Engineer

Subresource Integrity

Empreinte de contrôle d'un fichier pour le navigateur

Ce que c'est

Subresource Integrity permet à une page d'annoncer exactement les octets qu'un script ou une feuille de style doit contenir et de faire rejeter le fichier par le navigateur si le contenu reçu diffère. Son abréviation habituelle est SRI. La page inscrit une empreinte cryptographique du fichier dans un attribut `integrity` de l'élément `script` ou `link` qui le charge. À l'arrivée de la réponse, le navigateur applique la même fonction de hachage aux octets reçus et compare le résultat à l'empreinte de l'élément. S'ils sont égaux, le fichier s'exécute ou met en forme la page normalement. S'ils diffèrent, le navigateur traite la réponse comme une erreur réseau : le script ne s'exécute jamais et la feuille de style ne s'applique jamais.

La valeur se compose du nom d'une fonction de hachage, d'un tiret et de l'empreinte en base64. Une commande sur le fichier terminé la produit. L'exécuter deux fois montre combien peu suffit à changer le résultat :

```
$ printf "alert('Hello, world.');" > hello.js
$ openssl dgst -sha384 -binary hello.js | openssl base64 -A
H8BRh8j48O9oYatfu5AZzq6A9RINhZO5H16dQZngK7T62em8MUt1FLm52t+eX6xO
$ echo >> hello.js
$ openssl dgst -sha384 -binary hello.js | openssl base64 -A
qu8P7hCLFcBpas8BOnGmaANCkfXJIpFttDEaxOj0Pq9H6Q8NYjsMb+PVqwuJygVe
```

Précédée de `sha384-`, la première sortie devient la valeur de l'attribut pour ce script d'une ligne. Un seul saut de ligne ajouté à la fin produit une empreinte sans ressemblance avec la première ; un navigateur conservant celle-ci refuserait le second fichier.

La spécification reconnaît trois fonctions de hachage, SHA-256, SHA-384 et SHA-512, toutes de la famille SHA-2. Elle présente SHA-384 comme une bonne base au moment de sa rédaction. Un attribut peut contenir plusieurs empreintes séparées par des espaces. Le navigateur choisit la fonction la plus forte et accepte le fichier s'il correspond à l'une des empreintes produites avec cette fonction. Une page peut ainsi passer à un algorithme plus fort ou accepter deux versions d'une bibliothèque pendant une transition, sans période où toutes ses empreintes seraient incorrectes.

L'attribut existe sur `script` et `link`, et la même valeur peut figurer dans un en-tête de réponse `Link` qui précharge un fichier. Le texte actuel de la spécification est sa deuxième édition, un projet de travail du W3C publié par le Web Application Security Working Group. La première édition est devenue une recommandation du W3C en 2016.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que Subresource Integrity ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Des comptes, mots de passe et droits attribués selon les besoins exacts de chaque poste, puis retirés le jour du départ.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 1 000 règles RBAC pour sécuriser Linux — 1,000 règles d'accès écrites pour combler une faille de sécurité

1 000 règles RBAC pour sécuriser Linux Contactez-nous →

Pourquoi c'est important

Une page qui charge une bibliothèque depuis un réseau de diffusion de contenu, ou depuis un hôte que ses propriétaires n'exploitent pas, lui confie le pouvoir d'exécuter du code sur la page. HTTPS protège le fichier pendant son transport sur le réseau, mais ne dit rien de ce que l'hôte a choisi d'envoyer. Si le stockage du CDN est compromis, si un compte qui y publie est détourné ou si le fichier d'une adresse familière est discrètement remplacé, chaque page qui l'inclut exécute le nouveau code avec tous les pouvoirs de la page. Son propre code n'a aucun moyen de faire la différence.

SRI rend cette décision à la page. Le propriétaire du site calcule l'empreinte d'un fichier qu'il a lu ou construit ; elle est transmise dans la page depuis le serveur du site lui-même. Si un hôte sert autre chose, son fichier est refusé au lieu d'être

exécuté. La page perd une fonction jusqu'à ce que quelqu'un s'en occupe : une panne visible à la cause évidente, plutôt que l'exécution silencieuse du code d'un tiers.

L'empreinte fixe aussi une version. Une adresse comme `/lib/latest.js` peut se mettre à fournir une nouvelle version à une page à tout moment ; l'empreinte ne peut pas la suivre. Une mise à niveau jamais choisie par le propriétaire devient le refus d'un élément précis, signalé par un événement d'erreur.

Il faut décrire précisément les limites de SRI. Il compare un fichier à une empreinte ; il ne juge pas si le fichier est sûr. L'empreinte d'une bibliothèque contenant déjà une ligne hostile protège fidèlement cette ligne. SRI ne s'occupe pas des scripts intégrés directement dans la page : c'est le rôle d'une Content Security Policy. Enfin, il fait confiance à la page qui porte l'empreinte ; celle-ci doit donc toujours arriver intacte.

À quoi cela ressemble en pratique

Ce site l'utilise pour sa propre feuille de style. La page sur la sécurité mentionne Subresource Integrity sur celle-ci : le bundle reçoit une empreinte dans son nom et une empreinte de contrôle lors de la construction, pour que le navigateur du lecteur puisse vérifier qu'il arrive sans modification. Aucun JavaScript exécutable n'est publié et aucune requête ne vise un tiers. Le lecteur qui compte les hôtes dans le panneau réseau du navigateur n'en trouve qu'un, celui du site.

La page des crédits décrit le même fichier côté construction : un bundle CSS unique, concaténé et muni d'une empreinte dans son nom au moment de la construction, puis servi avec une empreinte SRI. Le fingerprinting inscrit un hachage du contenu dans le nom du fichier : une feuille modifiée reçoit une nouvelle adresse, et le navigateur ne prend jamais une ancienne copie en cache pour la version actuelle. La valeur d'intégrité lui permet, elle, de vérifier les octets derrière cette adresse. La construction produit les deux à partir des mêmes octets dans la même étape. Chaque version livre donc ensemble la feuille et son empreinte, ce qui laisse peu de place à la panne la plus courante décrite ci-dessous : un fichier modifié après le calcul de son empreinte.

Où les problèmes surviennent

Un intermédiaire entre la construction et le navigateur réécrit le fichier. Un CDN qui minifie les scripts, un proxy qui insère du contenu dans la réponse ou une étape de déploiement qui réécrit un fichier après calcul de l'empreinte modifie les octets du contenu. Le navigateur refuse alors le résultat, même sans attaque. Une recompression gzip ou Brotli qui préserve le contenu ne provoque pas cet écart : le contrôle porte sur le corps de réponse décodé, pas sur les octets compressés transmis sur le réseau. La spécification confie la responsabilité aux intermédiaires qui modifient le contenu : ils doivent maintenir l'empreinte en accord avec ce qu'ils servent, ou livrer l'original intact. Le serveur d'origine devrait envoyer `Cache-Control: no-transform` pour leur demander de ne pas toucher au fichier. L'ordre fiable consiste à calculer l'empreinte en dernier, sur les octets exacts du contenu décodé que le navigateur vérifiera.

Un fichier d'une autre origine arrive sans CORS. La spécification exige qu'un contrôle d'intégrité sur un fichier d'une autre origine utilise CORS, et qualifie une tentative sans CORS d'erreur logique. L'élément peut par exemple utiliser `crossorigin="anonymous"`, et l'hôte doit répondre avec un en-tête qui autorise le partage du fichier. `crossorigin="use-credentials"` utilise aussi CORS, avec des en-têtes serveur autorisant explicitement l'origine demandeuse et les informations d'authentification. Sans requête CORS et politique serveur correspondante, le navigateur bloque le fichier au lieu de le charger sans contrôle. La règle protège les autres sites : sans elle, un attaquant pourrait charger une réponse privée avec une empreinte devinée et déduire du résultat, par exemple, si le lecteur y est connecté.

L'empreinte arrive en HTTP non chiffré. SRI protège une page contre un hôte de fichiers compromis, jamais contre une personne sur le trajet réseau. Sur une page servie sans HTTPS, toute personne qui peut réécrire le fichier en transit peut aussi modifier ou supprimer l'empreinte dans la page. La spécification recommande donc de ne livrer les métadonnées d'intégrité qu'à un contexte sécurisé.

La fonction de hachage est ignorée par le navigateur. La spécification ne reconnaît que SHA-256, SHA-384 et SHA-512 et recommande aux navigateurs de refuser les fonctions connues comme faibles, telles MD5 et SHA-1. La surprise concerne un jeton extérieur à cette liste : le navigateur l'ignore, et un attribut composé uniquement de jetons ignorés équivaut à l'absence de métadonnées. Le fichier se charge donc sans contrôle. Un outil de construction capable de produire une empreinte MD5, comme Hugo, facilite cette erreur involontaire.

Un fichier refusé fait tomber la page avec lui. Lorsque le contrôle échoue, le navigateur déclenche un événement d'erreur sur l'élément, et le script ou la feuille de style manque simplement à la page. Pour une bibliothèque indispensable au fonctionnement, la spécification suggère d'intercepter l'événement et de charger une copie de repli depuis une source fiable, même si elle est plus lente, comme le serveur du site lui-même. Sans ce repli, la protection fonctionne comme prévu, mais le lecteur reçoit tout de même une page défectueuse.

Seuls les éléments portant une empreinte sont vérifiés. L'attribut d'intégrité d'un script ne dit rien du suivant ajouté sans lui, et examiner les scripts qui en possèdent un ne révélera jamais celui qui en est dépourvu. Le projet de norme ajoute un en-tête de réponse qui traite ce problème côté serveur. `Integrity-Policy` avec `blocked-destinations=(script)` bloque tout script externe arrivant sans métadonnées d'intégrité. `Integrity-Policy-Report-Only` les signale d'abord, pour permettre leur correction avant tout blocage.

Pour aller plus loin

- Subresource Integrity — le projet actuel, avec l'algorithme de vérification, les en-têtes `Integrity-Policy` et les notes sur les proxies, HTTP non chiffré et les fuites entre origines.
- Subresource Integrity, recommandation du W3C de 2016 — la première édition, recommandation stable dont est issu le projet actuel.
- Fetch Standard — l'endroit où les métadonnées d'intégrité d'une requête sont vérifiées sur le corps de réponse, et le protocole CORS dont dépend un contrôle entre origines.
- Hugo : resources.Fingerprint — comment un générateur de site statique calcule l'empreinte d'un fichier à la construction et renvoie la valeur d'intégrité correspondante.
- FIPS 180-4, Secure Hash Standard — la norme fédérale définissant SHA-256, SHA-384 et SHA-512.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu’est-ce que Subresource Integrity ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/sri/