HOST DISCOVERY

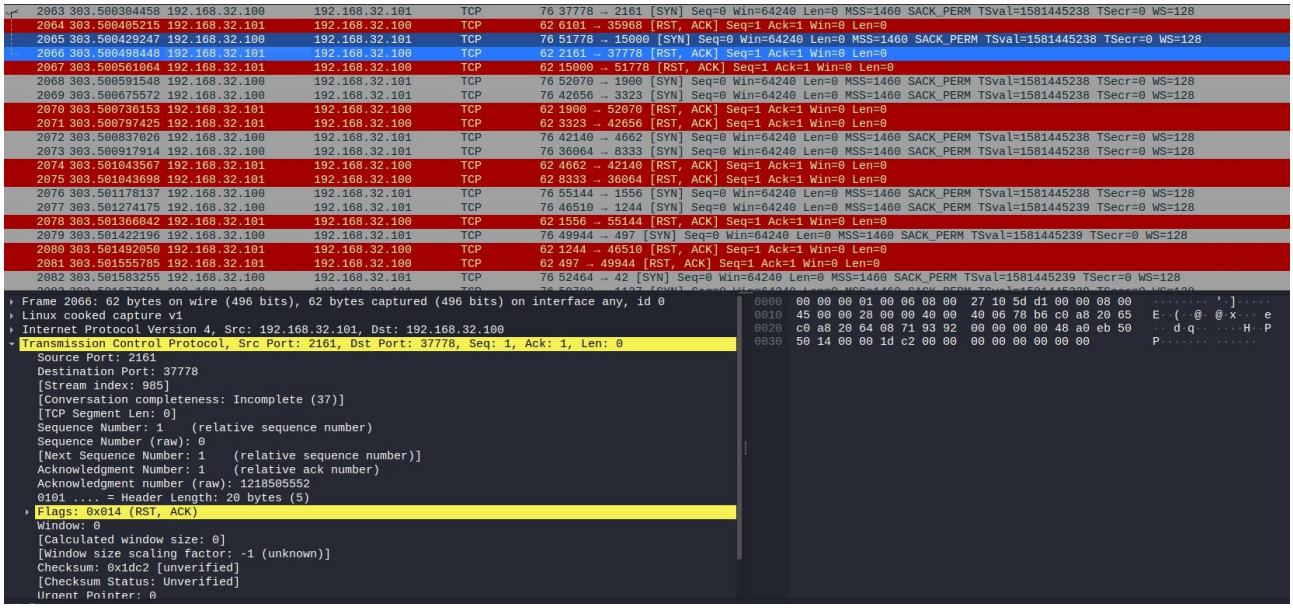
Per effettuare una host discovery sulla mia rete lan (formata da kali con ip 192.168.32.100 e metasploitable con ip 192.168.32.101), con nmap vado ad utilizzare il comando nmap **-sL** con target l'ip della macchina metasploitable, come segue. Riprovo con **-sn**, che controlla solo tramite ping senza scan invasivo e il risultato è comunque lo stesso in un tempo leggermente più breve.

```
┌──(riccbrun㉿kali)-[~]
└─$ nmap 192.168.32.101 -sL
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-18 15:15 CEST
Nmap scan report for 192.168.32.101
Nmap done: 1 IP address (0 hosts up) scanned in 13.03 seconds

┌──(riccbrun㉿kali)-[~]
└─$ nmap 192.168.32.101 -sn
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-18 15:15 CEST
Nmap scan report for 192.168.32.101
Host is up (0.00037s latency).
Nmap done: 1 IP address (1 host up) scanned in 13.01 seconds
```

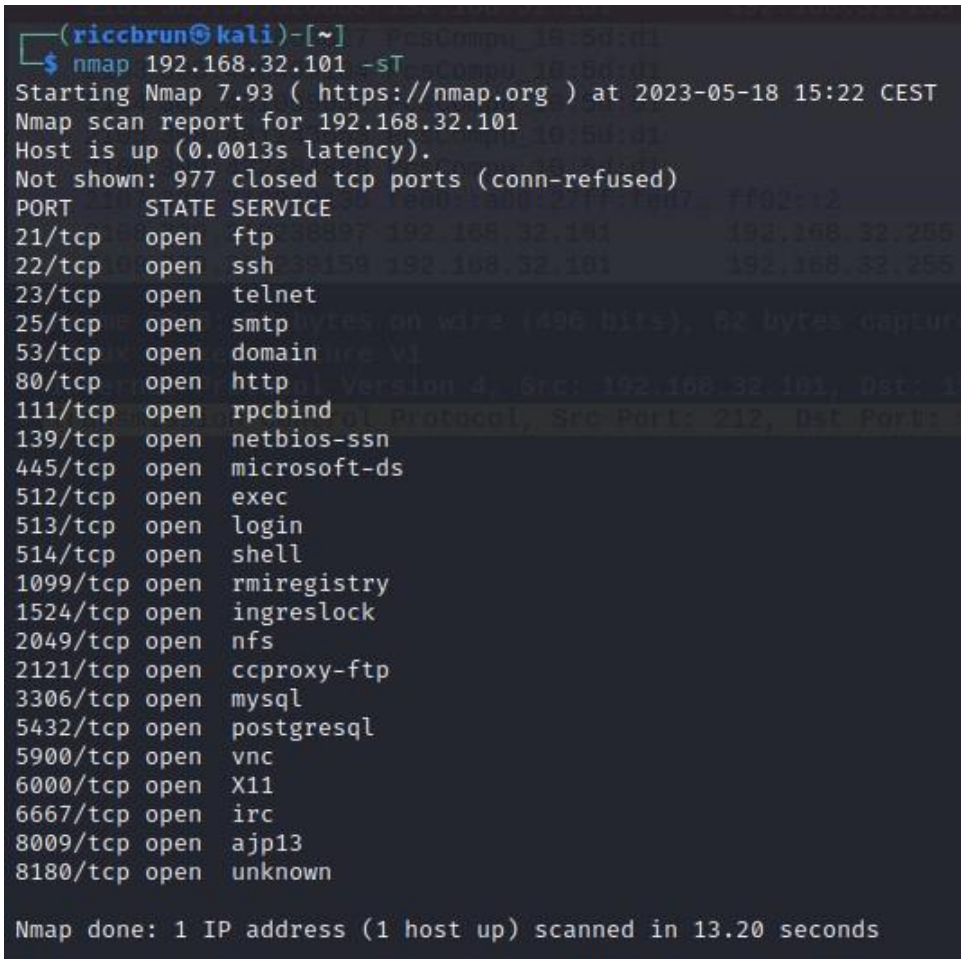
SCANSIONE TCP/SYN SULLE PORTE WELL-KNOWN

Con il comando **-sT**, metodo di scansione invasivo, nmap va a completare tutti i passaggi del three-way-handshake per instaurare una connessione TCP.

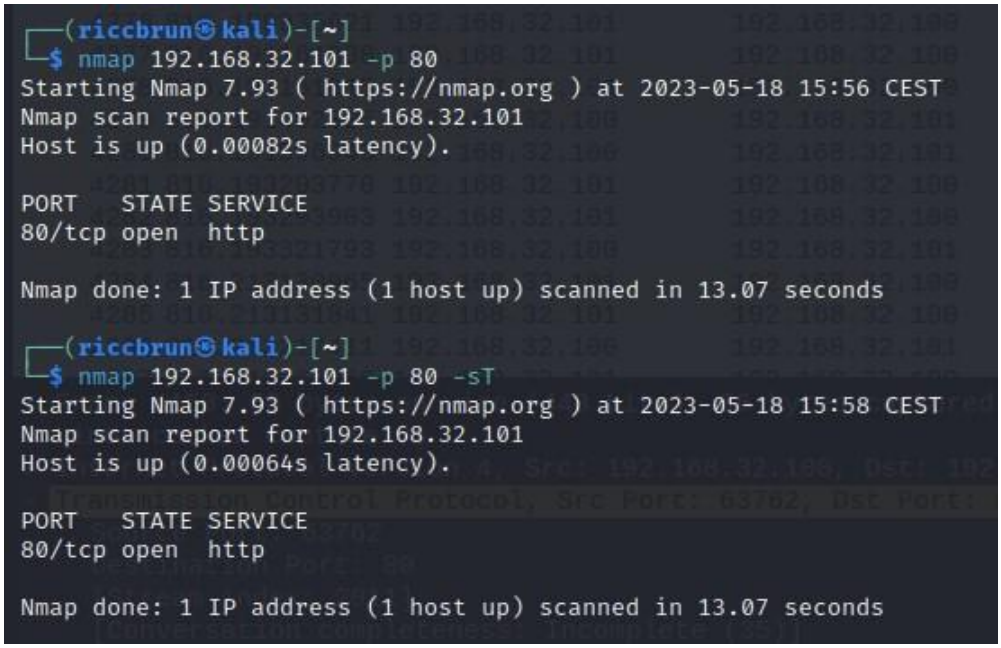
Senza specifiche, l'azione va effettivamente a scomodare una marea di porte, restituendo in gran parte risposte incomplete perché chiuse (la connessione viene rifiutata a priori). Ecco la schermata di wireshark:



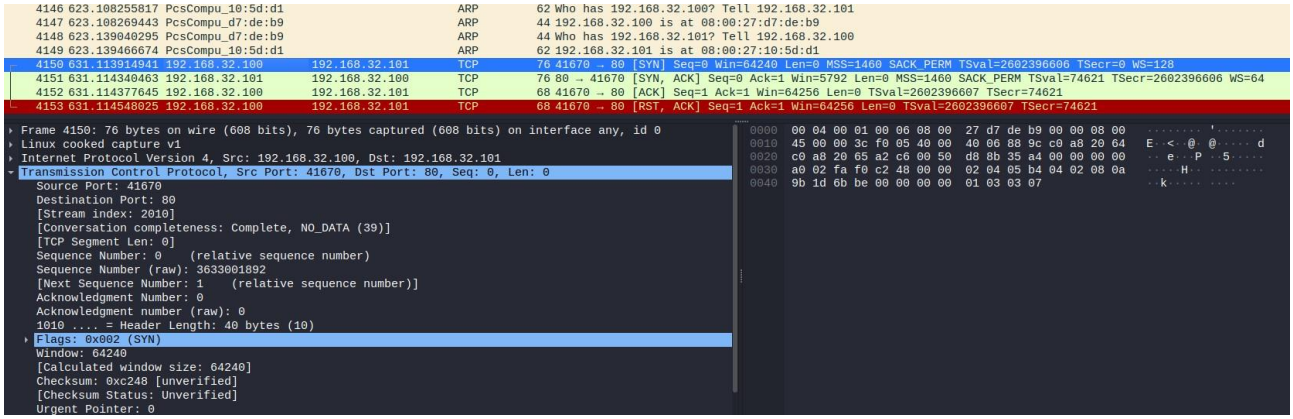
Alla fine dell'operazione comunque nmap mostra un elenco di porte aperte. **-sS** seppur in modo diverso porta allo stesso risultato.



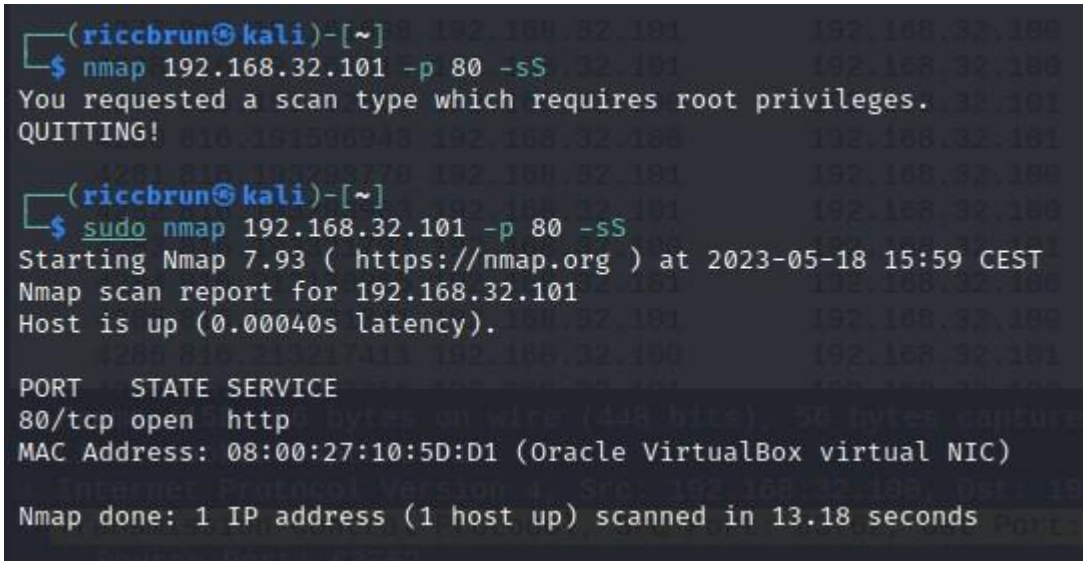
A questo punto la scelta migliore è isolare una porta e provare a rilanciare il comando specificando quella. Riprovo quindi a usare **-sT** inserendo come porta target la 80, una porta comune usata per il protocollo http.



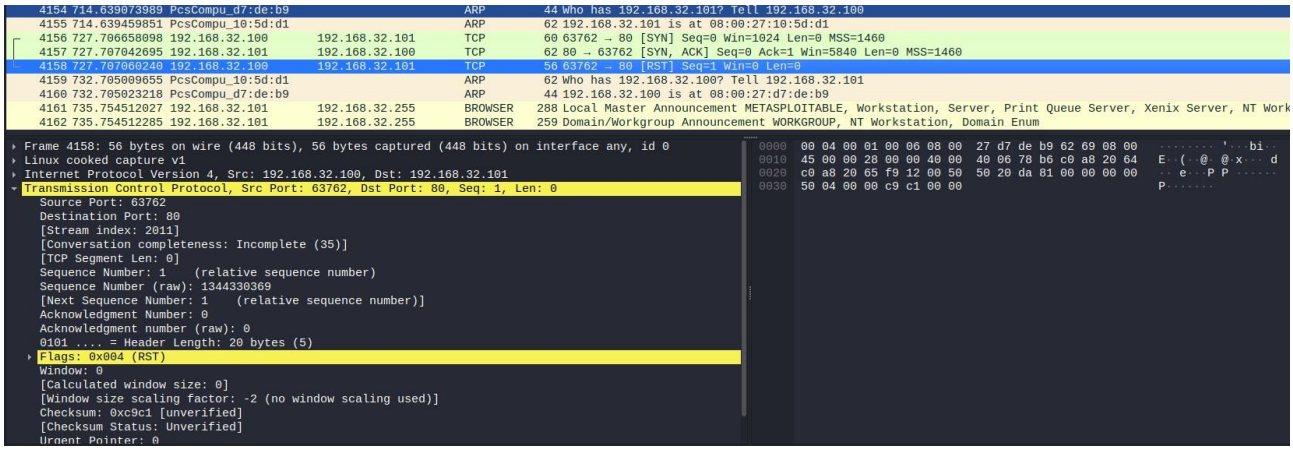
Adesso a differenza di prima, nel marasma di risposte negative riesco a visualizzare l'effettivo funzionamento del comando **-sT.** Dopo la chiamata ARP, la macchina kali con indirizzo 192.168.32.100: 41670 va a instaurare la connessione TCP con metasploitable all'ip 192.168.32.101:80 tramite il three-way-handshake visibile nell'immagine. Lo scambio di risposte SYN/ACK è completo.



Riprovo con la stessa porta con il comando **-sS** (che richiede privilegi di root):



In questo caso il risultato di wireshark è diverso, poiché l'**-sS** effettua una semplice scansione SYN, non va quindi a completare il three-way-handshake per instaurare la connessione TCP. Si ferma al primo passaggio e infatti la "conversazione" risulta incompleta:



SCANSIONE CON SWITCH-A SULLE PORTE WELL-KNOWN

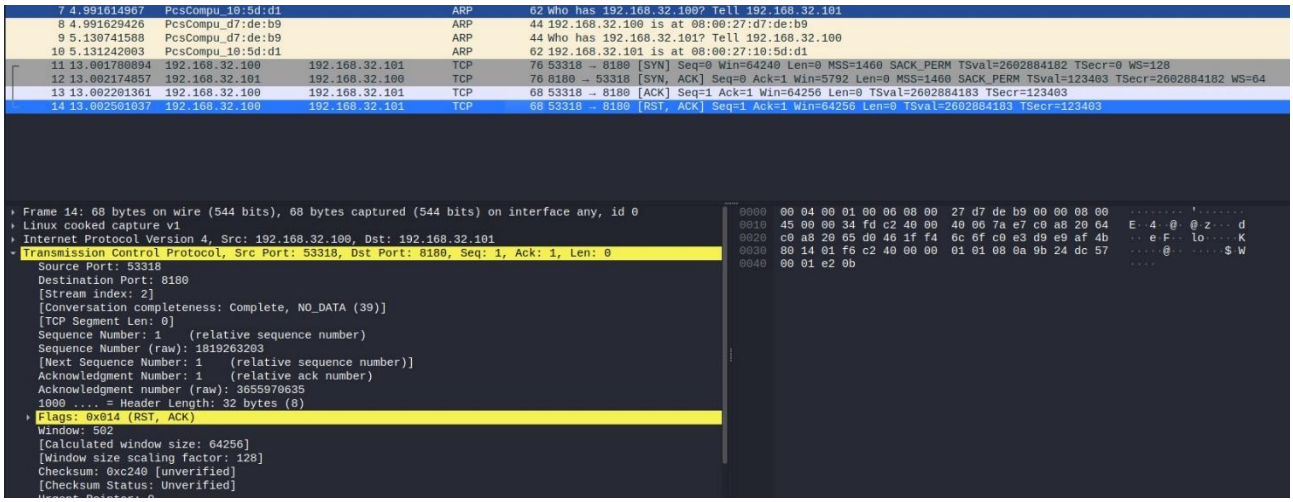
In ultimo voglio vedere una scansione con switch **-A** su questa porta well-known. Vado ad eseguire il comando e nmap è in grado di risalire al servizio attivo su quella porta.

```
┌──(riccbrun㉿kali)-[~]
└─$ nmap 192.168.32.101 -p 80 -A
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-18 16:00 CEST
Nmap scan report for 192.168.32.101
Host is up (0.00060s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.2.8 ((Ubuntu) DAV/2)
|_http-title: Metasploitable2 - Linux
|_http-server-header: Apache/2.2.8 (Ubuntu) DAV/2

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 23.75 seconds

┌──(riccbrun㉿kali)-[~]
└─$
```

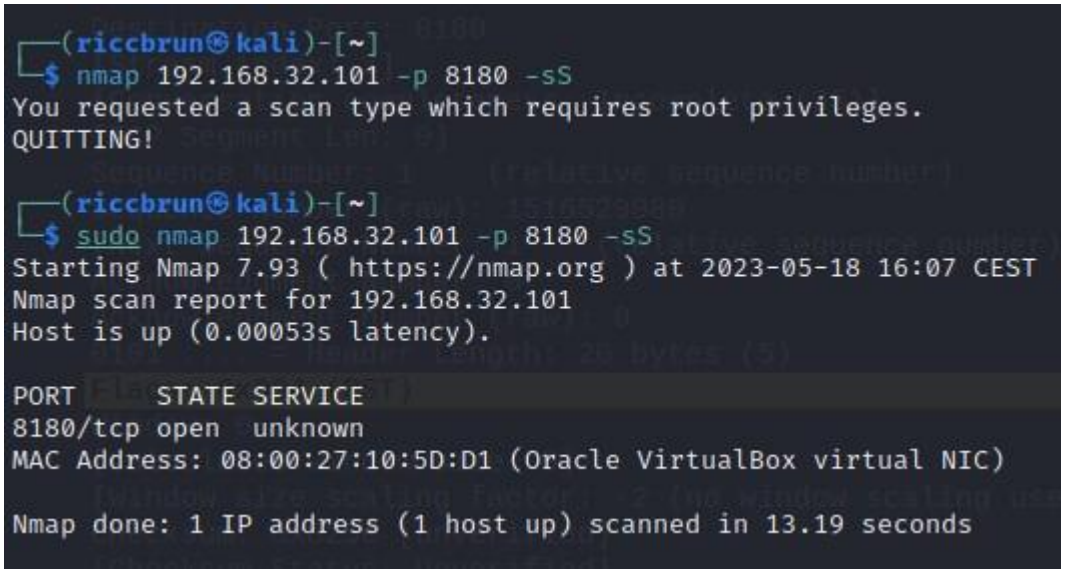
Per curiosità ho riprovato tutta la sequenza cambiando porta e inserendo quella “sconosciuta” 8180.

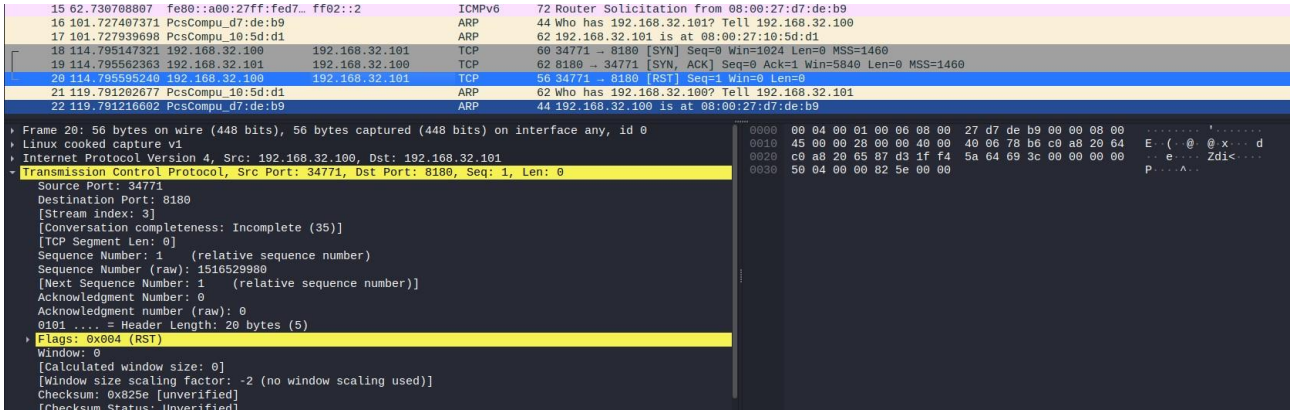
-sT e **-sS** riportano correttamente il protocollo completo

```
┌──(riccbrun㉿kali)-[~]
└─$ nmap 192.168.32.101 -p 8180 -sT
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-18 16:06 CEST
Nmap scan report for 192.168.32.101
Host is up (0.00060s latency).

PORT     STATE SERVICE
8180/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 13.08 seconds
```

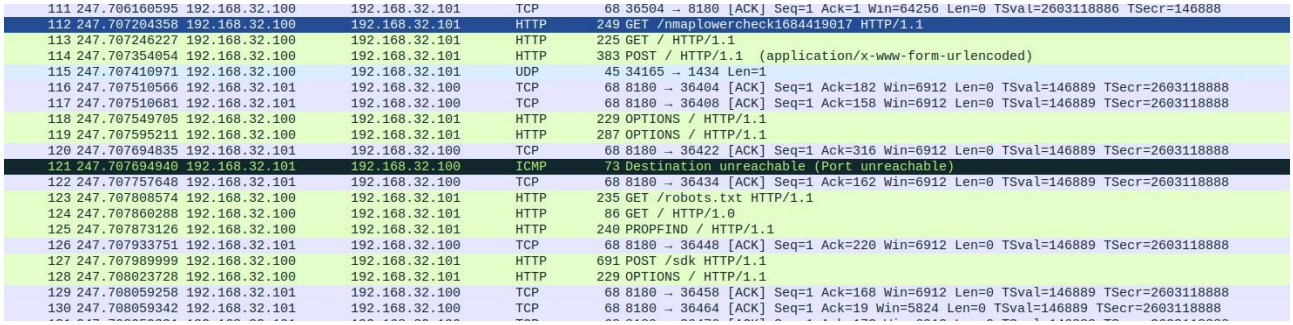


E incompleto:

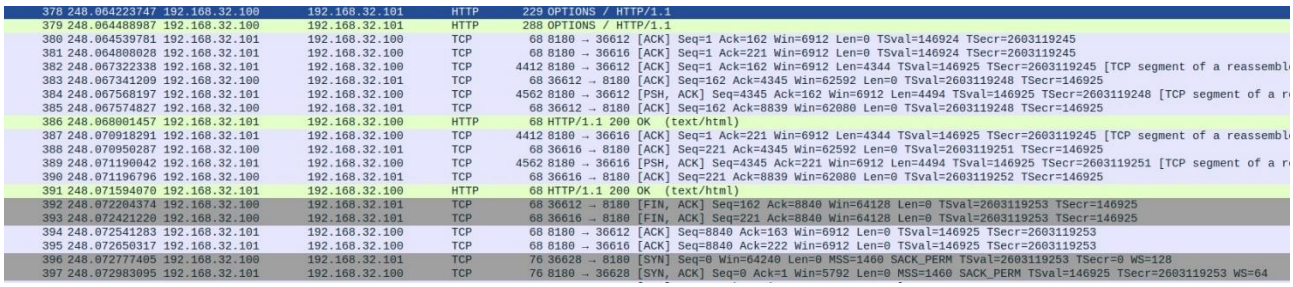




Eseguendo il comando **-A** invece risaliamo al servizio in ascolto sulla porta “sconosciuta”. Wireshark in questo frangente (come quello precedente) per ottenere le informazioni va a interrogare il target con delle richieste specifiche con diversi metodi http (GET, POST, OPTIONS, PR0PFIND…) e a cui segue anche una risposta OK 200 da metasploitable, come nell’immagine qui sotto:



111	247.706160595	192.168.32.100	192.168.32.101	TCP	68 36504 → 8180 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2603118886 TSecr=146888
112	247.707204358	192.168.32.100	192.168.32.101	HTTP	249 GET /nmaplowercheck1684419017 HTTP/1.1
113	247.707246227	192.168.32.100	192.168.32.101	HTTP	225 GET / HTTP/1.1
114	247.707354054	192.168.32.100	192.168.32.101	HTTP	383 POST / HTTP/1.1 (application/x-www-form-urlencoded)
115	247.707410971	192.168.32.100	192.168.32.101	UDP	45 34165 → 1434 Len=1
116	247.707510566	192.168.32.101	192.168.32.100	TCP	68 8180 → 36404 [ACK] Seq=1 Ack=182 Win=6912 Len=0 TSval=146889 TSecr=2603118888
117	247.707510681	192.168.32.101	192.168.32.100	TCP	68 8180 → 36408 [ACK] Seq=1 Ack=158 Win=6912 Len=0 TSval=146889 TSecr=2603118888
118	247.707549705	192.168.32.100	192.168.32.101	HTTP	229 OPTIONS / HTTP/1.1
119	247.707595211	192.168.32.100	192.168.32.101	HTTP	287 OPTIONS / HTTP/1.1
120	247.707694835	192.168.32.101	192.168.32.100	TCP	68 8180 → 36422 [ACK] Seq=1 Ack=316 Win=6912 Len=0 TSval=146889 TSecr=2603118888
121	247.707694940	192.168.32.101	192.168.32.100	ICMP	73 Destination unreachable (Port unreachable)
122	247.707757648	192.168.32.101	192.168.32.100	TCP	68 8180 → 36434 [ACK] Seq=1 Ack=162 Win=6912 Len=0 TSval=146889 TSecr=2603118888
123	247.707808574	192.168.32.100	192.168.32.101	HTTP	235 GET /robots.txt HTTP/1.1
124	247.707860288	192.168.32.100	192.168.32.101	HTTP	86 GET / HTTP/1.0
125	247.707873126	192.168.32.100	192.168.32.101	HTTP	240 PROPFIND / HTTP/1.1
126	247.707933751	192.168.32.101	192.168.32.100	TCP	68 8180 → 36448 [ACK] Seq=1 Ack=220 Win=6912 Len=0 TSval=146889 TSecr=2603118888
127	247.707989999	192.168.32.100	192.168.32.101	HTTP	691 POST /sdk HTTP/1.1
128	247.708023728	192.168.32.100	192.168.32.101	HTTP	229 OPTIONS / HTTP/1.1
129	247.708059258	192.168.32.101	192.168.32.100	TCP	68 8180 → 36458 [ACK] Seq=1 Ack=168 Win=6912 Len=0 TSval=146889 TSecr=2603118888
130	247.708059342	192.168.32.101	192.168.32.100	TCP	68 8180 → 36464 [ACK] Seq=1 Ack=19 Win=5824 Len=0 TSval=146889 TSecr=2603118888



378	248.064223747	192.168.32.100	192.168.32.101	HTTP	229 OPTIONS / HTTP/1.1
379	248.064488987	192.168.32.100	192.168.32.101	HTTP	288 OPTIONS / HTTP/1.1
380	248.064539781	192.168.32.101	192.168.32.100	TCP	68 8180 → 36612 [ACK] Seq=1 Ack=162 Win=6912 Len=0 TSval=146924 TSecr=2603119245
381	248.064808028	192.168.32.101	192.168.32.100	TCP	68 8180 → 36616 [ACK] Seq=1 Ack=221 Win=6912 Len=0 TSval=146924 TSecr=2603119245
382	248.067322338	192.168.32.101	192.168.32.100	TCP	4412 8180 → 36612 [ACK] Seq=1 Ack=162 Win=6912 Len=4344 TSval=146925 TSecr=2603119245 [TCP segment of a reassembl
383	248.067341209	192.168.32.100	192.168.32.101	TCP	68 36612 → 8180 [ACK] Seq=162 Ack=4345 Win=62592 Len=0 TSval=2603119248 TSecr=146925
384	248.067568197	192.168.32.101	192.168.32.100	TCP	4562 8180 → 36612 [PSH, ACK] Seq=4345 Ack=162 Win=6912 Len=4494 TSval=146925 TSecr=2603119248 [TCP segment of a r
385	248.067574827	192.168.32.100	192.168.32.101	TCP	68 36612 → 8180 [ACK] Seq=162 Ack=8839 Win=62080 Len=0 TSval=2603119248 TSecr=146925
386	248.068001457	192.168.32.101	192.168.32.100	HTTP	68 HTTP/1.1 200 OK (text/html)
387	248.070918291	192.168.32.101	192.168.32.100	TCP	4412 8180 → 36616 [ACK] Seq=1 Ack=221 Win=6912 Len=4344 TSval=146925 TSecr=2603119245 [TCP segment of a reassembl
388	248.070950287	192.168.32.100	192.168.32.101	TCP	68 36616 → 8180 [ACK] Seq=221 Ack=4345 Win=62592 Len=0 TSval=2603119251 TSecr=146925
389	248.071190042	192.168.32.101	192.168.32.100	TCP	4562 8180 → 36616 [PSH, ACK] Seq=4345 Ack=221 Win=6912 Len=4494 TSval=146925 TSecr=2603119251 [TCP segment of a r
390	248.071196796	192.168.32.100	192.168.32.101	TCP	68 36616 → 8180 [ACK] Seq=221 Ack=8839 Win=62080 Len=0 TSval=2603119252 TSecr=146925
391	248.071594070	192.168.32.101	192.168.32.100	HTTP	68 HTTP/1.1 200 OK (text/html)
392	248.072264374	192.168.32.100	192.168.32.101	TCP	68 36612 → 8180 [FIN, ACK] Seq=162 Ack=8840 Win=64128 Len=0 TSval=2603119253 TSecr=146925
393	248.072421220	192.168.32.100	192.168.32.101	TCP	68 36616 → 8180 [FIN, ACK] Seq=221 Ack=8840 Win=64128 Len=0 TSval=2603119253 TSecr=146925
394	248.072541283	192.168.32.101	192.168.32.100	TCP	68 8180 → 36612 [ACK] Seq=8840 Ack=163 Win=6912 Len=0 TSval=146925 TSecr=2603119253
395	248.072650317	192.168.32.101	192.168.32.100	TCP	68 8180 → 36616 [ACK] Seq=8840 Ack=222 Win=6912 Len=0 TSval=146925 TSecr=2603119253
396	248.072777405	192.168.32.100	192.168.32.101	TCP	76 36628 → 8180 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2603119253 TSecr=0 WS=128
397	248.072983095	192.168.32.101	192.168.32.100	TCP	76 8180 → 36628 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM TSval=146925 TSecr=2603119253 WS=64

Il servizio attivo sulla porta 8180 risulta essere un altro:

```
┌──(riccbrun㉿kali)-[~]
└─$ nmap 192.168.32.101 -p 8180 -A
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-18 16:09 CEST
Nmap scan report for 192.168.32.101
Host is up (0.00053s latency).

PORT     STATE SERVICE VERSION
8180/tcp open  http    Apache Tomcat/Coyote JSP engine 1.1
|_http-title: Apache Tomcat/5.5
|_http-favicon: Apache Tomcat
|_http-server-header: Apache-Coyote/1.1

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 35.39 seconds
```