config 2)

Per ottenere i dati necessari per il plot si è utilizzato uno script di bash, quest'ultimo accetta infatti come input un file di testo esportato da Wireshark e produce in output un file avente due colonne: la prima contiene il numero di tentativi effettuati per una certa porta mentre la seconda contiene il numero della porta. Per fare ciò utilizza, oltre alle funzioni bash già viste, la funzione uniq che con la sua opzione -c conta quante volte si ripete consecutivamente una determinata riga e la funzione sort per riordinarle (per permettere alla uniq di contarle correttamente).
Successivamente il file viene inserito in gnuplot, il quale mostra sull'asse X il numero della porta e sull'asse Y il numero di tentativi mostrando i tentativi effettuati dallo scan UDP di default.
Lo script è presente nell'appendice con i numeri 0.1 e 0.2.

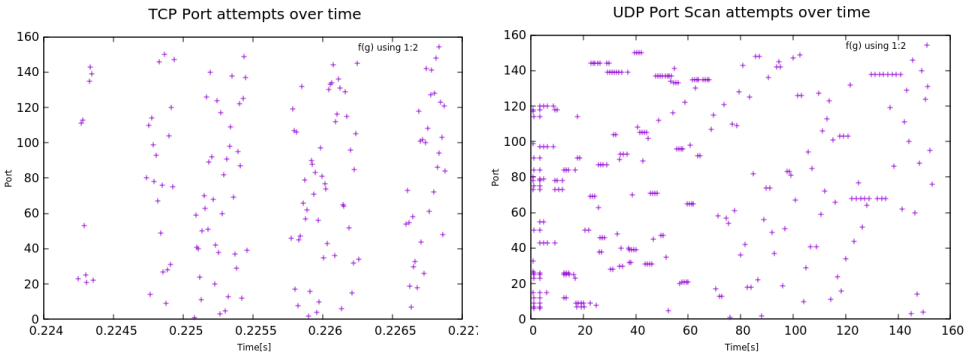


Dai grafici si nota come UDP sia un protocollo best effort e quindi vengano effettuate più ritrasmissioni (al più 8). Se si effettua uno zoom si può notare come le 8 ritrasmissioni appartengano a porte considerate come “open|filtered” proprio perché i pacchetti UDP inviati a quelle porte non ricevono risposta.
Es:




```
Not shown: 144 closed ports
PORT    STATE         SERVICE
68/udp  open|filtered dhcpc
123/udp open          ntp
135/udp open|filtered msrpc
137/udp open|filtered netbios-ns
138/udp open|filtered netbios-dgm
139/udp open|filtered netbios-ssn
```

Generazione ed elaborazione dati per i grafici per vedere l’andamento del port scanning nel tempo
Per quanto riguarda le trasmissioni di pacchetti nel tempo con gli script 1.1 e 1.2 si è giunti ad estrapolare tempo e porte di destinazione in modo da poterle rappresentare. Con il flag g si va a modificare titolo e file di input per cambiare da UDP Port Scan a TCP Port Scan. La TCP scan è stata eseguita con la configurazione 1 mentre la UDP scan con la configurazione 2.



Questi due grafici mostrano quanto affermato precedentemente ovvero che la scansione con UDP risulta essere più lenta (150 secondi circa contro i 0.22 della scansione TCP che nonostante le differenze di setup sono molti) e necessita di più trasmissioni rispetto a TCP (il numero di porte scansionate nei due grafici è lo stesso ma risultano più tentativi nel secondo grafico). Per quanto riguarda l’andamento della scansione nel tempo si noti che in ambedue i grafici l’ordine delle porte è casuale, così da essere meno individuabile, mentre per lo stesso motivo nel grafico TCP troviamo 3 “bande” verticali, infatti nmap non scansiona tutte le porte insieme (sarebbe facilmente individuabile) ma preferisce dividere la sua scansione in più sotto scansioni, ciascuna per un certo range di porte casuali.

APPENDICE

0.1:

```
file="porteUDPScan.txt"
cat $file | egrep -e "192.168.136.64" | tr -s " " | cut -d " " -f 7 | sort -k1,1 | uniq -c
```

0.2:

```
file="DataporteTCPScan.txt"
file1="DataporteUDPScan.txt"
g=0
f(s)=s==1?file:file1
title(s)=s==1?"TCP Port attempts":"UDP Port Scan attempts"
set title title(g) font "Verdana,15"
show title
set xlabel "Port"
set ylabel "Attempts"
set xtics font "Verdana,12"
set ytics font "Verdana,12"

plot f(g) using 2:1 title "Attempts"
```

1.1:

```
file="porteTCPScan.txt"
cat $file | egrep -e "172.16.22.2" | tr -s " " | cut -d " " -f 3,9
```

1.2:

```
file="DataporteTCPScanTime.txt"
file1="DataporteUDPScanTime.txt"
g=1
f(s)=s==1?file:file1
title(s)=s==1?"TCP Port attempts over time":"UDP Port Scan attempts over time"
set title title(g) font "Verdana,15"
show title
set xlabel "Time[s]"
set ylabel "Port"
set xtics font "Verdana,12"
set ytics font "Verdana,12"

plot f(g) using 1:2
```