Engineer

HTTPS

🔑 Dieselben Anfragen über eine verschlüsselte Verbindung

Was es ist

HTTPS ist HTTP, das durch eine TLS-Verbindung gesendet wird. Anfragen und Antworten bleiben dieselben Nachrichten mit denselben Methoden, Statuscodes und Headern. Was sich ändert, ist die Verbindung darunter: Sie ist jetzt verschlüsselt und authentifiziert. Das URI-Schema `https` ist in RFC 9110 definiert, der HTTP-Semantikspezifikation vom Juni 2022. Sie löste RFC 2818 ab, das kurze Dokument vom Mai 2000, das HTTP über TLS erstmals festhielt. Der Standardport ist 443, während unverschlüsseltes HTTP Port 80 verwendet.

RFC 9110 knüpft zwei Regeln an das Schema. Ein Client muss (MUST) bestätigen, dass die vom Server vorgelegte Identität hinreichend zum Host in der Adresse passt. Praktisch muss das Zertifikat diesen Namen abdecken; eine Verbindung zum falschen Namen ist fehlgeschlagen, wie stark ihre Verschlüsselung auch sein mag. Außerdem sind `http://example.com/` und `https://example.com/` unterschiedliche Origins mit getrennten Namensräumen. Aus Sicht des Protokolls teilen sie nur einen Hostnamen, sonst nichts.

HTTPS ist nicht an eine HTTP-Version gebunden. HTTP/1.1 und HTTP/2 laufen über TLS auf TCP. HTTP/3 läuft über QUIC, das den TLS-1.3-Handshake in den Transport einbettet. RFC 9114 schließt aus, damit einen unverschlüsselten `http://`-Server direkt zu erreichen. Die neueste Version kommt deshalb mit HTTPS oder überhaupt nicht.

Was die Verschlüsselung umfasst, zeigt eine Adresse am einfachsten. Bei `https://shop.example/orders/1234?ref=mail` werden der Pfad, die Abfragezeichenfolge, jeder Header einschließlich Cookies und `Authorization` sowie beide Nachrichtenbodys verschlüsselt. Das Netz zwischen Besucher und Server sieht weiterhin IP-Adresse und Port des Servers, den zeitlichen Verlauf und den Umfang des Datenverkehrs und bei den meisten Verbindungen auch

den Hostnamen `shop.example` : Der Browser hat ihn über DNS aufgelöst, und der Client nennt ihn in der ersten TLS-Nachricht, damit eine Adresse viele Websites bedienen kann. HTTPS verbirgt, wonach gefragt wurde, nicht, bei wem.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist HTTPS?“ 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

Für eine öffentliche Website lautet die aktuelle Frage, ob überhaupt noch etwas auf `http://` antworten sollte. Browser treffen inzwischen den größten Teil dieser Entscheidung selbst.

Der erste Hebel ist der sichere Kontext, ein W3C-Begriff für eine über HTTPS oder von der eigenen Maschine ausgelieferte Seite. Browser behalten eine wachsende Zahl von Funktionen sicheren Kontexten vor. Service Worker gehören dazu: `navigator.serviceWorker` ist für sichere Kontexte deklariert. Auf einer über unverschlüsseltes HTTP geladenen Seite fehlt die Eigenschaft deshalb einfach, und mit ihr jede darauf aufbauende Funktion, allen voran das Offline-Caching. Die Spezifikation Secure Contexts empfiehlt außerdem den Autoren neuer Spezifikationen, sensible Funktionen genauso abzusichern. Die Einschränkung ist für neue Entwicklungen damit der Standard statt einer Einzelfallausnahme.

Der zweite Hebel sind gemischte Inhalte. Sobald eine Seite selbst HTTPS verwendet, ordnet der Browser jeden ihrer Abrufe nach den W3C-Regeln für Mixed Content ein. Über `http://` angeforderte Bilder, Audiodateien und Videos werden automatisch auf `https://` umgestellt. Antwortet diese Adresse nicht, schlägt der Abruf fehl. Skripte, Stylesheets und Datenanfragen werden unmittelbar mit einem Netzwerkfehler blockiert. Eine einzige verbliebene unverschlüsselte Adresse in einer Vorlage kann deshalb eine ganze Funktion ausfallen lassen, nicht nur ein Bild.

Damit bleibt für Port 80 wenig übrig. Ein Listener, der jede Anfrage dauerhaft auf denselben Pfad unter `https://` umleitet, hat weiterhin seinen Platz, weil Menschen bloße Namen eingeben und alte Links noch `http://` verwenden. Auch die HTTP-Challenge der Zertifizierungsstelle kommt dort an, wenn ein Zertifikat ausgestellt oder erneuert wird. HSTS lässt einen wiederkehrenden Browser anschließend selbst diese Weiterleitung überspringen. Alles auf Port 80, was über diese Aufgaben hinausgeht, ist eine Seite ohne die genannten Garantien.

Loopback ist die bleibende Ausnahme für gewöhnliche Websites. Die Spezifikation stuft `localhost`, `127.0.0.1` und `::1` als potenziell vertrauenswürdig ein. Eine auf der eigenen Maschine des Entwicklers ausgelieferte Seite erhält deshalb Funktionen sicherer Kontexte ohne Zertifikat. Dieser Komfort für Entwickler verursacht einen der weiter unten beschriebenen Fehler.

Wie es in der Praxis aussieht

Die eigene Website des Unternehmens folgt dieser Regel. Auf dem Produktionshost beantwortet Caddy jede Webanfrage und erneuert das HTTPS-Zertifikat von Let’s Encrypt selbst. Das Zertifikat aktuell zu halten ist damit Aufgabe des Webservers statt eines Menschen. Dieselben Seiten werden als Tor-Onion-Dienst veröffentlicht, dessen Adresse bewusst mit `http://` beginnt. Eine Onion-Adresse kodiert den öffentlichen Identitätsschlüssel des Dienstes. Tor authentifiziert den Dienst und verschlüsselt die Verbindung, bevor HTTP ausgetauscht wird. Deshalb verwendet dieser Spiegel HTTP über Tor. Die Credits-Seite verlinkt auch Qualys SSL Labs, das die laufende HTTPS-Konfiguration für jeden Interessierten bewertet.

Kundenarbeit zeigt die Regel sicherer Kontexte von der anderen Seite. Für eine Plattform, deren Nutzer oft Seeleute und Außendienstmitarbeiter sind, die Mobiltelefone verwenden und häufig offline arbeiten, lieferte das Unternehmen vollständige Progressive-Web-App-Unterstützung. Ein Service Worker speichert Schriftarten, Bilder, Audio, Video und CDN-Ressourcen nach dem Cache-first-Prinzip. Weil Service Worker erst über HTTPS wirklich wie vorgesehen funktionieren, wurden Offline- und Installationsverhalten auf echten Geräten über HTTPS geprüft, statt sie vorauszusetzen. Derselbe Bedarf zeigt sich in den Werkzeugen eines Monorepos mit mehreren Programmiersprachen und 578 Task-Zielen. Die vom Unternehmen geschriebene go-task-Schicht stellt neben den nativen und Docker-Entwicklungsmodi eine HTTPS-Variante für PWA- und Mobiltests bereit.

Wo es schiefgeht

Nach der HTTPS-Umstellung laufen Skripte nicht mehr und einige Bilder verschwinden. Die Seiten enthalten noch `http://`-Adressen, fest in einer Vorlage kodiert, in CMS-Inhalten gespeichert oder mit einem Drittanbieter-Snippet eingefügt. Der Browser blockiert Skripte und Stylesheets und stellt Bilder auf HTTPS um. Diese fallen überall dort aus, wo der andere Host kein HTTPS anbietet. Die Browserkonsole nennt jede blockierte Adresse. Durchsuchen Sie Vorlagen und gespeicherte Inhalte nach dem alten Schema und ersetzen Sie jeden Treffer durch `https://` oder einen zur Website relativen Pfad:

```
grep -rnE '(src|href)="http://' templates/ content/
```

Ein Header `Content-Security-Policy: upgrade-insecure-requests` hilft den Seiten während der Bereinigung, lässt die gespeicherten Adressen aber unverändert.

Die Website leitet auf sich selbst um, bis der Browser aufgibt. Ein CDN oder Load Balancer beendet TLS und reicht die Anfrage über unverschlüsseltes HTTP an die Anwendung weiter. Diese sieht `http`, leitet auf `https` um, der Proxy reicht den neuen Versuch wieder als `http` weiter, und die Schleife endet mit dem Fehler „too many redirects“. Dieselbe Blindheit lässt die Anwendung absolute Links mit `http://` erzeugen. Der Proxy muss das ursprüngliche Schema weiterreichen, entweder im standardisierten Header `Forwarded: proto=https` aus RFC 7239 oder im älteren `X-Forwarded-Proto`. Die Anwendung sollte diesem Header nur vertrauen, wenn er von der eigenen Adresse des Proxys stammt, denn jeder Client kann ihn senden. Überlassen Sie die Weiterleitung anschließend einer einzigen Schicht, normalerweise der mit dem Zertifikat.

Ein über HTTPS gesetztes Sitzungscookie wird unverschlüsselt übertragen. RFC 9110 warnt, dass Cookies die dort gezogene Grenze zwischen den beiden Origins nicht beachten. Nach den Cookieregeln aus RFC 6265 wird ein Cookie ohne das Attribut `Secure` auch bei unverschlüsselten `http://`-Anfragen an denselben Host gesendet. Es genügt, dass ein Besucher den bloßen Namen eingibt. Ebenso genügt ein Angreifer im Netz, der eine beliebige andere unverschlüsselte Anfrage mit einer Weiterleitung auf die `http://`-Adresse der Website beantwortet. Setzen Sie `Secure` bei jedem von der Website ausgegebenen Cookie und ergänzen Sie HSTS, damit der Browser die unverschlüsselte Anfrage gar nicht erst stellt.

Der Service Worker läuft auf dem Laptop und fehlt auf dem Smartphone. Der Entwickler öffnet `http://localhost:3000`, einen sicheren Kontext, und alles funktioniert. Das Smartphone im selben Netzwerk öffnet `http://192.168.1.20:3000`, keinen sicheren Kontext. Deshalb ist `navigator.serviceWorker` undefiniert, eine Bibliothek mit entsprechender Prüfung überspringt die Registrierung, und der Offlinemodus fehlt unbemerkt. Liefern Sie auch den Entwicklungsbuild über HTTPS aus. Caddy etwa stellt für lokale Namen und IP-Adressen Zertifikate aus seiner eigenen lokalen Zertifizierungsstelle aus und versucht, deren Stammzertifikat zum Vertrauensspeicher der Maschine hinzuzufügen. Die Dokumentation warnt, dass dieser Schritt nicht garantiert ist und die Aufnahme in jeden Vertrauensspeicher Aufgabe des Administrators bleibt. Auf einem Testsmartphone muss dieses Stammzertifikat von Hand installiert werden.

Weiterführende Lektüre

- RFC 9110, Abschnitt 4.2.2: das URI-Schema https — Das Schema, sein Standardport und die Pflicht des Clients, die Serveridentität zu prüfen.
- Sichere Kontexte — Welche Seiten als sicher gelten, warum Loopback dazugehört und die Empfehlung, neue Funktionen einzuschränken.
- Gemischte Inhalte — Welche Anfragen einer HTTPS-Seite auf HTTPS umgestellt und welche blockiert werden.
- RFC 7239: Erweiterung zur Weiterleitung von HTTP-Anfragen — Der standardisierte Header, mit dem ein Proxy das vom Besucher verwendete Schema weiterreicht.
- Automatisches HTTPS — Caddys eigene Beschreibung seiner Zertifikate, seiner HTTP-Weiterleitung und seiner lokalen Zertifizierungsstelle.

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 HTTPS?“ 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/https/