Programming Assignment 5 - SEC 522: Cybersecurity Lab

**Sumaya Altamimi 442203026**

Finding, Exploiting, and Fixing Vulnerabilities

in Web Apps

Table of Contents

[Introduction 2](#_Toc68808806)

[Summary of the vulnerabilities 3](#_Toc68808807)

[Mutillidae 3](#_Toc68808808)

[1. Improper Neutralization of Input During Web Page Generation (Cross-site Scripting) 3](#_Toc68808809)

[2.(Persistent XSS) 4](#_Toc68808810)

[3.Cross-Site Request Forgery (CSRF) 6](#_Toc68808811)

[4. Improper Neutralization of Special Elements used in an OS Command (OS Command Injection) 8](#_Toc68808812)

[5. Information Exposure 9](#_Toc68808813)

[6. Improper Authentication 10](#_Toc68808814)

[7. Incorrect Permission Assignment for Critical Resource 11](#_Toc68808815)

[8. Improper Privilege Management 13](#_Toc68808816)

[9.Improper Input Validation 15](#_Toc68808817)

[DVWA 16](#_Toc68808818)

[10. Improper Neutralization of Special Elements used in an SQL Command (SQL Injection) 16](#_Toc68808819)

[11.Improper Neutralization of Special Elements used in an OS Command (OS Command Injection) 17](#_Toc68808820)

[12. Improper Limitation of a Pathname to a Restricted Directory (Path Traversal) 19](#_Toc68808821)

[13. Unrestricted Upload of File with Dangerous Type 20](#_Toc68808822)

[14. Improper Restriction of Excessive Authentication Attempts 23](#_Toc68808823)

[Metasploitable Network 27](#_Toc68808824)

[15. Improper Privilege Management 27](#_Toc68808825)

[References: 29](#_Toc68808826)

# 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 Metasploitable 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).
* My VM IP is 172.16. 235.2

|  |
| --- |
| You should find at least 15 (of the CWE/SANS Top 25) |
| How you discovered the vulnerability (tools, code analysis). |
| (you need to **actually** exploit the vulnerability). |
| How you mitigated (fixed) the vulnerability (description / code). how to fix any 10 of these. |

# **Summary of the vulnerabilities**

# Mutillidae

## 1. Improper Neutralization of Input During Web Page Generation (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.(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

**Fix**: The first solution to this is to escape the dynamic content which dimply means replacing any significant character with the HTML encoding, such as replacing ; with &#59. Also, having whitelist values will improve the security. For example, in adding a blog, there might be some predefined blogs for admin that he can select from instead of typing from scratch if applicable. One important concept is content security policy, this may help to control where javascript can be loaded from. In addition to that, implementing Http-only cookies is worth to consider.

## 3.Cross-Site Request Forgery (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

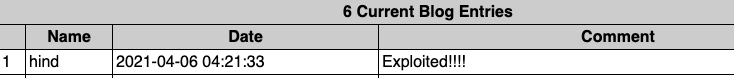
Description automatically generated

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:



**Fix**: Since CSRF depends on URL links, we can avoid malicious get requests by following REST design that force GET requests to be used only for (view) resources. Also, including secret tokens that called anti forgery token is important to prevent attack like this. SameSite Cookie Attribute would be additional layer of security that should be added to the set-cookie header. Finally, authentication for sensitive actions is required.

## 4. Improper Neutralization of Special Elements used in an OS Command (OS Command Injection)

Graphical user interface

Description automatically generated with low confidence

We can find all files listed

Text

Description automatically generated

## 5. Information Exposure

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

## 6. Improper Authentication

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

## 7. Incorrect Permission Assignment for Critical Resource

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.

## 8. Improper Privilege 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 privileges.

**Fix:** We have to make sure that cookies haven’t been tampered with. So digitally signing the data is important. So that in case of any modification, it will be detected after recalculating the signature. Also, not giving access to any resource before proper authentication.

## 9.Improper Input Validation

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

**Fix:** Disallowing any offsite redirects would be the first and the most important measure. Any site that is not in the list of acceptable site would be rejected after validation. Also, make sure the javascript is not vulnerable so it doesn’t take input from untrusted input.

# DVWA

## 10. Improper Neutralization of Special Elements used in an SQL Command (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

###### **Fix:** There are several ways to protect against this type of attack, one effective way is to escape the special characters properly. Also, sanitizing input and applying regular expressions in each input would definitely help. Another important protection method is the use of parametrized statements instead of string concatenation.

## 11.Improper Neutralization of Special Elements used in an OS Command (OS Command Injection)

Text

Description automatically generated

This indicates current path of DVWA is two folders after the root, to gain root access will do the following: 172.16.235.2 & dir ../../

Text

Description automatically generated

AS shown, there is a config file which seems interesting. We can use 172.16.235.2 & dir ../../config

Text

Description automatically generated

We can use 172.16.235.2 & type ../../config/config.inc.php to read the file but it is not readable. So will copy the content to another file using the command:

172.16.235.2 & cp ../../config/config.inc.php ../../mynewconfig

**Text

Description automatically generated**

Now we can access the file from the url:

[**http://172.16.235.2/dvwa/mynewconfig**](http://172.16.235.2/dvwa/mynewconfig)

**Graphical user interface, text, application

Description automatically generated**

We have all information about the database. We can connect to it remotely.

**Fix:** Restrict thePermitted Commands is one of the most effective solutions for this type of attacks. As a second line of defense, the principle of least privilege would help to limit the impact of command injection vulnerabilities as a second line of defense.

## 12. Improper Limitation of a Pathname to a Restricted Directory (Path Traversal)

Go to File inclusion

**Graphical user interface, text, application

Description automatically generated**

We can change the url from:

[**http://172.16.235.2/dvwa/vulnerabilities/fi/?page=include.php**](http://172.16.235.2/dvwa/vulnerabilities/fi/?page=include.php)

To:

[**http://172.16.235.2/dvwa/vulnerabilities/fi/?page=../../../../../etc/passwd**](http://172.16.235.2/dvwa/vulnerabilities/fi/?page=../../../../../etc/passwd)

And it worked:

**A screenshot of a computer

Description automatically generated**

In the same way, we can access other files. You can find the original directory to access to other files there and determine their path:

<http://172.16.235.2/dvwa/vulnerabilities/?page=index.php>. Php information file is another example:

**Application

Description automatically generated with medium confidence**

Fix: Run with restricted privileges. No process access files it doesn’t need. File names in the url should be carefully validated, especially those starting with ../ or ~/. Also, local and sensitive files shouldn’t be store in the same site public files.

## 13. Unrestricted Upload of File with Dangerous Type

Change the security level to medium. Go to file upload to upload jpg image.

**Graphical user interface

Description automatically generated with medium confidence**

This file can be accessed from the url:

**Graphical user interface, text, application

Description automatically generated**

Try ton upload non-jpg/jpeg file:

**Graphical user interface

Description automatically generated with medium confidence**

Because it only accepts jpeg images, as shown in the source code:

****

Now, will try to upload non-jpeg images. Upload a text file. Intercept the request using Burp Suite.

****

Change the content type to jpeg and forward the request. It will be uploaded and confirm that in the url.

****

And it worked. We could upload any script and any file type to harm the site.

**Graphical user interface

Description automatically generated with medium confidence**

To confirm, visit the url:

**Graphical user interface, text, application, email

Description automatically generated**

**Fix:** Additional and more sophisticated validation techniques is required. Avoid simply checking the file type and validate the extension of the file as well. Also, avoid executable files and scan files for viruses before upload. It is important to rename files before upload if necessary. Storing these file in a separate storage would be great enhancement of the security.

## 14. Improper Restriction of Excessive Authentication Attempts

Before performing Brute Force, try to login with fake credentials and intercept the request using Burp Suite.

**Graphical user interface, text, application, email

Description automatically generated**

Right click and send the request to intruder.

**Graphical user interface

Description automatically generated with low confidence**

In the position tab, select the type of the attack as Cluster bomb.

**Graphical user interface, text, application, email

Description automatically generated**

Click clear and add the username and password keywords:

**Graphical user interface, text, application

Description automatically generated**

At the payload tab, set the payload to 1 (username) and add common keywords.

**Graphical user interface, text, application, email

Description automatically generated**

Set the payload to 2 (password) and add common passwords.

**Graphical user interface, table

Description automatically generated**

At the options tab, set the keywords that will match the result in case of successful access:

**Graphical user interface, text, application

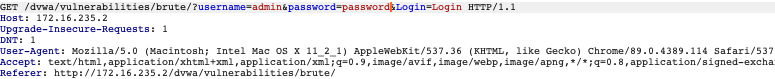
Description automatically generated**

In the same page click start attack. In my case, there are 36 probabilities and only one success:

**Table

Description automatically generated**

Now, confirm with sign-in to the server with the obtained passwords for admin:

****

After forwarding the request, it worked:

**Graphical user interface, text, application

Description automatically generated**

**Fix:** Limit the number of acceptable attempts. Adding another authentication factor to make sure the user is the one who claim to be.

# **Metasploitable Network**

## 15. Improper Privilege Management

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.

First, will use Nmap to discover open ports and services. One service I notices is vsftpd.

Open ports (Using nmap -sV 172.16.235.2)

PORT STATE SERVICE VERSION

21/tcp open ftp vsftpd 2.3.4

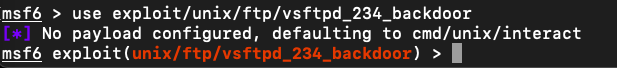
22/tcp open ssh OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)

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

Text

Description automatically generated

\*You have to make sure if it is the same version. Then use command:



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

And it worked, this is root:

Shape

Description automatically generated with medium confidence

**Fixes: Close unnecessary ports. Comprehensive testing to make sure that there are no backdoors. Authenticate before give access to any critical resource. Also, encrypt data and store them in a separate directory that is not in the client side. Authentication, authorization and permission checking are required.**

# References:

**[1]** [**https://owasp.org/www-project-top-ten/**](https://owasp.org/www-project-top-ten/)

**[2]** [**https://www.sans.org/top25-software-errors/**](https://www.sans.org/top25-software-errors/)

**[3]** [**https://www.hacksplaining.com/**](https://www.hacksplaining.com/)

**[4]**[**https://www.youtube.com/watch?v=2YD4vygeghM&list=LL&index=11&ab\_channel=HackerSploit**](https://www.youtube.com/watch?v=2YD4vygeghM&list=LL&index=11&ab_channel=HackerSploit)