Bild: Thorsten Hübner

Transportverschlüsselungs-verwirrung

Sichere Cipher-Suites für TLS auswählen

Transportverschlüsselung mit TLS macht das Internet sicherer, muss aber regelmäßig an den Stand der Technik angepasst werden. Die verfügbaren kryptografischen Verfahren werden in sogenannten Cipher-Suites zusammengefasst. Die richtigen Verfahren auszuwählen, ist die Aufgabe von Serverbetreibern. Ein Wegweiser durch den Cipher-Dschungel.

Von Jan Mahn

TLS ist Pflicht, wenn man Dienste im Internet veröffentlicht – das zweifelt heutzutage kaum noch jemand an, auch Websites ohne Login-Bereich und sensible Inhalte werden, anders als noch vor 10 Jahren, heute über HTTPS ausgeliefert. Mit kostenlosen Zertifikaten von Let's Encrypt und automatischer Verlängerung ist TLS heute mit wenigen Handgriffen ganz passabel eingerichtet. Doch wenn man mehr will als „ganz passabel" und tiefer in das Thema einsteigt, wird es kompliziert. In den Einstellungen von Webservern oder Reverse Proxies stößt man bei genauerem Hinsehen auf die Konfiguration der sogenannten Cipher-Suites und merkt: TLS ist nicht gleich TLS und die angebotenen Optionen sind nicht unbedingt selbsterklärend. Ist TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 die Einstellung der Wahl, oder sollte man lieber auf TLS_DHE_DSS_WITH_AES_128_CBC_SHA256 setzen? Qualifiziert beantworten kann man die Frage erst nach einem Ausflug in die Kryptografie. Wenn Sie sich nicht für das Warum interessieren und nur eine sichere Konfiguration kopieren wollen, springen

c't Hardcore

Sie direkt zum Abschnitt „Was bleibt“ in diesem Artikel.

Die folgenden Hinweise gelten nicht nur für Webserver, auch Mailserver (mit IMAP und SMTP) oder Exoten wie MQTT-Server sollten nur mit sicheren TLS-Einstellungen betrieben werden. Hat man viele Dienste zu veröffentlichen, ist man gut beraten, einen Reverse Proxy vorzuschalten, der die TLS-Abwicklung zentral für HTTP und alle anderen Protokolle erledigt. In den Einstellungen der Server oder Reverse Proxys mit TLS-Funktion kann man immer mehrere Cipher-Suites gleichzeitig aktivieren. In einer perfekten Welt würde eine einzige Cipher-Suite ausreichen, am besten die mit den längsten Schlüsseln und den besten Verfahren. Dass man sich als Serverbetreiber überhaupt Gedanken über das Thema machen muss, hängt damit zusammen, dass nicht alle Clients (Mailprogramme, Browser ...) das neueste und beste Verfahren verstehen und auch die Server nicht immer die neueste und beste Technik beherrschen. Teils weil die Entwickler noch nicht die neuesten Krypto-Bibliotheken eingebunden haben, teils weil die Nutzer hoffnungslos veraltete Software einsetzen und Updates ohnehin für ein lästiges Übel halten. Daher kann ein Server, der TLS beherrscht, immer mehrere Verfahren unterstützen und sich mit dem Client auf ein Verfahren einigen. Die Initiative geht vom Client aus, der dem Server mitteilt, welche Verfahren er kennt. Der Server entscheidet dann, welches der angebotenen Verfahren er nutzt. Das Problem: Wenn man neben sicheren auch unsichere Verfahren akzeptiert, um ganz alte Clients nicht auszuschließen, kann ein Angreifer, der als Man-in-the-Middle zwischen Client und Server sitzt, ein verwundbares Verfahren erzwingen. Dafür manipuliert er die Liste der unterstützten Verfahren, die der Client an den Server schickt (sogenannte Downgrade-Attacke). Die Abwägung besteht also darin, möglichst wenige (und nur sichere) Verfahren zuzulassen und gleichzeitig möglichst niemanden auszusperren.

TLS-Versionen

Los geht es mit einer einfachen Entscheidung: TLS ist eine Fortentwicklung des in grauer Vorzeit entstandenen SSL. Aktuell sicher und empfohlen sind die TLS-Versionen 1.2 und 1.3, alle Vorgänger (TLS 1.1 sowie SSL 2 und SSL 3) sollten Sie umgehend abschalten, sie enthalten gravierende – und allgemein bekannte – Sicherheitslücken und Probleme. Leider hindert das auch große Anbieter nicht, lange an den Altlasten hängenzubleiben: Dass sich gerade Mailhoster schwertun, die noch älteren Zöpfe TLS 1.0 und 1.1 abzuschneiden, zeigt eine ausführliche Untersuchung, die wir im Jahr 2021 durchgeführt haben [1].

Relevante Mailclients und Browser, die nicht mindestens TLS 1.2 sprechen, gibt es nicht mehr, selbst der Internet Explorer 11 schafft das; die anderen gängigen Browser seit Versionen etwa aus dem Jahr 2013, sodass Sie auch extrem updatefaule Nutzer nicht ausschließen. Ausschließlich auf TLS 1.3 können Sie im Web noch nicht ganz setzen, sofern Sie Internet-Explorer-Nutzern noch eine Chance geben wollen. Am 15. Juni 2022 schickt Microsoft den Problembrowser in Rente und für HTTPS per TLS 1.2 fällt dann ein Argument weg. Wenn Sie Ihr Geld nicht gerade mit einem Webshop verdienen und auf jeden Kunden angewiesen sind, wie alt sein Gerät auch sein mag, könnten Sie danach anfangen, TLS 1.2 abzuschalten; dringend nötig wird das aber wohl nicht sein, aktuell gilt auch TLS 1.2 als ausreichend sicher – die richtigen Cipher-Suites vorausgesetzt.

Cipher-Suites für TLS 1.2

Ist die Entscheidung für die TLS-Versionen gefallen, geht es ans Auswählen der Cipher-Suites. Eine Suite für TLS 1.2 heißt zum Beispiel TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256, die Bezeichnung folgt dem Schema

```
TLS_<Schlüsseltauschverfahren>_WITH_↵
↳<Verschlüsselungsverfahren>_↵
↳<Nachrichtenauthentifizierung>
```

Drei Komponenten gibt es also, die kombiniert werden können: ein Schlüsselaustauschverfahren, ein Verschlüsselungsverfahren und ein Hash-Verfahren, das in einem Message-Authentication-Code (MAC) genutzt wird, um die Nachricht zu authentifizieren.

Das Schlüsselaustauschverfahren erfüllt zwei Funktionen. Zum einen authentifiziert es den Server gegenüber dem Client. Der weiß so, dass er nicht mit einem fremden Server redet, der sich in den Verkehr gemogelt hat. Zum anderen dient das Verfahren – wie der Name nahelegt – den Parteien dazu, sich auf einen gemeinsamen Schlüssel zu einigen. Das bekannteste solche Verfahren ist der von Whitfield Diffie und Martin Hellmann (auf Basis von Arbeiten von Ralph Merkle) erfundene Diffie-Hellmann-Schlüsselaustausch. Ausführlich beschrieben und mit einem praktischen Beispiel zum Nachrechnen versehen haben wir das Verfahren in [2]. Die Kurzform: Bei einem solchen Schlüsselaustausch generieren zwei Systeme Zufallszahlen, rechnen damit und schicken die Rechenergebnisse über einen unsicheren Kanal. Am Ende einer festgelegten Abfolge von Berechnungen mit den Zahlen des jeweiligen Gegenübers haben beide Teilnehmer dasselbe Geheimnis, ein Beobachter des gesamten Nachrichtenaustauschs hat aber keine Chance, dieses Geheimnis zu erfahren oder selbst zu errechnen. Klingt irgendwie magisch und man hält es meist so lange für unmöglich, bis man es einmal mit kleinen Zahlen nachgerechnet hat.

Dass ein Schlüsselaustausch über eine unsichere Verbindung beim TLS-Verbindungsaufbau nötig ist, leuchtet ein – schließlich soll per TLS ein sicherer Kanal im per se unsicheren weltweiten Internet zustande kommen, und bis der Tunnel steht, können viele Akteure zuhören. Das ausgehandelte Geheimnis nutzen beide im Anschluss, um die Inhalte der Kommunikation mit einem symmetrischen Verschlüsselungsverfahren zu verschlüsseln. Mit dem Schlüsselaustausch allein ist es aber leider nicht getan. Wenn Sie in der Konfiguration Ihres Servers die Zeichenkette „ANON“ finden, herrscht dringender Handlungsbedarf: Diese Suite müssen Sie ersatzlos streichen. An dieser Position im Namen der Cipher-Suite sollte eigentlich ein Verfahren stehen, das den Server bei der Aushandlung authentifiziert. ANON steht für „keine Authentifizierung“ und ist ein Sicherheitsdesaster.

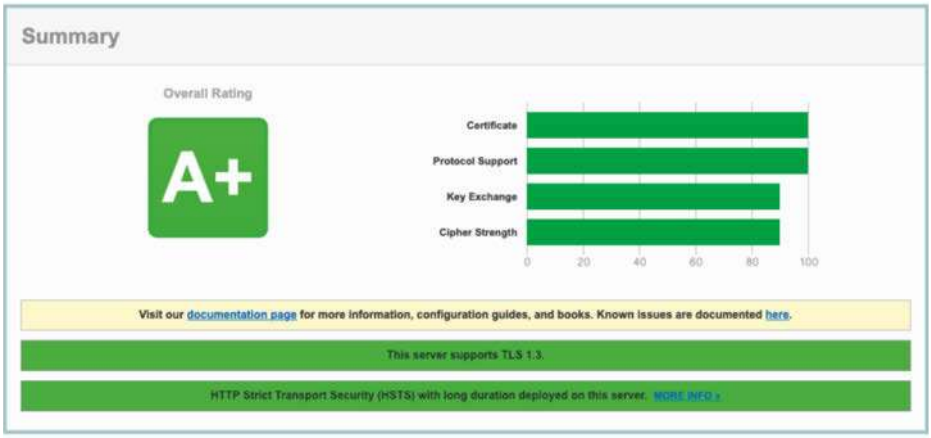
c't kompakt

- Betreiber von Servern, die Dienste per TLS anbieten, müssen sichere Cipher-Suites auswählen.
- Im Laufe der Jahre wurden viele kryptografische Verfahren als unsicher eingestuft. Sie sollte man zügig deaktivieren.
- Mit TLS 1.3 wird vieles einfacher – noch gibt es aber Browser, die den neuen Standard nicht verstehen.

Diffies und Hellmans Verfahren aus den Siebzigern ist zwar bis heute mathematisch sicher, aber war nicht als Schutzmaßnahme gegen ein anderes Problem konzipiert: Man-in-the-Middle-Angriffe. Wenn der Angreifer bei der Schlüsselaushandlung nicht nur lauschen, sondern selbst Nachrichten schicken kann, ist er unter ungünstigen Umständen in der Lage, selbst das gemeinsame Geheimnis mit der Gegenseite auszuhandeln und sich fortan als der berechtigte Empfänger auszugeben. Um das zu verhindern, muss das Zertifikat des Servers in die Aushandlung einbezogen werden. Die Prüfung von Zertifikat und Zertifizierungskette bis zu einer vertrauenswürdigen Stammzertifizierungsstelle ist eine der Kernfunktionen von TLS und wird vom Client gewissenhaft erledigt. Wenn der Schlüssel des Zertifikats genutzt wird, um die Schlüsselaushandlung zu authentifizieren, kann der Client sicher sein, dass er gerade ein gemeinsames Geheimnis mit dem angefragten Server und nicht etwa mit einem zugeschalteten Geheimdienstserver ausgehandelt hat.

Eine Kombination zweier kryptografischer Urgesteine, um per Zertifikat authentifizierten Schlüsselaustausch zu realisieren, heißt in den Cipher-Suites DH_RSA, bestehend aus klassischem Diffie-Hellmann, kombiniert mit RSA (erfunden von Rivest, Shamir und Adleman), dem bekanntesten asymmetrischen Kryptosystem, ausführlich erklärt in [2]. Dass beide Verfahren für IT-Verhältnisse recht alt sind, ist kein Problem. Mit ausreichend langem Schlüssel ist RSA bis heute sicher. Die Kombination DH_RSA bedeutet, dass der Server einen Schlüssel aus dem Zertifikat als Ausgangswert für Diffie-Hellmann nutzt und an den Client schickt. Das Zertifikat wiederum wurde vorab per RSA von einer Zertifizierungsstelle signiert – am Ende der Kette steht ein Stammzertifikat, dem der Client vertraut.

DH_RSA ist nicht unsicher, aber auch nicht das einzige und bei Weitem nicht das beste Verfahren. Welche Cipher-Suites überhaupt empfehlenswert sind, beschreibt das Bundesamt für Sicherheit in der Informationstechnik (BSI) in seiner Technischen Richtlinie TR-02102-2 (siehe ct.de/y2yq), die sich mit der „Verwendung von Transport Layer Security (TLS)“ beschäftigt. Deutsche Unternehmen, Behörden, Vereine und Privatpersonen können mit Fug und Recht angeben, nach Stand der Technik zu arbeiten, wenn sie diesem Dokument folgen.



Der Traum aller TLS-Admins: Ein Rating mit A+ im TLS-Test von Qualys. Dafür muss nicht nur TLS anständig konfiguriert sein.

Außer DH_RSA steht auch DHE_RSA in der BSI-Liste der sicheren Schlüsselaustauschverfahren. Das E steht für „ephemeral“, zu Deutsch flüchtig. Es bedeutet, dass der Server bei jedem Verbindungsaufbau neue Schlüssel für die Diffie-Hellmann-Aushandlung generiert. Damit der Client trotzdem die Authentizität des Servers prüfen kann, signiert der Server die Nachricht, die den frisch generierten DH-Schlüssel enthält, mit dem RSA-Schlüssel aus seinem Zertifikat. Mit den immer neuen DH-Schlüsseln setzt der Server das Konzept „Perfect Forward Secrecy“ (PFS) um, bestenfalls verarbeitet er den privaten Schlüssel nur im RAM und legt ihn nie auf der Festplatte ab. Der Vorteil von PFS liegt auf der Hand: Angenommen, ein Angreifer hat den gesamten Verkehr aufgezeichnet und hofft darauf, dass er irgendwann in den Besitz des privaten Schlüssels kommt (oder in 20 Jahren genug Rechenleistung hat, um ihn zu brechen). Mit aktiviertem PFS ist er aufgeschmissen, auch wenn er irgendwann einen Schlüssel herausfindet, weil jede TLS-Sitzung separat entschlüsselt werden müsste. Die Ephemeral-Varianten sind also in jedem Fall die bessere Wahl und auch die Kompatibilität ist kein Problem. Cipher-Suites mit DH_ (ohne E) im Namen können Sie guten Gewissens aus Ihrer Liste streichen.

RSA kann Ihnen noch an anderer Stelle begegnen: Das asymmetrische Kryptosystem kann man nicht nur nutzen, um einen Diffie-Hellmann-Schlüsselaustausch abzusichern, es kann auch selbst den Schlüsselaustausch übernehmen. In der Cipher-Suite steht es dann direkt an erster Stelle, zum Beispiel in TLS_RSA_WITH_AES_128_CBC_SHA256. Dann arbeitet der Server mit dem RSA-Schlüsselpaar des Zertifikats, also wieder mit einem statischen Schlüssel. Weil das Verfahren keine Perfect Forward Secrecy bietet, kann es ebenfalls raus aus Ihrer Konfiguration.

Das BSI empfiehlt außer den verschiedenen Varianten mit RSA noch Cipher-Suites mit DHE_DSS in seiner Richtlinie TR-02102-2. DSS ist die Abkürzung für „Digital Signature Standard“, der mal zum RSA-Ersatz werden sollte, aber bei TLS keinen Siegeszug antreten konnte. Lohnenswert ist der DSS-Einsatz nicht: Wenn Sie solche Suites einsetzen, gewinnen Sie keine Kompatibilität für weitere Clients, daher können Sie diese ebenfalls rauswerfen.

Mit Ellipse

Kryptografische Verfahren muss man nicht nur anpassen, wenn Sicherheitslücken entdeckt werden, sondern auch, weil die allgemein verfügbare Rechenleistung kontinuierlich wächst. Dadurch stehen auch potenziellen Angreifern immer mächtigere Supercomputer zur Verfügung. Typischerweise reagiert man darauf, indem alle paar Jahre die empfohlenen Schlüssellängen verdoppelt werden. Damit wächst aber auch der nötige Rechenaufwand bei jedem Verbindungsaufbau – und zwar noch schneller als die Schlüssellängen. RSA-Schlüssel mit mehr als 4096 Bit gelten als nicht mehr sinnvoll. Mögen die Berechnungen auf Client-Seite noch machbar sein, macht sich steigende Rechenarbeit beim Server durchaus bemerkbar, der viele TLS-Sitzungen parallel aufbauen muss. RSA-8192 wäre eine ungeheure Vernichtung von Rechenleistung. Einen Ausweg aus der Situation bietet Kryptografie mit elliptischen Kurven (Elliptic Curve Cryptography, ECC). Die ist

auch mit kurzen Schlüsseln sehr sicher, kostet weniger Rechenleistung und beschleunigt so die Abwicklung. Ein ECC-Schlüssel mit 224 Bit gilt als ebenso sicher wie ein 2048-Bit-RSA-Schlüssel. Auf Basis von elliptischer Kryptografie gibt es auch eine Diffie-Hellman-Variante – in den Cipher-Suites erkennt man Elliptic-Curve-Diffie-Hellmann an den Abkürzungen ECDH und ECDHE. Wie beim klassischen DH ist auch hier die ephemere Variante die erste Wahl, ECDH_ mit statischen Schlüsseln kommt also ebenfalls auf die Streichliste.

Verwirrend wird es bei der Wahl des zugehörigen Signaturverfahrens, wenn man ECDHE einsetzt. Man kann das neue Diffie-Hellmann-Verfahren sowohl mit RSA kombinieren als auch mit dem ebenfalls auf elliptischen Kurven basierenden Signaturverfahren ECDSA. Möglich sind also ECDHE_RSA und ECDHE_ECDSA. Entscheidend zu wissen: ECDHE selbst kann man unabhängig vom verwendeten Zertifikat nutzen, weil letzteres mit dem Schlüsselaustausch an sich nichts zu tun hat. Das Signaturverfahren dagegen muss zum eingesetzten Zertifikat passen. Wenn Sie zum Authentifizieren ECDSA statt RSA nutzen wollen, müssen Sie in die Details Ihres Zertifikats schauen, ob das auf elliptischen Kurven basiert; die meisten Zertifikate werden noch immer auf RSA-Basis ausgestellt.

Verschlüsselt und gehasht

Die letzten beiden Entscheidungen bei der Wahl der Suites betreffen das Verschlüsselungs- und Hashverfahren. Die (symmetrische) Verschlüsselung ist der Kern von TLS: Diffie-Hellmann oder seine elliptisch-kurvigen Verwandten werden genutzt, um Schlüsselmaterial auszutauschen. Das wird dann genutzt, um alle Inhalte im TLS-Tunnel symmetrisch verschlüsselt zu übertragen.

Das BSI ist bei der Empfehlung von Verschlüsselungsverfahren konservativ und setzt ausschließlich auf das altbewährte AES (Advanced Encryption Standard) – ein absoluter Krypto-Klassiker und bis heute eine solide Wahl; infrage kommen Schlüssellängen von 128 oder 256 Bit, beide gelten aktuell als sicher. Aber auch hier gibt es Tücken: AES ist eine Blockchiffre, die Daten werden in 128 Bit lange Abschnitte geteilt und nacheinander verschlüsselt. Unternimmt man nichts weiter, werden dieselben Klartextblöcke daher immer zu denselben Geheimtextblöcken. Diese AES-Betriebsart nennt man auch „Electronic Code Book“ (ECB). Anhand von Bilddateien kann man das Problem unverknüpfter Blöcke sehr anschaulich machen – das verschlüsselte Bild ist verzerrt, scheint aber weiterhin durch. Mit dem Kryptografie-Lehr- und Lernprogramm CrypTool2 können Sie diesen Anfängerfehler einfach nachspielen und Gegenmaßnahmen testen. Ausführlich beschrieben haben wir das in [3]. Bei einer Übertragung von sensiblen Inhalten im TLS-Tunnel wäre durchscheinender Klartext verheerend.

Für den Einsatz in TLS kommt ECB daher nicht infrage und ist in den Cipher-

Bewährtes Werkzeug für Webentwickler: caniuse.com zeigt auch, welche Browser TLS 1.3 verstehen. Der Internet Explorer gehört nicht dazu.

Suites zum Glück nicht vorgesehen. Anstatt die Blöcke stumpf nacheinander zu verschlüsseln, muss man sie verknüpfen, damit AES sicher ist. Jeder Block hat dann Einfluss auf den folgenden und identischer Klartext wird nicht mehr auf identischen Geheimtext abgebildet. Schon ein verändertes Bit am Anfang würde ein komplett anderes Chiffrat ergeben – so sollte eine Verschlüsselung aussehen. Eine populäre Blockverkettung ist der „Cipher Block Chaining Mode“ (CBC). Der ist grundsätzlich sicher, allerdings haben findige Sicherheitsforscher eine Schwachstelle ausgemacht – auszunutzen mit dem Lucky-13-Angriff. In der Praxis leicht umsetzbar ist der allerdings nicht. Es handelt sich um einen Timing-Angriff, den Forschern fielen unter Laborbedingungen minimale Zeitunterschiede beim Prüfen von Paketen auf, die sie dem Server schickten. Im echten Leben wäre es enorm schwer, diese Unterschiede zu ermitteln – man müsste möglichst nah vorm Server sitzen, sonst ist das Rauschen durch die üblichen Verzögerungen im Netzwerk (oder gar im weltweiten Internet) zu groß, um die winzigen Unterschiede zu erkennen. Dennoch besteht ein Restrisiko für erfolgreiche Lucky-13-Angriffe, das man nicht eingehen muss. Mit dem „Galois/Counter Mode“ (GCM) gibt es einen AES-Betriebsmodus, der resistent gegen Lucky-13-Angriffe ist. Streicht man CBC aus der Liste und aktiviert nur GCM, verliert man nichts.

Die letzte Entscheidung bei TLS 1.2 ist einfach: Zum Funktionsumfang von TLS gehört auch ein Verfahren, um die Integrität der übertragenen Daten zu sichern. Das Gegenüber kann dann sicher sein, dass die Daten auch wirklich vom Kommunikationspartner stammen und kein Angreifer den verschlüsselten Datenstrom manipuliert hat. Das ist genauso wichtig wie die Vertraulichkeit durch Verschlüsselung. Um das zu erreichen, generiert der Server aus dem Hash der Nachricht unter Zuhilfenahme des gemeinsamen Schlüssels einen MAC (Message Authentication Code). Der Client kann den prüfen und sicher sein, dass nur der Inhaber des gemeinsamen Schlüssels ihn erzeugt haben kann – die Nachricht muss dann genau so vom Server kommen. Wichtig dabei: Das Hashverfahren muss zuverlässig sein. Daher streicht man Cipher-Suites, die unzuverlässige Hashverfahren nutzen: Wenn Sie in Ihrer Konfiguration die Zeichenketten MD5 oder SHA (ohne Nummer) sehen, setzen Sie zügig den Rotstift daran an. Sicher (auch nach Auffassung des BSI) sind aktuell SHA-256 und SHA-384.

Was bleibt

Wenn Sie beim Lesen der vergangenen Abschnitte das Gefühl beschlichen hat, dass nicht mehr viele Optionen übrig bleiben, liegen Sie richtig. Die Liste schrumpft schnell zusammen auf sechs wesentliche Varianten. Haben Sie ein Zertifikat mit einem Schlüssel auf Basis elliptischer Kurven, bleiben nur die beiden Einträge

```
ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
```

Haben Sie einen RSA-Schlüssel im Zertifikat, bleiben folgende vier Zeilen:

```
ECDHE_RSA_WITH_AES_128_GCM_SHA256
ECDHE_RSA_WITH_AES_256_GCM_SHA384
DHE_RSA_WITH_AES_128_GCM_SHA256
DHE_RSA_WITH_AES_256_GCM_SHA384
```

Mit diesen wenigen Zeilen halten Sie sich strikt an die Richtlinie des BSI, haben keine Sicherheitslücken eingebaut, setzen PFS um, schützen sich vor Lucky-13 und schließen keine relevanten Clients aus. Der IE11 kann sich verbinden, ebenso antiquierte Telefone ab Android 4.4.2 oder Chrome-Browser ab Version 31 aus dem Jahr 2013 und Firefox ab Version 27 aus 2014.

Um dieses Wissen umzusetzen und Ihren Server zu konfigurieren, gibt es tatkräftige Unterstützung: Mozilla bietet unter der Adresse ssl-config.mozilla.org ein kleines Online-Werkzeug an, das Konfigurationsdateien für alle gängigen Webserver und Reverse Proxies von Apache über HAProxy und Nginx bis Traefik erzeugt. Wählen Sie in der Mitte oben „Intermediate“, um eine Konfigurationsdatei zu generieren, die TLS 1.2 noch berücksichtigt.

Die erzeugte Liste deckt sich weitgehend mit den oben abgedruckten sechs Optionen. Je nach Server variiert die Schreibweise der Cipher-Suiten etwas, einige wollen Binde-, andere Unterstriche und auch das „WITH“ oder „TLS“ am Anfang erwarten nur manche Server.

Mit diesen Feinheiten müssen Sie leben, und der Mozilla-Konfigurator nimmt Ihnen die Übersetzungsarbeit ab. Die größte inhaltliche Abweichung zwischen der BSI-konformen Liste oben und den Empfehlungen von Mozilla sind folgende Zeilen, die beim BSI fehlen:

```
ECDHE_ECDSA_WITH_CHACHA20_POLY1305
ECDHE_RSA_WITH_CHACHA20_POLY1305
```

Die Schlüsselaushandlung mit ECDHE ist oben schon erklärt, neu sind das MAC-Verfahren Poly1305 und das symmetrische Verschlüsselungsverfahren ChaCha20, das zu den Stromverschlüsselungsverfahren gehört. Verschlüsselt wird jedes Bit einzeln, also nicht blockweise wie bei AES. Daher enthält die Suite auch keine Angabe für einen Verkettungsmodus wie GCM oder CBC. Vorangebracht haben Google und Cloudflare dieses Verfahren, seit 2016 ist es im RFC 7905 verewigt und Internetstandard. ChaCha20 und Poly1305 gelten

als sicher und schnell, hinter beiden steht die Krypto-Koryphäe Daniel J. Bernstein. Insgesamt spricht viel dafür, diese Cipher-Suites zu aktivieren. Das BSI äußert sich in seiner Richtlinie bis heute noch gar nicht zu beiden, rät also weder dazu noch davon ab. Für das konservative deutsche Bundesamt ist die Technik offenbar noch zu jung, um sich festzulegen. Wer damit leben kann, dass er von den BSI-Empfehlungen abweicht, schaltet diese Suites ein, wer zu 100 Prozent konform sein will oder muss, lässt sie weg.

Wenn Sie dagegen irgendwo vom Stromverschlüsselungsverfahren RC4 als tolle Alternative zu AES hören, lassen Sie die Finger davon. Gegen dieses Verfahren gibt es wirkungsvolle Angriffe, einige Experten gehen davon aus, dass die NSA mit Werkzeugen ausgestattet ist, um RC4 nahezu in Echtzeit zu knacken und mitzulesen.

TLS 1.3

Die sichere Konfiguration von TLS 1.3 ist bei den meisten Servern erfrischend kurz. Das liegt daran, dass die Erfinder beim Planen von TLS 1.3 nicht nur Maßnahmen zur Beschleunigung eingebaut, sondern auch diverse Altlasten entfernt haben. MD5, SHA-1, CBC oder Diffie-Hellmann mit statischen Schlüsseln gibt es nicht mehr, PFS mit ephemeren Schlüsseln ist Pflicht. Im Hintergrund werden die Verfahren benutzt, die auch für TLS 1.2 empfohlen sind, in der Syntax von OpenSSL heißen die Cipher-Suites für TLS 1.3:

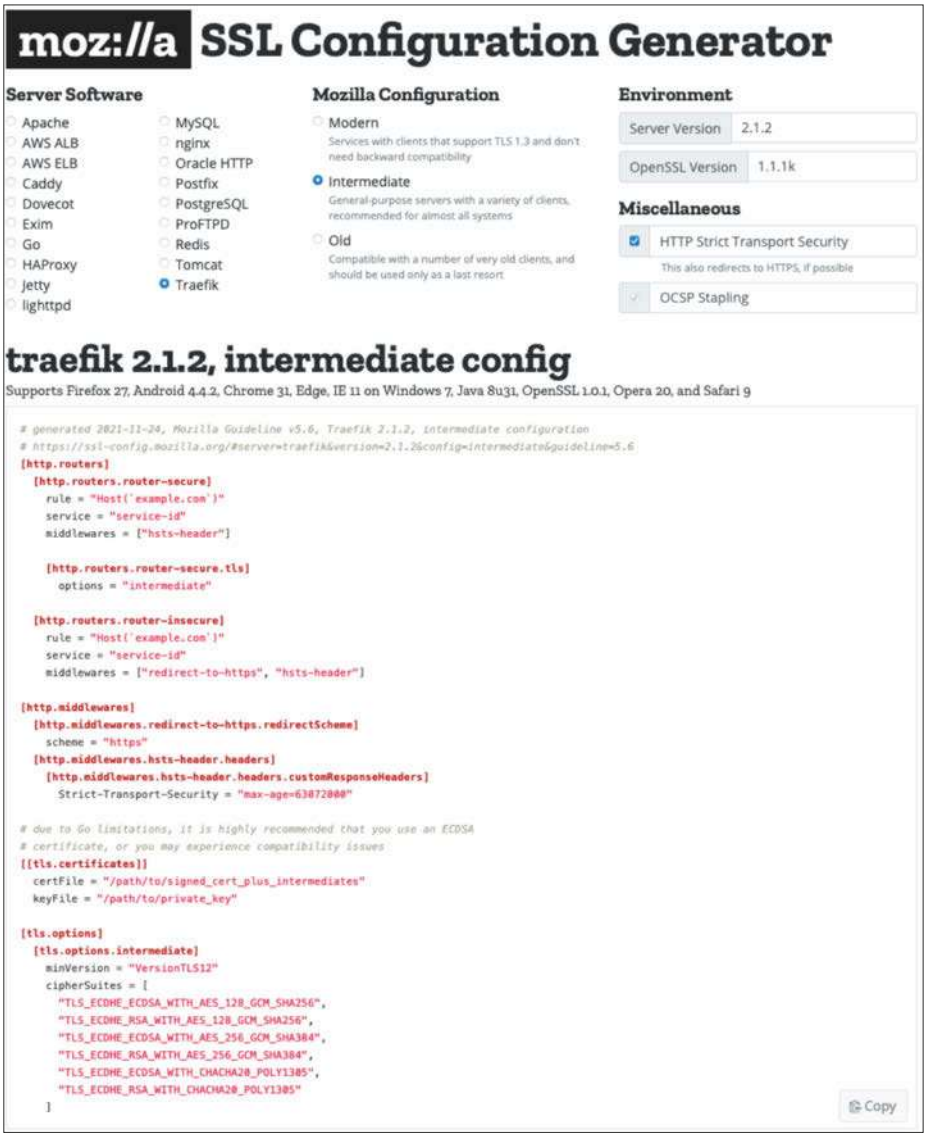
```
TLS13-AES-256-GCM-SHA384
TLS13-AES-128-GCM-SHA256
TLS13-CHACHA20-POLY1305-SHA256
```

Diese Information brauchen Sie als Anwender von gängigen Servern aber gar nicht. Bei Nginx etwa reicht die Zeile `ssl_protocols TLSv1.3;`, die Cipher-Suites müssen Sie nicht angeben. Vollständige Konfigurationen für den eigenen Server liefert das bereits vorgestellte Mozilla-Werkzeug, wenn man „Modern“ auswählt.

Es lohnt sich

Als Serverbetreiber ist man für die sichere Übertragung der Daten zu den Nutzern verantwortlich, kommt also nicht umhin, sich mit TLS auseinanderzusetzen. Mit dem Wissen zu Schlüsselübertragungs-, Verschlüsselungs- und Hashverfahren sowie dem Mozilla-Konfigurationshelfer ist man gut ausgestattet, um den eigenen Server sicher zu betreiben. Mit TLS 1.3 wird vieles einfacher, weil es viele unsichere Optionen nicht mehr gibt und man die Cipher-Suites meist nicht mehr selbst wählen muss. Wie bei jedem Server muss man regelmäßig Updates einspielen und darauf vertrauen, dass zukünftig für unsicher befundene Verfahren von den Entwicklern der Software über ein solches abgedreht werden.

Wenn Sie vor oder nach der Neukonfiguration Ihres TLS-Servers wissen wollen, wie gut sie dastehen, können Sie sich an einem beliebten Admin-Volkssport beteiligen und daran arbeiten, beim SSL-Server-Test unter ssllabs.com/ssltest eine Bewertung mit A+ zu ergattern. Neben Einstellungen an TLS mit optimalen Cipher-Suites muss man für diese Bestwertung zum Beispiel auch HTTP Strict Transport Security (HSTS) auf einem Webserver aktivieren. Damit weisen Server ihre Clients an, ausschließlich verschlüsselte Verbindungen zu nutzen. *(jam@ct.de)* c't



Mozilla macht das Erstellen von Konfigurationen mit sicheren Cipher-Suites einfach. Für gängige und weniger gängige Software erzeugt der SSL Configuration Generator Konfigurationsschnipsel.

Literatur

[1] Leo Dessani und Prof. Dr. Ronald Petrlic, Fehlender Vertrauensanker, Datenschutzprobleme bei Mailhosting-Anbietern, c't 13/2021, S. 132

[2] Dr. Jan Kopia, Sicherer Kanal, Symmetrische und asymmetrische Verschlüsselung, c't 7/2021, S. 56

[3] Nils Kopal und Bernhard Esslinger, Krypto ganz unkryptisch, Mit CrypTool 2 moderne Kryptografie ausprobieren und verstehen, c't 15/2021, S. 142

Internetstandards und BSI-Richtlinie: ***ct.de/y2yq***