Les Cookies http #RetourAuxSources

Speakers : Hubert Sablonnière (Clever Cloud)
Format : Conférence
Date : 19 avril 2018
Sites : http://cookies.rocks/ , http://example-foo.com et http://example-bar.com
Code source : https://github.com/hsablonniere/cookies.rocks

Les origines

L'origine des cookies remonte à 1994.
Hubert rend hommage à Lou Montulli qui travaille chez Netscape. Il met en place la 2ième webcam de l'histoire toujours accessible en ligne en 2018 sur http://www.fishcam.com/
Lou est co-créateur du navigateur en mode texte Lynx.
Hubert cite une anecdote sur Lynx : une discussion autour d'un verre a aboutie à la balise <blink> pour faire clignoter du texte.
1 an plus tard, le plugin Java est arrivé dans le navigateur. Il permet d'animer du texte.
Scott et Lou inventent le GIF animés pour accélérer ce type d'animation.

Lou est également l'inventeur des cookies http. Il souhaitait ajouter de l'état dans le navigateur.

Qu'est-ce qu'un cookie http ?

Définition limitée BFM TV : fichier stocké sur le navigateur.
Ne pas confondre les cookies avec la session côté serveur.

Définition pour Devoxxien(ne)s : **protocole pour maintenir l'état entre navigateur et serveur.**

Lors d'une requête http GET, le serveur renvoie une information contenant l'en-tête Set-Cookie : xxxxx
Le navigateur a une jarre à cookie où il les dépose.
Lorsqu'un utilisateur revient sur le site, le navigateur vérifie dans sa jarre si un cookie existe puis, si tel est le cas, l'envoie au serveur.

Depuis 24 ans, le fonctionnement d'un cookie n'a pas changé.
Leurs spécifications restent stables : spécs en 1994 puis 3 RFC en 1997 et 2000 puis 2011.

Combien de temps un cookie est stocké ?

De base, un cookie expire lors de la fermeture du navigateur.
Possibilité de préciser une date ou une durée en seconde.

Comment demander la suppression d'un cookie ?

Il faut préciser une date dans le passé (ex : epoch 01/01/1970).

Comment savoir à quel site web appartient à un site web ?

Information nécessaire pour savoir quand est-ce que le navigateur doit envoyer un cookie à un site.
Réponse : cela dépend de l'URL : protocole, domaine et Top Level Domain (TLD)
Domaines de 1er niveaux courant : .com, .fr, .org

Lorsqu'on dépose un cookie, on peut lui attribuer un domaine.

Domain=cookies.rock
L'attribut domaine= augmente la portée du cookie à tous les sous domaines
Si non spécifié, le cookie est associé au domaine de l'URL.

Comment gérer les .co.uk et reconnaître les TLD ? Problème très vieux (cf. Issue sur Bugzilla).
Liste http://publicsuffix.org cité dans la RFC liste les TLD (12000 lignes)
Les sources de Firefox, Safari et Chrome utilisent la liste

Dans l'URL, on parle désormais de suffixe public et non plus de TLD.

Subtilité de l'attribut Path=
De base, si ignoré, match tous les paths
Path= restreint la portée des cookies avec la subtilité du /
Pas très intéressant à utiliser

2 URL en http et https envoient le même cookie.
Si Man in the Middle et qu'un appel http est envoyé, le cookie part en clair
L'attribut Secure permet de n'envoyer les cookies qu'en HTTPS. Utiliser la fonctionnalité du serveur d'application.

Le header HSTS
Strict-Transport-Security : max-age=86400 ;
Attention à son utilisation. Se référer au site de l'OWASP.

Possibilité : dépôt de cookie dans un nom de domaine sécurisé depuis un domaine non sécurisé. Corrigé dans Firefox et Chrome.

Nouveau brouillon de RFC :
Utiliser le préfixe __Secure
Le navigateur ne déposera le cookie que si le le navigateur et le domaine sont en HTTPS.

Nouveau préfixe : __Host
Set-Cookie : __Host-id=42 ; Secure ; Path=/

Et le port ? Peut-on faire une vérification sur le port ?
Problème du cookie.
Same Origin Policy (SOP) qui s'applique aux cookies.
Origin : combinaison d'un protocole + hôte + port
Si même combinaison, on peut faire confiance. Mais les cookies ne vérifient pas le port.
Il existe bien une tentative de RFC pour pallier à ce problème, mais elle n'ira pas plus loin.

Lorsqu'une requête d'un site A appelle un site B, les cookies du site B sont envoyés. Cela permet des attaques de type CSRF.

Règle de base : ne pas modifier l'état côté serveur sur une requête GET.
Malgré tout, une attaque reste possible en POST.

Dans le nouveau brouillon de RFC 2017, nouvel attribut SameSite=
Permet d'empêcher l'envoi d'un cookie à un autre site.
SameSite=Lax ou Strict

Qui peut lire les cookies ?
L'API document.cookie du navigateur permet de récupérer tous les cookies
L'API navigateur la plus étrange possible. Permet des attaques XSS.
Si n'importe quel script JS s'exécute dans la page, il peut voler des cookies. Exemple d'injection d'une image <img> envoyant tous les cookies à un autre site.
Se référer à la page de l'OWASP car ce sujet est très compliqué.

L'Attribut HttpOnly permet d'empêcher au JavaScript d'accéder au cookie.
En général, c'est mieux de l'avoir.

Alternatives aux cookies ?
window.name : la valeur reste lorsqu'on change de page. Utilisé dans les années 1990.
WebStorage : localStorage et sessionStorage. Expiration à gérer manuellement.

Que dit la CNIL ? Nécessité de laisser aux utilisateurs le choix d'accepter ou non un cookie.

Comment améliorer l'XP utilisateur ?
Démo : requête cross-site sur un SVG envoyée sur cookies.rock. cookies.rock permet de tracer les URL d'un utilisateur + l'heure d'appel.
C'est ce qu'on appelle des cookies tiers.

Dans la navigateur, il y'a une possibilité de bloquer les cookies tiers.
Le header etag (checksum) permet de continuer à tracer l'utilisateur. On contourne le etag pour mettre un identifiant.

C'est quoi le referer ?
Header indiquant l'URL sur lequel on se trouve.
Referrer Policy : permet de ne pas envoyer le Referer ou une sous partie.

Traçace sans JavaScript : ETag, Date, HSTS Pinning, 301 redirect

Possibilité de régler les navigateurs pour bloquer les cookies tiers.
Une fois les cookies tiers bloqués (par défaut dans Safari), d'autres manières de pister arrivent. Jeu de chat à la souris.

Faut-il installer des plugins dans le navigateur ?

Problématique contradictoire : avoir des contenus gratuits mais ne pas vouloir être pisté.