Engineer

Subresource Integrity

хеш, за яким браузер перевіряє файл

Що це таке

Subresource Integrity дає сторінці змогу заздалегідь указати, які саме байти має містити скрипт чи таблиця стилів, і змушує браузер відкинути файл, якщо надійшло щось інше. Назву зазвичай скорочують до SRI. Сторінка записує криптографічний дайджест файлу в атрибут `integrity` на елементі `script` або `link`, який цей файл завантажує. Коли приходить відповідь, браузер проганяє отримані байти через ту саму хеш-функцію й порівнює результат зі значенням на елементі. Якщо вони збігаються, скрипт виконується або стилі застосовуються як завжди. Якщо різняться, браузер вважає відповідь мережевою помилкою: скрипт так і не виконується, а таблиця стилів так і не застосовується.

Значення складається з назви хеш-функції, дефіса й самого дайджесту в base64. Його дає одна команда над готовим файлом, а два її запуски показують, як мало треба, щоб відповідь змінилася:

```
$ 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
```

Із `sha384-` попереду перший результат і є значенням атрибута для цього однорядкового скрипту. Один доданий у кінці символ нового рядка дає дайджест, який нічим не схожий на перший, і браузер, що тримає перше значення, відкине другий файл.

Специфікація знає три хеш-функції, SHA-256, SHA-384 і SHA-512, усі з родини SHA-2, і на момент написання називає SHA-384 добрим базовим вибором. Атрибут може містити кілька дайджестів через пробіл. Браузер обирає серед них найсильнішу функцію й приймає файл, якщо він збігається з будь-яким дайджестом, обчисленим цією функцією. Так сторінка може перейти на сильніший алгоритм або під час переходу приймати дві версії бібліотеки, і жодної миті не нестиме самі лише хибні значення.

Атрибут є в `script` і `link`, а те саме значення може їхати в заголовку відповіді `Link`, який попередньо завантажує файл. Чинний текст специфікації — її друге видання, робочий проєкт W3C від робочої групи Web Application Security. Перше видання стало рекомендацією W3C у 2016 році.

Застосуйте поняття до рішення

Застосуйте «Що таке Subresource Integrity?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Облікові записи, паролі й права налаштовані так, щоб доступ відповідав роботі, а звільнені втрачали його того ж дня.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Написала 1 000 правил RBAC для Linux-серверів — 1,000 правил доступу написано, щоб усунути вразливість

Написала 1 000 правил RBAC для Linux-серверів Зв'язатися з нами →

Чому це важливо

Сторінка, яка вантажить бібліотеку з мережі доставки контенту чи з будь-якого хоста, яким її власники не керують, віддає цьому хосту владу виконувати код на сторінці. HTTPS захищає файл дорогою через мережу, але нічого не каже про те, що хост вирішив надіслати з самого початку. Якщо зламають сховище CDN, захоплять обліковий запис, з якого туди публікують, або тихо підмінять файл за звичною адресою, кожна сторінка, що його підключає, виконає новий код з усіма повноваженнями самої сторінки. Власний код сторінки ніяк не відрізнить одне від іншого.

SRI повертає це рішення сторінці. Дайджест обчислює власник сайту над файлом, який він прочитав або зібрав, і цей дайджест подорожує всередині сторінки, з власного сервера сайту. Хост, що віддає щось інше, отримує відмову замість виконання. Сторінка втрачає якусь функцію, доки хтось не погляне, і це помітна несправність з очевидною причиною, а не чужий код, що тихо працює.

Дайджест також фіксує версію. Адреса на кшталт `/lib/latest.js` будь-якої миті може почати віддавати нову версію під сторінкою, а дайджест за нею не піде. Оновлення, якого власник не обирав, перетворюється на відмову для одного конкретного елемента, про яку повідомляє подія error.

Варто точно розуміти, чого SRI не робить. Вона перевіряє файл щодо дайджесту, але не судить, чи безпечний цей файл. Дайджест бібліотеки, у якій уже є ворожий рядок, сумлінно захищає цей ворожий рядок. Вбудованих скриптів вона не торкається, це робота Content Security Policy. І вона довіряє сторінці, яка несе дайджест, тож сама сторінка все одно має дійти без змін.

Як це виглядає на практиці

Цей сайт застосовує її до власної таблиці стилів. Сторінка безпеки називає Subresource Integrity для таблиці стилів: під час збірки бандлу присвоюється відбиток і обчислюється хеш, тож браузер читача може перевірити, що він надійшов без змін. Виконуваний JavaScript не публікується, запитів до третіх сторін немає. Читач, який порахує хости на вкладці мережі в браузері, знайде один, власний хост сайту.

Сторінка подяк описує той самий файл з боку збірки: один CSS-бандл, який під час збірки склеюється з окремих файлів, отримує хеш вмісту в назві й віддається разом із хешем SRI. Відбиток вкладає хеш вмісту файлу в його назву, тож змінена таблиця стилів отримує нову адресу, і браузер ніколи не сплутає стару копію з кешу з чинною. Значення integrity водночас дає браузеру перевірити байти за цією адресою. Збірка створює і те, і те з тих самих байтів на одному кроці, тож кожен реліз відправляє таблицю стилів разом із її дайджестом, і найпоширеніша помилка з описаних нижче, файл, змінений після хешування, майже не має де статися.

Де виникають проблеми

Щось між збіркою та браузером переписує файл. CDN, що мініфікує скрипти, проксі, що вставляє щось у вміст відповіді, або крок розгортання, що переписує файл після обчислення дайджесту, змінює байти вмісту, і браузер відкидає результат, хоча ніхто нічого не атакував. Повторне стискання gzip або Brotli зі збереженням вмісту не спричиняє такої невідповідності: перевірка використовує декодоване тіло відповіді, а не стиснені байти, передані мережею. Специфікація покладає обов’язок на

посередників, які змінюють вміст: вони мають узгоджувати дайджест із тим, що віддають, або доставляти оригінал недоторканим, а серверу, що віддає файл, варто надсилати `Cache-Control: no-transform`, щоб попросити їх не чіпати його. Надійний порядок — брати дайджест останнім, саме з тих байтів декодованого вмісту, які перевірятиме браузер.

Файл з іншого походження приходить без CORS. Специфікація вимагає, щоб перевірка цілісності файлу з іншого походження йшла через CORS, і називає спробу без нього логічною помилкою. Наприклад, елемент може використовувати `crossorigin="anonymous"`, а хост має відповісти заголовком, який дозволяє поділитися файлом. `crossorigin="use-credentials"` також використовує CORS, із заголовками сервера, які явно дозволяють походження запиту й передавання облікових даних. Без CORS-запиту та узгодженої політики сервера браузер блокує файл, а не завантажує його без перевірки. Це правило захищає інші сайти: без нього зловмисник міг би завантажити приватну відповідь із вгаданим дайджестом і з результату дізнатися, наприклад, чи читач там увійшов в обліковий запис.

Дайджест приходить звичайним HTTP. SRI захищає сторінку від скомпрометованого хоста з файлами, але ніколи від когось на мережевому шляху. На сторінці без HTTPS той, хто може переписати файл дорогою, так само може переписати або видалити дайджест у сторінці, тому специфікація радить віддавати метадані цілісності лише в безпечний контекст.

Хеш-функція виявляється такою, яку браузер пропускає. Специфікація знає лише SHA-256, SHA-384 і SHA-512 і радить браузерам відкидати відомо слабкі функції, як-от MD5 і SHA-1. Несподіванка в тому, що стається зі значенням поза цим переліком: браузер його ігнорує, а атрибут, що складається лише з проігнорованих значень, вважається таким, що метаданих не має зовсім, тож файл завантажується без перевірки. Інструмент збірки, який уміє робити відбиток через MD5, як-от у Hugo, полегшує випадкову публікацію такого значення.

Відкинутий файл тягне за собою всю сторінку. Коли перевірка не проходить, браузер запускає на елементі подію `error`, і скрипта чи таблиці стилів просто немає. Для бібліотеки, без якої сторінка не працює, специфікація пропонує перехопити подію й завантажити запасну копію з довіреного, хоч і повільнішого джерела, наприклад із власного сервера сайту. Без такого запасу захист спрацьовує як задумано, а читач однаково отримує зламану сторінку.

Перевіряються лише елементи з дайджестом. Атрибут `integrity` на одному скрипті нічого не каже про наступний, який хтось додасть без нього, і перегляд скриптів, що мають дайджест, ніколи не виявить того, у якого його бракує. Проєкт специфікації додає заголовок відповіді, що закриває це з боку сервера. `Integrity-Policy` з `blocked-destinations=(script)` блокує кожен зовнішній скрипт, що приходить без метаданих цілісності, а `Integrity-Policy-Report-Only` спершу лише перелічує такі скрипти, щоб їх виправили до того, як щось заблокують.

Що почитати далі

- Subresource Integrity — чинний проєкт з алгоритмом перевірки, заголовками `Integrity-Policy` і примітками про проксі, звичайний HTTP та витоки між походженнями.
- Subresource Integrity, рекомендація W3C 2016 року — перше видання, стабільна рекомендація, з якої виріс проєкт.
- Fetch Standard — де метадані цілісності запиту звіряються з тілом відповіді, і протокол CORS, від якого залежить перевірка між походженнями.
- Hugo: resources.Fingerprint — як генератор статичних сайтів робить відбиток файлу під час збірки й повертає відповідне значення integrity.
- FIPS 180-4, Secure Hash Standard — федеральний стандарт США, що визначає SHA-256, SHA-384 і SHA-512.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке Subresource Integrity?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/sri/