

Cours de Sécurité Web ZZ2F5

Le navigateur

Définition

Un navigateur web est un logiciel conçu pour consulter et afficher le World Wide Web. Techniquement, c'est au minimum un client HTTP.

Pour en savoir plus

- ▶ [Navigateur web](#) – Wikipedia
- ▶ [moz://a](#) > L'histoire des navigateurs web

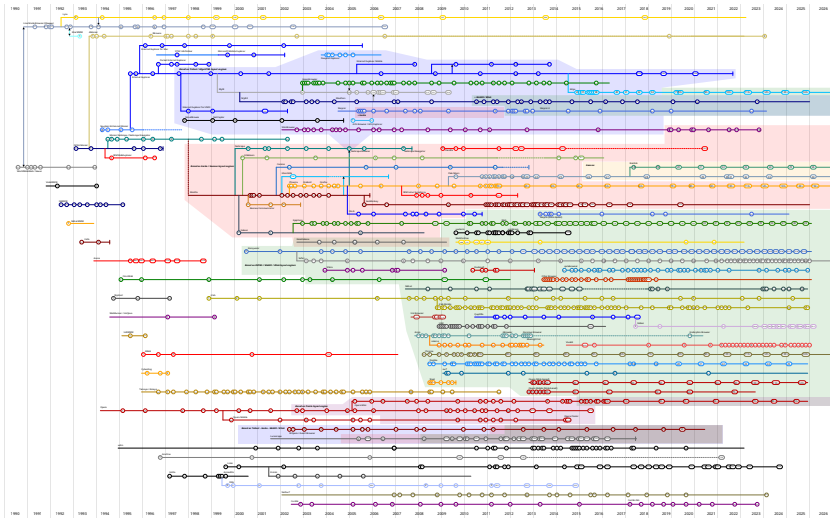
Point clé pour la sécurité

Dans la plupart des cas un navigateur embarque un **interpréteur JavaScript** : ce qui induit quelques garde-fous. . .



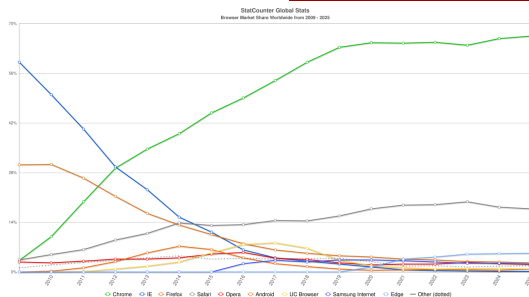
Part d'utilisation des navigateurs – les plus utilisés : Chrome (Chromium/Blink), Safari (WebKit), Edge (Chromium/Blink), Firefox (Gecko).

Arbre généalogique



Beaucoup d'évolution et de grandes familles

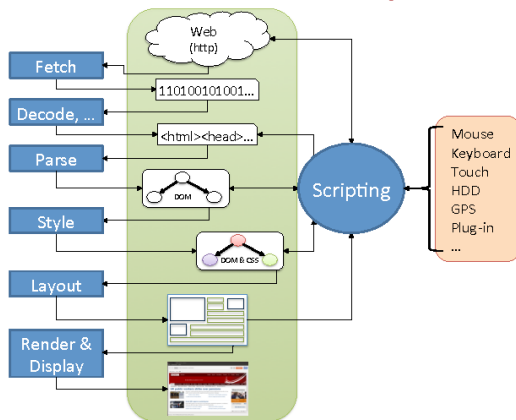
Part de marché (desktop)



Conséquences pour la sécurité

- ▶ la **domination de Chromium** (Chrome + Edge + Opera...) $\Rightarrow$ une faille commune affecte *un très grand nombre d'utilisateurs*
- ▶ les moteurs de rendu et JS sont les cibles privilégiées des *exploits*
- ▶ la sécurité du navigateur dépend de la *mise à jour* de ces moteurs

Les composants d'un navigateur



- ▶ **UI** : barre d'adresse, onglets, boutons. . .
- ▶ **moteur de rendu** (Blink, WebKit, Gecko) : HTML/CSS → pixels
- ▶ **interpréteur JS** (V8, SpiderMonkey, JavaScriptCore)
- ▶ **composant réseau** : les requêtes HTTP(S)
- ▶ **stockage** : cookies, localStorage, cache

Pourquoi plusieurs processus ?

- ▶ **isolation** : un onglet corrompu ne doit pas faire tomber tout le navigateur
- ▶ **sécurité** : le contenu non fiable (web) est isolé du reste
- ▶ **réactivité** : le rendu ne bloque pas l'interface

En pratique (Chromium, Firefox)

- ▶ un processus *browser* (UI, réseau, stockage) – *privilegié*
- ▶ un ou plusieurs “*processus renderer*” par **site** (sandboxés)
- ▶ des processus utilitaires (GPU, réseau, extensions)

Notion de sandbox

le code des pages web s'exécute dans une *bac à sable* : en cas de compromission, l'attaquant ne gagne a priori que le contrôle du “*processus renderer*”, pas de la machine. 7/9

Le code web est non fiable : il doit être contenu

- ▶ **moteur de rendu** : le HTML/CSS ne doit jamais pouvoir lire le disque, ouvrir des ports, ou accéder à d'autres sites
- ▶ **moteur JS** : l'API exposée au script est *restreinte* (pas de fs, pas de socket brut)
- ▶ **sandbox / processus** : en cas de bug, limiter ce que l'attaquant peut faire

Chaque mécanisme de restriction est un mécanisme de sécurité

- ▶ la **Same-Origin Policy** limite l'accès aux données d'un autre site
- ▶ le **CSP** (Content Security Policy) limite ce qu'une page peut charger/exécuter
- ▶ **HttpOnly**, **SameSite**, **Secure** protègent les cookies

⇒ autant de barrières que l'attaquant cherchera à contourner (XSS, CSRF, clique-jacking...).