Engineer

Subresource Integrity

Hashwert, mit dem ein Browser eine Datei prüft

Was es ist

Mit Subresource Integrity kann eine Seite vorab genau angeben, welche Bytes ein Skript oder Stylesheet enthalten muss. Der Browser verwirft die Datei, wenn etwas anderes ankommt. Meist wird der Begriff mit SRI abgekürzt. Die Seite schreibt einen kryptografischen Hashwert der Datei in ein `integrity`-Attribut des ladenden `script`- oder `link`-Elements. Sobald die Antwort eintrifft, wendet der Browser dieselbe Hashfunktion auf die empfangenen Bytes an und vergleicht das Ergebnis mit dem Hashwert am Element. Stimmen beide überein, wird die Datei wie gewohnt ausgeführt oder gestaltet die Seite. Unterscheiden sie sich, behandelt der Browser die Antwort als Netzwerkfehler: Das Skript wird niemals ausgeführt, das Stylesheet niemals angewendet.

Der Wert besteht aus dem Namen einer Hashfunktion, einem Bindestrich und dem Hashwert in Base64. Ein Befehl über die fertige Datei erzeugt ihn. Seine zweimalige Ausführung zeigt, wie wenig genügt, um das Ergebnis zu verändern:

```
$ 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
```

Mit vorangestelltem `sha384-` ist die erste Ausgabe der Attributwert für dieses einzeilige Skript. Ein einziger angehängter Zeilenumbruch erzeugt einen Hashwert, der mit dem ersten nichts gemeinsam hat. Ein Browser mit dem ersten Hashwert würde die zweite Datei ablehnen.

Die Spezifikation erkennt drei Hashfunktionen an: SHA-256, SHA-384 und SHA-512, alle aus der SHA-2-Familie. Zum Zeitpunkt ihrer Abfassung bezeichnet sie SHA-384 als gute Ausgangsbasis. Ein Attribut darf mehrere durch Leerzeichen getrennte Hashwerte aufführen. Der Browser wählt die stärkste enthaltene Funktion und akzeptiert die Datei, wenn sie einem der mit dieser Funktion erzeugten Hashwerte entspricht. So kann eine Seite zu einem stärkeren Algorithmus wechseln oder während einer Umstellung zwei Versionen einer Bibliothek akzeptieren, ohne dass zwischenzeitlich nur falsche Hashwerte auf der Seite stehen.

Das Attribut gibt es für `script` und `link`. Derselbe Wert kann auch in einem `Link`-Antwortheader stehen, der eine Datei vorlädt. Der aktuelle Spezifikationstext ist die zweite Ausgabe, ein W3C Working Draft der Web Application Security Working Group. Den Status einer W3C Recommendation erhielt die erste Ausgabe 2016.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Subresource Integrity?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Konten, Passwörter und Berechtigungen werden so geordnet, dass alle genau den beruflich erforderlichen Zugriff haben und ihn am Tag ihres Ausscheidens verlieren.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 1.000 RBAC-Regeln für sichere Linux-Server — 1,000 Zugriffsregeln zum Schließen einer Sicherheitslücke

1.000 RBAC-Regeln für sichere Linux-Server Kontakt aufnehmen →

Warum es wichtig ist

Eine Seite, die eine Bibliothek von einem Content Delivery Network oder einem anderen nicht selbst betriebenen Host lädt, überlässt diesem Host die Möglichkeit, Code auf der Seite auszuführen. HTTPS schützt die Datei auf ihrem Weg durch das Netzwerk, sagt aber nichts darüber aus, welchen Inhalt der Host zur Auslieferung ausgewählt hat. Wird in den Speicher des CDN eingebrochen, ein zur Veröffentlichung berechtigtes Konto übernommen oder die Datei hinter einer vertrauten Adresse unbemerkt ersetzt, führt jede einbindende Seite den neuen Code mit den vollen Befugnissen der Seite aus. Der eigene Code der Seite kann den Unterschied nicht erkennen.

SRI verlagert diese Entscheidung zurück zur Seite. Der Websitebetreiber berechnet den Hashwert aus einer gelesenen oder selbst gebauten Datei. Er wird innerhalb der Seite vom eigenen Server der Website übertragen. Liefert ein Host etwas anderes aus, wird seine Datei abgelehnt statt ausgeführt. Der Seite fehlt dann eine Funktion, bis jemand nachsieht: ein sichtbarer Fehler mit klarer Ursache, statt unbemerkt fremden Code auszuführen.

Der Hashwert legt auch eine Version fest. Eine Adresse wie `/lib/latest.js` kann einer Seite jederzeit eine neue Version liefern; der Hashwert kann ihr nicht folgen. Eine vom Betreiber nie gewählte Aktualisierung führt zur Ablehnung an einem bestimmten Element, gemeldet durch ein Fehlerereignis.

Die Grenzen von SRI sollten genau benannt werden. Es prüft eine Datei gegen einen Hashwert; es beurteilt nicht, ob die Datei sicher ist. Ein Hashwert einer Bibliothek, die bereits eine schädliche Zeile enthält, schützt diese Zeile zuverlässig mit. Inline-Skripte bleiben unberührt; dafür ist eine Content Security Policy zuständig. SRI vertraut außerdem der Seite, die den Hashwert enthält. Die Seite selbst muss deshalb weiterhin unversehrt ankommen.

Wie es in der Praxis aussieht

Diese Website verwendet SRI für ihr eigenes Stylesheet. Die Sicherheitsseite nennt Subresource Integrity für das Stylesheet: Beim Bauen der Website erhält das Bündel einen Fingerprint und einen Hashwert, sodass der Browser des Lesers prüfen kann, ob es unverändert angekommen ist. Es wird kein ausführbares JavaScript veröffentlicht, und es gibt keine Anfragen an Dritte. Wer die Hosts im Netzwerkfenster des Browsers zählt, findet einen: den der Website selbst.

Die Credits-Seite beschreibt dieselbe Datei aus Sicht des Builds: ein beim Build zusammengefügtes CSS-Bündel mit Fingerprint, das mit einem SRI-Hashwert ausgeliefert wird. Der Fingerprint setzt einen Hash des Dateiinhalts in den Dateinamen. Ein geändertes Stylesheet erhält dadurch eine neue Adresse, und der Browser verwechselt eine alte zwischengespeicherte Kopie nicht mit der aktuellen. Der Integritätswert wiederum ermöglicht die Prüfung der Bytes hinter dieser Adresse. Der Build erzeugt beides im selben Schritt aus denselben Bytes. Jede Veröffentlichung liefert deshalb Stylesheet und Hashwert gemeinsam aus; für den unten beschriebenen häufigsten Fehler, eine nach dem Hashen geänderte Datei, bleibt damit wenig Spielraum.

Wo es schiefgeht

Etwas zwischen Build und Browser schreibt die Datei um. Ein CDN, das Skripte minifiziert, ein Proxy, der Inhalte in die Antwort einfügt, oder ein Bereitstellungsschritt, der die Datei nach der Hashberechnung umschreibt, verändert die Inhaltsbytes. Der Browser lehnt das Ergebnis ab, obwohl niemand einen Angriff ausgeführt hat. Eine inhaltserhaltende erneute Komprimierung mit gzip oder Brotli verursacht diese Abweichung nicht: Geprüft wird der dekodierte Antwortinhalt, nicht die über das Netzwerk gesendeten komprimierten Bytes. Die Spezifikation nimmt inhaltsverändernde Zwischenstellen in die Pflicht: Sie müssen den Hashwert an den ausgelieferten Inhalt anpassen oder das Original unverändert liefern. Der Ursprungsserver sollte mit `Cache-Control: no-transform` darum bitten, die Datei unberührt zu lassen. Die verlässliche Reihenfolge lautet, den Hashwert zuletzt zu berechnen, aus genau den dekodierten Inhaltsbytes, die der Browser prüfen wird.

Eine Datei eines anderen Ursprungs kommt ohne CORS an. Die Spezifikation verlangt für die Integritätsprüfung einer Datei eines anderen Ursprungs CORS und bezeichnet einen Versuch ohne CORS als logischen Fehler. Das Element kann beispielsweise `crossorigin="anonymous"` verwenden; der Host muss mit einem Header antworten, der die Datei freigibt. Auch `crossorigin="use-credentials"` verwendet CORS, mit Serverheadern, die den anfragenden Ursprung und Zugangsdaten ausdrücklich erlauben. Ohne CORS-Anfrage und passende Serverrichtlinie blockiert der Browser die Datei, statt sie ungeprüft zu laden. Diese Regel schützt andere Websites: Ohne sie könnte ein Angreifer eine private Antwort mit einem geratenen Hashwert laden und aus dem Ergebnis etwa ableiten, ob der Leser dort angemeldet ist.

Der Hashwert kommt über unverschlüsseltes HTTP an. SRI schützt eine Seite vor einem kompromittierten Dateihost, niemals vor jemandem auf dem Netzwerkpfad. Wird eine Seite ohne HTTPS ausgeliefert, kann jeder, der die Datei unterwegs umschreiben kann, auch den Hashwert in der Seite umschreiben oder löschen. Deshalb empfiehlt die Spezifikation, Integritätsmetadaten nur in einem sicheren Kontext auszuliefern.

Der Browser überspringt die verwendete Hashfunktion. Die Spezifikation erkennt nur SHA-256, SHA-384 und SHA-512 an und empfiehlt Browsern, bekanntermaßen schwache Funktionen wie MD5 und SHA-1 abzulehnen. Überraschend ist der Umgang mit einem Token außerhalb dieser Liste: Der Browser ignoriert es. Ein Attribut, das nur aus ignorierten Tokens besteht, gilt als

völlig ohne Metadaten, sodass die Datei ungeprüft geladen wird. Ein Buildwerkzeug, das wie Hugo Fingerprints mit MD5 erzeugen kann, macht eine solche versehentliche Auslieferung leicht.

Eine abgelehnte Datei legt die Seite mit lahm. Scheitert die Prüfung, löst der Browser ein Fehlerereignis am Element aus, und das Skript oder Stylesheet fehlt schlicht. Für eine Bibliothek, ohne die die Seite nicht funktioniert, schlägt die Spezifikation vor, das Ereignis abzufangen und eine Ersatzkopie von einer vertrauenswürdigen, gegebenenfalls langsameren Quelle wie dem eigenen Server zu laden. Ohne eine solche Kopie arbeitet der Schutz wie vorgesehen, und der Leser erhält trotzdem eine defekte Seite.

Nur Elemente mit Hashwert werden geprüft. Ein Integritätsattribut an einem Skript sagt nichts über das nächste aus, das jemand ohne Attribut hinzufügt. Die Prüfung der Skripte mit Attribut wird das Skript ohne Attribut nie aufdecken. Der Entwurf ergänzt einen Antwortheader, der dies serverseitig abdeckt. `Integrity-Policy` mit `blocked-destinations=(script)` blockiert jedes externe Skript ohne Integritätsmetadaten. `Integrity-Policy-Report-Only` führt sie zunächst auf, damit sie korrigiert werden können, bevor etwas blockiert wird.

Weiterführende Lektüre

- Subresource Integrity — der aktuelle Entwurf mit dem Prüfalgorithmus, den `Integrity-Policy` -Headern und Hinweisen zu Proxys, unverschlüsseltem HTTP und ursprungsübergreifenden Informationslecks.
- Subresource Integrity, W3C Recommendation 2016 — die erste Ausgabe, die stabile Recommendation als Grundlage des Entwurfs.
- Fetch Standard — die Prüfung der Integritätsmetadaten einer Anfrage gegen den Antwortinhalt und das CORS-Protokoll, von dem eine ursprungsübergreifende Prüfung abhängt.
- Hugo: resources.Fingerprint — wie ein statischer Websitegenerator beim Build einen Fingerprint einer Datei erzeugt und den zugehörigen Integritätswert zurückgibt.
- FIPS 180-4, Secure Hash Standard — der US-Bundesstandard, der SHA-256, SHA-384 und SHA-512 definiert.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist Subresource Integrity?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/sri/