UNIVERSITY OF JYVÄSKYLÄ

Course Title: Introductory Penetration Testing and Security Assessment
Course Code: KYBS2001EN24K2

PENETRATION TESTING REPORT

Damn Vulnerable Web Application

Submitted to:

Andrei Costin
Senior Lecturer
Faculty of Information Technology
University of Jyväskylä

Submitted by:

Md Musfiqur Rahman Milton
Student ID: 2668749196
Master's Student of Tampere University
(Major: Information Security)

Date of Submission: June 15, 2024

Table of Contents

1 Executive Summary

This report provides a comprehensive assessment of the security aspect of DVWA Website. The aim of this test was to identify vulnerabilities, assess risks, and provide recommendations to enhance the security of the website. The scope of the assessment is as follows:

Website	Hostname
Damn Vulnerable Web App	193.167.189.112:2907

As a result, several vulnerabilities have been found on the website, such as cross site scripting, command injection, file upload, file inclusion, SQL injection, brute force and nmap scanning. Cross-site scripting (XSS) is an attack in which an attacker injects malicious executable scripts into the code of a trusted application or website. Command injection happens when an attacker finds a way to insert harmful commands into a system through an application. In a file upload attack, the hacker uploads malicious files to a website. File inclusion occurs when an attacker tricks a website into including files it shouldn't. SQL injection is when a hacker inserts harmful code into a query that a website sends to its database. Brute force attacks are like trying every possible combination of a lock until it opens. Nmap scanning is a technique hackers use to explore a network and find open doors. Some of them were critical like command injection and SQL injection, on the other hand some were information like nmap scanning. A brief mitigation strategy of each vulnerability was given in chapter 3.

2 Vulnerability Overview

Table 2.1 depicts all vulnerabilities found during the penetration test. They are categorized by their risk and potential and are differentiated in the categories informational, high and critical. Critical vulnerabilities are the most severe and demand immediate action. These flaws can be easily exploited by attackers and often lead to catastrophic consequences. High vulnerabilities represent serious security flaws that could be exploited by attackers to gain significant unauthorized access or cause major disruption. Informational vulnerabilities are the least severe and typically do not pose an immediate threat. These are more about providing insights or hints to security analysts about potential security weaknesses.

Risk	Vulnerability	Section	Page
High	Cross-Site Scripting	3.2	3
Critical	Command injection	3.3	5
High	File Upload	3.4	7
High	File Inclusion	3.5	9
Critical	SQL Injection	3.6	11
High	Brute Force	3.7	13
Informational	Nmap Scanning	3.8	16

Table 3.1: Vulnerability Information of the target domain

3 Results

3.1 DVWA Domain

Damn Vulnerable Web Application (DVWA) is a web application written in PHP and MySQL that is intentionally designed to have many vulnerabilities. Its main objective is to help web developers better understand the processes involved in securing web applications, to help security professionals test their knowledge and tools in a regulatory environment, and to help teachers and students learn about web application security in a controlled atmosphere. The purpose of DVWA is to provide a platform for practicing prevalent web vulnerabilities, ranging in difficulty levels, through a user-friendly and straightforward interface. Some system information about this domain explained in Table 3.1.1.

Hostname	193.167.189.112:2907
Server Address	173.0.156.3
Server Software	Apache/2.4.7 (Ubuntu)
System Information	Linux 5a98eeff0549 5.4.0-174-generic #193-Ubuntu SMP Thu Mar 7 14:29:28 UTC 2024 x86_64

Table 3.1: System Information of the target domain

3.2 DVWA XSS

Cross-Site Scripting (XSS) attacks are a kind of injection in which malicious scripts are inserted into otherwise legitimate and trustworthy websites. XSS attacks occur when an attacker uses a web application to deliver malicious code, usually in the form of a browser-side script, to a separate end user. Flaws that allow these attacks to succeed are common and arise whenever a web application uses input from users in its output without verifying or encoding it. Reflected attacks occur when the injected script is reflected off the web server, such as in an error message, search result, or other response that contains some or all of the input provided to the server as part of the request. Reflected attacks reach victims via a different channel, such as an e-mail message or another website. When a user is misled into clicking on a malicious link, filling out a tailored form, or simply navigating to a malicious site, the injected code goes to the vulnerable website and reflects the attack back to the user's browser. Subsequently, the browser executes the code on account of its origin from a "trusted" server. More information about this issue is presented in Table 3.2.

Description	When a user is tricked into clicking on a malicious link, submitting a form, or browsing a malicious site, the injected code travels to the site, reflecting the attack back to the user's browser, which executes the code as it came from a trusted server. XSS (reflected) was possible in the targeted domain. As in the low security level there is no input sanitization, it was possible execute this attack.
CVSS Base Score	7.3 [NVD NIST] CVSS v3.1 **Vector:** AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L
Exploitability	Low

Business impact	Low
Reference to Classification	OWASP A7:2017
	WASC – 08: Cross-site Scripting (DOM based)
	CWE-79: Improper Neutralization of Input During Web Page Generation
Affected input	;script(alert]151917084[;>script(
Affected output	http://193.167.189.112:2907/vulnerabilities/xss_r/?name=%3Bscript%28alert%5D151917084%5B%3B%3Escript%28#

Table 3.2 DVWA XSS (Reflected)

3.2.1 Minimal proof of concept

Because the system had a defense mechanism in place, it replaced the provided special characters with a different one. The payload was formed in such a way that it produces the output of <script>alert(151917084)</script>. Figure 3.2 displays the output of the script.

Payload	;script(alert]151917084[;>script(
Given Symbol	**Replaced Symbol**
;	<
(	>
]	(
[	)
>	/

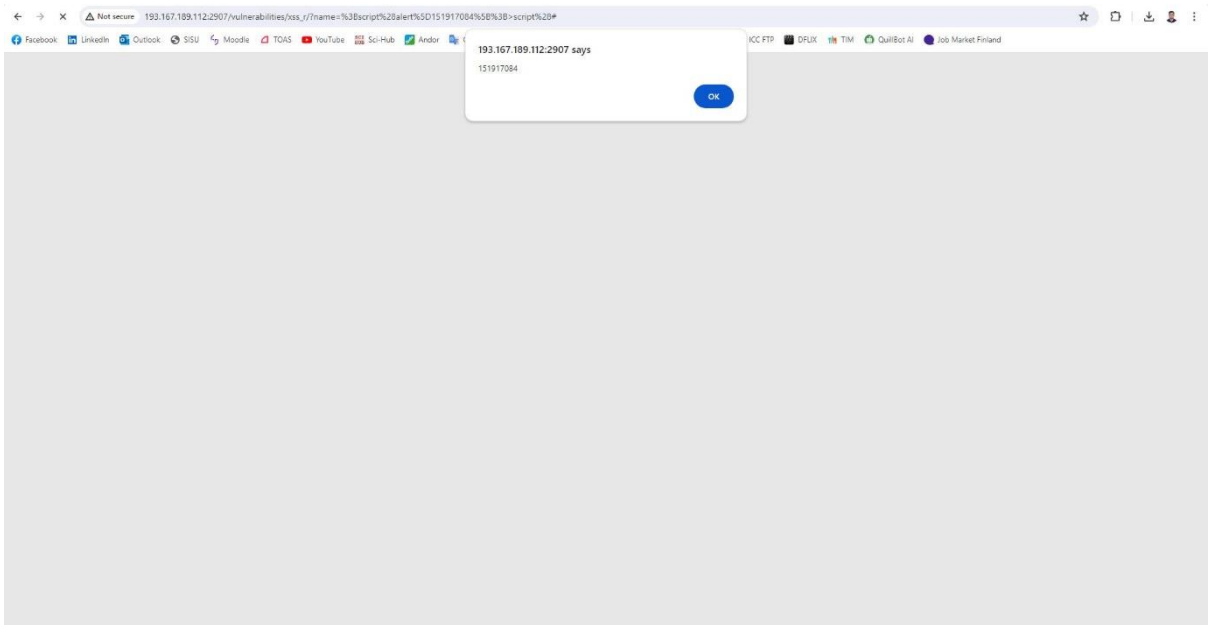

Figure 3.2: Output of XSS attack

3.2.2 Proposed solutions

Perform input validation and sanitization on input that originates from untrusted sources immediately upon reception. To ensure complete coverage, both inbound and outgoing input handling should be considered. Encode output as it escapes user input to make sure the browser perceives it as benign data rather than code. Use frameworks that automatically escape XSS by design, such as the latest Ruby on Rails, React JS. Enable a Content Security Policy (CSP) as a defense-in-depth mitigating control against XSS.

3.3 DVWA Command Execution (Injection)

OS command injection is also known as shell injection. It allows an attacker to execute operating system (OS) commands on the server that is running an application, and typically fully compromise the application and its data. Often, an attacker can leverage an OS command injection vulnerability to compromise other parts of the hosting infrastructure, and exploit trust relationships to pivot the attack to other systems within the organization. More information about this issue is presented in Table 3.3.

Description	If we check the source code, we can see that the code does not check if $target matches an IP Address. No filtering on special characters. **&& or ;** in Unix/Linux allows for commands to be separated.
CVSS Base Score	9.8 [NVD NIST] CVSS v3.1 **Vector:** AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H
Exploitability	Low
Business impact	Medium
Reference to Classification	OWASP A6:2017
	WASC – 31: Command Injection
	CWE-78: Improper Neutralization of Special Elements used in an OS Command
Affected input	127.0.0.1 && find / -iname 'secret' 127.0.0.1 && cat /etc/secret 127.0.0.1 && find / -iname 'runme_*.sh' 127.0.0.1 && cat /bin/runme_92275703.sh

Affected output	OWEzODRlZDgyYjI0NDg5Y2ViYWMxZmVmMTQwZmMzOGEyYzllODAwZA==
	
	#!/bin/bash
	
	if [[$# -eq 0]] ; then
	  echo 'No file given.'
	  exit 0
	fi
	
	["$1"] && SECRET="$1"
	base64 -d $SECRET

Table 3.3 DVWA Command Execution (Injection)

3.3.1 Minimal proof of concept

Figure 3.3.1: 127.0.0.1 && ls was used to check the contents of the current directory. Two files ‘secret’ and 'runme_*.sh' were required for this exercise and there weren’t present in the current directory.
Figure 2 & 3: 127.0.0.1 && find / -iname 'secret' was used to find secret file and was viewed using 127.0.0.1 && cat /etc/secret.
Figure 4 & 5: 127.0.0.1 && find / -iname 'runme_*.sh' was used to find secret file and was viewed using 127.0.0.1 && cat /bin/runme_92275703.sh. From this file it was perceived that base64 encoder is used to encode the secret. Then the secret was revealed using base64 decoder.

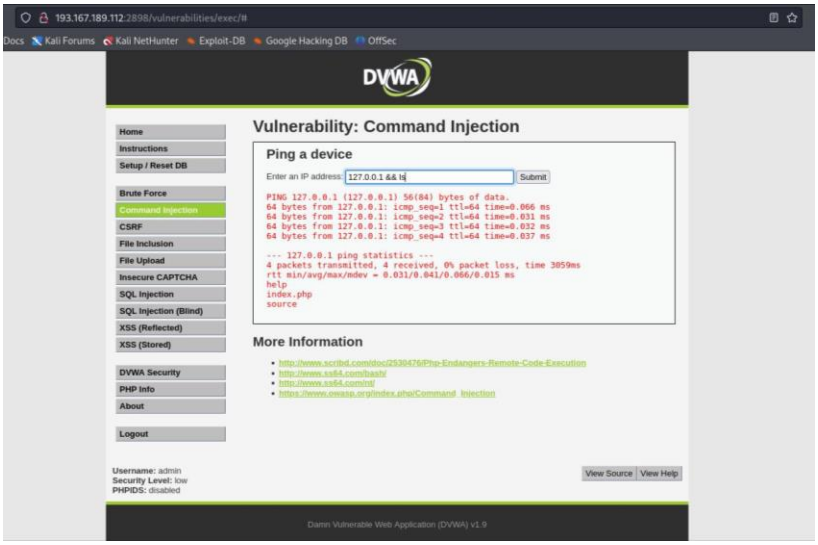

Figure 3.3.1: Content of current directory

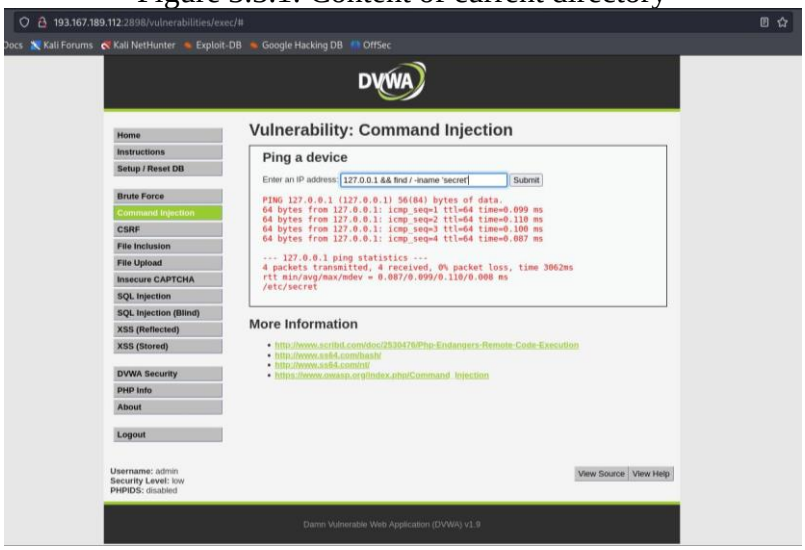

Figure 3.3.2: Location of the ‘secret’ file

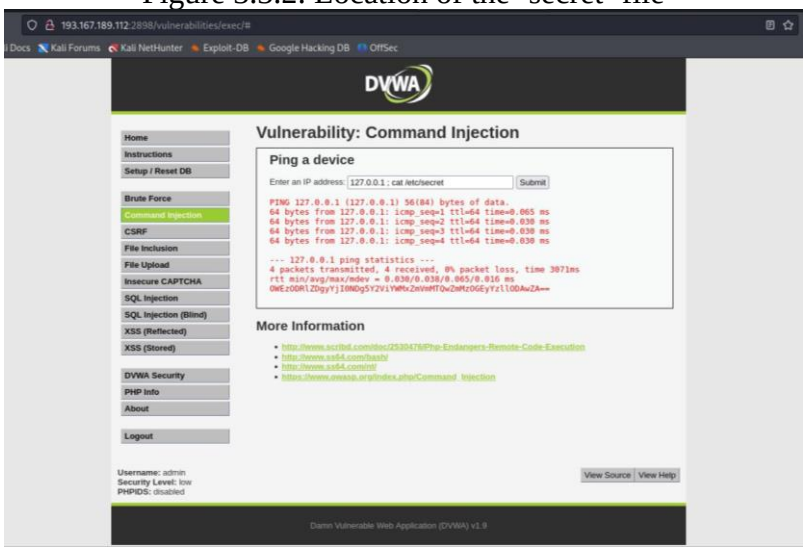

Figure 3.3.3: Content of the ‘secret’ file

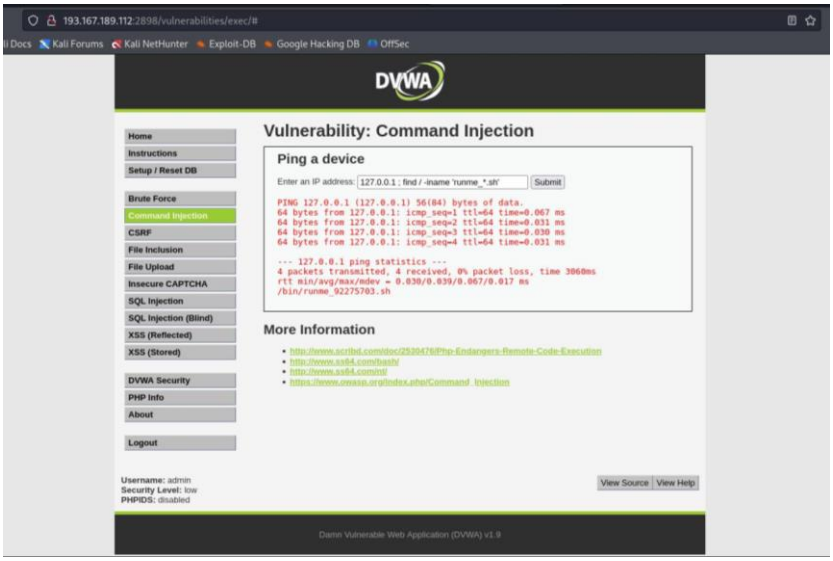

Figure 3.3.4: Location of the 'runme_*.sh' file

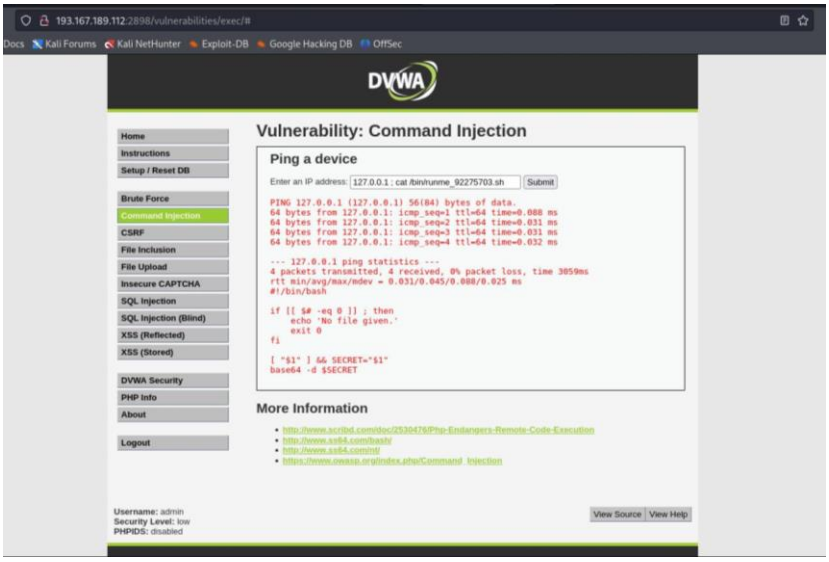

Figure 3.3.5: Content of the 'runme_*.sh' file

3.3.2 Proposed solutions

One of the most effective ways is to avoid calling all the system commands directly by taking from the user input. Using the built-in library functions is a good option, as it performs the tasks intended without tampering. From the source code it can be perceived, the "shell_exec()" function in PHP is being used for taking all our user inputs and passing them to the server. The escapeshellcmd() or escapeshellarg() can be used instead of "shell_exec()" to prevent command injection as it passes everything within a single quote ensuring to run all of one text rather than breaking it into multiple commands.

3.4 DVWA File Upload

File upload vulnerabilities are when a web server allows users to upload files to its filesystem without sufficiently validating things like their name, type, contents, or size. Failing to properly enforce restrictions on these could mean that even a basic image upload function can be used to upload arbitrary and potentially dangerous files instead. This could even include server-side

script files that enable remote code execution. In some cases, the act of uploading the file is enough to cause damage. Other attacks may involve a follow-up HTTP request for the file, typically to trigger its execution by the server. More information about this issue is presented in Table 3.4.

Description	From the source code of low level, it can be perceived that the code will not check the contents of the file being uploaded in any way. So, any valid PHP file with commands can be uploaded and executed.
CVSS Base Score	7.1 [NVD NIST] CVSS v3.1 **Vector:** AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:L/A:N
Exploitability	Low
Business impact	Low
Reference to Classification	WASC-33: Path Traversal
	CWE-434: Unrestricted Upload of File with Dangerous Type
Affected input	<?php readfile ("../Upload_1261.php"); ?>
Affected output	2accc07bc6c4f0480c3415a93b0e6efeab37dfd26f497b4e6ec8181fd0424fd2

Table 3.4 DVWA File Upload

3.4.1 Minimal proof of concept

Step 1: A JPG file was uploaded to learn about the file directory of the website.
Step 2: Once have knowledge about file directory, a PHP file with the command readfile was uploaded.
Step 3: The uploaded file was then executed to get the flag from the elements tab.

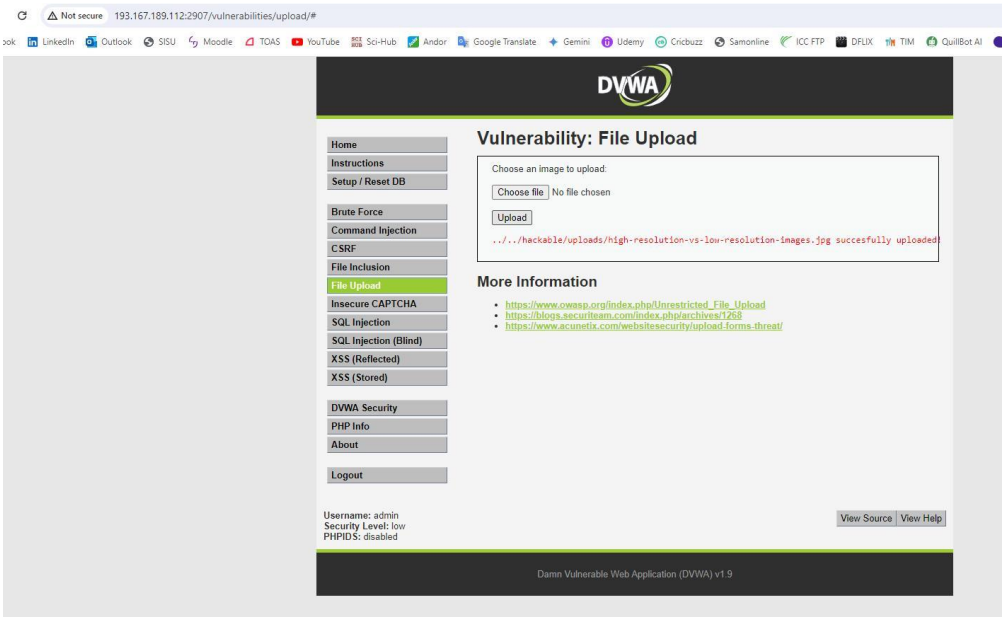

Figure 3.4.1: A JPG file was uploaded

Figure 3.4.2: Navigating through different directories the exercise

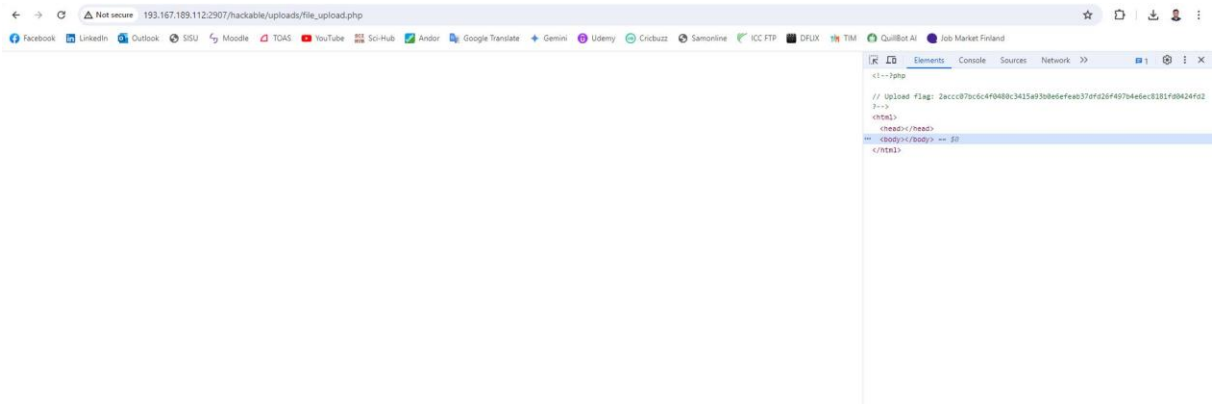

Figure 3.4.3: Capture the flag by executing PHP file.

3.4.2 Proposed solutions

The file types allowed to be uploaded should be restricted to only those that are necessary for business functionality. The application should perform filtering and content checking on any files which are uploaded to the server. Files should be thoroughly scanned and validated before being made available to other users. Mark all files as downloadable not executable.

3.5 DVWA File Inclusion

It is an attack that allows an attacker to include a file on the web server through a PHP script. This vulnerability arises when a web application lets the client submit input into files or upload files to the server. A file that includes vulnerability is distinct from a generic Directory Traversal Attack, in that directory traversal is a way of gaining unauthorized file system access, and a file inclusion vulnerability subverts how an application loads code for execution. Successful exploitation of a file that includes vulnerability will result in remote code execution on the web server that runs the affected web application. Local File Inclusion vulnerabilities allow an attacker to read and sometimes execute files on the victim machine. This can be very dangerous because if the web server is misconfigured and running with high privileges, the attacker may gain access to sensitive information. More information about this issue is presented in Table 3.5.

Description	From the source code of low level, it can be perceived that the code allows for direct input into one of many PHP functions that will include the content when executing.
CVSS Base Score	8.6 [NVD NIST] CVSS v3.1 **Vector:** AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:N/A:N
Exploitability	Low
Business impact	Medium
Reference to Classification	OWASP-A06:2021 - Vulnerable and Outdated Components
	CWE-98: Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion')
	WASC-05 - Remote File Inclusion
Affected input	http://193.167.189.112:2907/vulnerabilities/fi/?page=../../hackable/flags/46458.php
Affected output	ee81e15996311d6bc794622f6a7ef6048ee5fb8899faba29d0ac7a8067770d87

Table 3.5 DVWA File Inclusion

3.5.1 Minimal proof of concept

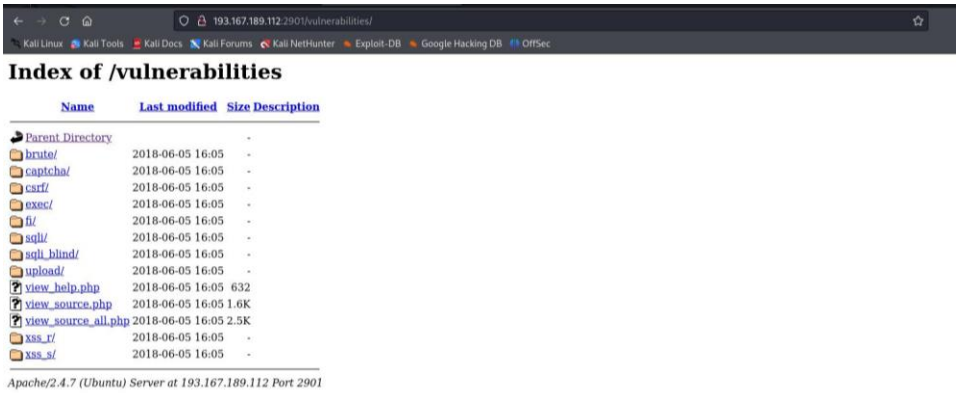

Figure 3.5.1: Navigating through different directories the exercise

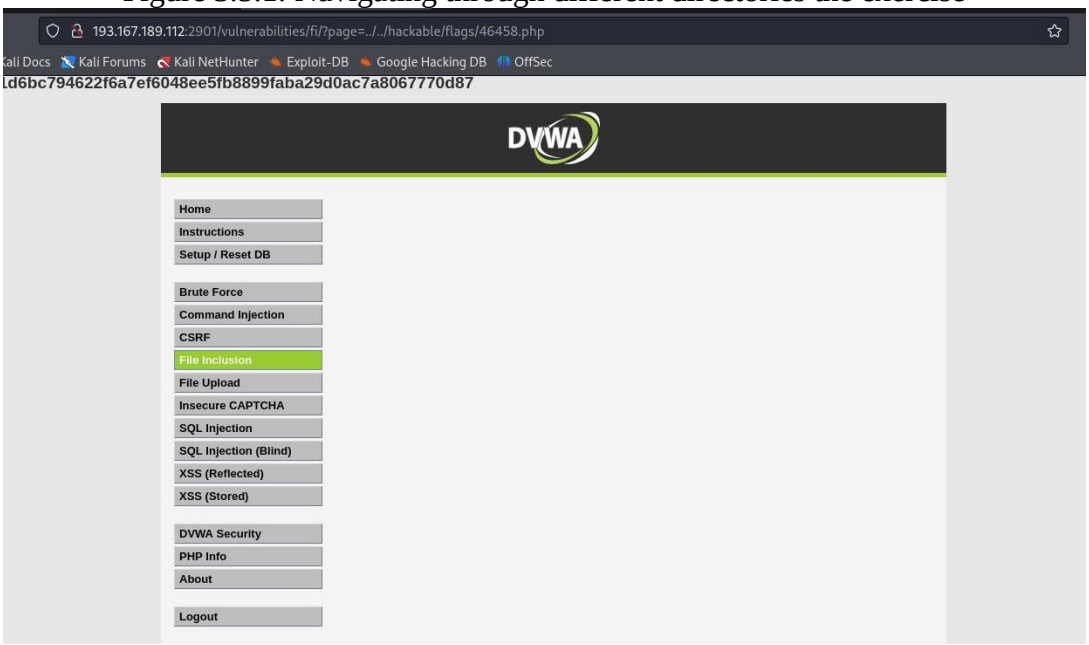

Figure 3.5.2: Capture the flag by executing PHP file.

Step 1: A JPG file was uploaded to learn about the file directory of the website.
Step 2: Once have knowledge about file directory, navigate to flag directory and check the given PHP file.
Step 3: The file was then executed as the following form:
?page=../../hackable/flags/46458.php

3.5.2 Proposed solutions

The most effective solution to eliminate file inclusion vulnerabilities is to avoid passing user-submitted input to any filesystem/framework API. If this is not possible the application can maintain an allow list of files, that may be included by the page, and then use an identifier to access to the selected file. Any request containing an invalid identifier must be rejected, in this way there is no attack surface for malicious users to manipulate the path.

3.6 DVWA SQL Injection without randomization

SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. This can allow an attacker to view data that they are not normally able to retrieve. This might include data that belongs to other users, or any other data that the application can access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior. In some situations, an attacker can escalate a SQL injection attack to compromise the underlying server or other back-end infrastructure. It can also enable them to perform denial-of-service attacks. More information about this issue is presented in Table 3.6.

Description	From the source code of low level, it can be perceived that, the variable $id is retrieved from the user input without any validation or sanitization. It is then directly concatenated into the SQL query string. This allows an attacker to manipulate the value of $id and inject malicious SQL code.
CVSS Base Score	9.4 [NVD NIST] CVSS v3.1 **Vector:** AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L
Exploitability	Low
Business impact	Medium
Reference to Classification	OWASP-A03:2021- Injection
	WASC - 19: SQL Injection
	CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
Affected input	' Union Select email, NULL FROM dvwa.VIEWS # ' UNION SELECT table_name, NULL FROM information_schema.tables # # ' Union Select email, NULL FROM dvwa.vips #
Affected output	Table 'dvwa.VIEWS' doesn't exist ID: ' Union Select email, NULL FROM dvwa.vips # First name: 4samma.fl@custmon.net Surname:

Table 3.6 DVWA SQL Injection

3.6.1 Minimal proof of concept

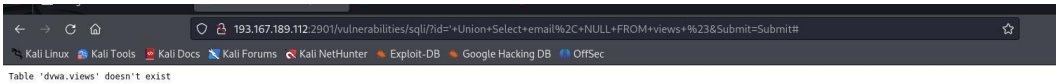

Figure 3.6.1: Executing a query to get the information about database

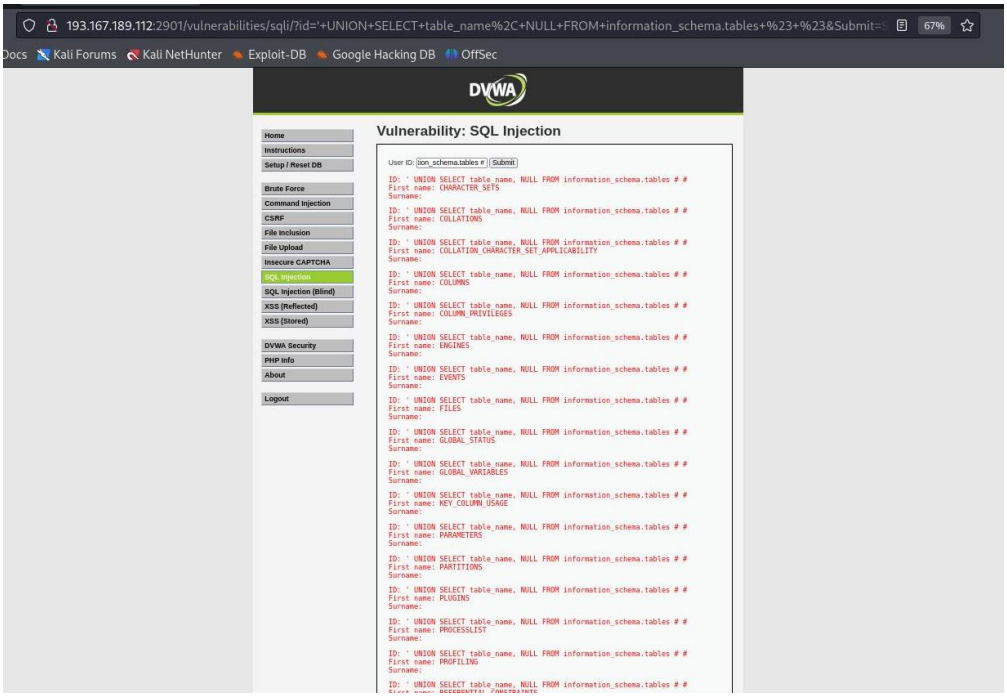

Figure 3.6.2: Executing a query to get the information about tables

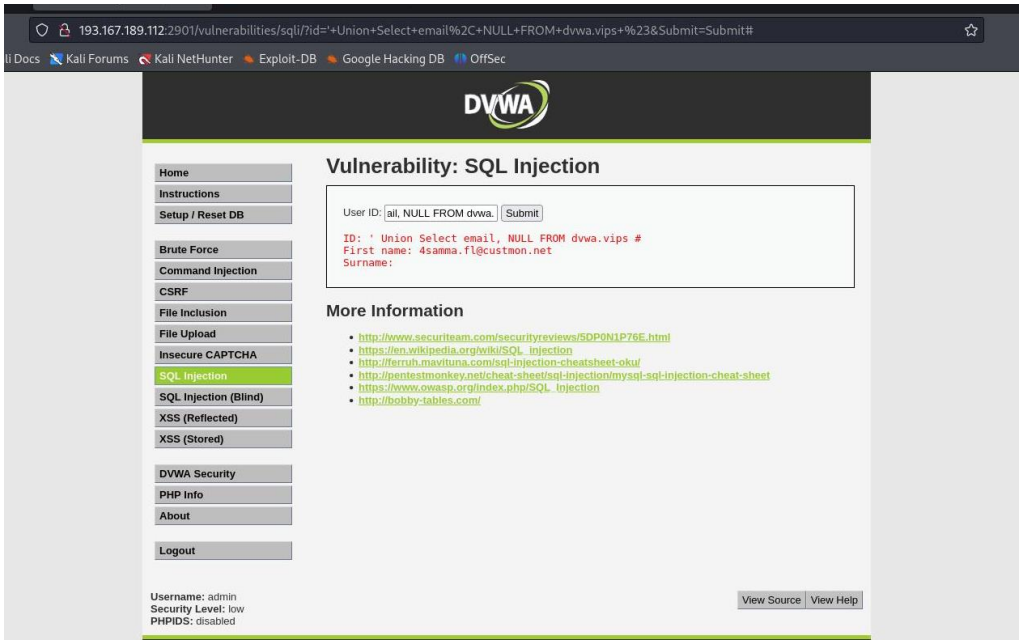

Figure 3.6.3: Executing the query to get the information from retrieved table and database

Step 1: Form a query *' Union Select email, NULL FROM dvwa.VIEWS #* to know about the database. By executing this query, the database name was identified which is dvwa.
Step 2: Then the second query *' UNION SELECT table_name, NULL FROM information_schema.tables # #* was executed to get table information which is vips.
Step 3: In this step, using information the above step, *' Union Select email, NULL FROM dvwa.vips #* required information was retrieved.

3.6.2 Proposed solutions

Most instances of SQL injection can be prevented using parameterized queries instead of string concatenation within the query. These parameterized queries are also known as "prepared statements". Though stored procedures are not always safe from SQL injection, developers can use certain standard stored procedure programming constructs. This approach has the same effect as the use of parameterized queries if the stored procedures are implemented safely. Another approach is escaping all user input before putting it in a query. A naive counter measure for low level would be to look for special keywords such as ;, #, -- or ' in the user input.

3.7 DVWA Brute-Force Login

A brute-force attack is an attempt to discover a password by systematically trying every possible combination of letters, numbers, and symbols until you discover the one correct combination that works. If your web site requires user authentication, you are a good target for a brute-force attack. An attacker can always discover a password through a brute-force attack, but the downside is that it could take years to find it. Depending on the password's length and complexity, there could be trillions of possible combinations. To speed things up a bit, a brute-force attack could start with dictionary words or slightly modified dictionary words because most people will use those rather than a completely random password. These attacks are called dictionary attacks or hybrid brute-force attacks. Brute-force attacks put user accounts at risk and flood your site with unnecessary traffic. Hackers launch brute-force attacks using widely available tools that utilize wordlists and smart rulesets to intelligently and automatically guess user passwords. Although such attacks are easy to detect, they are not so easy to prevent. More information about this issue is presented in Table 3.7.

Description	From the source code of low level, it can be perceived that, the developer has completely missed out any protection methods, allowing for anyone to try as many times as they wish, to login to any user without any repercussions.
CVSS Base Score	7.8 [NVD NIST] CVSS v3.1 **Vector:** AV:L/AC:H/PR:L/UI:N/S:C/C:H/I:H/A:H
Exploitability	Low
Business impact	Medium
Reference to Classification	OWASP-A02:2017 - Broken Authentication, A07:2021 - Identification and Authentication Failures
	WASC - 11: Brute Force
	CWE-307: Improper Restriction of Excessive Authentication Attempts
Affected input	sudo ifconfig

	sudo nmap -F 173.0.156.0/29 hydra 173.0.156.4 -F -V -L /usr/share/usernames -P /usr/share/common.txt http-get-form "/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:F=Username and/or password incorrect.:H=Cookie: PHPSESSID=692ne26g1lge594nodj76c5792; security=low"
Affected output	Host: 173.0.156.4 Login: anna Password: unreal

Table 3.3.1 DVWA Brute Force Login

3.7.1 Minimal proof of concept

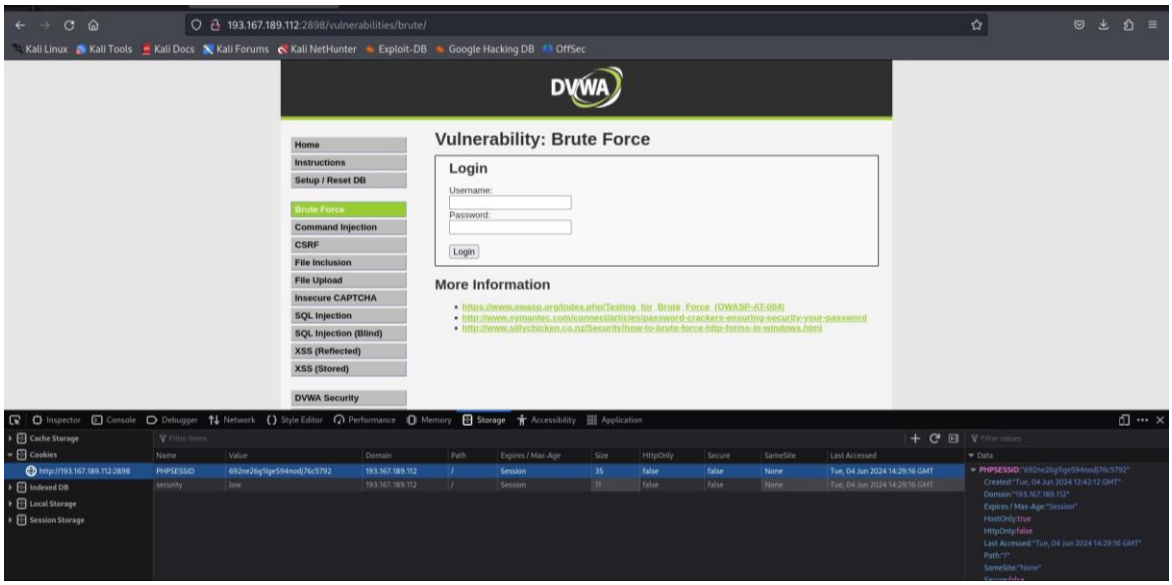

Figure 3.7.1: Capture the cookie from developer console tab

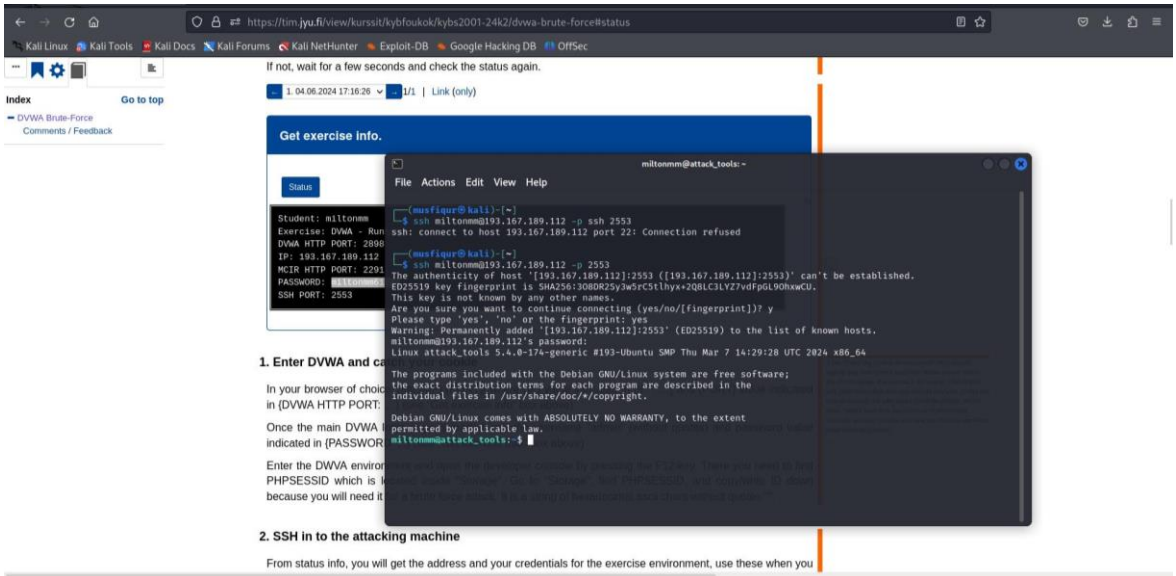

Figure 3.7.2: SSH into the attacking machine

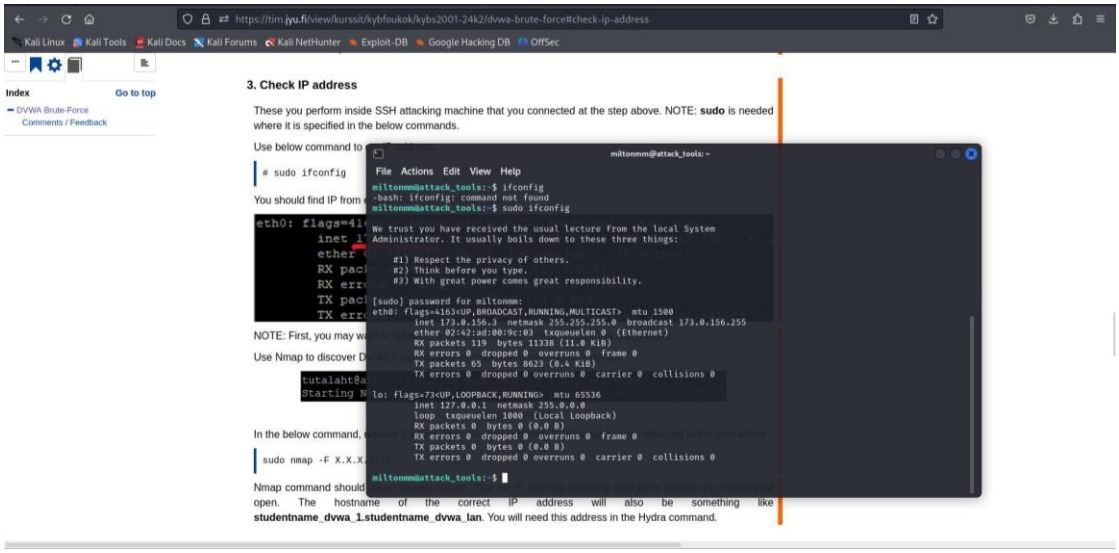

Figure 3.7.3: Check the IP address of the system

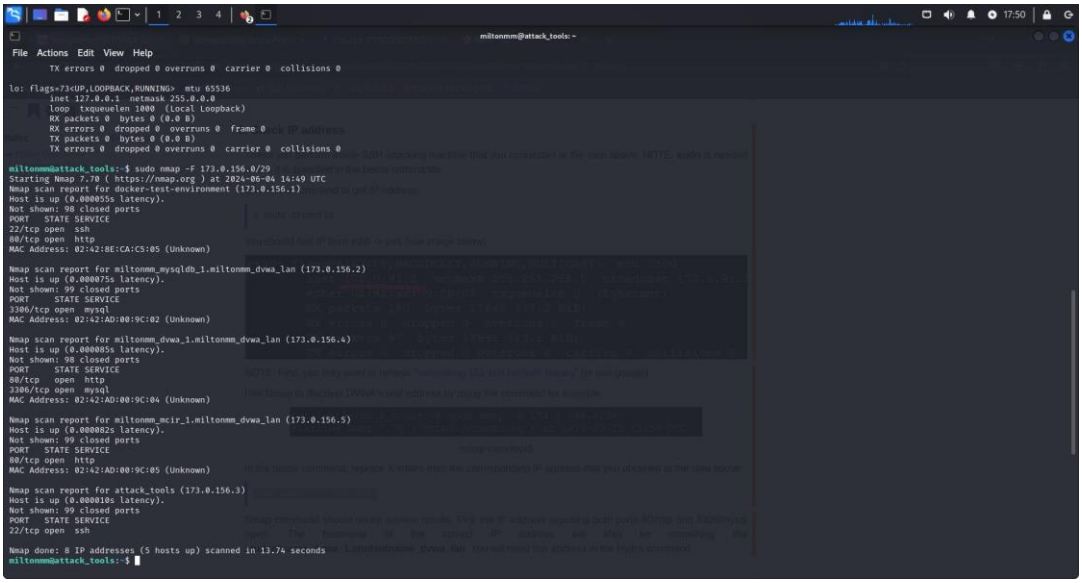

Figure 3.7.4: Using Nmap to scan the sub-network

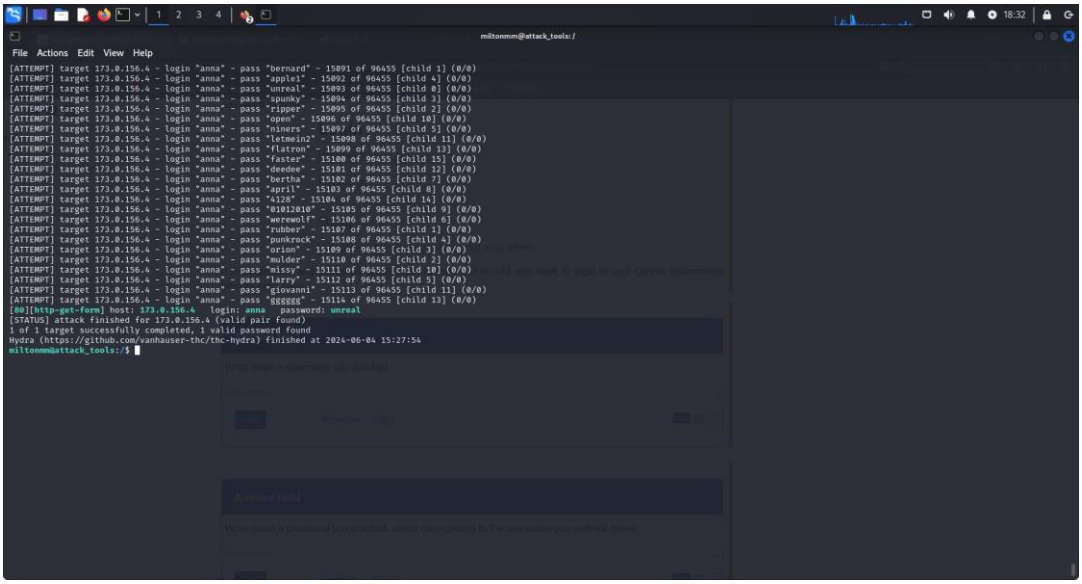

Figure 3.7.5: Using hydra to obtain the credentials

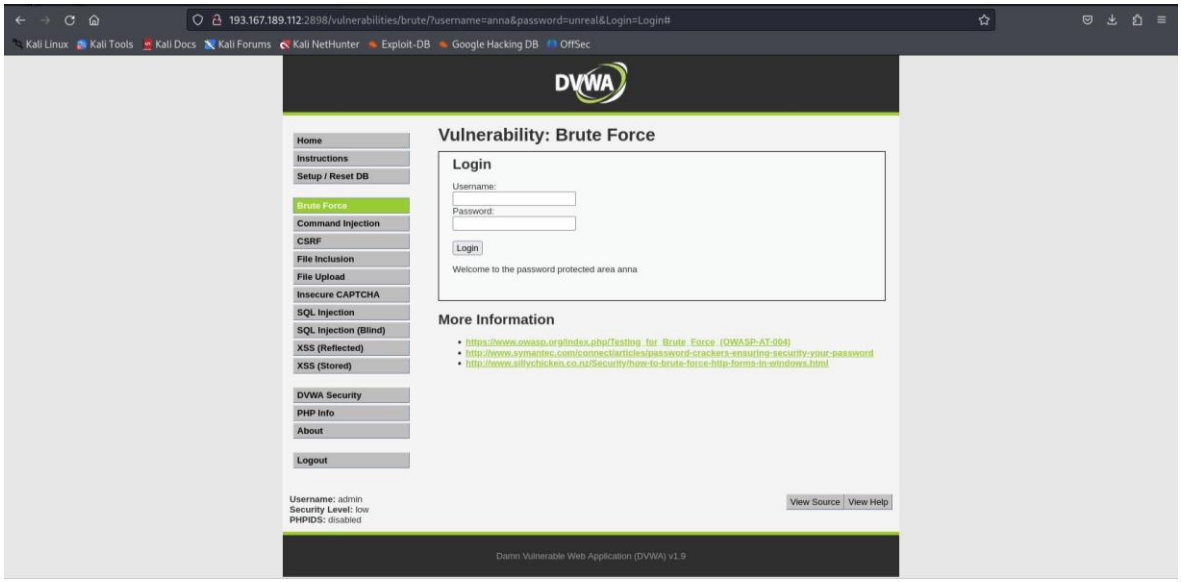

Figure 3.7.6: Using obtained credentials to access the system

Step 1: Obtained the cookie from the developer console tab. It will be used later to run the attack.
Step 2: SSHed into target machine and checked its IP address. It will be used later to run the nmap scan. In the nmap scan, network address of the subnet is used.
Step 3: Form and run hydra attack to IP address that has both http and mysql ports opened.
Step 4: Once credentials is obtained, it is then used to access the system.

3.7.2 Proposed solutions

The most obvious way to block brute-force attacks is to simply lock out accounts after a defined number of incorrect password attempts. Account lockouts can last a specific duration, such as one hour, or the accounts could remain locked until manually unlocked by an administrator. However, account lockout is not always the best solution, because someone could easily abuse the security measure and lock out hundreds of user accounts. Assign unique login URLs to blocks of users so that not all users can access the site from the same URL. Use a CAPTCHA to prevent automated attacks.

3.8 NMAP scan

Nmap ("Network Mapper") is a free and open-source utility for network exploration and security auditing. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services those hosts are offering, what operating systems they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. It was designed to rapidly scan large networks but works fine against single hosts. More information about this issue is presented in Table 3.8.

Description	Nmap offers various scanning techniques. A TCP SYN scan is a stealth scan used to determine if ports on a target system are open, closed or filtered. A version detection scan gathers details about the services and applications running on identified open ports. An OS detection scan discovers which OS a target network or computer is

	running. A vulnerability scan detects Common Vulnerabilities and Exposures (CVEs) to discover if a target is vulnerable to attacks.
CVSS Base Score	Informational
Exploitability	Informational
Business impact	Informational
Reference to Classification	CWE-319: Cleartext Transmission of Sensitive Information
Affected input	ssh miltonmm@193.167.189.112 -p 2553 sudo nmap -F 173.0.156.0/29 nmap -T5 -sV 173.0.156.2 -p 1025-65535
Affected output	STATE SERVICE 80/tcp open http 111/tcp open rpcbind 5900/tcp open vnc 6000/tcp open X11 Target Port: 28000

Table 3.3.1 DVWA Command Execution (Injection)

3.8.1 Minimal proof of concept

Step 1: SSHed into target machine and checked its IP address. It will be used later to run the nmap scan. In the nmap scan, network address of the subnet is used.
Step 2: Once the result is obtained, identify the target by checking the number of open TCP port.
Step 3: Using the formed command, scanned the target and found the required port.

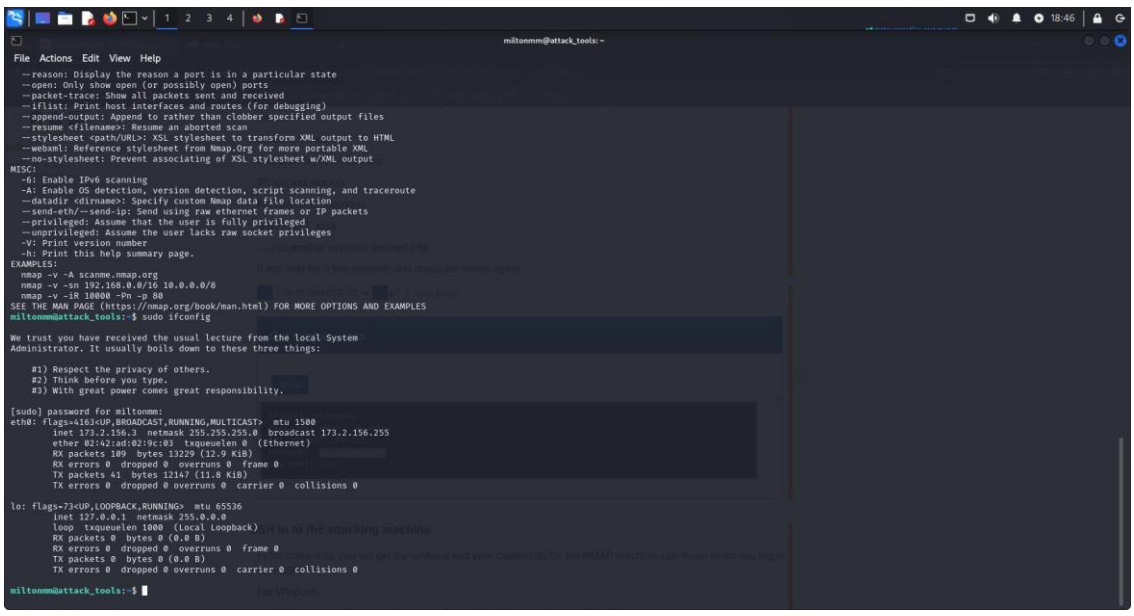

Figure 3.8.1: Check the IP address of the system

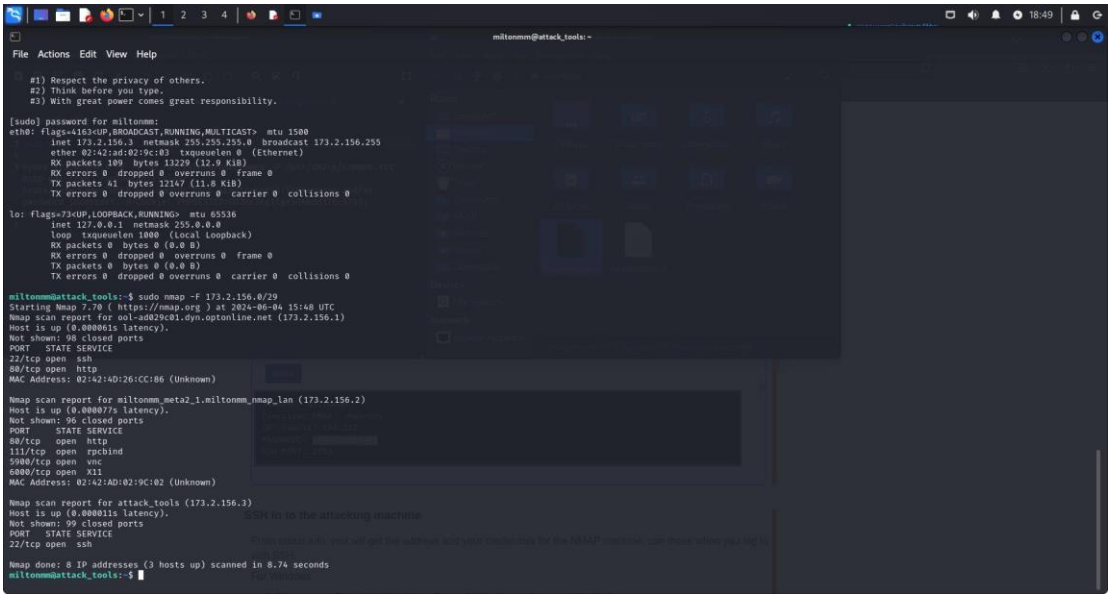

Figure 3.8.2: Using Nmap to scan the sub-network

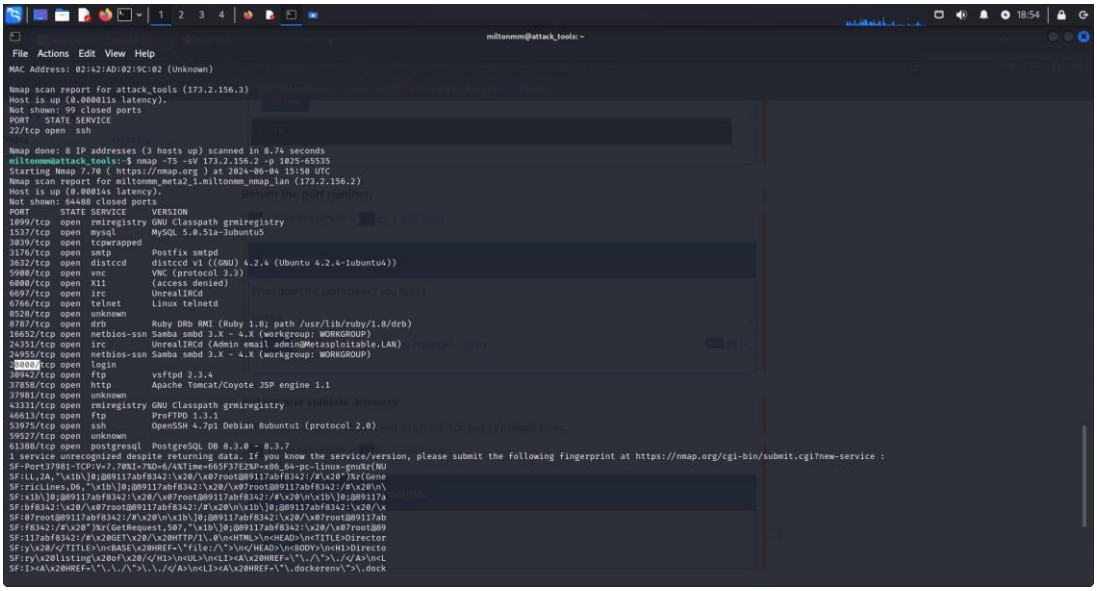

Figure 3.8.3: Using Nmap command to scan the target

3.8.2 Proposed solutions

Possible defences include blocking the probes, restricting information returned, slowing down the Nmap scan, and returning misleading information. Firewalls are the most effective way to block Nmap scans and other unauthorized access attempts. Most operating systems and routers come with built-in firewalls.