Jürgen Schmidt

SSL für Fortgeschrittene

Verschlüsselung mit HSTS und Pinning absichern

Let's Encrypt erzeugt eine SSL-Konfiguration mit sehr guten Voreinstellungen. Allerdings weist Transport Layer Security (TLS) konzeptionelle Probleme auf, über die ein entschlossener Angreifer diese Verbindungen immer noch belauschen kann. Das ist keineswegs nur Theorie und Panikmache: Solche Angriffe wurden in freier Wildbahn beobachtet. Doch mit ein paar Handgriffen kann man sie nahezu unmöglich machen.

Ein grundsätzliches Problem ist die nur optionale Verschlüsselung. Ein hartnäckiger Angreifer wird versuchen, den Aufbau einer verschlüsselten Verbindung zu verhindern und stattdessen eine Klartext-Übertragung der Daten erzwingen. Dazu gibt es Tools wie sslstrip, die aus allen HTTPS-Links das „s" herausoperieren und damit zum Aufruf einer ungesicherten HTTP-Seite führen.

Konkret sieht das so aus, dass ein Anwender etwa in seinem Browser „meinemail.de" eintippt und dort auf den Login-Button klickt, der ihn normalerweise auf die gesicherte Anmeldeseite des E-Mail-Dienstes führt. Wenn da nicht der Angreifer wäre: Der sorgt mit sslstrip heimlich im Hintergrund dafür, dass der Anwender auf einer ungesicherten HTTP-Seite landet und seine Login-Daten und danach die Mails ungesichert im Klartext über die Leitung schickt.

Dagegen schützt auch kein automatischer Redirect des Servers auf seine verschlüsselten Seiten. Denn der kommt – sslstrip sei Dank – beim Browser gar nicht mehr an. Dass die aktuelle Seite nicht verschlüsselt ist, sieht man zwar in der Adresszeile des Browsers. Aber wer überprüft die schon bei jedem Seitenaufruf; Smartphones zeigen sie oft nicht einmal an.

Manchmal genügt es sogar, dass der Angreifer in eine ungesicherte Verbindung zu einer beliebigen Web-Seite einen Link auf ein Bild auf http://meinemail.de einschleust. Der Browser wird versuchen, dieses Bild von dort zu laden und dabei sein Session-Cookie an den Angreifer schicken, das nicht explizit mit dem secure-Flag gesichert wurde. Mit diesem Session-Cookie kann der Lauscher dann die Sitzung seines Opfers übernehmen und etwa dessen Mails lesen.

All diese Angriffe beruhen darauf, dass der Browser ungesicherte HTTP-Verbindungen zum Server aufbaut, obwohl er es gar nicht müsste. Dass etwa die Login-Seite auch – vielleicht sogar ausschließlich – als sichere HTTPS-Seite verfügbar ist, weiß er ja nicht.

Genau da setzt HTTP Strict Transport Security (HSTS) an. Damit erklärt ein Server „sicheremail.de" dem Browser: „Hey Browser, all meine Seiten sind auch verschlüsselt verfügbar. Und das wird auch in den nächsten zwölf Monaten so sein." Mit dieser Information kann der Browser den Angriff verhindern. Denn er wird in den nächsten zwölf Monaten jeden Aufruf von sicheremail.de automatisch auf die HTTPS-Seiten leiten.

Das Verfahren ist seit 2009 standardisiert (RFC 6797). Firefox, Chrome, Opera und Safari nutzen es bereits seit einigen Jahren, um Verbindungen zu Web-Servern besser zu sichern. Microsoft hat immerhin Edge und Internet Explorer 11 mit HSTS ausgestattet; ältere IE-Versionen ignorieren es.

Voraussetzung für den Einsatz von HSTS auf Server-Seite ist lediglich, dass tatsächlich alle Seiten der Domain via HTTPS zur Verfügung stehen. Die Umsetzung ist denkbar einfach. Der Server muss in seine Antworten nur eine zusätzliche Header-Zeile wie

```
Strict-Transport-Security: max-age=31536000;↗
                           includeSubDomains
```

einbauen – aber nur und ausschließlich via HTTPS. Die Lebensdauer des HSTS-Eintrags ist in Sekunden anzugeben; obiger Wert entspricht in etwa einem Jahr. Die Option includeSubDomains bedeutet, dass der HSTS-Eintrag auch für Subdomains wie www.sicheremail.de und chat.sicheremail.de gilt. Wer weitverzweigte Domains betreibt, sollte sich unbedingt erst einen Überblick verschaffen, bevor er versehentlich Spezialseiten ohne HTTPS-Unterstützung lahmlegt.

Zertifikate fixieren

Selbst wenn der Browser verschlüsselte HTTPS-Verbindungen erzwingt, sind hartnäckige Angreifer noch nicht außen vor. Die TLS-Verschlüsselung ist zwar in der Praxis kaum zu knacken – aber es gibt eine elegantere und deutlich weniger aufwendige Möglichkeit, an die Daten im Klartext zu gelangen: Der Angreifer gibt sich als die angewählte Gegenstelle aus.

Bei diesem sogenannten Man-in-the-Middle-Angriff klinkt sich der Angreifer in eine gesicherte Verbindung ein. Da der Browser dann die Schlüssel mit ihm aushandelt statt mit dem richtigen Server, kann der Angreifer in der Folge alles mitlesen. Genau das sollen eigentlich die Unterschriften von Zertifizierungsstellen verhindern. Die versichern dem Browser nämlich, dass der Server, mit dem er gerade spricht, tatsächlich sicheremail.de ist.

Doch die Browser glauben weit über hundert Zertifizierungsstellen, darunter viele amerikanische, aber auch solche der Regierungen von China, Tunesien und Macao. Man darf also getrost davon ausgehen, dass jeder Geheimdienst, der etwas auf sich hält, zumindest ein Intermediate-CA-Zertifikat unter seiner Kontrolle hat. Damit kann er sich beliebige Zertifikate ausstellen, denen die Browser blindlings vertrauen. Die chinesische Regierung und auch afrikanische Regimes wurden bereits dabei erwischt, dass sie Oppositionelle mittels solcher Man-in-the-Middle-Angriffe mit gefälschten Zertifikaten überwacht haben.

Dagegen hilft eine recht neue Technik namens Zertifikats-Pinning. Dabei gibt der Betreiber eines Dienstes dem Browser zusätzliche Informationen zu seinen Zertifikaten. So wissen Chrome und Firefox, dass ein Google-Zertifikat entweder von GeoTrust oder einer Google-CA unterschrieben sein muss. Weist sich ein Server etwa mit einem Google-Zertifikat von einer chinesischen CA aus, schlagen diese Browser Alarm. Genauso wurde die chinesische Regierung in flagranti erwischt. Die Domains von Twitter, Facebook und Dropbox sind ähnlich gesichert.

Zwar passen nicht alle Internet-Server auf diese statischen Pin-Listen der Browser. Aber Sie können Ihrem eigenen Server einen ähnlichen Schutz angedeihen lassen. Analog zu HSTS erklärt der Server mit HTTP Public Key Pinning (HPKP) etwas wie: „Wenn du in der nächsten Woche wieder hier vorbeikommst, wird mein Zertifikat von einer der folgenden CAs unterschrieben sein."

Wenn ein Browser innerhalb des angegebenen Zeitraums erneut auf die Seite zugreifen will, prüft er zuvor die präsentierte Zertifikatskette darauf, ob einer der angegebenen Schlüssel vorkommt. Ist das nicht der Fall, meldet er einen Pinning-Fehler – selbst wenn das Zertifikat von einer der angeblich vertrauenswürdigen CAs unterschrieben und somit eigentlich gültig wäre. Das beherrschen immerhin Chrome, Firefox und Opera.

Auch HPKP setzt ein Admin über einen zusätzlichen Header um:

```
Public-Key-Pins: pin-sha256="YLh1...uihg="; ...;
                 max-age=2592000; includeSubDomains
```

Für erste Experimente empfiehlt es sich, mit einer kurzen Lebensdauer von wenigen Minuten zu starten und diese erst bei reibungslosem Betrieb auf einige Wochen oder höchstens Monate zu erhöhen. Schließlich will man ja einen Schlüsselwechsel in endlicher Zeit über die Bühne bringen. Auch hier ist Vorsicht mit den includeSubDomains geboten.

Mit unbedachten HPKP-Einträgen kann man den Zugang zu einer Seite nahezu unmöglich machen. Haben Browser Pins für eine Domain gespeichert, ist es dem Anwender unmöglich, HTTPS-Webseiten ohne diese aufzurufen. In der Kombination mit einem langlebigen HSTS-Eintrag ist das eine fatale Kombination.

Deshalb muss die Wahl der Pins mit Bedacht erfolgen. Der HPKP-Standard erfordert, dass man man mindestens zwei davon angibt. Damit hat man einen Fallback, wenn ein Zertifikat aus irgendwelchen Gründen nicht mehr zum Einsatz kommen kann. Ignorieren Sie diese Vorgabe nicht leichtfertig, indem Sie einen zufällig ausgewürfelten Pin einsetzen. Besser ist es, mit

```
openssl genrsa -out reserve.key 2048
```

einen Reserve-Schlüssel zu erzeugen, den man als Backup-Pin angibt und anschließend sicher verwahrt. Aus ihm kann man im Fall der Fälle jederzeit ein zum Pinning passendes Zertifikat erstellen.

Nur die eigenen Schlüssel zu pinnen, macht es Dritten unmöglich, Zertifikate auf diese Domain auszustellen. Das klingt sicher, bedeutet, aber, dass man keine Möglichkeit mehr hat zu reagieren, wenn diese Schlüssel aus irgendwelchen Gründen nicht mehr zum Einsatz kommen können.

Gute Admins scheuen das Risiko und werden deshalb eher das Intermediate-Zertifikat ihrer CA pinnen, die damit dann im Zweifelsfall ein neues, zum Pinning passendes Zertifikat ausstellen kann. Erfahrene Webmaster gehen auf Nummer sicher und pinnen lieber gleich die Root-CA ihres Zertifikats, weil diese nicht so schnell wechseln. Und weil der Teufel ein Eichhörnchen ist, wählen sie noch eine zweite CA, die sie auch noch in die Pin-Liste mit aufnehmen.

In einem Header sind beliebig viele Pins erlaubt. Generell gilt: je mehr desto besser. Hat man seine Wahl getroffen, kann man die Hash-Werte zu Schlüsseln in einem Zertifikat, einem Zertifikats-Antrag oder einem Key-File komfortabel mit einem Skript wie hpkp-gen (siehe Link unten) berechnen. Das spuckt gleich komplette HPKP-Header-Zeilen aus, die man in der SSL-Sektion seiner Server-Konfiguration einträgt.

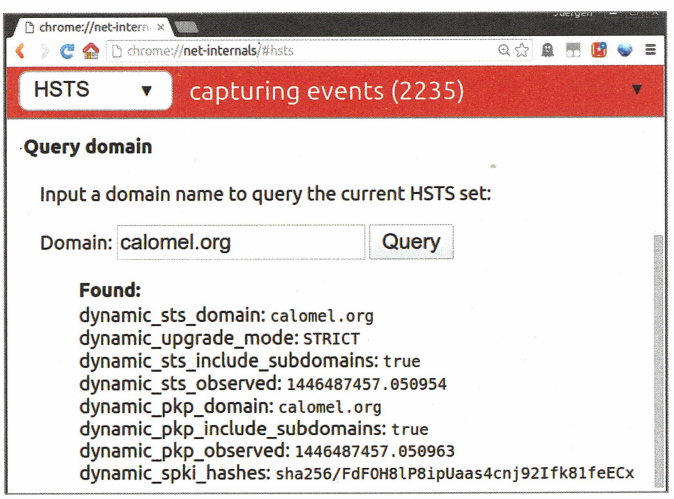
Zum Schluss ein kleiner Dämpfer: Weder HSTS noch HPKP sind wasserdicht. Beide vertrauen darauf, dass der erste Aufruf einer HTTPS-Web-Seite schon irgendwie sicher sein wird (Trust On First Use, TOFU). Außerdem hat das Pinning den Pferdefuß, dass die Browser es abschalten, wenn lokal installierte Root-CA-Zertifikate etwa von Antiviren-Software zum Einsatz kommen. Doch jeder einzelne Server, der Pinning einsetzt, erhöht die Wahrscheinlichkeit, dass gezielte Angriffe auf HTTPS-Verbindungen auffliegen. Das macht MitM-Angriffe für Spione, die im Geheimen operieren müssen, auf Dauer zu gefährlich – und genau das wollen wir doch. (ju@ct.de)

Literatur

[1] Jürgen Schmidt; Festgenagelte Zertifikate, TLS wird sicherer durch Certificate Pinning, c't 23/15, S. 118

[2] Jürgen Schmidt; Sicher mit Pin, Zertifikats-Pinning auf dem eigenen Server, c't 23/15, S. 122

c't Tools für HSTS und HPKP: ct.de/yg7k



Wenn Sie in die Adresszeile chrome://net-internals/#hsts eingeben, zeigt Chrome die gespeicherten HSTS- und HPKP-Einträge einer Domain. Dort können Sie auch selber welche setzen oder löschen.