# **Programming Assignment 5 - SEC 522: Cybersecurity Lab**

# **Sumaya Altamimi 442203026**

# Finding, Exploiting, and Fixing Vulnerabilities in Web Apps

# Introduction

In this lab, I am going to use Metasploitable Linux virtual machine. Which is a vulnerable for the purpose of practicing penetration testing techniques

## Instructions:

* You can download it through this link <https://sourceforge.net/projects/metasploitable/files/Metasploitable2/>.
* Use the default login and password msfadmin,msfadmin.
* Never expose this VM to an untrusted network (use NAT or Host-only mode if you have any questions what that means).

# Find the IP address of the VM

# Look for vulnerabilities in the web services

(Enter IP into web browser)

# Using static analysis tools to assist the search.

# Summary of the vulnerabilities

at least 15 different types of vulnerabilities

## Discovery

How you discovered the vulnerability (tools, code analysis).

## Attack vector

Description of the attack vector exploiting the vulnerability (you need to actually exploit the vulnerability).

## Mitigation

How you mitigated (fixed) the vulnerability (description / code).

## Screenshots

# Find an actual exploit for each vulnerability

You should find at least 15 (of the CWE/SANS Top 25) different types of vulnerabilities you discovered in the site (VM image).

# Fix the code

in order to prevent the vulnerability from reoccurring.

You should show how to fix any 10 of these.

Hint: The **Mutillidae** web service on this image (a link can be found when you enter the VM IP address in your browser) has each of the OWASP Top 10 vulnerabilities. You only need to find 5 others.

Other useful tools that will help you include **FireBug, Burp-Suite, Wireshark, etc**.

Use the hints in the application as a last resort (your analysis cannot be a regurgitation of the hints).

Also, be mindful of the Security setting in the application as it can make your job harder if set too high.

In addition, please avoid simply repeating the examples shown in class using Damn Vulnerable Web Application (DVWA).

# Information Gathering

## Metasploitable IP (Using Ifconfig)

172.16.235.2

# Open ports (Using nmap -sV 172.16.235.2)

**PORT STATE SERVICE VERSION**

21/tcp open ftp vsftpd 2.3.4

22/tcp open ssh OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)

23/tcp open telnet Linux telnetd

25/tcp open smtp Postfix smtpd

53/tcp open domain ISC BIND 9.4.2

80/tcp open http Apache httpd 2.2.8 ((Ubuntu) DAV/2)

111/tcp open rpcbind 2 (RPC #100000)

139/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)

445/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)

512/tcp open exec netkit-rsh rexecd

513/tcp open login

514/tcp open tcpwrapped

1099/tcp open java-rmi GNU Classpath grmiregistry

1524/tcp open bindshell Metasploitable root shell

2049/tcp open nfs 2-4 (RPC #100003)

2121/tcp open ftp ProFTPD 1.3.1

3306/tcp open mysql MySQL 5.0.51a-3ubuntu5

5432/tcp open postgresql PostgreSQL DB 8.3.0 - 8.3.7

5900/tcp open vnc VNC (protocol 3.3)

6000/tcp open X11 (access denied)

6667/tcp open irc UnrealIRCd

8009/tcp open ajp13 Apache Jserv (Protocol v1.3)

8180/tcp open http Apache Tomcat/Coyote JSP engine 1.1

# Using msfconsole

## Search

Now, will use msfconsole to search for **vsftpd**

Text

Description automatically generated

\*You have to make sure if it is the same version.

## Use

## 

Set our host IP and run “Show options” command.

Text

Description automatically generated

## Exploit

Text

Description automatically generated

# List files

A picture containing text

Description automatically generated

## This is root

Shape

Description automatically generated with medium confidence

# **Summary of the vulnerabilities**

# **Metasploitable Network**

## (Information Exposure/ Root access)

I found the ftp port open, with service called vsftpd 2.3.4. This version allowed me to gain root access of the machine. The reason is that it might contain backdoor by the attacker.

# Mutillidae

## 1.(Cross-site Scripting)

Go to the directory below:

Graphical user interface, application

Description automatically generated

Write javascript code to test the service, I wrote simple alert:

Graphical user interface, application

Description automatically generated

and it worked:

A picture containing shape

Description automatically generated

In order to make a real exploit will try to display cookie instead of random characters in the alert.

*<script>alert(document.cookie)</script>*

and It worked:

Text

Description automatically generated

Session IDs could be used to steal someone else session, by sending an email with a link

asking to login, that also activate the script when he arrived at it. The victim will click on a link and run the scrip on his computer, attacker can take the session id and can act on behalf of the victim.

## 2.Command Injection

Graphical user interface

Description automatically generated with low confidence

We can find all files listed

Text

Description automatically generated

## Fix:

Escaping charactres, whitelist acceptable values.

## 3.(Insecure Direct Object References)

Graphical user interface, text

Description automatically generated

Inspect the file menu:

Graphical user interface, text, application

Description automatically generated

Change the option value to “/etc/passwd”

Text

Description automatically generated

and it worked:

Graphical user interface, text, application, email

Description automatically generated

## Fix:

Require authentication for each object in the web server and check for privileges.

## 4.(Injection to extract user info)

Before you inject, you need to replace ‘metasploitable’ with ‘owasp10’ in the virtual machine /var/www/mutillidae/config.inc. Then go the User info tab.

Text

Description automatically generated

Write the following command:

Graphical user interface, text, application

Description automatically generated

And it worked:

Table

Description automatically generated with low confidence

## 5.(Injection to login)

Go to Login

A screenshot of a computer

Description automatically generated with medium confidence

Type the following command:

Graphical user interface, text, application

Description automatically generated

and It worked:

Graphical user interface, text

Description automatically generated

## 6.(Persistent XSS)

Go to Add to your blog

Graphical user interface, text, application

Description automatically generated

Table

Description automatically generated with low confidence

So, whenever you go to the blogs page, the alert will pop up as follow:

Shape

Description automatically generated with low confidence

## 7.(Broken Authentication and session management)

Install some cookie editor (browser plugin) and register to make new account.

Graphical user interface, text, application

Description automatically generated

Note that your name appeared at the corner:

Graphical user interface, text, application

Description automatically generated

Go to Broken authentication page and select login.

Graphical user interface, application

Description automatically generated

In the cookie editor, chane the uid to 1 (typically it is either 0 or 1)

Graphical user interface, text, application, email

Description automatically generated

Save the change by using the save icon.

Graphical user interface, application

Description automatically generated

Refresh the page, note the new name:

Graphical user interface, text, application

Description automatically generated

Now, you have all admin permissions.

## 8.(CSRF)

Go to add blog

Graphical user interface, application

Description automatically generated

Capture the traffic using Burp Suite and Add a blog:

Graphical user interface, text, application, email

Description automatically generated

Note the added traffic in Burp Suite:

Graphical user interface, text, application, email

Description automatically generated

Sign-in to a different user (or later send the url to another user). Change the blog content as follow:

csrf-token=SecurityIsDisabled&blog\_entry=Exploited!!!!&add-to-your-blog-php-submit-button=Save+Blog+Entry

Graphical user interface, text, application, email

Description automatically generated

By right click and send the request, the url request will be automatically generated by Burp Suite. Select current browser session.

Graphical user interface, application

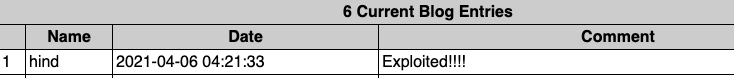
Description automatically generated

Stop capturing and open the copied url in your browser. This shows the second account that you signed in (the victim account):

Graphical user interface, text, application

Description automatically generated

This is the new blog added:



9.(Failure to restrict URL Access)

Go to Secret administrative pages:

Graphical user interface, text, application

Description automatically generated

10.(Unvalidated forward)

Go to credits

Graphical user interface, text, application

Description automatically generated

Start capturing Using Burp Suit. Click on Owasp link and go back to Burp Suit.

Change the forward value to another link of your choice. In my case it is:

<http://172.16.235.2/mutillidae/index.php?page=redirectandlog.php&forwardurl=http://www.google.org>

By right click and send the request, the url request will be automatically generated by Burp Suite. Select current browser session and copy the url.

Graphical user interface, application

Description automatically generated

Stop capturing and Sign-in to a different account to use that url. It will redirect you to the attacker url.

Graphical user interface, text, application

Description automatically generated

11.(Misconfigure)

Go to misconfigure

By manually guessing the php pages, I found that phpinfo.php gave overview of the configuration:

<http://172.16.235.2/mutillidae/phpinfo.php>

Text

Description automatically generated

# DVWA

## SQL injection

Text

Description automatically generated with medium confidence

Try to guess table names:

' or 1=1 UNION SELECT \* FROM TABLES#

Take advantages of error messages:

A picture containing text

Description automatically generated

When tried ' or 1=1 UNION SELECT \* FROM users# :



Which means table exist.

After several trials, this worked as follow:

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

Background pattern

Description automatically generated