Universität Hamburg

Fachbereich Informatik

Entwurf vom
20. Mai 2015

Exposé

Bachelorarbeit über TLS (1.3)

vorgelegt von

Tom Petersen

geb. am 13. Dezember 1990 in Hannover

Matrikelnummer 6359640

Studiengang Informatik

eingereicht am 20. Mai 2015

1 SSL und TLS - ein Überblick

SSL (Secure Socket Layer) bzw. TLS[1] (Transport Layer Security) ist ein zustandsbehaftetes Protokoll, das auf dem TCP-Protokoll der Transportschicht des TCP/IP-Protokollstapels aufbaut[2]. Es bildet also eine Schicht zwischen Transport- und Anwendungsschicht. Viele Protokolle der Anwendungsschicht nutzen TLS zur sicheren Datenübertragung, so beispielsweise HTTPS oder FTPS.

SSL wurde von der Firma Netscape entwickelt und nachdem es starke Verbreitung gefunden hatte, durch die IETF als TLS 1.0 in RFC 2246 standardisiert (TLS 1.0 entspricht hierbei SSL 3.1). Aktuell ist die TLS-Version 1.2 und an Version 1.3 wird gearbeitet. [Sch09]

Hauptaufgaben von TLS sind Authentifikation der Kommunikationspartner, symmetrische Verschlüsselung der Kommunikation sowie die Sicherstellung der Integrität der übertragenen Nachrichten. Die hierbei verwendeten kryptographischen Verfahren werden erst zu Beginn der Kommunikation festgelegt. [Eck13]

TLS 1.0 RFC 2246 - http://tools.ietf.org/html/rfc2246

TLS 1.1 RFC 4346 - http://tools.ietf.org/html/rfc4346

TLS 1.2 RFC 5246 - http://tools.ietf.org/html/rfc5246

TLS Extensions RFC 3546 - http://tools.ietf.org/html/rfc3546,
RFC 3466 - http://tools.ietf.org/html/rfc4366,
RFC 6066 - http://tools.ietf.org/html/rfc6066

TLS 1.3 Draft - https://tools.ietf.org/html/draft-ietf-tls-tls13-05

1. Im weiteren Verlauf dieser Arbeit wird der Einfachheit lediglich von TLS gesprochen, es ist jedoch ebenso SSL gemeint. Bei etwaigen Unterschieden wird explizit auf diese eingegangen werden.
2. Es gibt auch DTLS (Datagram Transport Layer Security), ein auf TLS basierendes Protokoll, dass auch per UDP Daten übertragen kann

2 Funktionsweise und Teilprotokolle

Zum Einstieg grobe Funktionsbeschreibung, evtl. schon mit Grafik über Verbindungsaufbau?, Grafik der TLS-Protokolle

2.1 Untere Schicht

TLS besteht selbst aus zwei Schichten. In der unteren Schicht befindet sich das *Record-Protokoll*, das die Daten von den Teilprotokollen der oberen Schicht entgegennimmt, diese Anwendungsdaten fragmentiert (maximale Paketgröße 2^{14} Byte) und optional komprimiert. Danach wird je nach aktuell verhandelten krypographischen Funktionen die Integrität der Daten durch Berechnen und Anhängen eines MACs gesichert und die Nachricht verschlüsselt[1].

Nach http://crypto.stackexch schlägt Schneier in Cryptography Engineering MAC-then-encrypt aus Gründen der Komplexheit von Encrypt-then-MAC vor. Nachlesen! Gibt es in der Informatik-Bibliothek: T FER 45399

Der Mac wird auf die folgende Weise berechnet[2]:

$$\begin{aligned} MAC = hash(&MAC\_KEY|pad_2| \\ &hash(MAC\_KEY|pad_1|seq\_num|SSLCompressed.type| \\ &SSLCompressed.length|SSLCompressed.fragment)) \end{aligned}$$

Ich vermute mal, dass das noch das alte Verfahren ist, das in TLS durch HMAC abgelöst wird. Nachschauen, ebenso SSLCompressed vmtl jetzt anders?

wobei $pad_1 = (0x36)\ldots(0x36)$ (48 mal für MD5 und 40 mal für SHA) und $pad_2 = (0x5c)\ldots(0x5c)$ (48 mal für MD5 und 40 mal für SHA) gilt. Diese Werte entsprechen ipad und opad aus dem HMAC-Verfahren ([KBC97]).

Der Record-Header enthält Informationen über die verwendete SSL/TLS-Version, den Content-Type (Handshake, Alert, ChangeCipherSpec, ApplicationData) und die Länge des Klartextfragments.

Aus dem nach Ausführung des Handshake-Protokolls (siehe nächsten Abschitt) beidseitig bekannten Master-Secret werden Schlüssel für die Erstellung des MACs sowie für die Kommunikation zwischen Client und Server berechnet (je einer pro Seite). Dazu werden solange Schlüsselblöcke nach dem folgenden Verfahren erstellt, bis alle Schlüssel konstruiert werden können:

Wahrscheinlich auch das hier ein bisschen anders?

$$\begin{aligned} key\_block = &MD5(ms|SHA('A'|ms|R_C|R_S))| \\ &MD5(ms|SHA('BB'|ms|R_C|R_S))| \\ &MD5(ms|SHA('CCC'|ms|R_C|R_S)) \end{aligned}$$

1. Achtung: hier wird MAC-then-Encrypt angewendet. Laut [BN00] ist Encrypt-then-MAC vorzuziehen.
2. Entnommen aus [Eck13].

2.2 Obere Schicht

In der oberen Schicht sind vier Teilprotokolle spezifiziert: *Handshake-, ChangeCipherSpec-, Alert-* und *ApplicationData-Protokoll*.

Das *Handshake-Protokoll* dient zur Vereinbarung kryptographischer Verfahren und zur Aushandlung eines Schlüssels für die symmetrische Verschlüsselung der später gesendeten Daten. Der Handshake kann folgendermaßen ablaufen (abhängig von optionalem Clientzertifikat, Preshared Key, erneute Verbindung, ...; entnommen aus [Eck13]):

als ordentliche Grafik ohne Clientverifikation

- –>client-hello mit
  $H_C = (4$ Byte Zeitstempel, 28 Byte Zufallszahl, Sitzungsidentifikator (!= Null bei bereits existierender Sitzung), Cipher Suite$)$
- <– server-hello mit
  $H_S = (4$ Byte Zeitstempel, 28 Byte Zufallszahl, Sitzungsidentifikator (falls vom Client gewünschte Sitzung bedient werden kann), Cipher Suite, die vom Server unterstützt wird$)$
- <– Server Zertifikat (inklusive öffentlichem Schlüssel des Servers, meist nach X.509v3)
- Client: Zertifikatverifikation
- –> Generierung und Senden von pre-master-secret (48 Byte) verschlüsselt mit öffent. Schlüssel des Servers (bei RSA) oder Diffie-Hellman-Verfahren
- Client und Server: aus Zufallszahl des Clients R_C aus H_C, Zufallszahl des Servers R_S aus H_S und pre-master-secret wird das master-secret berechnet

so wahrscheinlich < TLS 1.2, danach Pseudozufallsfunktion -> Nachschauen

$$\begin{aligned}\text{master-secret} = & MD5(Pre|SHA('A'|Pre|R_C|R_S))| \\ & MD5(Pre|SHA('BB'|Pre|R_C|R_S))| \\ & MD5(Pre|SHA('CCC'|Pre|R_C|R_S))\end{aligned}$$

- –> Change Cipher Spec-Nachricht (nicht Bestandteil des Handshake-Protokolls -> eigener Record)
- –> Handshakeabschluss: finished mit MD5 und SHA-MAC über alle bisher ausgetauschten Nachrichten (mit Ausnahmen)
- <– Change Cipher Spec-Nachricht (nicht Bestandteil des Handshake-Protokolls -> eigener Record)
- <– Handshakeabschluss: finished mit MD5 und SHA-MAC über alle bisher ausgetauschten Nachrichten (mit Ausnahmen)

Das *ChangeCipherSpec-Protokoll* dient dazu, die vereinbarten kryptographischen Verfahren zu ändern. Es enthält lediglich eine Nachricht mit dem Wert 1, die für das Übernehmen der während des Handshakes ausgehandelten Verfahren steht. [Eck13]

Das *Alert-Protokoll* dient dazu, auftretende Fehler oder Warnungen zu versenden, die während des Datenaustausches auftreten.

Übersicht

Das *ApplicationData-Protokoll* ist zuständig für das Durchreichen von Anwendungsdaten, die von der Anwendungsschicht gesendet werden sollen.[Sch06]

2.3 Sitzungs- und Verbindungskonzept

TLS erstellt beim ersten Handshake eine Sitzung zwischen Client und Server. Hierbei wird ein Sitzungsidentifikator erstellt, der beim server-hello mitgesendet wird. Weiterhin wird sich in der Sitzung das Zertifikat des Gegenübers, optional das Kompressionsverfahren, die Cipher Suite und das master-secret gemerkt.

Ein Client kann nun, wenn er den Sitzungsidentifikator beim client-hello mitschickt, eine alte Sitzung in Form einer neuen Verbindung wiederaufnehmen oder mehrere Verbindungen parallel aufbauen. Eine Verbindung wird dabei durch Client- und Server-Zufallszahlen R_C und R_S, die generierten Schlüssel, einen Initialisierungsvektor sowie aktuelle Sequenznummern beschrieben.

Wie wird sich auf den geeinigt? Übertragen? Aus master-secret?

Beim Verbindungsaufbau kann so ein verkürzter Handshake genutzt werden, bei dem weniger Nachrichten gesendet werden müssen. Es kann dabei auf Neuberechnung des master-secret, Server- und Client-Validierung und Aushandlung der Cipher Suite verzichtet werden.

Grafik des verkürzten Handhakes

2.4 Ciphersuites

3 Angriffe auf SSL und TLS

Eine gute Übersicht zu bisherigen Angriffen auf TLS findet sich in [MS13]. Viele Schwächen früherer Protokollversionen bis SSL 3 sind in [WS96] zu finden.

JEDEN Angriff auf angreifbare Version und Änderungen in neuen Versionen überprüfen.

3.1 Version Rollbacks [nette Übersetzung?]

Ein Angreifer kann eine SSL 3-konforme client-hello-Nachricht so modifizieren, dass der Server eine SSL 2-Verbindung aufbaut. So kann der Angreifer alle Schwächen der älteren Protokollversion ausnutzen. Abhilfe schafft die Einbindung der SSL-Version in das per RSA übertragene pre-master-secret.

Ein Schwachpunkt könnte laut [WS96] immer noch die Wiederaufnahme einer SSL 3-Sitzung durch eine SSL 2-client-hello-Nachricht sein. Dieses sollte in Implementationen verhindert werden.

3.2 Ciphersuite Rollback

Ein für SSL 2.0 bestehender Angriff ermöglichte aktiven Angreifern die während des Handshake-Protokolls übertragenen Listen von unterstützten Cipher Suites zu verändern, so dass schwache kryptographische Verfahren erzwungen werden konnten (oftmals exportgeschwächte Verfahren mit kürzeren Schlüsselängen).

In SSL 3.0 wird dieser Angriff dadurch verhindert, dass die finished-Nachrichten von Client und Server jeweils einen mit dem master-secret berechneten MAC über die Nachrichten des *Handshake*-Protokolls enthalten, der die Integrität dieser Nachrichten bestätigt.

Eine detaillierte Übersicht ist in [WS96] zu finden.

3.3 Verhindern der Change Cipher Spec-Nachricht

Im Sonderfall einer SSL-Verbindung, die lediglich die Integrität der Nachrichten schützen soll, aber nicht verschlüsselt, lässt sich ausnutzen, dass der in der finished-Nachricht gesendete MAC die Change Cipher Spec-Nachricht nicht mit einschließt. Dadurch kann ein aktiver Angreifer diese Nachrichten abfangen und nicht weiterleiten, sodass die Verbindungspartner die Integritätsprüfung nicht einsetzen. Ein Angreifer ist so in der Lage, gesendete Nachrichten zu verändern.

Theoretisch wäre der Angriff unter bestimmten Voraussetzungen und schwacher Kryptographie auch bei verschlüsselten Verbindungen möglich, unter praktischen Gesichtspunkten aber eher unwahrscheinlich.

Die TLS 1.0-Spezifikation verhindert diesen Angriff dadurch, dass sie eine Change Cipher Spec-Nachricht vor der finished-Nachricht explizit vorschreibt.

Details zu diesem Angriff sind in [WS96] zu finden.

3.4 Schwache Cipher Suites

3.5 Bleichenbacher-Angriff und darauf basierende Angriffe

Daniel Bleichenbacher stellte 1998 in [Ble98] einen Adaptive Chosen Ciphertext Angriff gegen RSA-basierte Protokolle vor.

Der Angriff basiert auf dem festen Format nach PCKS #1 formatierter Nachrichten.

3.6 Vaudenay in 2002? Padding Oracle

In [Vau02] beschreibt der Autor einen Angriff zur Erlangung des Klartextes, bei dem das für Blockchiffren nötige Padding im CBC-Modus ausgenutzt wird. Durch das vorgegebene Format des Paddings und da das Padding bei TLS nicht durch den MAC geschützt ist (MAC - then PAD - then Encrypt) ermöglicht es theoretisch in einer relativ kleinen Zahl von Anfragen die Berechnung des Klartextes. Praktisch konnte das Verfahren nicht eingesetzt werden, da SSL 3.0 für Padding- und Entschlüsselungsfehler gleiche Fehlermeldungen ausgibt (TLS 1.0 nicht?) und bei Paddingfehlern mit einem decryption_failed-error die Sitzung abbricht.

Mal gucken

nachschauen!

Begirff: Padding Oracle einbauen

In [CHVV03] beschreiben die Autoren eine Umsetzung des Angriffs auf TLS-gesicherte IMAP-Verbindungen zur Erlangung von Passwörtern. Hierbei wird das Problem ununterscheidbarer und verschlüsselter Fehlermeldungen durch einen Timing-Angriff umgangen. Außerdem bedenken die Autoren das Abbrechen der Sitzung durch Nutzung vieler paralleler Sitzungen mit dem gleichen verschlüsselten Aufruf (wie es bei der Authentifizierung im IMAP-Protokoll der Fall ist).

3.7 Lucky Thirteen

In [AFP13] stellen die Autoren weitere auf [Vau02] basierende Angriffe vor, die ebenfalls auf Timing-Attacken zur Erkennung falschen Paddings und mehrere Verbindungen setzen.

3.8 Bard - IV? Chosen Plaintext Attacks

In [Bar04] stellt der Autor einen Angriff vor, der die Art ausnutzt, wie die für den CBC-Modus nötigen Initialisierungsvektoren (IV) von TLS bereitgestellt werden. Durch die Nutzung des letzten Ciphertextblocks der letzten Nachricht als IV der neuen Nachricht lässt sich unter bestimmten Vorraussetzungen ein Chosen-Plaintext-Angriff durchführen. Der Autor beschreibt eine Möglichkeit unter Nutzung von Browser-Plugins über HTTPS übertragene Passwörter oder

PINs herauszufinden. In [Bar06] verbessert der Autor seinen Angriff durch nie Nutzung von Java-Applets anstelle von Browser-Plugins.

Seit TLS 1.1 werden explizite IV vorgeschrieben. Hierzu besteht jede Nachricht aus einem Ciphertextblock mehr als Klartextblöcken. Dieser erste Block bildet den IV für die restliche Verschlüsselung. Da dieser IV nicht vor dem Senden der Nachricht bekannt ist, wird der hier beschriebene Chosen-Plaintext-Angriff verhindert.

3.9 BEAST

In [DR11] und in einem Konferenzbeitrag auf der ekoparty Security Conference 2011 wurde von den Autoren das Tool BEAST vorgestellt, dass die Ideen aus [Bar04] aufgreift. Hier wurde jedoch auch die praktische Umsetzbarkeit am Beispiel des Entschlüsselns einer über HTTPS gesendeten Session-ID gezeigt.

3.10 CRIME

3.11 Poodle

3.12 Heartbeat?

3.13 FREAK

+weakdh.org?

3.14 Probleme bei Zertifikatvalidierung und bei CAs

3.15 Diebstahl privater Schlüssel

4 Protokollversion 1.3

Literaturverzeichnis

[AFP13] N. J. Al Fardan and K. G. Paterson. Lucky thirteen: Breaking the tls and dtls record protocols. *2014 IEEE Symposium on Security and Privacy*, 0:526–540, 2013.

[Bar04] Gregory V. Bard. Vulnerability of ssl to chosen-plaintext attack. Technical report, 2004.

[Bar06] Gregory V. Bard. A challenging but feasible blockwise-adaptive chosen-plaintext attack on ssl. In *SECRYPT 2006, PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON SECURITY AND CRYPTOGRAPHY, SET'UBAL*, pages 7–10. INSTICC Press, 2006.

[Ble98] Daniel Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1. In *Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '98, pages 1–12, London, UK, 1998. Springer-Verlag.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In *Advances in Cryptology - ASIACRYPT 2000*. Springer Berlin Heidelberg, 2000.

[CHVV03] Brice Canvel, Alain P. Hiltgen, Serge Vaudenay, and Martin Vuagnoux. Password interception in a ssl/tls channel. In *Advances in Cryptology - CRYPTO 2003*, volume 2729, pages 583–599. Springer, 2003.

[DR11] Thai Duong and Juliano Rizzo. Here come the xor ninjas. 2011.

[Eck13] Claudia Eckert. *IT-Sicherheit - Konzepte, Verfahren, Protokolle*. Oldenbourg Verlag, München, 2013.

[KBC97] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, 1997.

[MS13] Christopher Meyer and Jörg Schwenk. SoK: Lessons Learned From SSL/TLS Attacks. In *The 14th International Workshop on Information Security Applications*, 2013.

[Sch06] Bruce Schneier. *Angewandte Kryptographie - Der Klassiker. Protokolle, Algorithmen und Sourcecode in C*. Pearson Studium, München, 2006.

[Sch09] Klaus Schmeh. *Kryptographie - Verfahren, Protokolle, Infrastrukturen*. dpunkt.verlag, Heidelberg, 2009.

[Vau02] Serge Vaudenay. Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS... In *Advances in Cryptology - EUROCRYPT 2002*, pages 534–545. Springer, 2002.

[WS96] David Wagner and Bruce Schneier. Analysis of the SSL 3.0 Protocol. In *The Second USENIX Workshop on Electronic Commerce*, pages 29–40. USENIX Association, 1996.