Certified Ethical Hacking With Penetration Testing
CEHWPT

LABS Course

CEHWPT LAB7 Web application Penetration Testing

Prepared by Eng. Khaled Gamo

23-6-2019

Web application Penetration Testing LAB

In this LAB we will attacking vulnerable web application system this system is called DVWA.

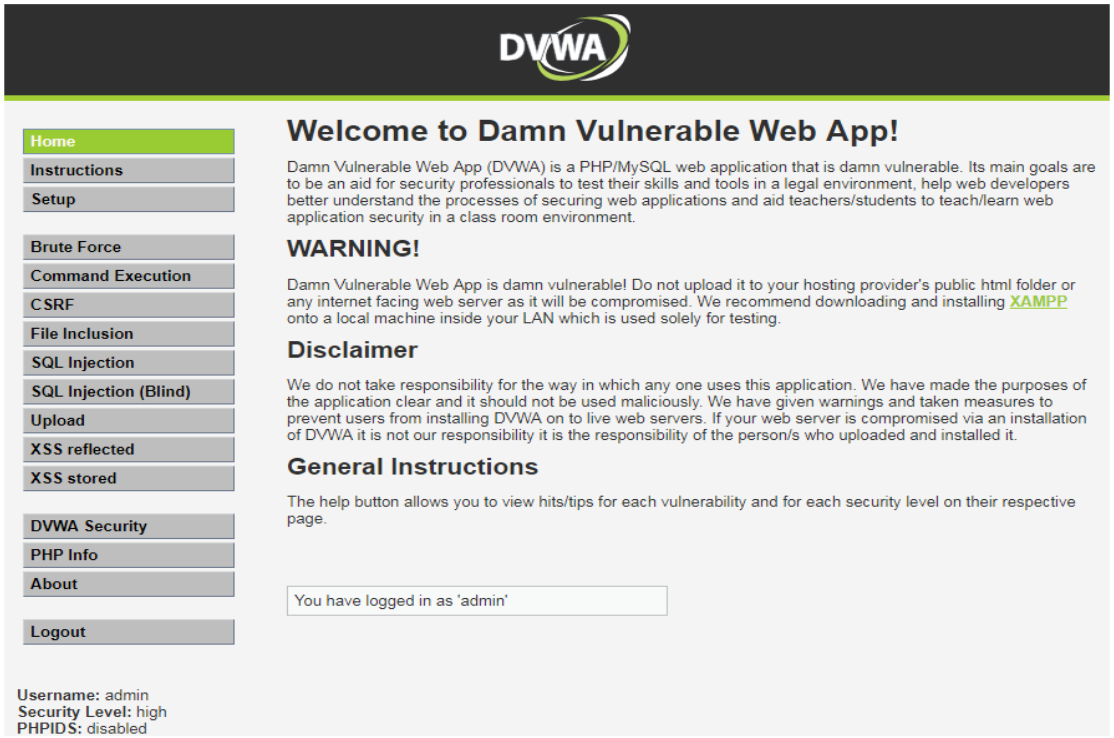
Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

Part1 setup the LAB

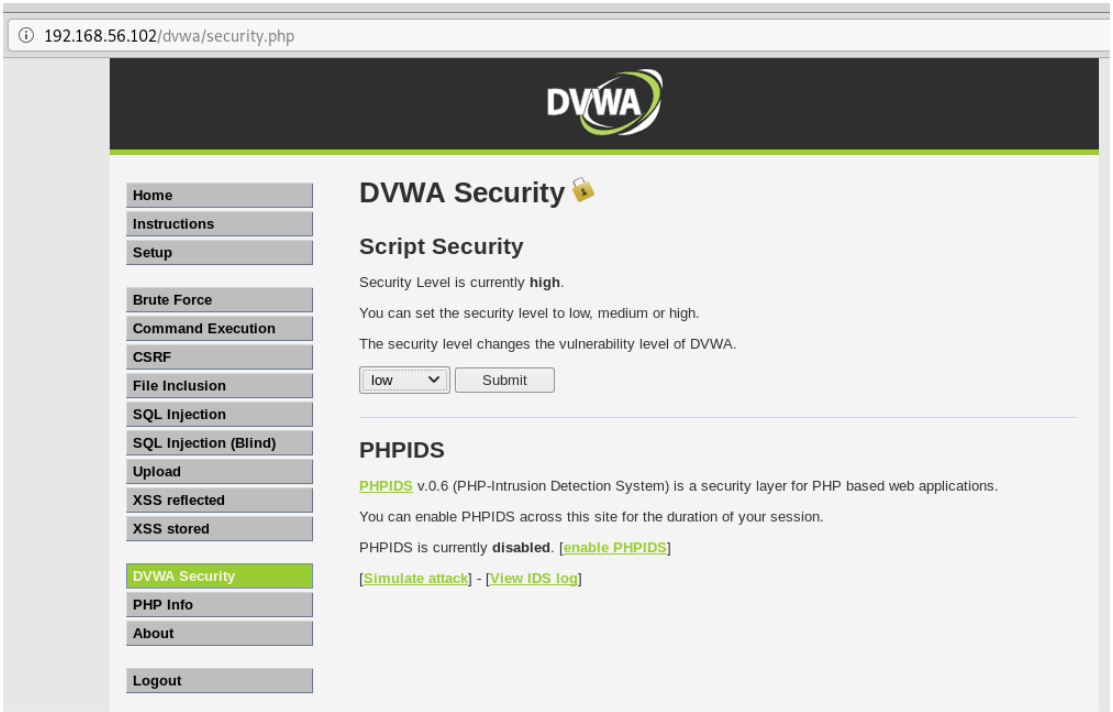
In order to complete the exercises we need a kali Linux & DVWA which already included in Metasploitatable VM.

Power up your Kali & Metasploitable , they should be configured as Viral box host only adapter, first we get the IP address of the our target 192.168.56.102 we will check DVWA service by open http://192.168.56.102

Logging to DVWA using username admin password is password we will get the following page.



We will start with DVWA security level low

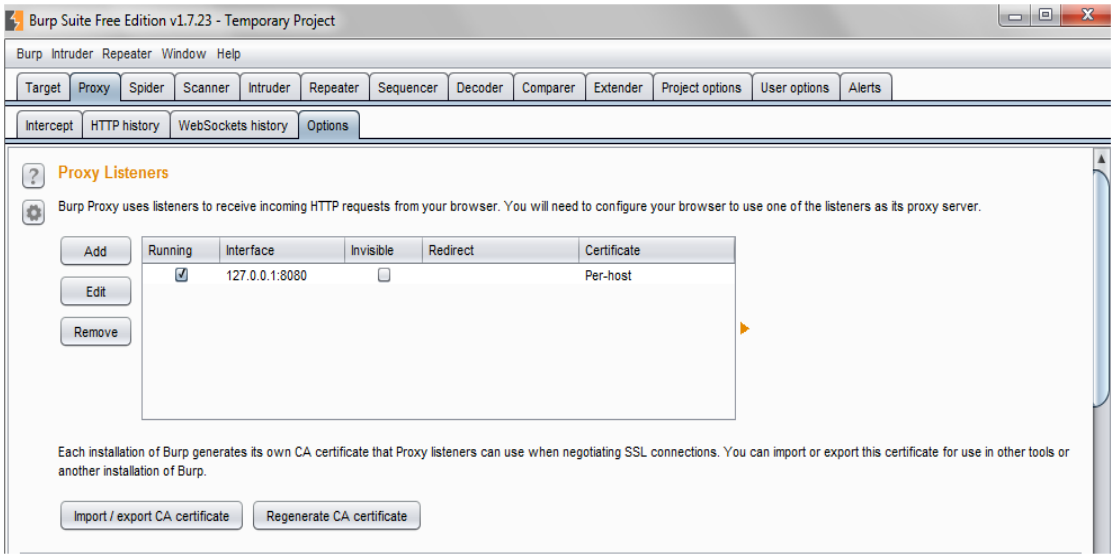


Attack 1 Brute Force

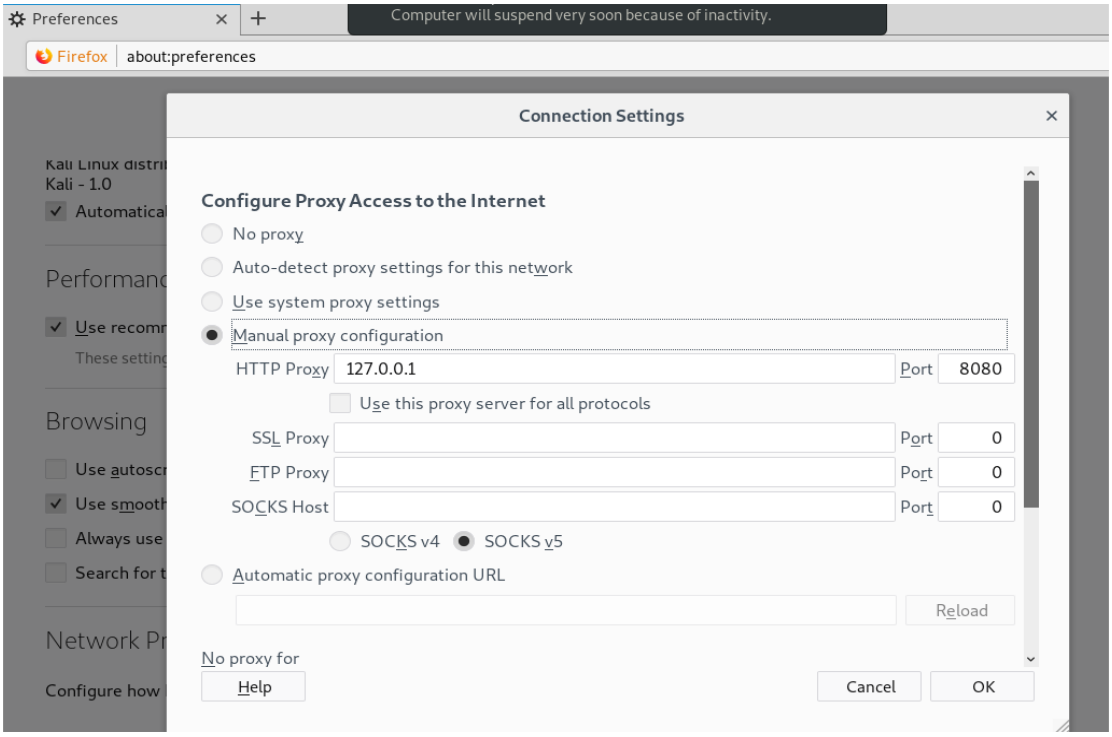
We will create random password file called worldlist.txt

Contain many password one of them password.

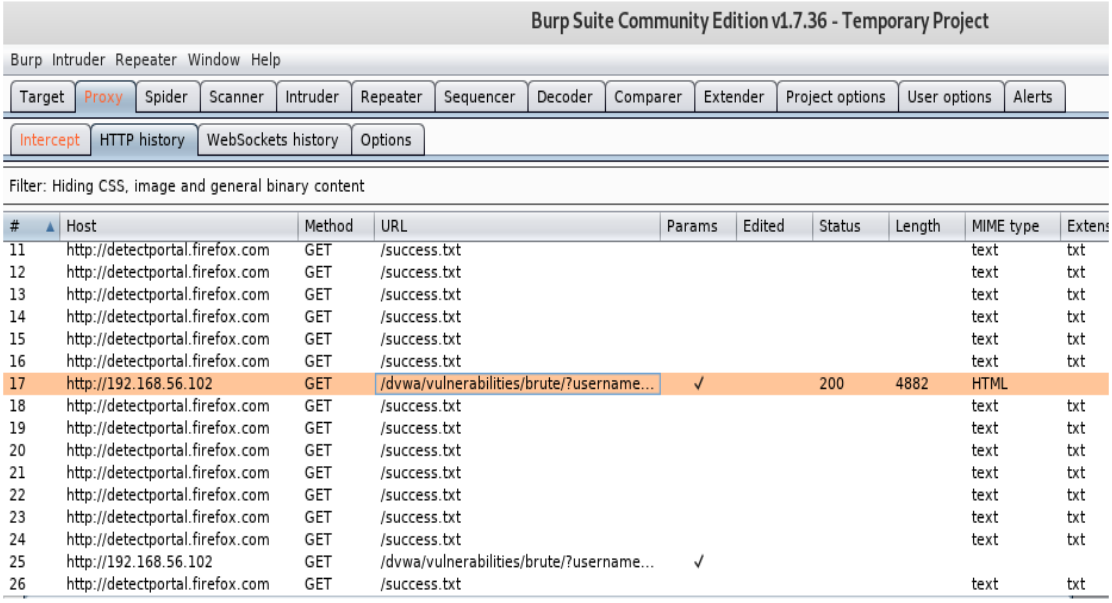
In this lab we will use BURPSUITE proxy to attack the server after running the BURP we should configure the proxy options in to 127.0.0.1:8080



In Firefox browser in kali we should use BURP as proxy so go to preference and select network setting to 127.0.0.1:8080

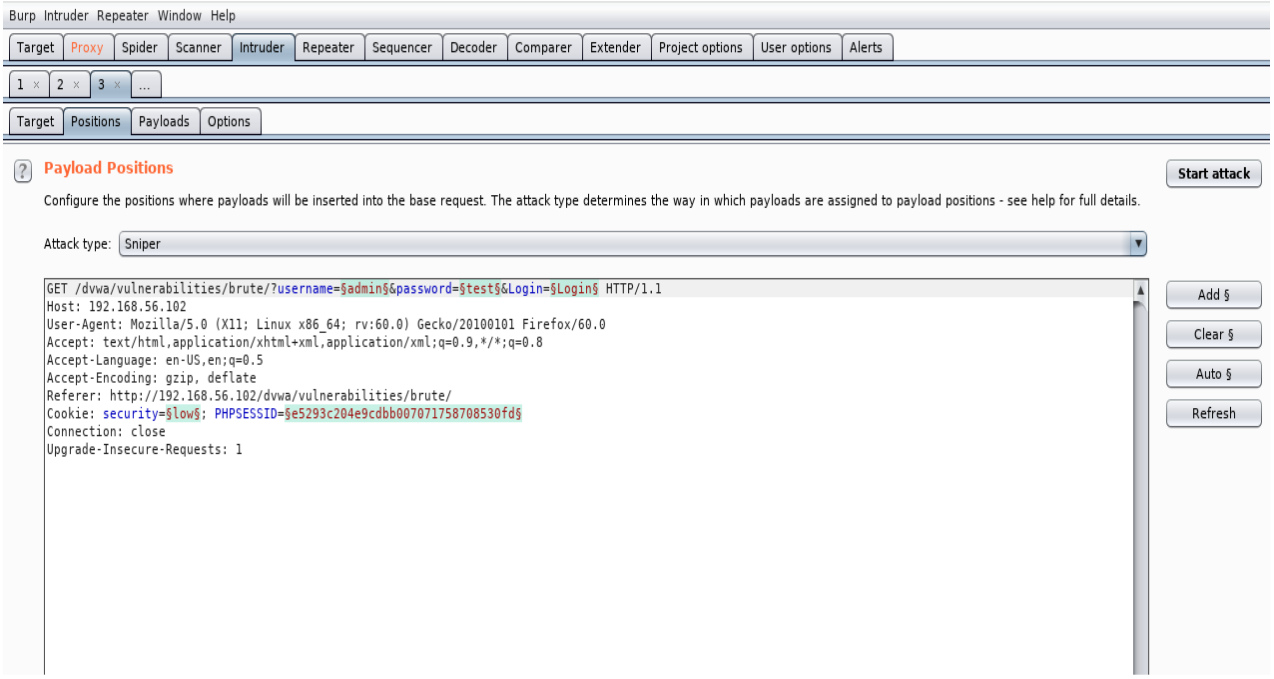


Make sure intercept is off in BURP open the brute force link in DVWA and put username admin password any password like test then turn on the intercept then refresh the page and go to Burp proxy history and find the requested page as following

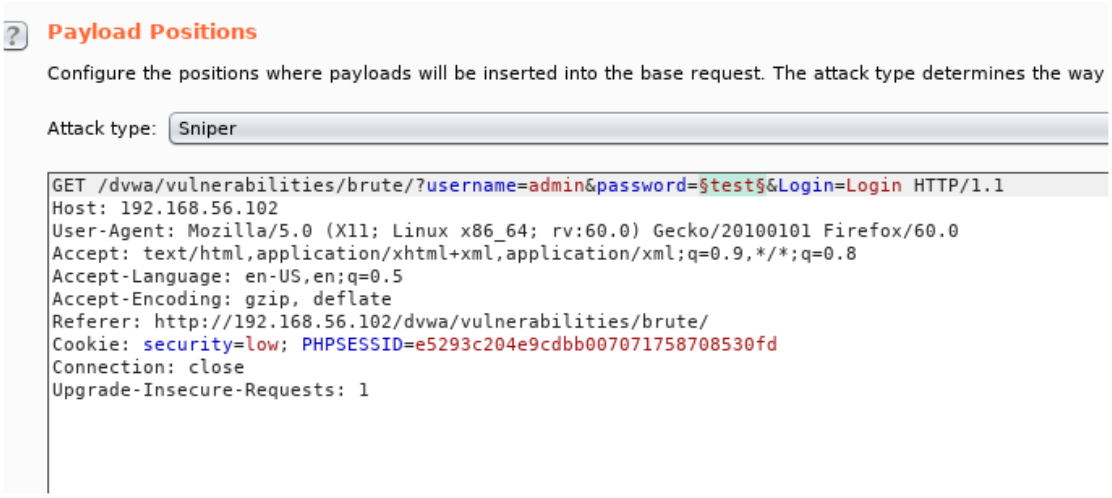


#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extens
11	http://detectportal.firefox.com	GET	/success.txt					text	txt
12	http://detectportal.firefox.com	GET	/success.txt					text	txt
13	http://detectportal.firefox.com	GET	/success.txt					text	txt
14	http://detectportal.firefox.com	GET	/success.txt					text	txt
15	http://detectportal.firefox.com	GET	/success.txt					text	txt
16	http://detectportal.firefox.com	GET	/success.txt					text	txt
17	http://192.168.56.102	GET	/dvwa/vulnerabilities/brute/?username...	✓		200	4882	HTML	
18	http://detectportal.firefox.com	GET	/success.txt					text	txt
19	http://detectportal.firefox.com	GET	/success.txt					text	txt
20	http://detectportal.firefox.com	GET	/success.txt					text	txt
21	http://detectportal.firefox.com	GET	/success.txt					text	txt
22	http://detectportal.firefox.com	GET	/success.txt					text	txt
23	http://detectportal.firefox.com	GET	/success.txt					text	txt
24	http://detectportal.firefox.com	GET	/success.txt					text	txt
25	http://192.168.56.102	GET	/dvwa/vulnerabilities/brute/?username...	✓					
26	http://detectportal.firefox.com	GET	/success.txt					text	txt

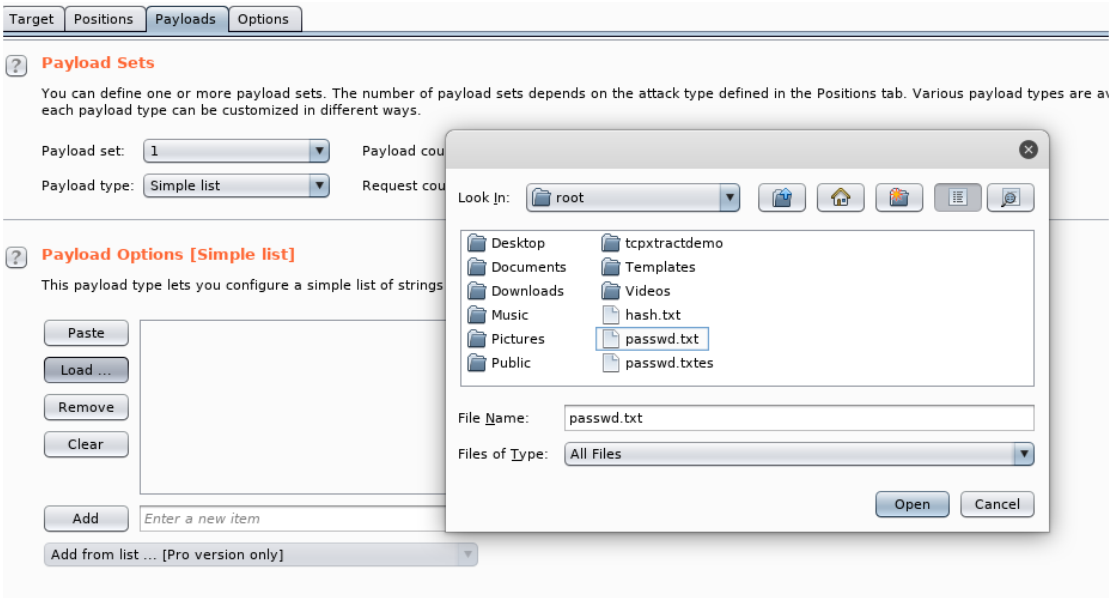
Right click on the link and send it to intruder the go to intruder go to Position snd select attack type sniper



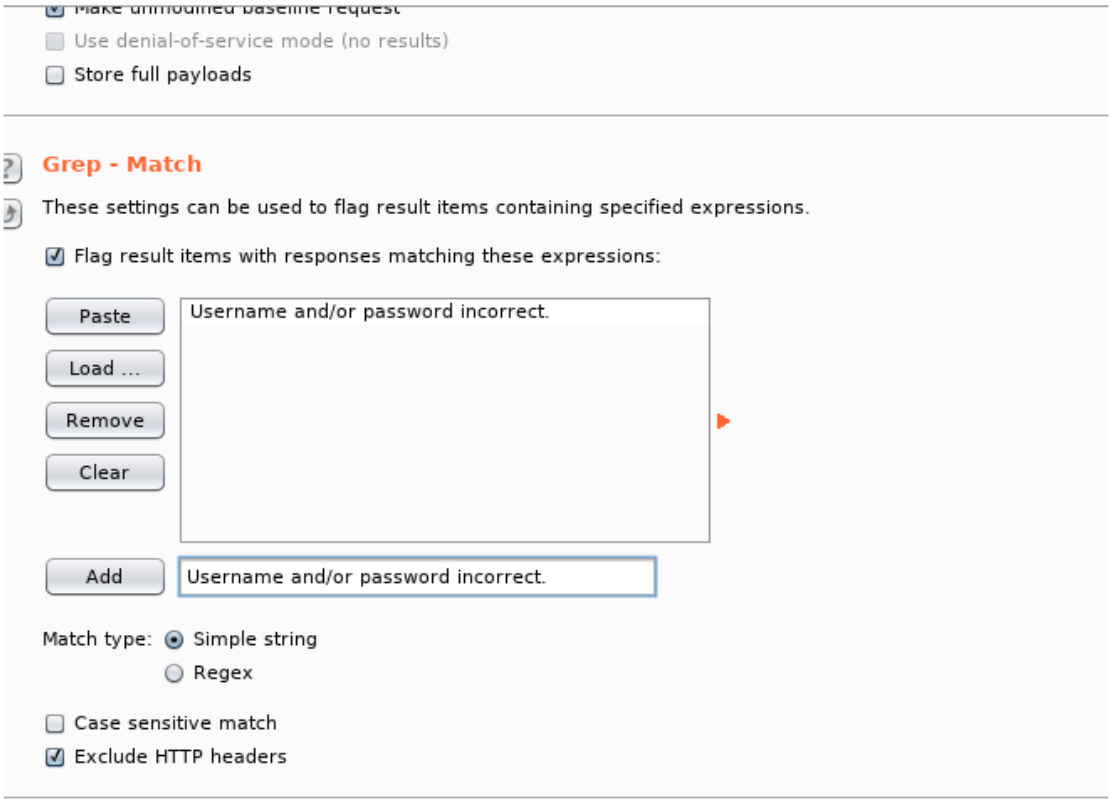
Press clear button to clear all selected fields, then we are interested in password filed. select the word test and press select



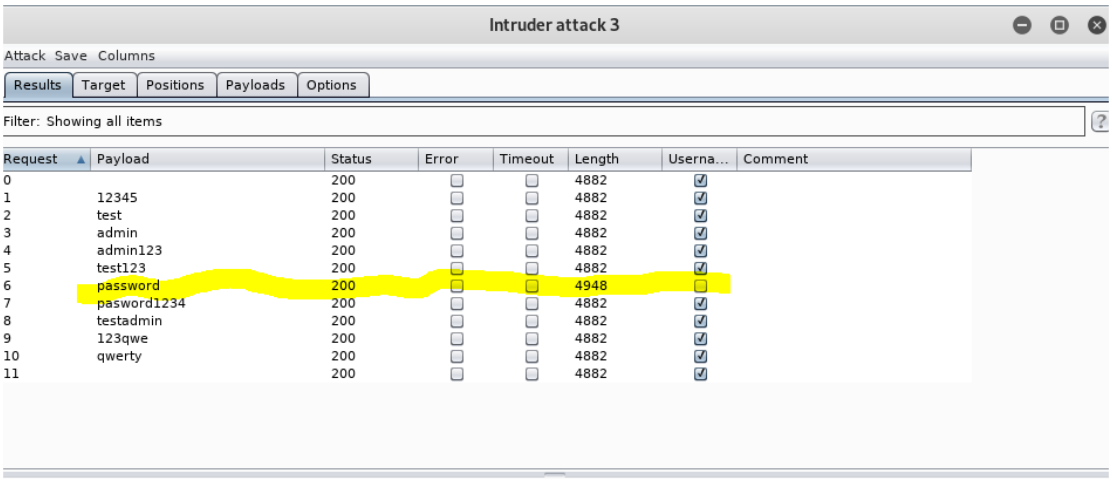
Then we will configure the Payload as a following we will choose passwd.txt file be loaded.



In the option filed we can configure the Grep match as we received from the website



Then start the attack, we found the password which is password.



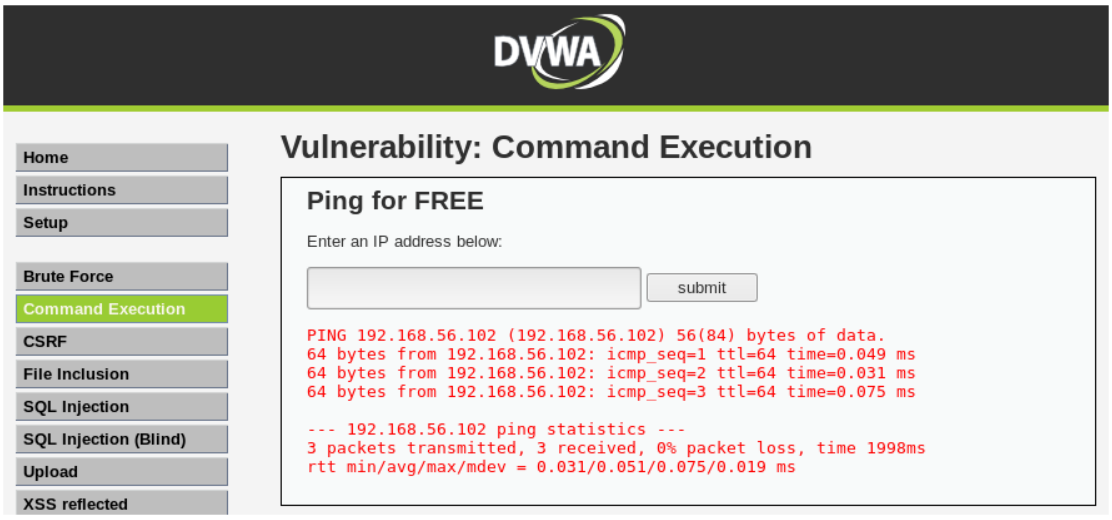
Request	Payload	Status	Error	Timeout	Length	Userna...	Comment
0		200			4882		
1	12345	200			4882		
2	test	200			4882		
3	admin	200			4882		
4	admin123	200			4882		
5	test123	200			4882		
6	password	200			4948		
7	pasword1234	200			4882		
8	testadmin	200			4882		
9	123qwe	200			4882		
10	qwerty	200			4882		
11		200			4882		

Attack 2 Command Execution

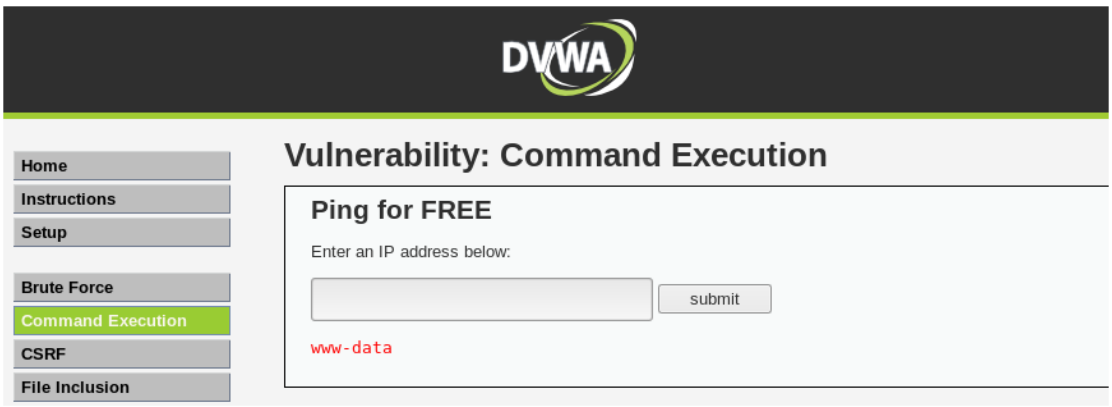
Overview

Command injection is an attack in which the goal is execution of arbitrary commands on the host operating system via a vulnerable application. Command injection attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell. In this attack, the attacker-supplied operating system commands are usually executed with the privileges of the vulnerable application. Command injection attacks are possible largely due to insufficient input validation.

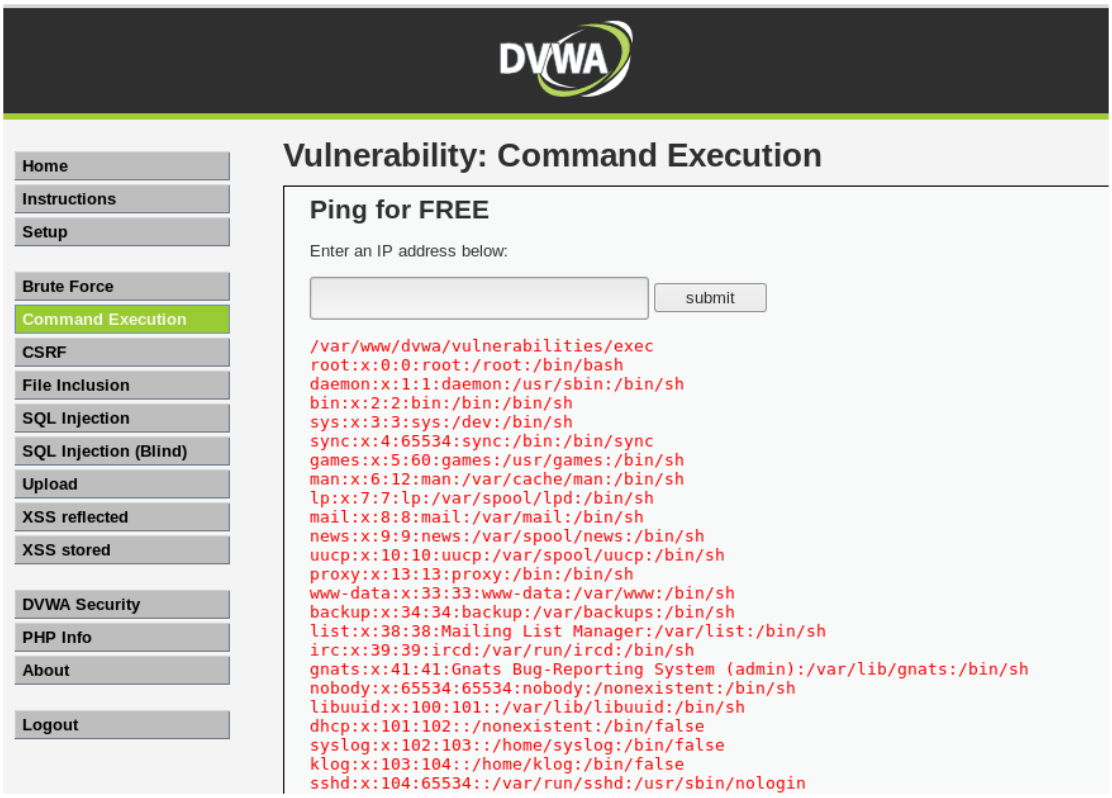
In this senior the form configured to execute ping command so let us try it using 192.168.56.102 and press submit the result is as following.



We can run arbitrary code by typing ; whoami which will tell us the server is running as the www-data user. The semicolon is a way to stack commands in Linux, so here we use it to end the previous command (which was the ping functionality), and we insert a new command of our choosing to be run by the vulnerable server.



We can use many commands by using ; like ;pwd; cat /etc/passwd



Let's change the DVWA security to the medium and trying command

; whoami and press submit the command can't executed so let's try other command || whoami .

The difference between the operators is that ; runs both commands irrespective of the first command's status, whereas || executes the second command only if the previous one failed.

Attack 3 Cross site request forgery CSRF

CSRF is an attack which forces an end user to execute unwanted actions on a web application in which he/she is currently authenticated. With a little help of social engineering (like sending a link via email/chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. A successful CSRF exploit can compromise end user data and operation in case of normal user. If the targeted end user is the administrator account, this can compromise the entire web application**.**

First view the page source and copy the form that responsible to change the password and past it in text editor.

```
<form action="#" method="GET">    New password:<br>
<input type="password" AUTOCOMPLETE="off" name="password_new"><br>
Confirm new password: <br>
<input type="password" AUTOCOMPLETE="off" name="password_conf">
<br>
<input type="submit" value="Change" name="Change">
</form>
```

So let's try to change the password for the user that already logged by using the following link

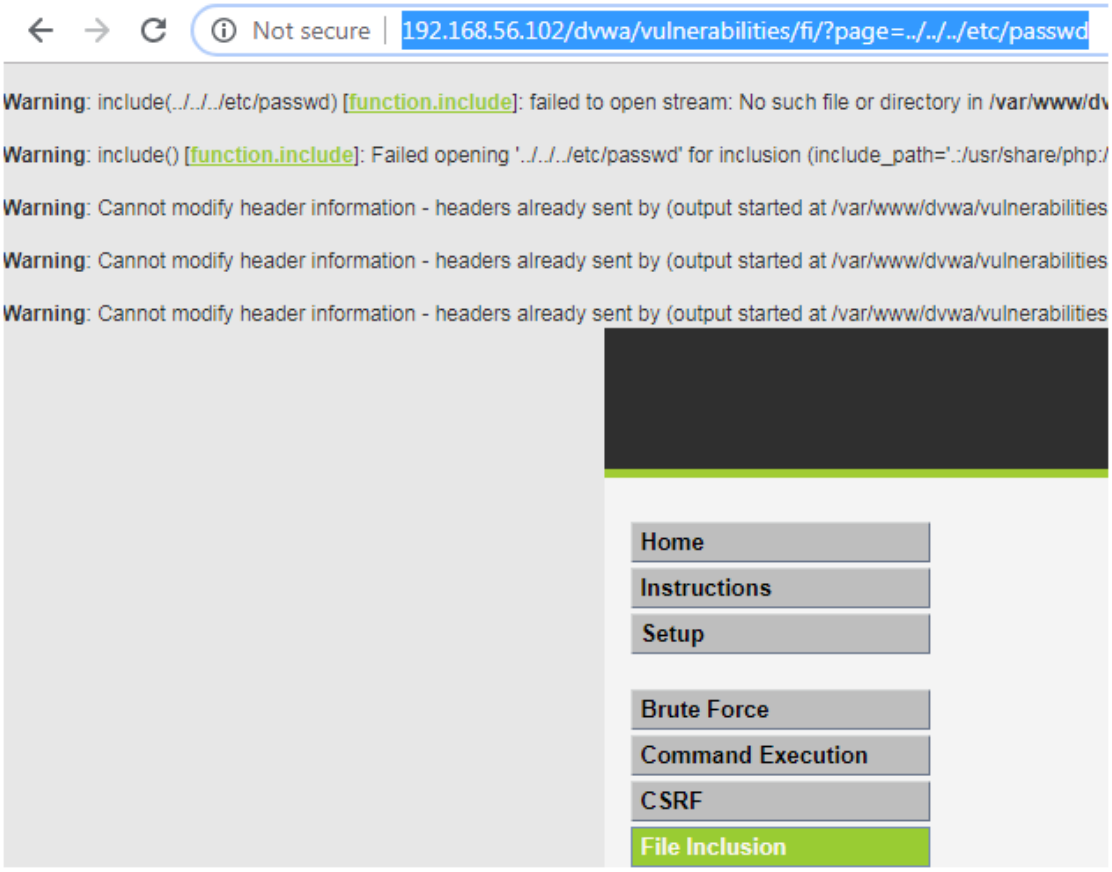
http://192.168.56.102/dvwa/vulnerabilities/csrf/?password_new=1234&password_conf=1234&Change=Change

The result show us that the password is changed the user now can't use his password since he rest his password as the attacker manipulating him. Try login by new password 1234.

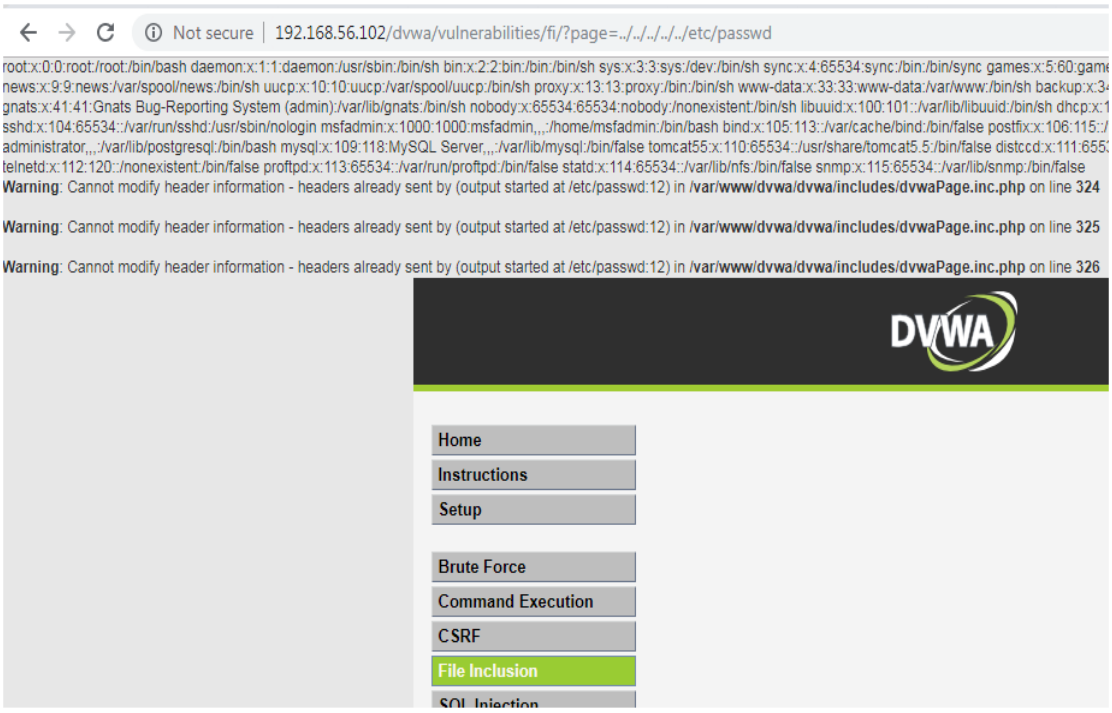
Attack 4 File inclusion

The application loads data from an attacker-controlled resource at runtime, enabling a variety of malicious activities. Either the source address or the resource itself (or both) may be under the attacker's control.

In this attack we will try to replace include.php by ../../../etc/passwd and try it, check the result.



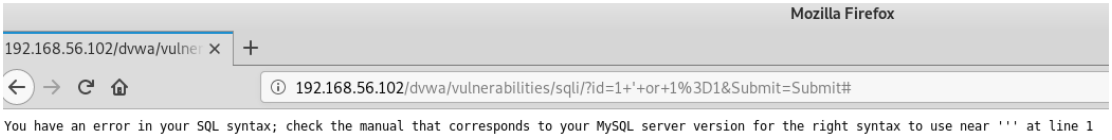
Let's try page=../../../../../etc/passwd
As you can see we get access to passsd file.



Attack 5 SQL injection

A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to effect the execution of predefined SQL commands.

The vulnerability is straightforward. On low security, injecting 'or 1=1 the following error.

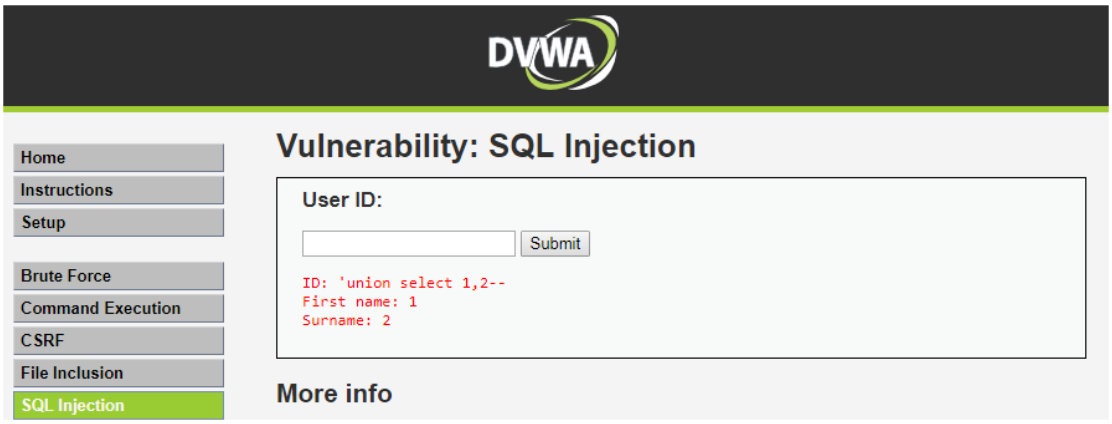


You can find the user data base by try ID 1,2,3,4 and 5.

Discover the number of columns

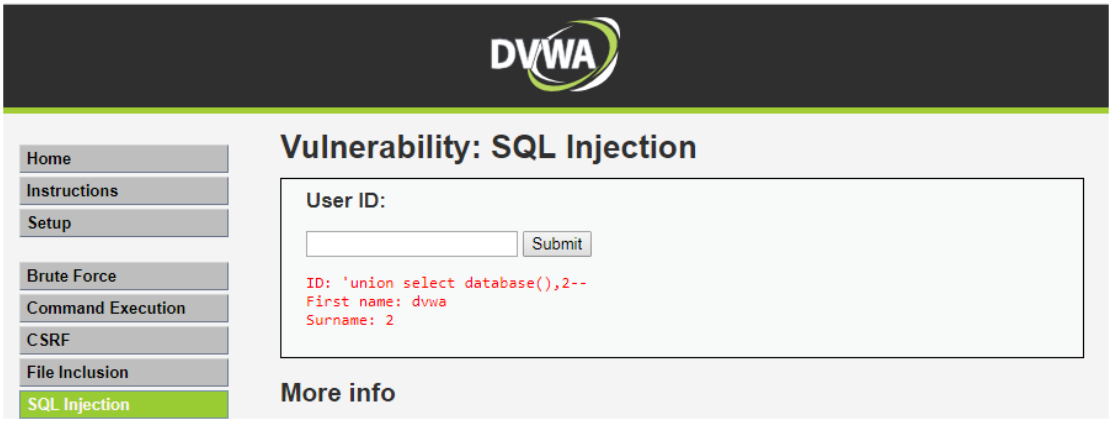
Using the following command, after – you should press space and then submit

'union select 1,2—



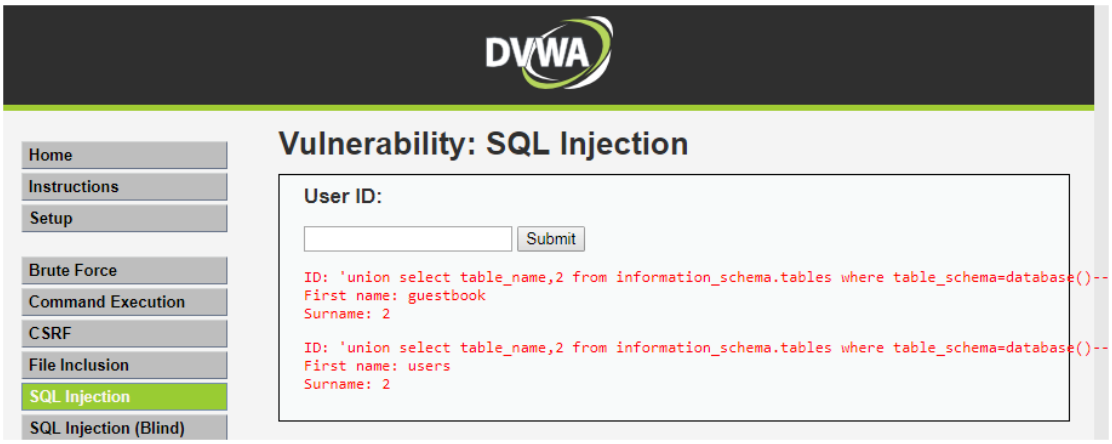
Get database name

'union select database(),2—



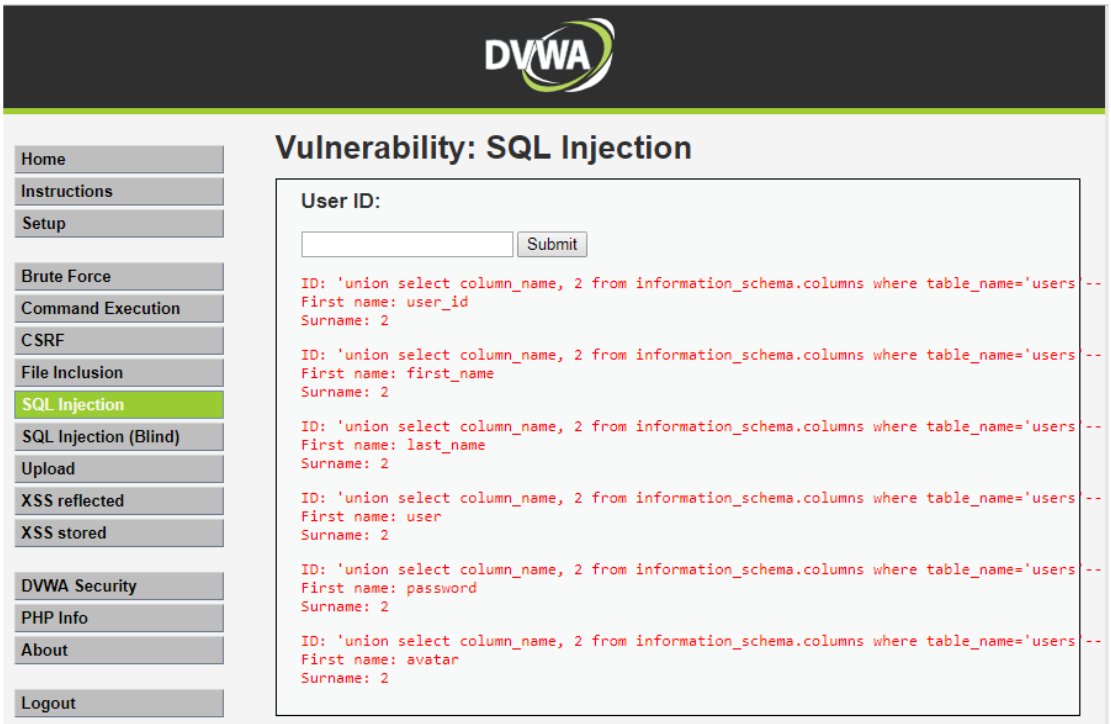
Get table names for the current database

'union select table_name,2 from information_schema.tables where table_schema=database()—



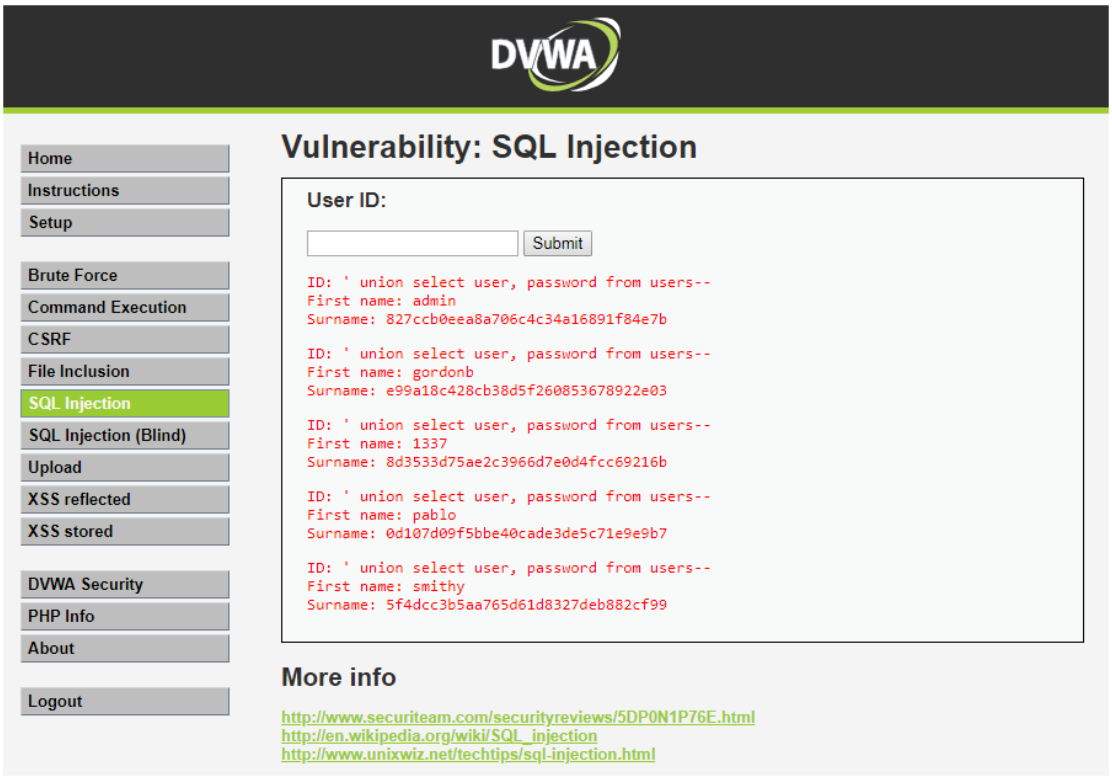
Get column names for the users table

'union select column_name, 2 from information_schema.columns where table_name='users'--



Get password hashes

' union select user, password from users--



Attack 6 File Upload

Uploaded files represent a significant risk to applications. The first step in many attacks is to get some code to the system to be attacked. Then the attack only needs to find a way to get the code executed. Using a file upload helps the attacker accomplish the first step.The consequences of unrestricted file upload can vary, including complete system takeover, an overloaded file system or database, forwarding attacks to back-end systems, and simple defacement. It depends on what the application does with the uploaded file and especially where it is stored.

First we are going to create php file that run command uname –a

In the text editor write the following

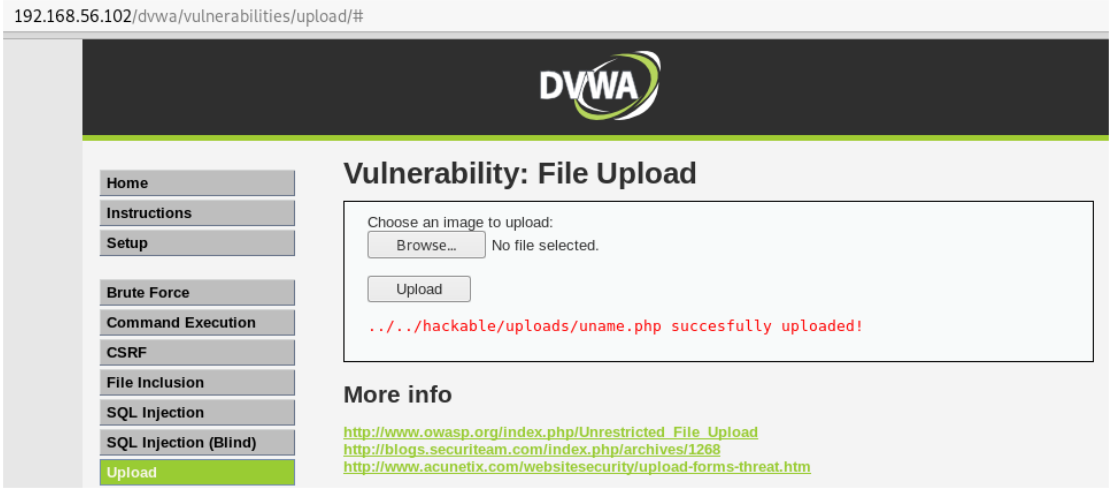
```
<?php

$cmd = system('uname –a');
echo $cmd;

?>
```

And save the file as uname.php

Then go to the DVWA server and upload the file



As you can see the uname.php has been uploaded in the path /../../hackable/uplodas

So let's open the file in the browser

http://192.168.56.102/dvwa/vulnerabilities/upload/../../hackable/uploads/unmae.php

We should get the system name info as below

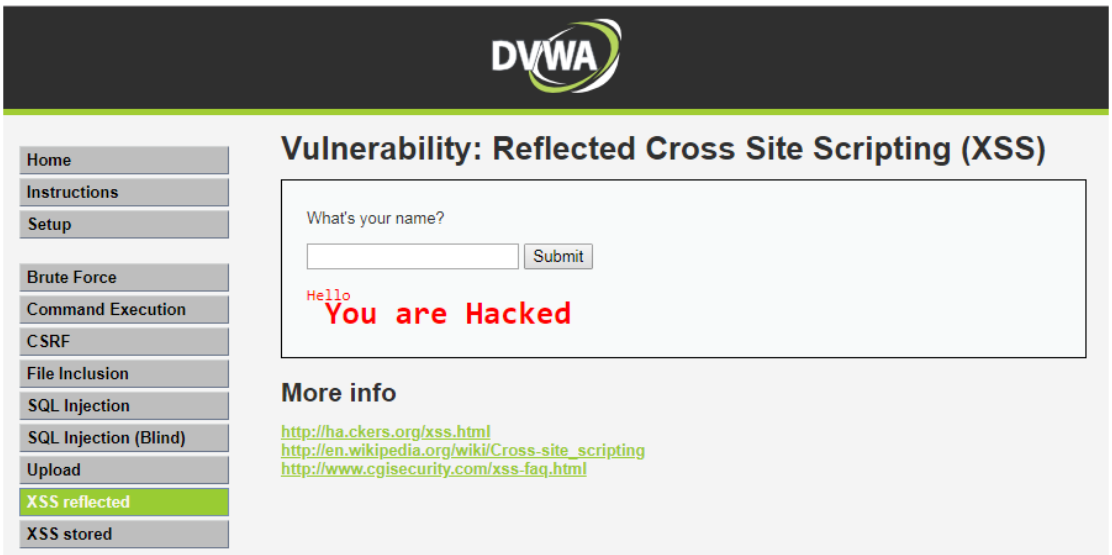
Attack 7 XSS reflected

Reflected attacks are those where the injected script is reflected off the web server, such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request. Reflected attacks are delivered to victims via another route, such as in an e-mail message, or on some other web site. When a user is tricked into clicking on a malicious link, submitting a specially crafted form, or even just browsing to a malicious site, the injected code travels to the vulnerable web site, which reflects the attack back to the user's browser. The browser then executes the code because it came from a

“trusted” server. Reflected XSS is also sometimes referred to as Non-Persistent or Type-II XSS.

Let's try the following

<h1> You are Hacked </h1>



Let' s try java script code <script>alert("hello")</script>