

Port Scanner e Verb Scanner

```
PORT SCANNER -
VERB SCANNER

—— PER QUESTO PROGRAMMA L'UNICO INDIRIZZO FUNZIONALE E' 192.168.50.101 ——

Inserisci l'indirizzo IP del sistema target: 192.168.50.101
Scansionamento indirizzo IP: 192.168.50.101

Inserisci la porta di inizio range: 1
Inserisci la porta di fine range: 300

Inizio scansione: 2024-02-15 06:36:53.293390
[+] Porta 21 aperta
[+] Porta 22 aperta
[+] Porta 23 aperta
[+] Porta 25 aperta
[+] Porta 53 aperta
[+] Porta 80 aperta
[+] Porta 111 aperta
[+] Porta 139 aperta

Fine scansione: 2024-02-15 06:36:54.099087

Inserisci la porta del sistema target: 80

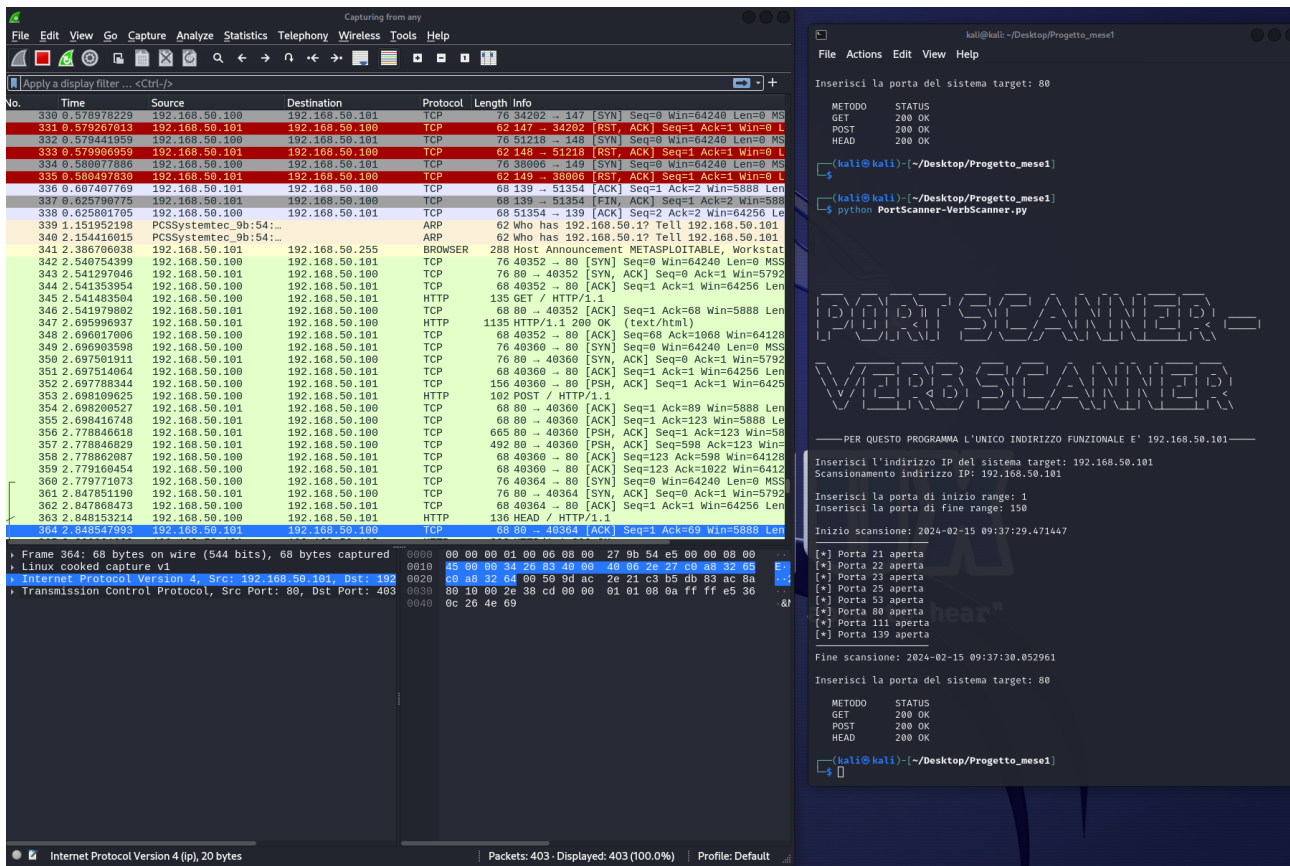
METODO    STATUS
GET        200 OK
POST       200 OK
HEAD       200 OK
```

(Schermata del codice in esecuzione)

Una funzione avanzata alla quale abbiamo pensato e sviluppato è stata semplicemente pensare alla user experience. Abbiamo lavorato seguendo questo modello per facilitarne l'utilizzo. I codici sviluppati singolarmente "Scanner delle porte" e "Enumerazione dei metodi HTTP" sono stati quindi uniti per migliorarne l'esperienza d'uso.

Il codice inizia chiedendo all'utente di inserire un indirizzo IP (che in questo specifico caso è solo quello specificato a inizio del programma, in quanto viene indicato per la macchina di Metasploitable) e quale range di porte vuole scansionare. Una volta trovate le porte aperte selezioniamo quella su cui vogliamo lavorare (che nel nostro caso specifico è la porta 80) per poi lanciare una serie di richieste di GET, POST e HEAD, facendoci visualizzare se la connessione è andata a buon fine e il relativo status. Questo poi ci permetterà di concentrarci sull'attacco Brute Force da eseguire nella fase successiva essendo già a

conoscenza dei metodi che possiamo utilizzare per comunicare con la porta aperta precedentemente rilevata.



(Nella parte sinistra della foto sopra Wireshark che ci conferma l'invio dei pacchetti lanciati dal programma che possiamo vedere in esecuzione nella parte destra della foto)

330	0.578978229	192.168.50.100	192.168.50.101	TCP	76 34202 → 147 [SYN] Seq=0 Win=64240 Len=0 MS
331	0.579267013	192.168.50.101	192.168.50.100	TCP	62 147 → 34202 [RST, ACK] Seq=1 Ack=1 Win=0 L
332	0.579441959	192.168.50.100	192.168.50.101	TCP	76 51218 → 148 [SYN] Seq=0 Win=64240 Len=0 MS
333	0.579906959	192.168.50.101	192.168.50.100	TCP	62 148 → 51218 [RST, ACK] Seq=1 Ack=1 Win=0 L
334	0.580077886	192.168.50.100	192.168.50.101	TCP	76 38006 → 149 [SYN] Seq=0 Win=64240 Len=0 MS
335	0.580497830	192.168.50.101	192.168.50.100	TCP	62 149 → 38006 [RST, ACK] Seq=1 Ack=1 Win=0 L
336	0.607407769	192.168.50.101	192.168.50.100	TCP	68 139 → 51354 [ACK] Seq=1 Ack=2 Win=5888 Len
337	0.625790775	192.168.50.101	192.168.50.100	TCP	68 139 → 51354 [FIN, ACK] Seq=1 Ack=2 Win=588
338	0.625801705	192.168.50.100	192.168.50.101	TCP	68 51354 → 139 [ACK] Seq=2 Ack=2 Win=64256 Le
339	1.151952198	PCSSystemtec_9b:54:...		ARP	62 Who has 192.168.50.1? Tell 192.168.50.101
340	2.154416015	PCSSystemtec_9b:54:...		ARP	62 Who has 192.168.50.1? Tell 192.168.50.101
341	2.386706038	192.168.50.101	192.168.50.255	BROWSER	288 Host Announcement METASPLOITABLE, Workstat
342	2.540754399	192.168.50.100	192.168.50.101	TCP	76 40352 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS
343	2.541297046	192.168.50.101	192.168.50.100	TCP	76 80 → 40352 [SYN, ACK] Seq=0 Ack=1 Win=5792
344	2.541353954	192.168.50.100	192.168.50.101	TCP	68 40352 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len
345	2.541483504	192.168.50.100	192.168.50.101	HTTP	135 GET / HTTP/1.1
346	2.541979802	192.168.50.101	192.168.50.100	TCP	68 80 → 40352 [ACK] Seq=1 Ack=68 Win=5888 Len
347	2.695996937	192.168.50.101	192.168.50.100	HTTP	1135 HTTP/1.1 200 OK (text/html)
348	2.696017006	192.168.50.100	192.168.50.101	TCP	68 40352 → 80 [ACK] Seq=68 Ack=1068 Win=64128
349	2.696903598	192.168.50.100	192.168.50.101	TCP	76 40360 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS
350	2.697501911	192.168.50.101	192.168.50.100	TCP	76 80 → 40360 [SYN, ACK] Seq=0 Ack=1 Win=5792
351	2.697514064	192.168.50.100	192.168.50.101	TCP	68 40360 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len
352	2.697788344	192.168.50.100	192.168.50.101	TCP	156 40360 → 80 [PSH, ACK] Seq=1 Ack=1 Win=6425
353	2.698109625	192.168.50.100	192.168.50.101	HTTP	102 POST / HTTP/1.1
354	2.698200527	192.168.50.101	192.168.50.100	TCP	68 80 → 40360 [ACK] Seq=1 Ack=89 Win=5888 Len
355	2.698416748	192.168.50.101	192.168.50.100	TCP	68 80 → 40360 [ACK] Seq=1 Ack=123 Win=5888 Le
356	2.778846618	192.168.50.101	192.168.50.100	TCP	665 80 → 40360 [PSH, ACK] Seq=1 Ack=123 Win=58
357	2.778846829	192.168.50.101	192.168.50.100	TCP	492 80 → 40360 [PSH, ACK] Seq=598 Ack=123 Win=
358	2.778862087	192.168.50.100	192.168.50.101	TCP	68 40360 → 80 [ACK] Seq=123 Ack=598 Win=64128
359	2.779160454	192.168.50.100	192.168.50.101	TCP	68 40360 → 80 [ACK] Seq=123 Ack=1022 Win=6412
360	2.779771073	192.168.50.100	192.168.50.101	TCP	76 40364 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS
361	2.847851190	192.168.50.101	192.168.50.100	TCP	76 80 → 40364 [SYN, ACK] Seq=0 Ack=1 Win=5792
362	2.847868473	192.168.50.100	192.168.50.101	TCP	68 40364 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len
363	2.848153214	192.168.50.100	192.168.50.101	HTTP	136 HEAD / HTTP/1.1
364	2.848547993	192.168.50.101	192.168.50.100	TCP	68 80 → 40364 [ACK] Seq=1 Ack=69 Win=5888 Len
Frame 364: 68 bytes on wire (544 bits), 68 bytes captured					0000 00 00 00 01 00 06 08 00 27 9b 54 e5 00 00 08 00
Linux cooked capture v1					0010 45 00 00 34 26 83 40 00 40 06 2e 27 c0 a8 32 65
Internet Protocol Version 4, Src: 192.168.50.101, Dst: 192					0020 c0 a8 32 64 00 50 9d ac 2e 21 c3 b5 db 83 ac 8a
Transmission Control Protocol, Src Port: 80, Dst Port: 403					0030 80 10 00 2e 38 cd 00 00 01 01 08 0a ff ff e5 36
					0040 0c 26 4e 69

(Nella foto sopra Wireshark che conferma l'invio dei pacchetti con relativo metodo GET, POST ed HEAD verso la porta 80)