# Se protéger des DOM Based XSS avec les Trusted Types

**Meetup OWASP Paris**

8 novembre 2021

There is a better way



OCTO Part of Accenture © 2021 - All rights reserved

# Sommaire





OCTO Part of Accenture © 2021 - All rights reserved

01

# DOM Based XSS





OCTO Part of Accenture © 2021 - All rights reserved

# L'origine des DOM Based XSS

Un code **JavaScript**, exécuté dans un **navigateur**,
peut créer une vulnérabilité **XSS**
lorsqu'il **insère** des **données** dans la **page**.

Ce type de XSS est appelé **DOM-based XSS** or **DOM XSS**, mais **browser-based XSS** serait peut être plus approprié.



OCTO Part of Accenture © 2021 - All rights reserved

# Exemple de DOM Based XSS

```
1   <html>
2
3   <body>
4     <h1>DOM Based XSS Demo</h1>
5     <div id="data" class="element"></div>
6
7     <script>
8       const xss = `Hello!<img src="none" onerror="alert('This data has become code!')">`;
9       let div = document.getElementById("data");
10      div.innerHTML = xss;
11    </script>
12  </body>
13
14  </html>
```

Cet exemple est simplifié pour la présentation.
Vous pouvez retrouver l'exemple complet sur Stackblitz :
https://stackblitz.com/edit/auth0-guestblog-trustedtypes-xss



OCTO Part of Accenture © 2021 - All rights reserved

# Injection Sinks

- **HTML injection sinks:**
  - `Element.innerHTML`
  - `Element.outerHTML`
  - `Document.write`
  - …

- **DOM XSS injection sinks**
  - Setters for `HTMLScriptElement.src`
  - Setters for `HTMLScriptElement.text`
  - Functions that execute code directly like `eval`
  - Navigation to '`javascript:`' URLs
  - …

https://w3c.github.io/webappsec-trusted-types/dist/spec/#injection-sinks



OCTO Part of Accenture © 2021 - All rights reserved

# Comment s'en premunir ?

1. Utiliser les **API sécurisées** (mais cela est parfois impossible)

2. Activer les **Trusted Types**



OCTO Part of Accenture © 2021 - All rights reserved

# Exemple avec une API sécurisées

```
1   <html>
2
3   <body>
4     <h1>DOM Based XSS Demo</h1>
5     <div id="data" class="element"></div>
6
7     <script>
8       const xss = `Hello!<img src="none" onerror="alert('This data has become code!')">`;
9       let div = document.getElementById("data");
10      div.textContent = xss;
11    </script>
12  </body>
13
14  </html>
```



OCTO Part of Accenture © 2021 - All rights reserved

02

# Trusted Types





OCTO Part of Accenture © 2021 - All rights reserved

# Trusted Types

- Standard en cours de définition par le W3C (Draft, 8 October 2021)
- Compatibilité:

| IE | Edge * | Firefox | Chrome | Safari | Opera | Safari on iOS * | Opera Mini * | Android Browser * | Opera Mobile * |
|---|---|---|---|---|---|---|---|---|---|
| | 12-81 | | 4-81 | | 10-68 | | | | |
| 6-10 | 83-94 | 2-93 | 83-94 | 3.1-14.1 | 69-80 | 3.2-14.8 | | 2.1-4.4.4 | 12-12.1 |
| 11 | 95 | 94 | 95 | 15 | 81 | 15 | all | 95 | 64 |
| | | 95-96 | 96-98 | TP | | | | | |

https://caniuse.com/trusted-types



OCTO Part of Accenture © 2021 - All rights reserved

# Activer les trusted-types

```
<head>
    <meta
        http-equiv="Content-Security-Policy"
        content="require-trusted-types-for 'script'">
</head>
```



OCTO Part of Accenture © 2021 - All rights reserved

# Type Error


Le navigateur refusera toute injection de simple `string` dans un `sink`.
Seul un `trusted-type` sera autorisé.





OCTO Part of Accenture © 2021 - All rights reserved

# Trusted Types

- TrustedHTML
- TrustedScript
- TrustedScriptURL

Le type doit être approprié au contexte !



OCTO Part of Accenture © 2021 - All rights reserved

# Trusted Types policy

```
<script>
  if (window.trustedTypes && trustedTypes.createPolicy) {
    trustedTypes.createPolicy('default', {
      createHTML: string => DOMPurify.sanitize(string, {RETURN_TRUSTED_TYPE: true})
    });
  }
</script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/dompurify/2.2.7/purify.min.js"></script>
```



OCTO Part of Accenture © 2021 - All rights reserved

03

# Bilan



OCTO Part of Accenture © 2021 - All rights reserved

# Bilan

- Inconvenients
  - Standard n'est pas encore validé
  - Peu supporté par les navigateurs (seulement chromium based)
  - Complexe à implémenter avec un framework front (react, vuejs, ...)
  - Rappel le X-XSS-PROTECTION qui fut un échec et dont le support fut retiré de chromium
- Avantages:
  - Renforce la protection contre les DOM Based XSS
  - Supporté par Angular depuis la version 11



OCTO Part of Accenture © 2021 - All rights reserved

Source de cette presentation:
https://auth0.com/blog/securing-spa-with-trusted-types/



OCTO Part of Accenture © 2021 - All rights reserved




OCTO Part of Accenture © 2021 - All rights reserved