Kai Jendrian

TLS – Dos and Don'ts

TLS – ein Sicherheitsmonstrum

Nicht erst seit den Enthüllungen von Edward Snowden hat die Absicherung von grundsätzlich unsicheren Internet-Protokollen wie HTTP, SMTP, POP3 oder IMAP auf der Transportschicht an Bedeutung gewonnen. Hierzu wird in der Praxis vor allem auf die inzwischen 20 Jahre alte Familie von Verschlüsselungsprotokollen zurückgegriffen, die unter dem Kürzel SSL/TLS (Secure Sockets Layer/Transport Layer Security) bekannt ist. Mit Hilfe von SSL/TLS sollen vor allem die Vertraulichkeit, Integrität und Authentizität der übertragenen Daten geschützt werden.

Das Secure Sockets Layer (SSL) Protokoll wurde bis Version 3.0 weiterentwickelt, bevor es 1999 mit der Standardisierung durch das IETF in Transport Layer Security (TLS) umbenannt wurde. In der Praxis hat sich die Schreibweise SSL/TLS etabliert, häufig wird aber auch abkürzend nur von SSL gesprochen. Seit der vor kurzem bekannt gewordenen POODLE-Schwachstelle [1] muss allerdings von einem Einsatz von SSLv3 abgeraten werden; nur noch TLS-Versionen gelten als angemessen sicher. In diesem Beitrag wird daher nur noch das Kürzel TLS verwendet, um diesem Fakt Rechnung zu tragen – es sei denn, es wird explizit auf eine Version von SSL verwiesen.

Im Verlauf der 20jährigen Geschichte hat sich TLS zu einer komplexen Protokollfamilie entwickelt, die leider von zahlreichen Sicherheitsschwachstellen [2] betroffen war. BEAST, CRIME, TIME, BREACH, Heartbleed, Lucky 13 und RC4-Schwachstellen sind zurzeit in aller Munde. Sie betreffen vor allem ältere Versionen von SSL und die zugehörigen Cipher-Suites. Bei der Vielzahl von Schwachstellen fällt es auch Experten schwer, die Übersicht zu behalten und eine sichere Konfiguration vorzunehmen – besonders dann, wenn es Kompatibilitätsanforderungen gibt, die das Festhalten an unsicheren Einstellungen erzwingen. Da gilt es, Sicherheit gegen die Unterstützung alter Software abzuwägen – mit Blick auf die Schwachstellen kann es empfehlenswert sein, alte Zöpfe abzuschneiden.

TLS ist einfach zu aktivieren. Nicht ganz so einfach ist es allerdings, TLS korrekt zu konfigurieren. In diesem Gateway wird eine Auswahl wichtiger Empfehlungen für eine sichere Nutzung von TLS nach dem derzeitigen Stand der Technik gegeben. Für interessierte Leser bieten die Literaturhinweise zusätzliches Futter: Die Absicherung der Transportschicht füllt ganze Bücher.

Es ist zu erwarten, dass neue Schwachstellen in Protokollen und Implementierungen aufgedeckt werden. Daher ist es wichtig, aktuelle Sicherheitsentwicklungen im Blick zu behalten, die eigene Konfiguration von TLS regelmäßig zu überprüfen und einen Prozess zu etablieren, um die Konfiguration stets auf dem Stand der Technik zu halten.

Konfiguration der Server

TLS kommt in sehr unterschiedlichen Produkten zum Einsatz. In jedem Einsatzszenario muss TLS möglichst sicher konfiguriert werden. Im Folgenden werden einige Produkt unabhängige Empfehlungen vorgestellt. Für die konkrete technische Umsetzung empfiehlt sich ein Blick in das Buch *Bulletproof SSL and TLS* [3], das *OpenSSL Cookbook* [4], die *SSL/TLS Deployment Best Practices* [5] oder die Empfehlungen *Applied Crypto Hardening* [6] sowie die zahlreichen Software- und Produktdokumentationen.

Schlüssel und Zertifikate

Ein wesentlicher Baustein für den Einsatz von TLS sind Schlüsselpaare für asymmetrische Kryptografie und die Beglaubigungen der öffentlichen Schlüssel in Form von Zertifikaten. Dabei gilt es ein paar wichtige Grundsätze zu beachten.

Schlüsselpaare sollten immer in einer sicheren Umgebung erzeugt werden und sicher gespeichert werden. RSA-Schlüssel sollten eine Mindestlänge von 2048 bit haben. Der Zugriff auf die privaten Schlüssel sollte auf das Notwendige beschränkt werden. Auf keinen Fall sollten Schlüsselpaare genutzt werden, die nicht vollständig unter eigener Kontrolle erzeugt wurden oder auf die unbefugt zugegriffen werden konnte.

Werden Schlüssel über Diffie-Hellman-Verfahren ausgetauscht ist zu beachten, dass die entsprechenden Parameter dem Sicherheitsniveau der Länge der gewählten Schlüssel entsprechen. Nicht jede Software lässt sich hier einfach konfigurieren – beispielsweise passt der Apache-Webserver erst ab Version 2.4.7 die Parameter automatisch an.

Nach der Erzeugung der Schlüssel müssen die öffentlichen Schlüssel beglaubigt werden. Hierzu hat sich ein PKI-Vertrauensmodell mit zahlreichen *Certificate Authorities* im Internet etabliert, welches viele eigene Herausforderungen mit sich bringt, die aber hier nicht diskutiert werden können. Um ein Zertifikat zu beschaffen ist zunächst ein *Certificate Signing Request* zu erstellen, mit dem der öffentliche Schlüssel verschiedenen (Domain-) Namen zugeordnet wird. Dabei sollte beachtet werden, dass alle (Domain-) Namen abgedeckt sind, unter denen die zu sichernden Dienste angeboten werden.

Fehlerhaft ausgestellte Zertifikate (z. B. eine falsche Zuordnung von Namen und Diensten) sind ein möglicher Grund für Warnungen an den Benutzer, die vermieden werden sollten. Wenn nötig sollten alternative Namen im Zertifikat hinterlegt oder *Wildcard*-Zertifikate genutzt werden. Die früher erforderliche eindeutige Zuordnung von IP-Adressen zu Zertifikaten ist mit moderner Software dank *Server Name Indication* (SNI) nicht mehr erforderlich. So lassen sich virtuelle Hosts mit überschaubarem Aufwand absichern.

Die Zertifikatsart sollte passend zum erforderlichen Vertrauensniveau in den Dienst gewählt werden: Die einfachste Variante, ein *Domain Validation* (DV) Zertifikat bestätigt nur die Kontrolle des Zertifikatsinhabers über eine Internet-Domäne. Beim *Organisation Validation* (OV) Zertifikat führt die *Certificate Authority* eine grundlegende Prüfung zum Zertifikatsantragsteller durch. Beim *Extended Validation* (EV) Zertifikat werden ausführliche, standardisierte Prüfungen zum Antragsteller durch-

geführt, so dass mit einem solchen Zertifikat das höchste Vertrauensniveau erreicht werden kann. Intensivere Prüfungen haben allerdings in der Regel auch ihren Preis.

Bei neuen Zertifikaten ist zu beachten, dass für die Erstellung der Signatur nicht der Hash-Algorithmus SHA-1 verwendet wird, da dieser zumindest vom Chrome-Browser ab Frühjahr 2015 als unsicher beanstandet wird [7].

Auch abgelaufene Zertifikate führen zu Sicherheitswarnungen beim Benutzer. Es sollte daher einen Prozess zur rechtzeitigen Verlängerung und zum Austausch der Zertifikate geben, um Benutzer nicht unnötig zu verunsichern. Für den Fall einer Kompromittierung der privaten Schlüssel sollte ein Prozess zur Sperrung des zugehörigen Zertifikats etabliert werden.

Schließlich sollte immer die vollständige Zertifikatskette in der richtigen Reihenfolge ausgeliefert werden.

Protokolleinstellungen

Für SSL/TLS gibt es einige wenige wichtige Empfehlungen. SSL ist tot: SSLv2 und SSLv3 sollten auf keinen Fall mehr aktiviert werden. Wenn möglich sollte auch auf TLS v1.0 verzichtet werden. Lediglich für TLS v1.1 und TLS v1.2 sind zurzeit keine Schwachstellen bekannt.

Aktuelle Server unterstützen *Secure Renogiation*, so dass es kaum noch einen Grund geben sollte, *Insecure Renogitiation* zu aktivieren. Nur noch stark veraltete Clients erfordern diese Option. Zur Vermeidung von Angriffen über die Kompression von TLS sollte diese grundsätzlich deaktiviert sein.

Auswahl von Cipher-Suites

Eine der größten Herausforderung bei der Konfiguration von TLS besteht in der Auswahl passender Cipher-Suites. Bei TLS besteht eine Cipher-Suite immer aus verschiedenen kryptografischen Primitiven für Schlüsselaustausch und authentisierung, Verschlüsselung und Betriebsmodus sowie (optional) MAC oder Pseudozufallsfunktionen. Alle gültigen Cipher-Suites sind bei IANA standardisiert [8]. Viele dieser Cipher-Suites gelten heute als unsicher und sollten nicht mehr verwendet werden.

Über die optimale Auswahl streiten sich die Experten, auch hier ist es wichtig, eine ausgewogen Auswahl zu treffen und bewusst zu entscheiden, wie viel Rückwärtskompatibilität zwingend erforderlich ist. Es bietet sich an, Cipher-Suites zu bevorzugen, die ein paar Grundregeln genügen:

- Es sollten keine NULL-Cipher, keine EXPORT-Cipher, keine anonymen (ADH, AECDH) und keine schwachen Verschlüsselungsverfahren (RC2, RC4, DES, MD5) eingesetzt werden.
- Exoten sollten möglichst vermieden werden (PSK, SRP, SEED, IDEA, ARIA), da deren Sicherheit in der Regel nicht sehr intensiv öffentlich bzw. wissenschaftlich überprüft worden ist.
- Der Schlüsselaustausch sollte dynamisch über Diffie-Hellman erfolgen, dabei ist der Austausch über Elliptische Kurven performanter; damit ist *Forward Secrecy* möglich [18].
- GCM sollte als Betriebsmodus bevorzugt werden.
- Der Server sollte das Verschlüsselungsverfahren nach seiner Priorisierung auswählen und nicht die Priorisierung des Clients berücksichtigen.
- Aktuelle Cipher-Suites von TLS v1.2 sollten bevorzugt werden.
- Bei notwendiger Unterstützung veralteter Browser müssen ggf. unsichere Verschlüsselungsverfahren unterstützt werden. Da-

bei gilt – wegen der bekannten Schwächen von RC4 – die Suite 3DES_EDE_CBC3 als sicherer als RC4_128_SHA.

Für einige Server bietet der *Mozilla SSL Configuration Generator*[1] einen guten Einstieg in die Auswahl passender Cipher-Suites. Weitere detaillierte Empfehlungen finden sich, neben den oben genannten Quellen, in der *Technischen Richtlinie TR-02102-2, Kryptographische Verfahren: Empfehlungen und Schlüssellängen, Teil 2 – Verwendung von Transport Layer Security (TLS)* [9].

Best Practices – TLS

Über die Konfiguration der Dienste hinaus gibt es einige Best Practices, deren Berücksichtigung das Sicherheitsniveau von TLS im Betrieb signifikant erhöht.

- Bei einer Webanwendung sollten alle Seiten geschützt mit TLS ausgeliefert werden. Sogenannter Mixed-Content sollte vermieden werden.
- Zugriff auf Cookies sollte nur für geschützte Seiten und nicht für Javascript ermöglicht werden. Hierfür sollten die Cookie-Flags *secure* und *HttpOnly* durchgängig gesetzt sein.
- Es sollte HTTP *Strict Transport Security* (HSTS) eingeschaltet sein. Der entsprechende Header sollte konfiguriert sein.
- Software sollte in möglichst aktuellen Versionen eingesetzt werden. Das gilt besonders für kryptografische Bibliotheken (wie OpenSSL) und Webserver (wie Apache). Beispielsweise fehlt im Versionszweig 2.2.x von Apache die Unterstützung Elliptischer Kurven.

Fazit

Zusammenfassend lässt sich festhalten, dass SSL/TLS über die Zeit zahlreiche Sicherheitsprobleme hatte und daher die Konfiguration sorgfältig vorgenommen werden sollte. Aktuelle Entwicklungen zu TLS sollten regelmäßig beobachtet werden. Dabei sollte es einen Plan für die Reaktion auf neue Schwachstellen geben.

Ganz wichtig ist es auch, die eigene Konfiguration regelmäßig zu testen. Werkzeuge wie *SSL Labs*[2] oder *O-Saft*[3] leisten dabei gute Dienste.

1 https://mozilla.github.io/server-side-tls/ssl-config-generator/
2 https://www.ssllabs.com/ssltest/
3 https://www.owasp.org/index.php/O-Saft

Literatur

[1] Bodo Möller, Thai Duong, Krzysztof Kotowicz; This POODLE Bites: Exploting the SSL 3.0 Fallback; https://www.openssl.org/~bodo/ssl-poodle.pdf
[2] Mozilla; Attacks on SSL and TLS, https://wiki.mozilla.org/Security/Server_Side_TLS#Attacks_on_SSL_and_TLS
[3] Ivan Ristić; Bulletproof SSL and TLS
[4] Ivan Ristić; OpenSSL Cookbook
[5] Ivan Ristić; SSL/TLS Deployment Best Practices, https://www.ssllabs.com/downloads/SSL_TLS_Deployment_Best_Practices.pdf
[6] Diverse Autoren; Applied Crypto Hardening, https://bettercrypto.org/static/applied-crypto-hardening.pdf
[7] Chris Palmer; Gradually sunsetting SHA-1, http://googleonlinesecurity.blogspot.de/2014/09/gradually-sunsetting-sha-1.html
[8] IANA TLS Cipher Suite Registry, http://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml#tls-parameters-4
[9] BSI; Technische Richtlinie TR-02102-2, Kryptographische Verfahren: Empfehlungen und Schlüssellängen, Teil 2 – Verwendung von Transport Layer Security (TLS). https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102-2_pdf.pdf?__blob=publicationFile
[10] OWASP; Transport Layer Protection Cheat Sheet, https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet
[11] OWASP; HTTP Strict Transport Security https://www.owasp.org/index.php/HTTP_Strict_Transport_Security
[12] OWASP; Certificate and Public Key Pinning, https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning
[13] Mozilla Wiki – Security TLS, https://wiki.mozilla.org/Security/Server_Side_TLS
[14] Certificate Transperency, http://www.certificate-transparency.org/
[15] Let's encrypt; https://letsencrypt.org/
[16] Apache mod_ssl Dokumentation;
http://httpd.apache.org/docs/2.4/mod/mod_ssl.html
[17] ENISA; Algorithms, key size and parameters report 2014, http://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-size-and-parameters-report-2014
[18] Fox, Dirk: Perfect Forward Secrecy (PFS). DuD 11/2013, S. 729
[19] Achim Hoffmann, Torsten Gigler; O-Saft – Richtig verschlüsseln mit SSL/TLS; OWASP Day Germany 2014, https://www.owasp.org/images/1/19/Richtig_verschluesseln_mit_SSL%2BTLS_-_Achim_Hoffmann%2BTorsten_Gigler.pdf