

共筆連結

SQL injection 、 XSS

SQL injection 、 XSS

CONTENTS

0x01 SQL injection

0x02 跨站腳本攻擊 (XSS)

CONTENTS

0x01 SQL injection

先備知識

攻擊介紹與認識

攻擊與防禦繞過

0x02 跨站腳本跨站 (XSS)

CONTENTS

0x01 SQL injection

0x02 跨站腳本攻擊(XSS)

先備知識

攻擊介紹與認識

攻擊與防禦繞過

為什麼SQL INJECTION跟XSS很重要？

除了是OWASP TOP 10公布的網頁十大風險
這兩個攻擊更是簡單又暴力的攻擊
甚至至今都還常出現在許多網頁

SQL injection



SQL INJECTION

是什麼？

藉由在輸入字串中夾帶SQL指令，攻擊資料庫(查詢、刪除.....等)，進而產生其他網頁威脅。





SQL INJECTION

可以幹嘛？

1. 資料外洩
2. 破壞資料庫 (例：DELETE、DROP可以刪除資料)
3. 破壞網頁





SQL INJECTION

的種類？

- 從回應網頁看到執行結果/錯誤訊息

Union-Based、Error-Based

- 僅可以知道有沒有成功

Boolean-Based、Time-Based



SQL INJECTION

的種類？

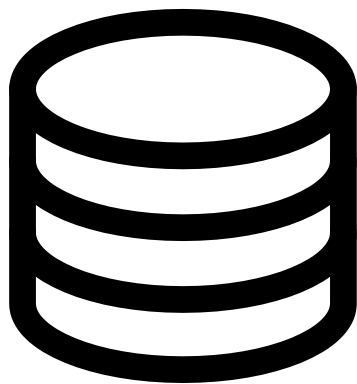
- Time-Based：應用沒有明確的回顯，只能使用特定的時間函數來判斷
- Boolean-Based：只能從應用返回中推斷語句執行後的布林值
- Error-Based：應用會顯示全部或者部分的報錯信息
- Union-Based：有的應用可以加入;後一次執行多條語句

複習一下資料庫與SQL語言.....

- **Structured Query Language**：結構化查詢語言
- SQL的範圍包括資料新增、查詢、更新和刪除，資料庫模式建立和修改，以及資料存取控制。



資料庫



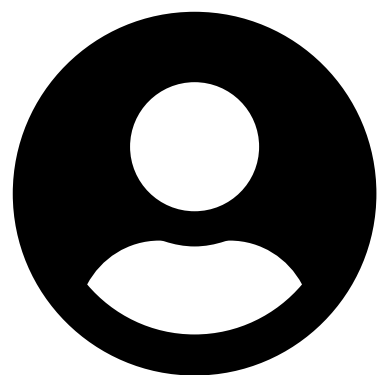
資料庫

- 帳戶資料：帳號、密碼、.....等

資料表

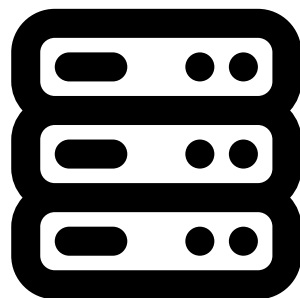
資料欄位

資料庫



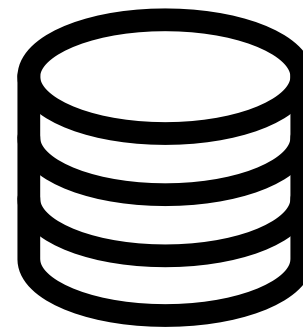
user

操作



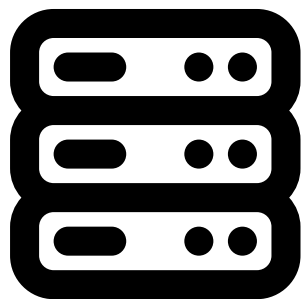
伺服器(後端)

SQL 語句
傳送指令與資料



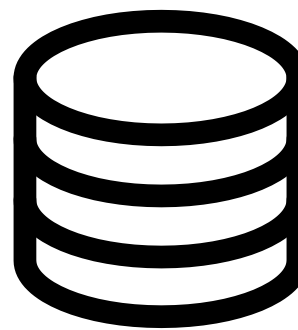
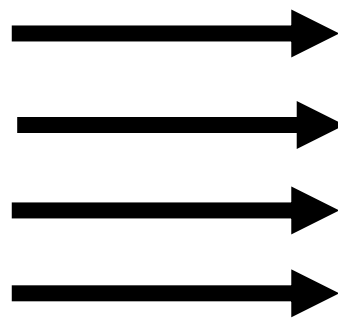
資料庫

資料庫



伺服器

新增
查詢
更新
刪除



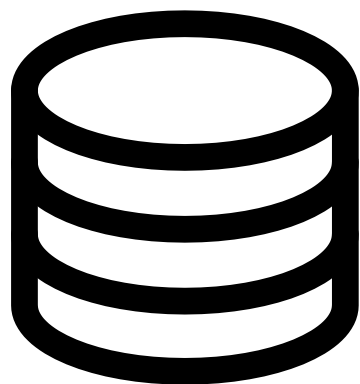
資料庫

SQL 語法

- 查詢: `SELECT * FROM TABLE`
- 新增: `INSERT INTO "表格名" ("欄位1", "欄位2", ...) VALUES ("值1", "值2", ...);`
- 更新: `UPDATE "表格名" SET "欄位1" = [新值] WHERE "條件";`
`UPDATE users SET age= 140 WHERE name= "hahaha"`
- 刪除: `DELETE FROM, DROP TABLE, ...`

資料庫

```
SELECT * FROM user WHERE id =1;
```

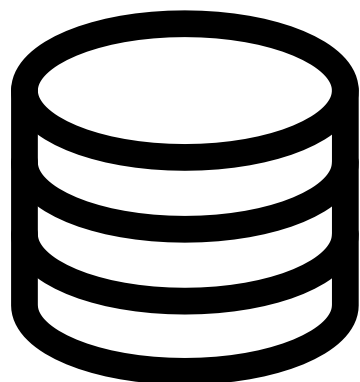


資料庫

id	username	password	Create_date
1	Kevin	1qaz2wsx	2020/01/23
2	Jonathan	p@@@sw0rddd	2020/09/18
3	Peter	14mp3t3r	2021/08/23

資料庫

```
SELECT * FROM user WHERE id =2;
```

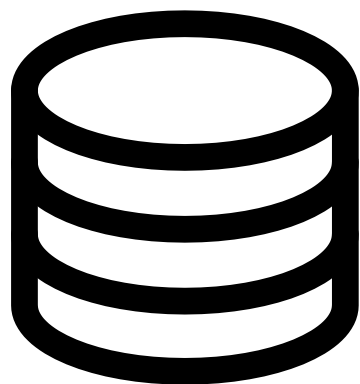


資料庫

id	username	password	Create_date
1	Kevin	1qaz2wsx	2020/01/23
2	Jonathan	p@@@sw0rddd	2020/09/18
3	Peter	14mp3t3r	2021/08/23

資料庫

```
SELECT * FROM user WHERE id =3;
```

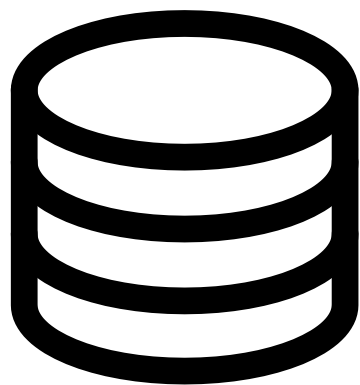


資料庫

id	username	password	Create_date
1	Kevin	1qaz2wsx	2020/01/23
2	Jonathan	p@@@sw0rddd	2020/09/18
3	Peter	14mp3t3r	2021/08/23

資料庫

SELECT * FROM user WHERE id =3; DROP TABLE user



資料庫

id	username	password	Create_date
1	Kevin	1qaz2wsx	2020/01/23
2	Jonathan	p@@@sw0rddd	2020/09/18
3	Peter	14mp3t3r	2021/08/23



資料庫

id	password	Create_date
1		2020/01/23
2	Jonat	2020/09/18
3	Peter	2020/09/23

user WHERE id =3; DROP TABLE user

PWNED !!!



練習一下基本的SQL查詢語法.....

選擇所有

```
SELECT * FROM Users WHERE Username=' ' + txt.User.Text+' ' AND  
Password=' '+ txt.Password.Text+' '
```





練習一下基本的SQL查詢語法.....

從Users這個表格裡

```
SELECT * FROM Users WHERE Username=' ' + txt.User.Text+' ' AND  
Password=' '+ txt.Password.Text+' '
```





練習一下基本的SQL查詢語法.....

其中Username是

```
SELECT * FROM Users WHERE Username=' ' + txt.User.Text+' ' AND  
Password=' '+ txt.Password.Text+' '
```





練習一下基本的SQL查詢語法.....

(使用者輸入他的Username)

```
SELECT * FROM Users WHERE Username=' ' + txt.User.Text+' ' AND  
Password=' '+ txt.Password.Text+' '
```





練習一下基本的SQL查詢語法.....

且(這個很重要)

```
SELECT * FROM Users WHERE Username=' ' + txt.User.Text+' ' AND  
Password=' '+ txt.Password.Text+' '
```





練習一下基本的SQL查詢語法.....

```
SELECT * FROM Users WHERE Username=' ' + txt.User.Text+' ' AND  
Password=' '+ txt.Password.Text+' '
```

密碼是






練習一下基本的SQL查詢語法.....

```
SELECT * FROM Users WHERE Username=' ' + txt.User.Text+' ' AND  
Password=' '+ txt.Password.Text+' '
```

(使用者的密碼)



[illegible]

Username : Jonathan

SQL語言對database查詢：

A decorative word cloud in the top right corner of the slide, featuring various names and words in different sizes and orientations, including 'JANA', 'DUST', 'NCH', 'ALEX', 'JOSE', 'BOSCAR', 'ZOEY', 'ADISON', 'COLE', 'WILLIAM', 'JOSEPH', 'JOHN', 'BYRON', 'ESTHER', 'WILLIAM', 'JOSEPH', 'JOHN', 'BYRON', 'ESTHER'.

駭客想幹的事.....

駭客的 input :

Username : admin ' or 1=1--

Password : (隨意輸入)

SQL語言對database查詢：

```
SELECT * FROM Users WHERE Username= 'admin' or 1=1--' AND
```

```
Password= (任意密碼)
```

A decorative word cloud in the bottom right corner of the slide, featuring various names and words in different sizes and orientations, including 'WILLIAM', 'JOSEPH', 'JOHN', 'BYRON', 'ESTHER', 'WILLIAM', 'JOSEPH', 'JOHN', 'BYRON', 'ESTHER'.

A decorative header at the top of the slide featuring a dense, overlapping cloud of names in various sizes and orientations, including names like COLTON, BRIELLA, JOSE, ALEXANDER, and others.

這是在幹嘛？

A decorative footer at the bottom of the slide featuring a smaller, more compact cloud of names, including WILLIAM, JOSEPH, and JOHN.

駭客想幹的事.....

駭客的 input :

Username : admin ' or 1=1--

Password : (隨意輸入)

admin 是管理員的意思

or 就是或

1=1 是個成立的等式

-- 是 SQL 語言中的註解符號

SQL語言對database查詢：

```
SELECT * FROM Users WHERE Username= 'admin' or 1=1--' AND
```

```
Password= (任意密碼)
```



也就是說.....



駭客想幹的事.....

駭客的 input :

Username : admin ' or 1=1--

Password : (隨意輸入)

admin或1=1只要一個成立就好

SQL語言對database查詢：

然而1=1就是個恆成立的等式

```
SELECT * FROM Users WHERE Username= 'admin' or 1=1--AND
```

```
Password=(任意密碼)
```

-- 後全部都被註解掉了



所以.....

駭客在繞過語法後，輸入隨意密碼的情況下，
拿到了網頁管理員的權限



WILLIAM
JOSEPH
JOHN
BYRON

[illegible]



UNION-BASED SQL INJECTION

```
UNION SELECT 1,2 --
```

```
UNION SELECT null,2 --
```



學了sql injection的我

UNION-BA

UNION SELECT 1,

UNION SELECT nu





Lab 0x01 Irish-Name-Repo 1


Lab 0x02 Irish-Name-Repo 2

Lab 0x0


Lab 0x02

事情才不是你想的那麼簡單

卑鄙源之助



這麼簡單的攻擊
網管會不知道嗎？



SQL injection filter bypass

A decorative word cloud in the top right corner of the slide, featuring various names and words in different sizes and orientations, including 'COLTON', 'BRIELLA', 'JOSEKAL', 'BOSCAR', 'ALEXANDER', 'NATHAN', 'ZOEY', 'ADISON', 'COLE', 'JULIA', 'XAVIER', 'CABERON', 'DAVID', 'ALEX', 'JOHN', 'JOSEPH', 'WILLIAM', 'ELIE', 'JOHN', 'JOSEPH', 'WILLIAM', 'ELIE'.

字串過濾

假設網頁限制了 admin，駭客可以.....：

1. 大小寫替換：

例：AdMiN、aDmIn.....等

A decorative word cloud in the bottom right corner of the slide, featuring various names and words in different sizes and orientations, including 'WILLIAM', 'ELIE', 'JOHN', 'JOSEPH', 'WILLIAM', 'ELIE', 'JOHN', 'JOSEPH'.



字串過濾

假設網頁限制了 admin，駭客可以.....：

3. 編碼：

admin的ASCII編碼：97 100 109 105 110

admin的unicode編碼：\u0061\u0064\u006d\u0069\u006e



A decorative word cloud in the top right corner of the slide, featuring various names and words in different sizes and orientations, including 'COLTON', 'BRIELLA', 'JOSEKAL', 'BOSCAR', 'ALEXANDER', 'NATHAN', 'ZOEY', 'ADISON', 'COLE', 'JULIA', 'XAVIER', 'CARRON', 'BOB', 'DAVID', 'ADAM', 'JOHN', 'JOSEPH', 'WILLIAM', 'ELIE', 'JOHN', 'JOSEPH', 'WILLIAM', 'ELIE'.

字串過濾

假設網頁限制了 admin，駭客可以.....：

3. 雙關鍵字繞過：

ADadminMIN

A decorative word cloud in the bottom right corner of the slide, featuring various names and words in different sizes and orientations, including 'WILLIAM', 'ELIE', 'JOHN', 'JOSEPH', 'WILLIAM', 'ELIE', 'JOHN', 'JOSEPH'.



字串過濾

假設網頁限制了 admin，駭客可以.....：


4. 註釋繞過：

Oracle	--comment
Microsoft	--comment /*comment*/
PostgreSQL	--comment /*comment*/
MySQL	#comment -- comment [Note the space after the double dash] /*comment*/



A decorative header at the top of the slide featuring a dense, overlapping cloud of names in various colors and orientations, including names like COLTON, BRIELLA, JOSE, and ALEXA.

其他符號限制繞過

1. 用 ; 代替 -- 、 # 去繞過
 2. 空白繞過 (%00 、 /**/ 、 + 、 %20)
- 
- A decorative footer at the bottom of the slide featuring a smaller, more concentrated cluster of names, including WILLIAM, JOSEPH, and JOHN.



Lab 0x03 Web Gauntlet



跨站腳本攻擊XSS

(cross-site scripting)

反射型 XSS

- 又稱為非儲存型
- 將惡意的 **script** (參數型、非持久型 XSS腳本)注入在網頁開發時所留下的漏洞 (常見的是 **URL** 網址列中)，並透過 **GET** 傳遞。
- 藏有惡意腳本的**URL**當然容易被發現，所以通常駭客會再對惡意的參數進行編碼(**ex : URL encode**)
- 此方法只能做一次性攻擊，且需要受害者主動點擊連結。
- 攻擊的應用：釣魚信件，可以竊取 **cookie**、帳號密碼...等機密資訊。

g.hacq.me/challenges/baby01.php?payload= <script>alert%28"xss"%29<%2Fscript> +



已匯入



未匯入

xss.challenge.training.hacq.me 顯示

xss

確定

產生一個alert框

直接將payload輸入在URL列中

```
https://xss.challenge.training.hacq.me/challenges/baby01.php?payload=<script>alert("xss")</script>
```

將payload裡的特殊字元編碼(url encode)後輸入在URL列中

```
/xss.challenge.training.hacq.me/challenges/baby01.php?payload=%3Cscript%3Ealert%28%22xss%22%29%3C%2Fscript%3E%20|
```

GET把參數皆在後方

靶場位置:

<https://xss.challenge.training.hacq.me/challenges/baby01.php>

儲存型 XSS

- 直接將惡意的腳本(javascript)儲存到資料庫中，當受害者瀏覽藏有惡意腳本網頁時，**server** 端會將惡意腳本取出，之後將惡意腳本傳回瀏覽器。
- 駭客常會將惡意腳本放在留言板中，等到下一個瀏覽這個留言板的受害者，並進行攻擊。

DOM 型 XSS

- DOM (Document Object Model)，它可以使 javascript 跑起來，而不用透過 server 端。
- 如果 javascript 在跑起來的過程中，沒有詳細檢查資料使得操作 DOM 的過程代入了惡意指令。

跨站腳本攻擊XSS

Bypass XSS filter

XSS繞過手法

- 過濾掉script字串的話 → 關鍵字繞過

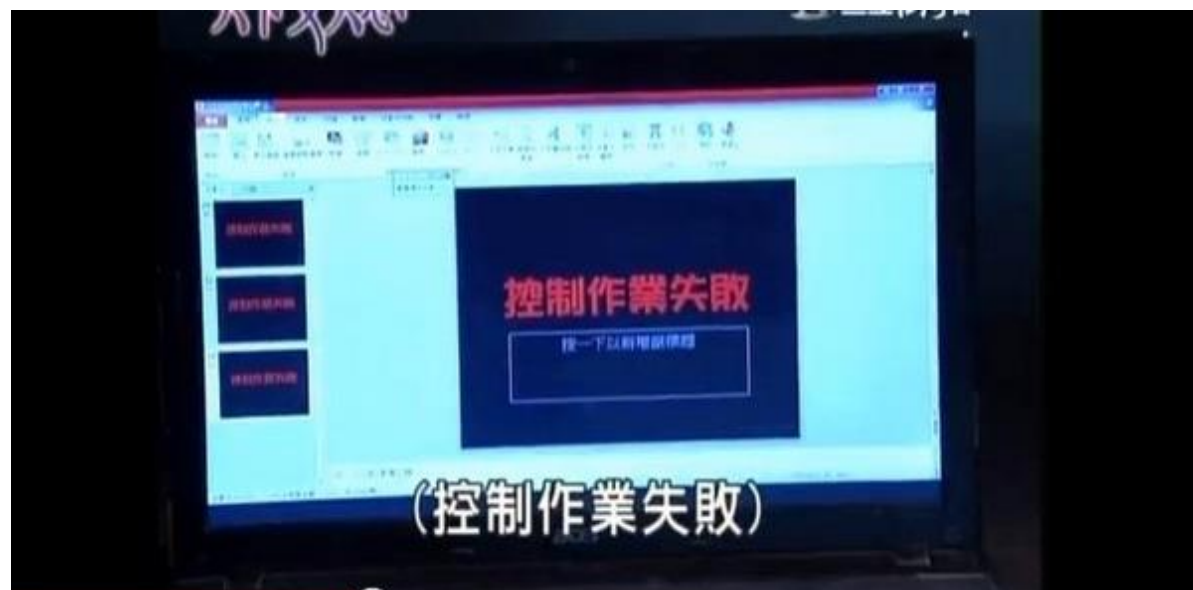
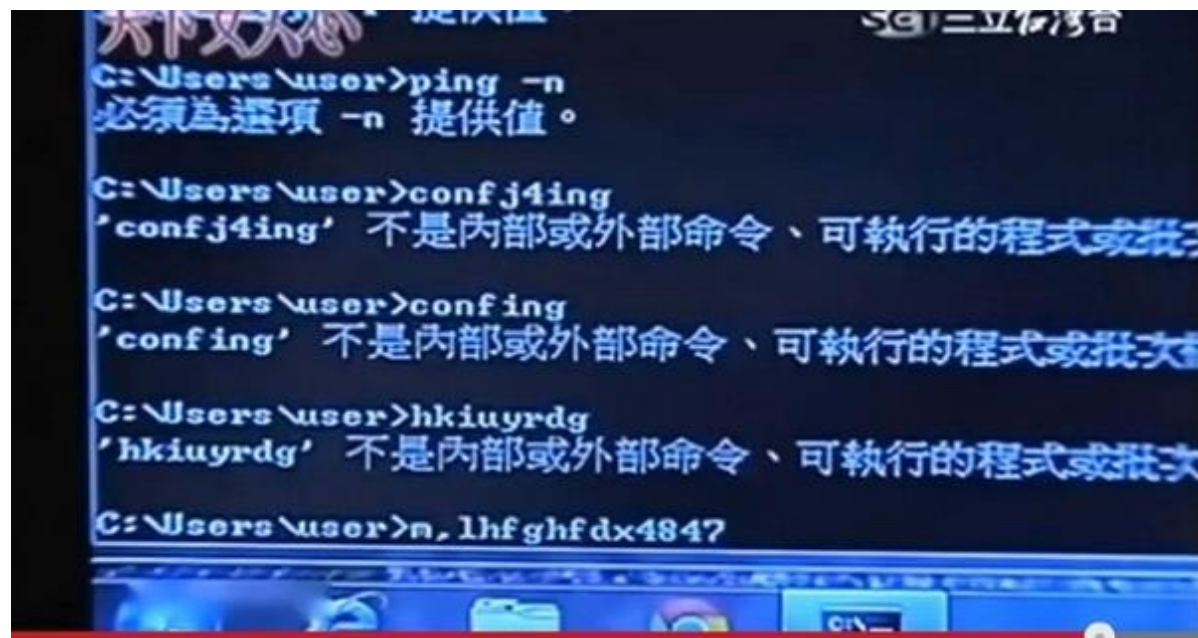
<ScRiPt></scRlpt>

<sc<script>ript></sc<script>ript>

- 過濾掉引號 → 語法編碼繞過

<script>alert(String.fromCharCode(88,83,83))</script> = <script>alert("XSS")</script>

XSS攻擊練習



解題思路



尋找注入點與截斷點



思考注入的tag與內容



特定題目需要透過設計
payload來繞過某些字元

Warning: You are entering the XSS game area

Welcome, recruit!

Cross-site scripting (XSS) bugs are one of the most common and dangerous types of vulnerabilities in Web applications. These nasty buggers can allow your enemies to steal or modify user data in your apps and you must learn to dispatch them, pronto!

At Google, we know very well how important these bugs are. In fact, Google is so serious about finding and fixing XSS issues that we are paying mercenaries up to \$7,500 for dangerous XSS bugs discovered in our most sensitive products.

In this training program, you will learn to find and exploit XSS bugs. You'll use this knowledge to confuse and infuriate your adversaries by preventing such bugs from happening in your applications.

There will be cake at the end of the test.

Let me at 'em!

?

<https://xss-game.appspot.com/>

剛學會XSS準備大展身手的我




看到第一題的我



Target code (toggle)

```
33
34 def get(self):
35     # Disable the reflected XSS filter for demonstration purposes
36     self.response.headers.add_header("X-XSS-Protection", "0")
37
38     if not self.request.get('query'):
39         # Show main search page
40         self.render_string(page_header + main_page_markup + page_footer)
41     else:
42         query = self.request.get('query', '[empty]')
43
44         # Our search engine broke, we found no results :-(
45         message = "Sorry, no results were found for <b>" + query + "</b>."
46         message += " <a href='?'>Try again</a>."
47
48         # Display the results page
49         self.render_string(page_header + message + page_footer)
50
51     return
52
53 application = webapp.WSGIApplication([ ('.*', MainPage), ], debug=False)
```



message = "Sorry, no results were found for <script>alert("xss")</script>."





- (載入一張圖片，但這個圖片來源是錯的來源，所以無法成功載入圖片，因此產生一個alert)

Target code (toggle)

```
5 <script src="/static/game-frame.js"></script>
6 <link rel="stylesheet" href="/static/game-frame-styles.css" />
7
8 <!-- Load jQuery -->
9 <script
10   src="//ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js">
11 </script>
12
13 <script>
14   function chooseTab(num) {
15     // Dynamically load the appropriate image.
16     var html = "Image " + parseInt(num) + "<br>";
17     html += "<img src='/static/level3/cloud" + num + ".jpg' />";
18     $('#tabContent').html(html);
19
20     window.location.hash = num;
21
22     // Select the current tab
23     var tabs = document.querySelectorAll('.tab');
24     for (var i = 0; i < tabs.length; i++) {
25       if (tabs[i].id == "tab" + parseInt(num)) {
26         tabs[i].className = "tab active";
27       }
28     }
29   }
30 }
```

我們輸入的東西會被這個參數讀取



```
html += "<img src='/static/level3/cloud' + num + '.jpg' />";
```

後面的單引號輸入，使前面的單引號閉合



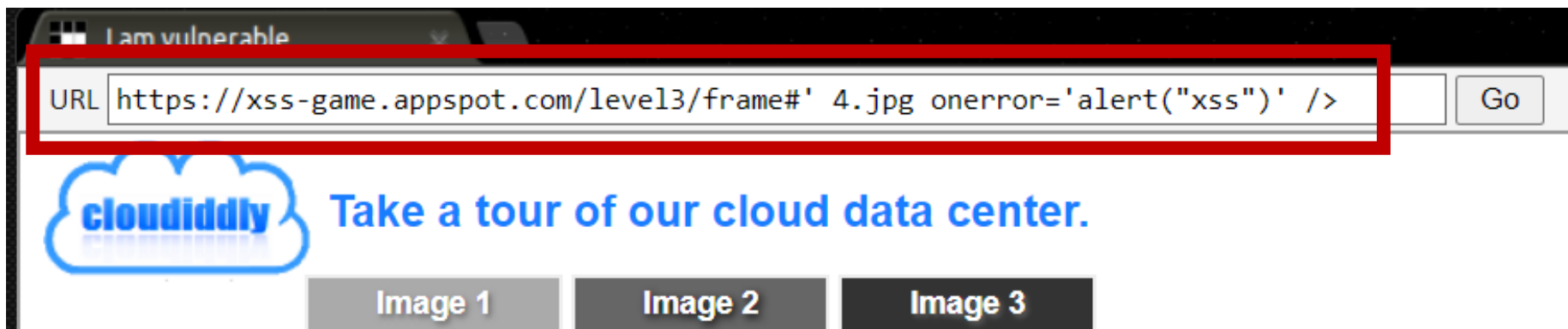
自己把這個html閉合，讓後面讀不到

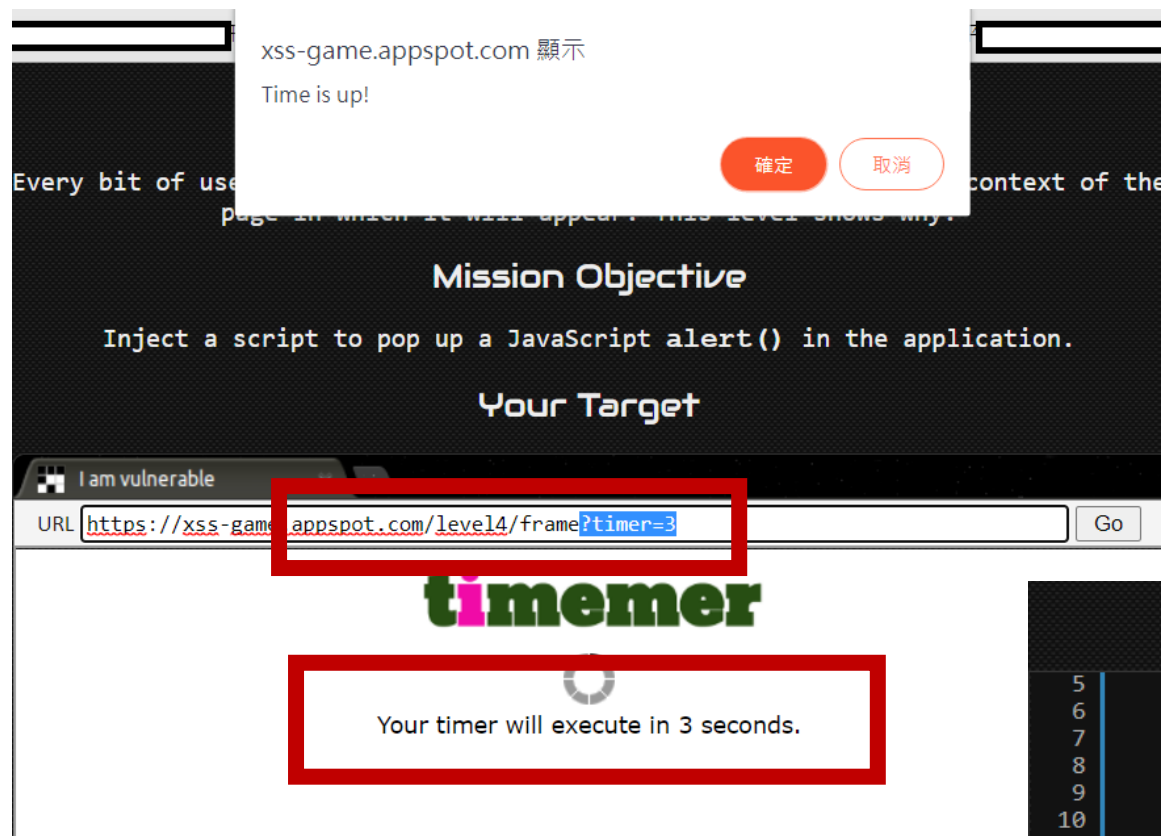


```
html += "<img src='/static/level3/cloud' 4.jpg onerror='alert(\"xss\")' />.jpg' />";
```



輸入一個假的.jpg檔，騙瀏覽器讀取







要選哪個為截斷點？



輸入的東西會被{{ timer }}這個參數接收



```

```

) 閉合掉前面的(' , ;使後面的alert被執行
('xss 恰巧能與本來tag裡面的 ')合成

```

```



這就是漏洞的藝術~

這邊似乎沒有可以注入的地方，按下sign up



注意到 url 上 **Signup?next=confirm**，f12打開開發人員工具
(下面的code不會跟著網頁而做改變)



Mission Description

Cross-site scripting isn't *just* about correctly escaping data. Sometimes, attackers can do bad things even without injecting new elements into the DOM.

Mission Objective

Inject a script to pop up an alert() in the context of the application.

Your Target

URL

Go

Groovy
Reader 2.0

Enter email:

[Next >>](#)

```
Elements Console Sources Network Performance >> 6
<!-- Internal game scripts/styles, mostly boring stuff -->
<script src="/static/game-frame.js"></script>
<link rel="stylesheet" href="/static/game-frame-styles.css">
</head>
<body id="level5">
  
  <br>
  <br>
  <!-- We're ignoring the email, but the poor user will never know! -->
  " Enter email: "
  <input id="reader-email" name="email" value>
  <br>
  <br>
  <a href="/confirm">Next >></a> == $0
</body>
</iframe>
</div>
<h2>...</h2>
<iframe id="source-frame" src="/level5/source" style="display: inline;">...</iframe>
<h2>...</h2>
<div id="hints">...</div>
<a href="http://tools.ietf.org/html/draft-hoehrmann-javascript-scheme-00"> </a>
</body>
</html>

html body div#game-frame-container iframe.game-frame html body#level5 a
```



也就是.....



{{ next }} 放的就是confirm

Target code (toggle)

confirm.html level.py **signup.html** welcome.html

```
1 <!doctype html>
2 <html>
3   <head>
4     <!-- Internal game scripts/styles, mostly boring stuff -->
5     <script src="/static/game-frame.js"></script>
6     <link rel="stylesheet" href="/static/game-frame-styles.css" />
7   </head>
8
9   <body id="level5">
10    <br><br>
11    <!-- We're ignoring the email, but the poor user will never know! -->
12    Enter email: <input id="reader-email" name="email" value="">
13
14    <br><br>
15    <a href="{{ next }}">Next >></a>
16  </body>
17 </html>
```





因此我們把CONFIRM換成我們要
注入的SCRIPT



`Next >>`

本來的 confirm 會在按下 next 之後被導向到其他頁面
但被換成了 javascript 後，瀏覽器會以為這是需要被執行的 javascript，因而執行這段 script。

`Next >>`



記得按下 Next 才會有 alert 喔~

url 中的 # 後可以放一個路徑



Hints 4/4 (show)

1. See how the value of the location fragment (after #) influences the URL of the loaded script.
2. Is the security check on the gadget URL really foolproof?
3. Sometimes when I'm frustrated, I feel like screaming.
4. If you can't easily host your own evil JS file, see if `google.com/jsapi?callback=foo` will help you here.



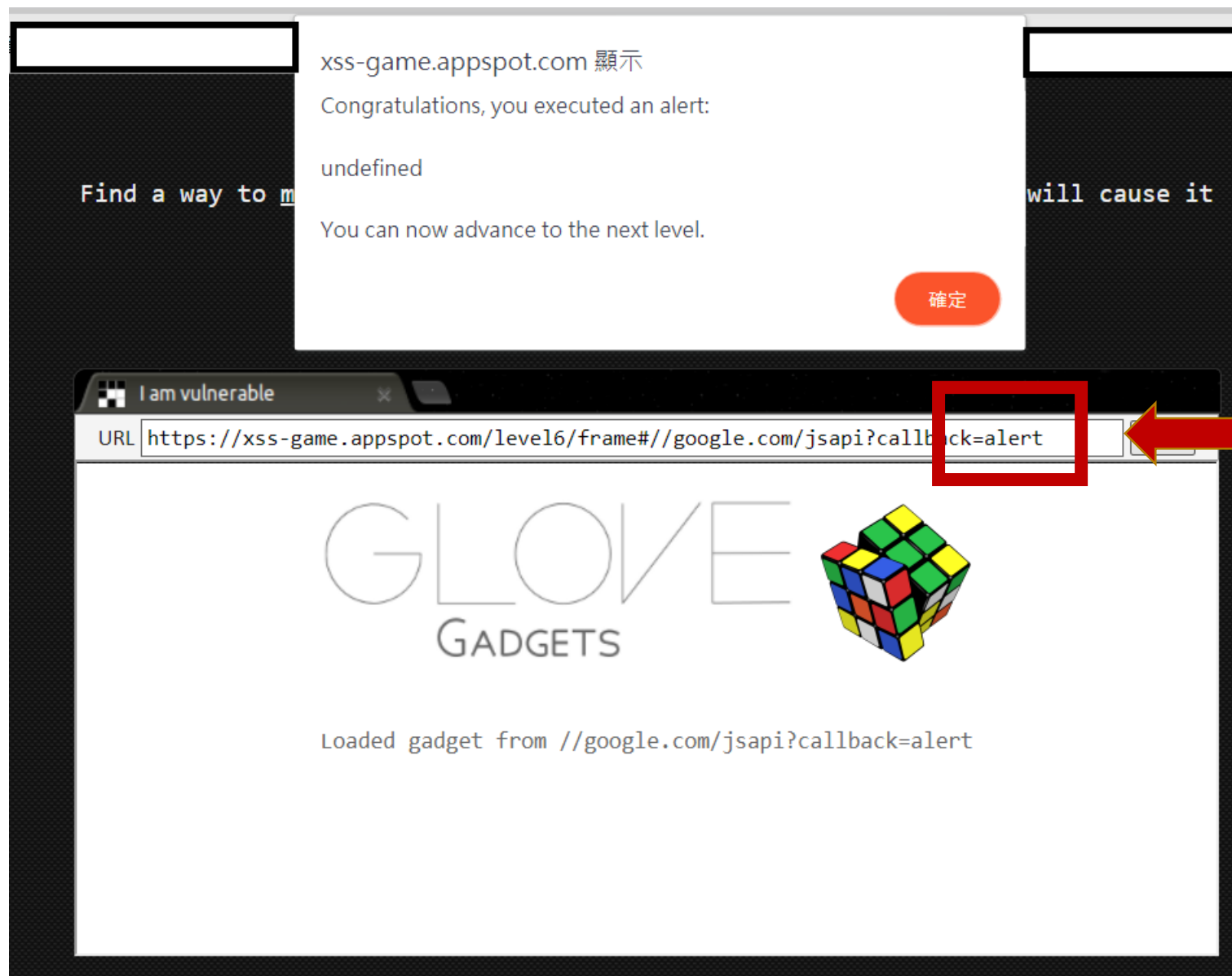
他請你訪問一個 url

網頁不能 load http



去掉 http 後還是沒東西，但可以知道 = 後放一個參數





= 後參數改成 alert
大功告成!!!!



簽到表單