Dirk Fox

Perfect Forward Secrecy (PFS)

Hintergrund

Datenübertragungsprotokolle, die die Vertraulichkeit der (Inhalts-) Daten mittels Verschlüsselung schützen, wie z. B. das *Transport Layer Security* Protokoll (TLS, traditionell als „SSL" bezeichnet) [1], verwenden symmetrische Verschlüsselungsverfahren, bei denen für die Ver- und Entschlüsselung derselbe Schlüssel zur Anwendung kommt. Dieser Schlüssel wird als *Session Key* zu Beginn einer Verbindung zwischen Client und Server vereinbart und nach deren Ende gelöscht.

Die Schlüsselvereinbarung beim Verbindungsaufbau erfolgt dabei mittels eines Schlüsselaustausch- oder Schlüsseltransportprotokolls, das asymmetrische kryptografische Verfahren verwendet. Dieses Protokoll stellt sicher, dass dabei weder der vereinbarte *Session Key* einem Dritten zur Kenntnis gelangen noch das Protokoll (bspw. durch einen *Man-in-the-Middle*-Angriff) manipuliert werden kann.

Dieses Vorgehen – Schlüsseltransport mit asymmetrischer Kryptografie und anschließender symmetrischer Verschlüsselung – kombiniert die Vorteile der beiden Ansätze:

- Symmetrische Verschlüsselungsverfahren erreichen einen deutlich höheren Durchsatz als asymmetrische Verfahren.
- Asymmetrische Kryptoverfahren arbeiten mit Schlüsselpaaren aus einem öffentlichen und einem privaten Schlüssel und erfordern daher keinen vertraulichen Schlüsselaustausch zwischen den Kommunikationspartnern.
- Die Verwendung jeweils einmalig genutzter und zufällig ausgewählter *Session Keys* verhindert, dass ein Angreifer aus mehreren Verbindungen große Mengen an Schlüsseltext erhält, der ihm möglicherweise einen Angriff auf das verwendete symmetrische Verschlüsselungsverfahren erleichtert.

Verwendet ein System für den Schlüsseltransport und die anschließende verschlüsselte Übertragung ausschließlich anerkannt starke kryptografische Verfahren (wie bspw. RSA respektive AES-GCM), ausreichend lange Schlüssel [2], die zufällig gewählt wurden (und damit nicht „vorhersagbar" sind), ein standardisiertes Protokoll (wie bspw. SSL/TLS) und eine fehlerfreie Implementierung, dann sind die übermittelten Daten wirksam vor dem unberechtigten Zugriff Dritter geschützt.

Grenzen

Zwei Restrisiken bleiben allerdings selbst dann, wenn man alles richtig gemacht hat:

- Werden die verwendeten kryptografischen Verfahren eines (hoffentlich fernen) Tages gebrochen, dann kann ein Angreifer oder Nachrichtendienst, der die Datenübertragung aufgezeichnet und gespeichert hat, sie nachträglich entschlüsseln.
- Kann ein Angreifer oder Nachrichtendienst sich Zugriff auf den geheimen Schlüssel des Servers verschaffen, dann kann er – bei einer aufgezeichneten Datenübertragung auch nachträglich – das Schlüsseltransportprotokoll aus Sicht des Servers entschlüsseln und erhält damit Kenntnis von dem verwendeten symmetrischen *Session Key*.

Abhilfe

Zwar lässt sich das Brechen eines kryptografischen Verfahrens durch die Verwendung gut untersuchter und unter Experten als sicher angesehener Algorithmen und möglichst langer Schlüssel erschweren, aber nicht grundsätzlich ausschließen.

Anders hingegen sieht es beim Schutz des *Session Keys* aus. Hier entscheidet das verwendete Schlüsselvereinbarungsprotokoll darüber, ob ein Dritter, der das Protokoll abhört, mit Kenntnis des privaten Schlüssels des Servers nachträglich den *Session Key* gewinnen kann. Denn es gibt Schlüsselvereinbarungsprotokolle, die die Eigenschaft *Perfect Forward Secrecy* (PFS) besitzen und damit verhindern, dass der *Session Key* rückwirkend aus einem Protokollmitschnitt gewonnen werden kann.

Das Prinzip hinter PFS geht auf eine Idee von Diffie und Hellman zurück, die diese bereits 1976 in ihrem wegweisenden Aufsatz „New directions in cryptography" publizierten; derselbe Aufsatz, in dem sie auch die Idee der asymmetrischen Kryptographie vorstellten [3]: Wählen die beiden Kommunikationspartner, Client und Server, beide eine Zufallszahl a bzw. b und schicken einander $A = g^a \bmod p$ bzw. $B = g^b \bmod p$ zu, so können beide daraus denselben *Session Key*

$$K = A^b \bmod p = g^{ab} \bmod p = g^{ba} \bmod p = B^a \bmod p$$

berechnen (mit $1 < g < (p\text{-}1)$, p Primzahl und a, b aus $\{1, \ldots, p\text{-}1\}$). Ähnlich funktioniert das Diffie-Hellman-Verfahren auf Grundlage elliptischer Kurven anstelle einer Primzahl. Bei ausreichend großem p (einige hundert Dezimalstellen) können aus A und B weder a noch b bestimmt werden – und damit auch nicht der *Session Key* K. Das per Zertifikat bestätigte RSA- oder DSA-Schlüsselpaar des Servers wird bei Verfahren mit PFS lediglich zur Signatur von B verwendet, um einen *Man-in-the-Middle*-Angriff auszuschließen. Löschen Client und Server ihre zufällig gewählten Exponenten a und b sowie den *Session Key* nach erfolgreicher verschlüsselter Datenübertragung, so kann niemand, nicht einmal Client oder Server, selbst mit Kenntnis des privaten Serverschlüssels den verwendeten symmetrischen *Session Key* zurückgewinnen.

Einige Webserver verwenden in der PFS-Standardeinstellung für Diffie-Hellman-Schlüssel eine deutlich kürzere Schlüssellänge (bspw. 1.024 bit für p) als für den eigentlichen, zertifizierten Serverschlüssel (meist mindestens 2.048 bit für RSA) – das sollte bei der Konfiguration korrigiert werden.

Literatur

[1] Esslinger, Bernhard; Müller, Maik: *Secure Sockets Layer (SSL) Protokoll*, DuD 12/1997, S. 691-697.

[2] Fox, Dirk: *Mindestlängen von Passwörtern und Schlüsseln*. DuD 10/2009, S. 620-623.

[3] Diffie, Whitfield; Hellman, Martin: *New Directions in Cryptography*. IEEE Transactions on Information Theory. Bd. 22, Nr. 6, 1976, S. 644-654.