

NMAP

Su questo esercizio vedremo come utilizzare nmap su kali linux.
Per prima cosa, apriamo le macchine virtuali Kali e metasploitable.

Dopo questo, apriamo kali, e sul terminale lanciamo il comando di nmap, la funzione che vogliamo, se sS o sT, sapendo che il comando sS lancia solo una SYN, quindi questo comando risulta meno “aggressivo” quindi in poche parole andrà solo a controllare se la porta è aperta, e non altro. Invece il comando sT lancia uno scan TCP, questo è più aggressivo, risultando nel controllare tutte le porte aperte e tutte le sue vie.

Come in immagine, eseguiamo il comando di nmap -sS:

```
(kali@kali)-[~]
$ sudo nmap -sS 192.168.50.101 -p 1-1024
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-07 14:04 EST
Nmap scan report for 192.168.50.101
Host is up (0.00092s latency).
Not shown: 1012 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
MAC Address: 08:00:27:A8:A0:9A (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.86 seconds
```

Come in immagine, eseguiamo il comando di nmap -sT:

```
(kali@kali)-[~]
$ nmap -sT 192.168.50.101 -p 1-1024
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-07 13:59 EST
Stats: 0:00:10 elapsed; 0 hosts completed (0 up), 1 undergoing Ping Scan
Parallel DNS resolution of 1 host. Timing: About 0.00% done
Nmap scan report for 192.168.50.101
Host is up (0.0048s latency).
Not shown: 1012 closed tcp ports (conn-refused)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell

Nmap done: 1 IP address (1 host up) scanned in 13.98 seconds
```

No.	Time	Source	Destination	Protocol	Length	Info
221	89.841446645	192.168.50.100	192.168.50.101	TCP	74	42498 → 530 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=313...
222	89.841880772	192.168.50.100	192.168.50.101	TCP	74	41546 → 625 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=313...
223	89.842208124	192.168.50.101	192.168.50.100	TCP	60	337 → 53552 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
224	89.842208348	192.168.50.101	192.168.50.100	TCP	60	739 → 59884 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
225	89.842613538	192.168.50.101	192.168.50.100	TCP	60	285 → 54386 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
226	89.842613753	192.168.50.101	192.168.50.100	TCP	60	530 → 42498 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
227	89.842613802	192.168.50.101	192.168.50.100	TCP	60	625 → 41546 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
228	89.843063911	192.168.50.100	192.168.50.101	TCP	74	48576 → 447 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=313...
229	89.843486060	192.168.50.100	192.168.50.101	TCP	74	36532 → 418 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=313...
230	89.845808501	192.168.50.100	192.168.50.101	TCP	74	60250 → 547 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=313...
231	89.844355768	192.168.50.101	192.168.50.100	TCP	60	447 → 48576 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
232	89.844355985	192.168.50.101	192.168.50.100	TCP	60	418 → 36532 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
233	89.844483559	192.168.50.100	192.168.50.101	TCP	74	44954 → 452 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=313...
▶ Frame 2028: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, id 0						
▶ Ethernet II, Src: 08:00:27:a8:a0:9a (08:00:27:a8:a0:9a), Dst: 08:00:27:cb:7e:f5 (08:00:27:cb:7e:f5)						
▶ Internet Protocol Version 4, Src: 192.168.50.101, Dst: 192.168.50.100						
▶ Transmission Control Protocol, Src Port: 54, Dst Port: 50222, Seq: 1, Ack: 1, Len: 0						

eth0: <live capture in progress>

Packets: 2105 · Displayed: 2105 (100.0%)

Profile: Default

No.	Time	Source	Destination	Protocol	Length	Info	
628	13.231289218	192.168.50.101	192.168.50.100	TCP	60	266 → 48689 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	
629	13.231289257	192.168.50.101	192.168.50.100	TCP	60	277 → 48689 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	
630	13.231289294	192.168.50.101	192.168.50.100	TCP	60	572 → 48689 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	
631	13.23135559	192.168.50.101	192.168.50.101	TCP	58	48689 → 836 [SYN] Seq=0 Win=1024 Len=0 MSS=1460	
633	13.23161949	192.168.50.100	192.168.50.101	TCP	58	48689 → 966 [SYN] Seq=0 Win=1024 Len=0 MSS=1460	
633	13.23155066	192.168.50.100	192.168.50.101	TCP	58	48689 → 981 [SYN] Seq=0 Win=1024 Len=0 MSS=1460	
634	13.23165347	192.168.50.100	192.168.50.101	TCP	58	48689 → 986 [SYN] Seq=0 Win=1024 Len=0 MSS=1460	
635	13.23191544	192.168.50.101	192.168.50.100	TCP	60	997 → 48689 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	
636	13.23190265	192.168.50.100	192.168.50.101	TCP	58	48689 → 204 [SYN] Seq=0 Win=1024 Len=0 MSS=1460	
637	13.231915570	192.168.50.101	192.168.50.100	TCP	60	604 → 48689 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	
638	13.231915615	192.168.50.101	192.168.50.100	TCP	60	600 → 48689 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	
639	13.231915733	192.168.50.101	192.168.50.100	TCP	60	714 → 48689 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	
640	13.232126185	192.168.50.101	192.168.50.101	TCP	58	48689 → 937 [SYN] Seq=0 Win=1024 Len=0 MSS=1460	
641	13.232210985	192.168.50.100	192.168.50.101	TCP	58	48689 → 471 [SYN] Seq=0 Win=1024 Len=0 MSS=1460	
642	13.232290930	192.168.50.100	192.168.50.101	TCP	58	48689 → 393 [SYN] Seq=0 Win=1024 Len=0 MSS=1460	
▶ Frame 628: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, id 0 ▶ Ethernet II, Src: 08:00:27:a8:a0:9a (08:00:27:a8:a0:9a), Dst: 08:00:27:cb:7e:f5 (08:00:27:cb:7e:f5) ▶ Internet Protocol Version 4, Src: 192.168.50.101, Dst: 192.168.50.100 ▶ Transmission Control Protocol, Src Port: 266, Dst Port: 48689, Seq: 1, Ack: 1, Len: 0							0000 08 00 27 0010 00 28 00 0020 32 64 01 0030 00 00 80

Invece con la scansione sS noi inviamo solamente un SYN e la macchina bersaglio ci risponde soltanto con una RST