Protéger sites web et téléservices accessibles via HTTP

Octobre 2017 - Mois européen de la cybersécurité

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE DE L'ÉCONOMIE
ET DES FINANCES

MINISTÈRE DE L'ACTION
ET DES COMPTES PUBLICS

TABLE DES MATIERES

MENACES

Interception et altération des communications

Depuis l'éclatement de la « bulle Internet », le web est de facto devenu le canal de déploiement d'applications le plus important, aussi bien en interne vis-à-vis des agents, qu'en externe vis-à-vis des usagers. L'intégrité, la confidentialité, et plus généralement la sécurité de ces sessions sont donc des cibles d'attaques dont les grandes familles ont été identifiées et cataloguées depuis une décennie par l'OWASP (Open Web Application Security Project).

Injection de code et de contenu

Pour les applications web, les attaques d'injection de code (via un composant compromis d'une page) et de cross site scripting (XSS) sont parmi les plus dangereuses [6] parce qu'elles permettent des usurpations de sessions, des escalades de privilèges ou des prises de contrôles à distance des navigateurs.
Au sein du Top10 d'OWASP, elles sont donc rangées respectivement en première et troisième position.

Les tests d'intrusion détectent régulièrement des failles de cette nature qui pourraient être rendues difficilement exploitables avec très peu d'effort par l'utilisation des méthodes de sécurisation entre les serveurs et les navigateurs modernes via l'émission d'entêtes appropriés.

Fondamentalement, les attaquants cherchent le moyen de s'insérer entre le serveur et le navigateur, soit pour injecter du code, soit pour récupérer des informations sur la session, informations que l'attaquant pourra utiliser ou rejouer en back-office pour usurper l'identité de l'utilisateur légitime. C'est donc toute la chaîne de distribution entre le serveur et le client qui doit être sécurisée.

ETAT DE L'ART

De la même façon que les attaquants affinent leurs méthodes d'intrusion, les acteurs de la filière, éditeurs de navigateurs, de serveurs ou d'ateliers de développement ont peaufiné des moyens de lutter contre les attaques, notamment via des entêtes http complémentaires.

HTTPS

Au gré des révélations successives démontrant que des intermédiaires pouvaient manipuler les connexions web pour des raisons diverses (introduction de publicité, piratage et espionnage de toutes natures, surveillance étatique), **la mise en œuvre systématique de la version chiffrée de http** s'est imposée depuis 2010 (entre autres via les efforts de Google ou de Electronic Frontier Foundation [1]) car elle répond directement au besoin de confidentialité mais aussi d'intégrité des communications.

A strict minima, elle est indispensable chaque fois que des informations d'authentification sont échangées. Les objections historiques à son déploiement généralisé étaient centrées sur des questions de performances (mais l'argument ne tient plus avec des générations récentes de processeurs dotés d'instructions de chiffrement / déchiffrement natif d'AES) ou sur le coût des certificats (mais l'argument ne tient plus non plus avec le développement d'offres gratuites comme Let's Encrypt [16]).
A l'inverse, l'emploi de https implique pour le navigateur de faire l'impasse sur un certain nombre de fonctions dangereuses. La balance coûts / avantages s'est donc inversée [2].

Http Strict Transport Security (HSTS)

Pour faciliter leur transition vers HTTPS, certains sites grand public ont présenté dans un premier temps un contenu similaire via HTTP et HTTPS. Cela les exposait néanmoins à des attaques qui, dans un premier temps, visent à dégrader la connexion vers la version non chiffrée, puis dans un deuxième temps, à la manipulation de contenus.

En 2010, Chrome a introduit le dispositif HSTS **qui permet à un site de déclarer qu'il ne sera accessible qu'en HTTPS.** Cette information est mise en cache (généralement pendant une longue durée) par le navigateur pour détecter les tentatives futures de dégradation. Typiquement le header http inséré ressemble à :

```
Strict-Transport-Security: max-age=31536000
```

Pour éviter les interceptions à la première connexion, Chrome utilise également une liste explicite de pré-déclarations de sites ne devant être appelés qu'en https [4].

Marquage des cookies

Comme les cookies sont essentiellement utilisés comme « témoins de connexion » (c'est d'ailleurs la dénomination administrative officielle [3]), en particulier d'authentification, leur protection est d'une importance capitale pour les applications web.
Dès IE6 (2001, concomitant à Windows XP), les éditeurs ont adopté des drapeaux qui permettent d'indiquer pour les cookies sensibles **qu'ils ne doivent être transportés que sur des connexions https**, et d'autre part **qu'ils ne sont pas manipulables via javascript** mais uniquement pour le moteur interne du navigateur.

```
Set-Cookie : <name>=<value> ; secure ; HttpOnly
```

Autres entêtes

Outre l'approche très structurée de CSP (détaillé ci-dessous), les différents navigateurs ont implémenté des entêtes ad-hoc. Même si leur impact n'est pas toujours clair et inter-opérable, il est recommandé de les implémenter pour profiter des efforts de sécurisation de Microsoft, Google et autres.

```
X-Content-Type-Options:nosniff
X-Frame-Options:SAMEORIGIN
X-Xss-Protection:1; mode=block
```

Content Security Policy (CSP)

Les prémisses de Content Security Policy (CSP) datent de l'époque 2011-2012. Ce standard [7], de portée beaucoup plus large que les marquages de cookies par exemple, vise directement le problème des injections de contenu, code compris. Comme d'habitude, le principe de CSP consiste à pré-déclarer comme liste blanche les origines des composants de la page : media (images, fichiers attachés), Javascript, CSS, fonts et toutes autres ressources.

Fin 2014, Google a annoncé avoir totalement mis Gmail sous contrôle CSP, alors que ce webmail est une application particulièrement complexe [10].

Pour identifier les ressources problématiques, ce standard a aussi mis en place un système de reporting par les navigateurs. Voici un exemple plus complet :

```
Content-Security-Policy : default-src 'self' https;
form-action 'self';
upgrade-insecure-requests; block-all-mixed-content;
referer origin-when-cross-origin;
reflected-xss block;
report-uri https://monsite.report-uri.io/report/<id>
```

Pour faciliter les mises en œuvre, il est aussi possible d'utiliser `Content-Security-Policy-Report-Only` en lieu et place de `Content-Security-Policy` durant une phase de tests ou pour superviser durablement une application.

NB : évidemment pour ne pas dévoyer ce dispositif, il faut se détourner des directives `unsafe-inline` et `unsafe-eval`.

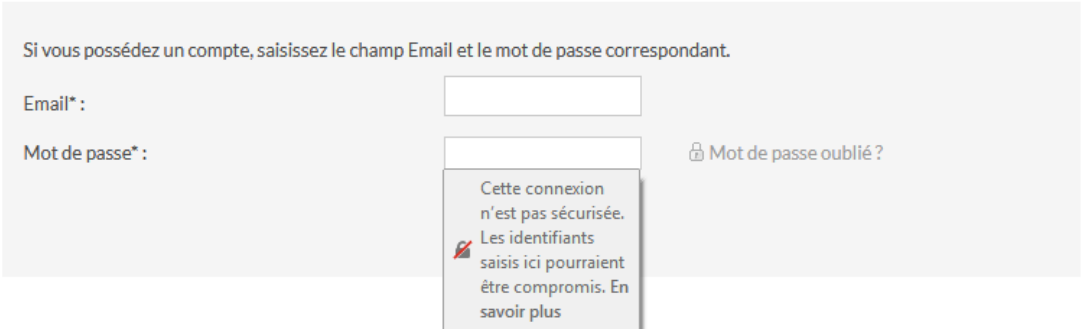
RECOMMANDATIONS PRATIQUES

Les attaquants n'utilisent plus les mêmes techniques qu'en 2010 ou 2000. Il est donc temps pour les organisations de standardiser leurs pratiques, mettre à niveau leurs développements et de déploiements de sites et portails web.

Mettre en œuvre HTTPS et HSTS

Au regard de l'état de l'art, de la tendance de fond dans cette direction, la mise en œuvre de https doit devenir systématique, aussi bien en externe qu'en interne, dès lors qu'une connexion entre machines (client – serveur, mais aussi serveur – serveur) quitte un site d'hébergement unique. Cette règle de sécurisation au niveau applicatif permet de réduire considérablement les hypothèses de sécurité vis-à-vis des réseaux entre le navigateur et les serveurs.

L'objectif d'un emploi large de HTTPS/HSTS, aussi bien pour un site web d'information que pour les téléservices applicatifs, devient d'autant plus nécessaire, si vous ne voulez pas que les utilisateurs s'interrogent sur la sécurisation du site web. Quand les navigateurs détectent par exemple un formulaire sur la page, leurs versions récentes se mettent à déclencher le déclenchement d'alertes variées, dites interstitielles.



A partir du moment où on fait l'effort d'exposer une application web en HTTPS, il n'y pas de raison après quelques mois d'observation de ne pas profiter du surcroît de protection assuré par HSTS.

Activer le marquage des cookies et les entêtes HTTP

Les marquages des cookies ne présentent pas d'options à géométrie variable, et n'ont aucun effet de bord autre que le renforcement de la sécurité. En conséquence, c'est aujourd'hui un dispositif à incorporer systématiquement.
Le déploiement des entêtes HTTP (X-FO, X-CTO, X-XSS) doit faire partie des nouveaux automatismes.

Un marquage `SameSite` est en cours de développement comme puissant frein aux attaques CSRF. Il n'est encore déployé que sur quelques navigateurs (dont celui qui a la plus grosse « part de marché ») et il peut avoir des effets de bord sur des systèmes d'authentification unifié quand il est implémenté dans sa version `strict`. Cet usage

avancé ne peut donc être considéré comme faisant partie de la panoplie standard en 2018 mais dans la foulée des marques HttpOnly et Secure, rajouter `SameSite=Lax` est une précaution pour l'avenir.

Définir une politique CSP

La mise sous contrôle CSP pourrait devenir un exercice complexe pour un journal en ligne (Le Monde, Le Figaro[1]) qui utilisent de nombreux composants tiers : publicités, analytics, animations, etc, réalisés à force d'`iframe`, etc.).

Mais pour des applications professionnelles, une politique minimaliste est un bon point de départ :

```
Content-Security-Policy : default-src 'self'
```

Il existe maintenant de nombreux sites d'information [8,9,11,19, 20] et surtout une extension Firefox [18] comme aides à la formalisation d'une politique adéquate pour une application en ligne ou un site.
De plus, ce dispositif de sécurité peut être mis en place en mode *reporting* avant de passer à un mode blocage. Il est donc recommandé de commencer à s'y mettre en 2018.

Tester

Depuis la première version de ce document (à l'origine interne aux ministères économiques et financiers), les équipes sécurité de Mozilla ont mis sur pied un service web permettant de vérifier la bonne mise en œuvre de la plupart des mesures décrites ci-dessus [17].
Si vous voulez garder des tests confidentiels, la Fondation met à disposition l'outil en Open Source.

Rappels

Avant même de mettre en œuvre les marquages et entêtes cités ci-dessus, il convient de rappeler quelques « nettoyages » élémentaires :

- supprimer les références de version applicative qui révèlent une vieille version ayant des failles référencées ; en particulier les entêtes `Server`, `X-Powered-By` des réponses HTTP, les pages/URL auto-générées `/server-status` (Apache mod_status), `/manager/status` (Apache Tomcat), `/CHANGELOG.txt` (Drupal), etc.
- désactiver le support des méthodes HTTP comme `TRACE` et `TRACK`
- et pour mémoire, SSLv2 et v3 ont été supplantés par TLS et ne sont plus considérés comme fiables (attaque POODLE notamment) ; il convient de supprimer ces versions anciennes de connexions cryptées et même d'avoir un plan d'actions pour supprimer TLSv1.0.

Dans le même cadre de rappel de bon sens, l'insertion des entêtes visant la sécurité est grandement facilitée par l'usage d'un mandataire inverse (reverse-proxy) en

[1] Mais ceux-ci n'ont pas mis en place HTTPS contrairement à des alter-ego internationaux, New York Times, Washington Post, The Times, The Guardian, Die Welt…

frontal du serveur applicatif. Cet équipement (qui peut être logiciel, virtualisé et mutualisé entre serveurs https) fournit également des services WAF (Web Application Firewall), bien utiles pour mettre en place des mesures d'urgence quand une nouvelle faille, générique ou spécifique, est découverte dans le CMS (Content Management System) utilisé par le site.

EN RESUME

En même temps que l'usage du numérique et du web en particulier se diffuse, les technologies murissent.
Depuis 2005, les éditeurs de navigateurs et de serveurs web ont introduit une panoplie de plus en plus complète de dispositifs de protection. Encore faut-il les activer et qu'ils deviennent les « valeurs par défaut ». Le systématisme est grandement facilité par la disponibilité d'outils gratuits de vérification en ligne.
Dès lors, il en est de la responsabilité de chaque organisation.

REFERENCES

[1] https://www.eff.org/https-everywhere
[2] https://scotthelme.co.uk/still-think-you-dont-need-https/
[3] http://www.culture.fr/franceterme/result?francetermeSearchTerme=cookie&francetermeSearchDomaine=0&francetermeSearchSubmit=rechercher&action=search
[4] https://hstspreload.appspot.com/
[5] https://www.owasp.org/index.php/List_of_useful_HTTP_headers
[6] https://fin1te.net/articles/xss-on-facebook-via-png-content-types/
[7] http://www.w3.org/TR/CSP3/
[8] http://www.html5rocks.com/en/tutorials/security/content-security-policy/
[9] https://scotthelme.co.uk/content-security-policy-an-introduction/
[10] http://gmailblog.blogspot.fr/2014/12/reject-unexpected-content-security.html
[11] https://scotthelme.co.uk/migrating-from-http-to-https-ease-the-pain-with-csp-and-hsts/
[12] https://diogomonica.com/2015/12/29/from-double-f-to-double-a/
[13] https://hacks.mozilla.org/2016/02/implementing-content-security-policy/
[14] https://scotthelme.co.uk/csp-cheat-sheet/
[15] https://securityheaders.io/
[15] http://www.ssi.gouv.fr/uploads/IMG/pdf/NP_Securite_Web_NoteTech.pdf
[16] https://letsencrypt.org/
[17] https://observatory.mozilla.org/
[18] https://addons.mozilla.org/en-US/firefox/addon/laboratory-by-mozilla/
[19] https://www.alsacreations.com/article/lire/1723-tour-horizon-https-et-en-tetes-de-securite.html
[20] https://github.com/nico3333fr/CSP-useful/blob/master/README.md

Contact : dssi.shfds (at) finances.gouv.fr
Diffusion Internet : https://www.economie.gouv.fr/hfds/cybersecurite-et-politique-ministerielle-ssi

HFDS Bercy