Engineer

Cookie

ein Wert, den der Browser speichert und an den Server sendet

Was es ist

Ein Cookie, auch HTTP-Cookie genannt, besteht aus einem kurzen Namen und einem Wert. Ein Webserver schickt beides in einem Antwortheader an den Browser, der es mit späteren Anfragen an denselben Server zurücksendet. So bekommt HTTP ein Gedächtnis, obwohl das Protokoll jede Anfrage als unbekannt behandelt: eine Anmeldung, die den nächsten Klick übersteht, ein noch gefüllter Warenkorb an der Kasse, eine einmal gewählte und beibehaltene Sprache.

Der Server setzt jedes Cookie mit einem eigenen `Set-Cookie`-Header. Der Browser schickt alle passenden Cookies in einem einzigen `Cookie`-Header zurück. Hier bilden die ersten drei Zeilen eine Antwort; der Rest ist eine spätere Anfrage desselben Browsers:

```
HTTP/1.1 200 OK
Set-Cookie: __Host-sid=4f2c9a71e0b8; Path=/; Secure; HttpOnly; SameSite=Lax
Set-Cookie: lang=da; Max-Age=31536000; Path=/

GET /account/ HTTP/1.1
Host: shop.example
Cookie: __Host-sid=4f2c9a71e0b8; lang=da
```

Alles nach dem ersten Semikolon ist ein Attribut, also eine Anweisung dazu, wie lange das Cookie aufzubewahren und wohin es zu senden ist. `Expires` oder `Max-Age` legt eine Höchstlebensdauer fest. Fehlen beide, bleibt das Cookie erhalten, bis der Browser die Sitzung für beendet erklärt. `Domain` erweitert es auf eine ganze Domain und deren Subdomains. Ohne dieses Attribut geht das Cookie nur an den Host zurück, der es gesetzt hat. `Path` beschränkt es auf einen Teil der Website. `Secure` hält es von unverschlüsselten Verbindungen fern, `HttpOnly` verbirgt es vor Skripten, die `document.cookie` lesen, und `SameSite` entscheidet, ob es bei von einer anderen Website ausgelösten Anfragen mitgeschickt wird.

Auf dem Rückweg werden nur Namen und Werte übertragen. Der Server kann daraus deshalb weder das Ablaufdatum noch die Verwendung von `Secure` noch den letzten Verfasser des Cookies erkennen.

Die Regeln stehen im 2011 veröffentlichten RFC 6265. Seine unter dem Namen rfc6265bis bekannte Überarbeitung ergänzt `SameSite`, die Namenspräfixe `__Secure-` und `__Host-` und die Anforderung, dass Browser jede Lebensdauer auf höchstens 400 Tage begrenzen. Zum Zeitpunkt des Schreibens ist sie genehmigt und wartet darauf, vom RFC Editor als Ersatz für RFC 6265 veröffentlicht zu werden.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist ein Cookie?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Eine schnelle, auffindbare Website, die Ihre Mitarbeitenden selbst bearbeiten können, ohne für jede Textänderung eine Rechnung zu bekommen.

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. 100 Webanwendungen mit HTML/HTML5 entwickelt — 100 entwickelte Webanwendungen

100 Webanwendungen mit HTML/HTML5 entwickelt Kontakt aufnehmen →

Warum es wichtig ist

Meist ist ein Cookie das Einzige, was die Anfrage eines angemeldeten Nutzers von einer anonymen unterscheidet. Nach der Anmeldung steht die Sitzungskennung im Cookie aus Sicht des Servers für den Nutzer: Wer sie vorlegt, erhält Zugriff auf das Konto, ohne erneut nach einem Passwort gefragt zu werden. Die meisten Cookieprobleme entstehen, wenn jemand anderes ein Cookie liest, kopiert oder verwendet.

Letzteres macht der Browser leicht. Er hängt Cookies an jede passende Anfrage an, unabhängig davon, wer sie ausgelöst hat. Ein Formular auf einer anderen Website, das Daten an einen Shop sendet, kommt mit den Cookies des Besuchers an. Der Server sieht eine Anfrage eines angemeldeten Nutzers. RFC 6265 nennt das ambient authority, also eine ohne ausdrückliche Übergabe verfügbare Berechtigung, und benennt die dadurch möglichen Angriffe: Cross-Site Request Forgery und das Confused-Deputy-Problem.

Cookies sind auch das älteste Trackingwerkzeug im Web. Ein auf Tausenden Websites eingebettetes Skript kann ein Cookie seiner eigenen Domain setzen, das der Browser von jeder dieser Websites zurücksendet. Ein Unternehmen kann so die Besuche einer Person über alle diese Websites hinweg zusammenführen. Die Überarbeitung beschreibt diese Nutzung von Drittanbieter-Cookies, hält fest, dass die meisten Browser sie inzwischen blockieren oder partitionieren, und empfiehlt eine so restriktive Richtlinie, wie die Kompatibilität erlaubt.

Das Recht knüpft an die Speicherung an, nicht an die Technik. In der EU erlaubt Artikel 5(3) der ePrivacy-Richtlinie einer Website nur dann, Informationen auf dem Gerät eines Besuchers zu speichern oder dort gespeicherte Informationen zu lesen, wenn nach klarer und umfassender Information eine Einwilligung erteilt wurde. Ausgenommen ist Speicherung, die ausschließlich der Übertragung einer Nachricht dient oder für einen vom Besucher ausdrücklich angeforderten Dienst unbedingt erforderlich ist. Dieser Artikel ist der Ursprung von Einwilligungsbannern und erwähnt Cookies nicht einmal namentlich. Die Leitlinien des Europäischen Datenschutzausschusses zu seinem technischen Anwendungsbereich wenden ihn auch auf Trackingpixel und Trackinglinks an.

Wie es in der Praxis aussieht

Diese Website setzt keine Cookies. Ihre Cookie-Seite erklärt den Grund: Es gibt keine Konten, keinen Warenkorb und nichts, womit Besucher verfolgt werden. Deshalb gibt es kein Einwilligungsbanner, weil es nichts gibt, worin einzuwilligen wäre. Die Seite nennt auch die anderen Stellen, an denen eine Website etwas hinterlassen kann, von `localStorage` bis zum Service Worker, und erklärt, dass auch keine davon verwendet wird.

Der Verzicht auf Cookies verändert, wie eine Website etwas über ihre Leser erfährt. Das Zählen der Leser allein anhand des Zugriffslogs misst die Leserschaft ohne Cookie, Skript oder Pixel aus dem eigenen Log des Webservers. Der `Cookie`-Anfrageheader wird ebenso wie der Autorisierungsheader beim Schreiben aus jeder Zeile entfernt. Das Ergebnis ist absichtlich grob: Weder die Zahl unterschiedlicher Besucher noch der Weg eines Besuchers durch die Website lassen sich ermitteln. Die Seite nennt diese Grenzen direkt neben den Zahlen.

Braucht eine Anwendung tatsächlich ein Cookie, lautet die nützliche Frage, was ihm anvertraut wird. Auf einer Plattform, auf der eine Person für mehrere Organisationen handelt, wird die aktive Organisation im Browser gehalten und in

einem serverseitigen Cookie gespiegelt, damit serverseitig gerenderte Seiten denselben Kontext zeigen. Keine der beiden Kopien entscheidet über Zugriffsrechte. Der Server prüft die Autorisierung bei jeder Anfrage. Wer den Wert ändert, verändert deshalb nur die Anzeige auf dem eigenen Bildschirm. Das ist die sichere Form für jedes Cookie: Der Wert darf bestimmen, was gerendert wird. Die Entscheidung darüber, was jemand tun darf, bleibt beim Server.

Wo es schiefgeht

Ein Skript auf der Seite liest die Sitzung. Ohne `HttpOnly` kann ein von einem Angreifer eingeschleustes Skript `document.cookie` lesen und die Sitzungskennung an ein beliebiges Ziel senden. `HttpOnly` entfernt das Cookie aus dieser API. Es verhindert den Diebstahl, nicht den Missbrauch: Das eingeschleuste Skript kann weiterhin Anfragen senden, und der Browser hängt das Cookie weiterhin an. Deshalb ergänzt `HttpOnly` eine Content Security Policy, statt sie zu ersetzen.

Das Cookie wird über unverschlüsseltes HTTP übertragen. Ein Cookie ohne `Secure` wird bei jeder `http://` -Anfrage an seinen Host mitgesendet und ist für jeden auf dem Netzwerkpfad lesbar. Nach RFC 6265 schützt `Secure` nur die Vertraulichkeit. Ein Angreifer im Netzwerk kann ein sicheres Cookie weiterhin über eine unsichere Verbindung überschreiben. Die Überarbeitung verkleinert diese Lücke, indem sie über unverschlüsseltes HTTP gesetzte `Secure` -Cookies ignoriert und ein Cookie über unverschlüsseltes HTTP ablehnt, das ein sicheres Cookie überlagern würde.

Eine benachbarte Subdomain setzt das Cookie. `Domain=example.com` sendet ein Cookie an jeden Host unter dieser Domain. Jeder dieser Hosts kann ein Cookie für die ganze Domain setzen, das die Hauptwebsite dann wie ihr eigenes liest. Auch `Path` ist keine Sicherheitsgrenze. Das Präfix `__Host-` begegnet diesem Problem: Ein Browser akzeptiert ein solches Cookie nur, wenn es `Secure` ist, `Path=/` hat und kein `Domain` besitzt. Dadurch bleibt es an den einen Host gebunden, der es gesetzt hat.

Eine andere Website löst die Anfrage aus. `SameSite=Strict` beschränkt das Cookie auf Anfragen von derselben Site. `Lax` ergänzt Navigationen auf oberster Ebene mit sicheren Methoden, etwa das Folgen eines Links. `None` sendet es überallhin und wird ignoriert, wenn das Cookie nicht zugleich `Secure` ist. Die Überarbeitung behandelt ein Cookie ohne `SameSite` weitgehend wie `Lax` . Ein

Server, der darauf angewiesen ist, sollte es ausdrücklich setzen. Eine Site besteht für diesen Zweck aus Schema und registrierbarer Domain. `SameSite` hilft deshalb nicht gegen eine feindliche benachbarte Subdomain.

Der Server glaubt dem Wert. Alles in einem Cookie kann von der Person geändert werden, in deren Browser es liegt. RFC 6265 empfiehlt Servern, Cookieinhalte zu verschlüsseln und zu signieren, warnt aber, dass das Cookie auch dann noch für einen Replay-Angriff wiederverwendet werden kann. Das robustere Muster ist eine zufällige Sitzungskennung, die auf serverseitig gespeicherten Zustand verweist und bei der Anmeldung ersetzt wird. Damit ist eine vorher von einem Angreifer platzierte Kennung nach der Anmeldung wertlos. Vor diesem Angriff, der Session Fixation, warnt der RFC.

Der Browser verwirft es. Ein Ablaufdatum ist eine Obergrenze. Browser entfernen Cookies, um ihre Grenzen einzuhalten, und ein Angreifer kann Cookies verdrängen, indem er viele eigene setzt. Server sollten sich nicht darauf verlassen, dass Browser Cookies behalten. Auch das Löschen hat eine Falle: Ein `Set-Cookie` mit einem Ablaufdatum in der Vergangenheit entfernt das Cookie nur, wenn `Path` und `Domain` mit den Werten bei seiner Erstellung übereinstimmen.

Weiterführende Lektüre

- RFC 6265: HTTP State Management Mechanism — der Standard für `Set-Cookie` und `Cookie` mit Attributregeln und einem Sicherheitsabschnitt zu ambient authority, schwacher Integrität und Sitzungskennungen.
- Cookies: HTTP State Management Mechanism (rfc6265bis) — die genehmigte Überarbeitung mit `SameSite`, den Präfixen `__Secure-` und `__Host-`, der Obergrenze von 400 Tagen und einem Abschnitt über Drittanbieter-Cookies.
- Richtlinie 2002/58/EG, konsolidierter Text — die 2009 geänderte ePrivacy-Richtlinie mit Artikel 5(3) zur Speicherung und zum Lesen von Informationen auf dem Gerät eines Nutzers.
- EDSA-Leitlinien 2/2023 zum technischen Anwendungsbereich von Artikel 5(3) — die Anwendung der Regel auf Trackingpixel, Trackinglinks, lokale Verarbeitung und andere Techniken jenseits von Cookies.



Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Arbeitsablauf, die vorhandenen Schnittstellen und die Nutzer, die die Veränderung benötigen. Wir können eine prüfbare Lieferung definieren und die Übergabe so gestalten, dass Ihr Team die gelieferten Bestandteile warten kann.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist ein Cookie?“ 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/cookie/