Universität Hamburg

Fachbereich Informatik

Entwurf vom
10. Juli 2015

Bachelorarbeit

Funktionsweise, Angriffe und Abwehrmechanismen von SSL/TLS

vorgelegt von

Tom Petersen

geb. am 13. Dezember 1990 in Hannover

Matrikelnummer 6359640

Studiengang Informatik

eingereicht am 10. Juli 2015

Betreuer: Dipl.-Inf. Ephraim Zimmer

Erstgutachter: Prof. Dr.-Ing. Hannes Federrath

Zweitgutachter: Dr. Dominik Herrmann

Aufgabenstellung

Die Protokollfamilie SSL/TLS umfasst Techniken zum Schutz von Kommunikationsdaten in IP-basierten Netzen. Ihre weite Verbreitung und Wichtigkeit für die IT-Sicherheit ist historisch gewachsen, und ihr Einsatz erstreckt sich über mittlerweile weit mehr Protokolle der Anwendungsschicht, als nur das ursprünglich anvisierte HTTP. Diese weite Verbreitung hat zwei wesentliche Konsequenzen. Zum einen wurden sowohl die Spezifikation der Protokollfamilie als auch praktische Implementierungen von SSL/TLS Gegenstand zahlreicher Angriffe. Zum zweiten sind ein grundlegendes Verständnis der Funktionsweise von SSL/TLS und der erwähnten Angriffe obligatorisch bei der Entwicklung und Implementierung von verteilter Software, Internetdiensten und Protokollimplementierungen auf der Anwendungsschicht, die mittels SSL/TLS abgesichert werden sollen.

In dieser Bachelorarbeit soll unter Einbeziehung aktueller Entwicklungen und Forschungsergebnisse die Funktionsweise von SSL/TLS, bedeutende Angriffe auf diese Protokollfamilie sowie daraus erarbeitete Anpassungen der Protokollspezifikation und Abwehrmechanismen erläutert und speziell für den Einsatz in der Hochschullehre aufbereitet werden. Darüber hinaus soll ein modular aufgebautes Tool zur Veranschaulichung der SSL/TLS-Funktionsweise sowie deren Angriffe und Abwehrmechanismen entwickelt und prototypisch umgesetzt werden. Der Fokus des Tools liegt in der Demonstration von SSL/TLS und dessen Schwächen mit beliebiger Verständnisvertiefung, sollte allerdings auch um weitere IT-Sicherheitsprotokolle erweiterbar sein.

Zusammenfassung

Abstract schreiben

Für den eiligen Leser sollen auf etwa einer halben, maximal einer Seite die wichtigsten Inhalte, Erkenntnisse, Neuerungen bzw. Ergebnisse der Arbeit beschrieben werden.

Durch eine solche Zusammenfassung (im engl. auch Abstract genannt) am Anfang der Arbeit wird die Arbeit deutlich aufgewertet. Hier sollte vermittelt werden, warum der Leser die Arbeit lesen sollte.

Inhaltsverzeichnis

nette Übersetzung?

1 Einführung

In diesem Exposé werde ich darauf eingehen, warum ich mich in meiner Bachelorarbeit gerne mit dem TLS-Protokoll befassen würde und eine kurze Einführung in das Thema geben. Zuerst werde ich kurz erklären, was mich dazu motiviert hat, mich mit dem Thema zu befassen, und in welche Richtung eine mögliche Bachelorarbeit gehen könnte.

Danach folgt eine Übersicht über die Funktionsweise des Protokolls und bisherige Angriffe gegen aktuelle und frühere Versionen des TLS- bzw. SSL-Protokolls.

Auf geeignete Literatur bzw. Veröffentlichungen wird an den entsprechenden Stellen der Ausarbeitung eingegangen.

1.1 Motivation

TLS ist das wohl am meisten genutzte Sicherheitsprotokoll im Internet. Aus diesem Grund wurde es im Laufe seiner Entwicklung oft untersucht und angegriffen. Dabei sind viele einfache und elegante Angriffe gefunden worden, die zeigen, wie wirksam die kleinsten Schwächen in Protokollen ausgenutzt werden können, und das auch Entscheidungen in scheinbar unbedenklichen Bereichen zu Sicherheitslücken führen können.

In den verschiedenen SSL- und TLS-Versionen wurden viele Änderungen vorgenommen, um diese Angriffe zu verhindern. Daher bietet TLS auch gute Beispiele für Dinge, die bei der Erstellung eines Protokolls bedacht werden müssen, und für wirksame Gegenmaßnahmen gegen bestimmte Angriffe.

1.2 Richtung der Bachelorarbeit

Eine Bachelorarbeit, die sich mit TLS befasst, könnte neben der grundsätzlichen Funktionsweise und einer Übersicht über bisherige Angriffe auf die Änderungen in TLS 1.3 eingehen. Für diese Version liegt ein Draft in 5. Version vom 9. März 2015 vor ([Res15]).

Auch ein konstruktiver Teil wäre denkbar, der sich mit der Überprüfung von TLS-gesicherten Servern oder der Implementation von Angriffen (oder verwundbaren Systemen) zum Beispiel für die Lehre befassen könnte.

2 SSL und TLS - ein Überblick

2.1 Geschichte von TLS

Dieser Überblick basiert in großen Teilen auf [Sch09].

SSL (Secure Socket Layer) bzw. TLS[1] (Transport Layer Security) ist ein zustandsbehaftetes Protokoll, das auf dem TCP-Protokoll[2] der Transportschicht des TCP/IP-Protokollstapels aufbaut.

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

Viele Protokolle der Anwendungsschicht nutzen TLS zur sicheren Datenübertragung, so beispielsweise HTTPS oder FTPS und auch viele Anwendungen übertragen ihre Daten TLS-gesichert.

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

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

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

2.2 Implementierungen

1. Im weiteren Verlauf dieser Arbeit wird der Einfachheit halber lediglich von TLS gesprochen. Bei etwaigen Unterschieden wird explizit auf diese eingegangen.
2. DTLS (Datagram Transport Layer Security) ist ein auf TLS basierendes Protokoll, dass auf UDP aufsetzt.

3 Funktionsweise und Teilprotokolle

Die Informationen in diesem Abschnitt stammen überwiegend aus der TLS 1.2-Spezifikation ([DR08]). Für einen ersten Überblick wurde [Eck13] genutzt.

In der oberen Schicht sind vier Teilprotokolle spezifiziert: Handshake Protocol, Change Cipher Spec Protocol, Alert Protocol und Application Data Protocol.

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

3.1 Record Protocol

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

Kapitel evtl. hinter den Handshake? Und vorher auf Schlüsselberechnung eingehen?

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

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

Aus dem nach Ausführung des Handshake Protocol (siehe nächsten Abschnitt) beidseitig bekannten master-secret werden Schlüssel für die Erstellung des MACs sowie für die Kommunikation zwischen Client und Server berechnet (also insgesamt vier Schlüssel). Dazu werden solange Schlüsselblöcke nach dem folgenden Verfahren erstellt, bis alle Schlüssel konstruiert werden können.

PDUs einbauen

```
key_block = PRF(SecurityParameters.master_secret,
                "key expansion",
                SecurityParameters.server_random +
                SecurityParameters.client_random);
```

Hierbei werden die folgenden Funktionen genutzt:

```
PRF(secret, label, seed) = P_hash(secret, label + seed)

P_hash(secret, seed) = HMAC_hash(secret, A(1) + seed) +
                       HMAC_hash(secret, A(2) + seed) +
                       HMAC_hash(secret, A(3) + seed) + ...

A(0) = seed
A(i) = HMAC_hash(secret, A(i-1))
```

1. Achtung: hier wird MAC-then-Encrypt angewendet. Laut [BN00] ist Encrypt-then-MAC vorzuziehen. In [Kra01] wird dieses Ergebnis bestätigt, aber auch die Sicherheit von authenticate-then-encrypt unter bestimmten Voraussetzungen gezeigt.

```
Client                                                  Server

                              <--------          HelloRequest*
ClientHello                   -------->
                                                   ServerHello
                                                  Certificate*
                                              ServerKeyExchange*
                                             CertificateRequest*
                              <--------       ServerHelloDone
Certificate*
ClientKeyExchange
CertificateVerify*
[ChangeCipherSpec]
Finished                      -------->
                                              [ChangeCipherSpec]
                              <--------               Finished
Application Data              <------->       Application Data
```

Abbildung 3.1: Nachrichtenverlauf beim vollständigen TLS-Handshake. Entnommen aus [DR08].

Die Integrität der Daten, die aus einem der Protokolle der oberen Schicht an das Record Protocol gesendet werden, wird zuerst mit einem MAC geschützt. Dieser wird folgendermaßen berechnet:

```
MAC(MAC_write_key, seq_num +
                   TLSCompressed.type +
                   TLSCompressed.version +
                   TLSCompressed.length +
                   TLSCompressed.fragment);
```

Bei den in TLS verwendeten Cipher-Suites wird das HMAC-Verfahren zur Berechnung des MACs genutzt. Details hierzu sind in [KBC97] zu finden. Die für dieses Verfahren verwendete Hashfunktion wird in der Cipher-Suite angegeben. Bei SSL 3.0 wurde hier noch eine HMAC-ähnliche Konstruktion verwendet.

In Versionsunterschiede auslagern?

Danach wird der Klartext zusammen mit dem MAC optional mit Padding versehen (bei Nutzung von Blockchiffren), verschlüsselt und dann verschickt.

3.2 TLS-Handshake

Das Handshake Protocol dient zur Herstellung einer gesicherten Verbindung. Hierbei werden kryptographische Verfahren zwischen den Kommunikationspartnern vereinbart, optional ihre Identitäten authentifiziert und ein gemeinsames Geheimnis (das sogenannte pre-master-secret) für die bereits beschriebene Generierung der während der eigentlichen Kommunikation verwendeten Schlüssel übertragen oder berechnet. Eine Übersicht über die während eines vollständigen Handshakes ausgetauschten Nachrichten bietet Abbildung 3.1. Im Folgenden werden diese Nachrichten und ihr Aufbau im Detail betrachtet.

HelloRequest*

ClientHello

Die client-hello-Nachricht hat das folgende Format:

```
struct {
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suites<2..2^16-2>;
    CompressionMethod compression_methods<1..2^8-1>;
    select (extensions_present) {
      case false:
          struct {};
      case true:
          Extension extensions<0..2^16-1>;
    };
} ClientHello;
```

Hierbei enthält **client_version** die neueste vom Client unterstützte TLS-/SSL-Version (z.B. 3.3 für TLS 1.2). **random** besteht aus einem 4-Byte großen Zeitstempel (UNIX-Format) und 28 zufälligen Bytes. Die **session_id** dient zur Identifikation einer Sitzung. Sie ist bei dem ersten Handshake leer und kann später dazu verwendet werden, bestehende Sitzungen wieder aufzunehmen (vgl. Abschnitt 3.5). Die Cipher-Suite-Liste enthält alle vom Client unterstützten Cipher-Suites in Reihenfolge seiner Präferenz. Ebenso wird eine Liste von unterstützten Kompressionsalgorithmen übertragen. Optional kann auch eine Liste von gewünschten TLS-Extensions angegeben werden (vgl. refbla).

Abschnitt über TLS extensions

ServerHello

```
struct {
    ProtocolVersion server_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suite;
    CompressionMethod compression_method;
    select (extensions_present) {
      case false:
          struct {};
      case true:
          Extension extensions<0..2^16-1>;
    };
} ServerHello;
```

In **server_version** steht die höchste Version, die Server und Client unterstützen und die damit für die Kommunikation verwendet wird. **random** besteht äquivalent zur client-hello-Nachricht aus einem 4-Byte Zeitstempel und 28 zufälligen Bytes. Die **session_id** enthält entweder eine neu generierte ID, die ID einer wieder aufgenommen Sitzung oder kann auch leer sein, um anzugeben, dass die Sitzung nicht wieder aufgenommen werden kann. In **cipher_suite** und **compression_method** überträgt der Server die von ihm aus den vom

Client übertragenen Listen ausgewählte Cipher-Suite bzw. den Kompressionsalgorithmus. In der Extensionliste gibt der Server alle vom Client gewünschten Extensions an, die er unterstützt.

ServerCertificate*

Server Zertifikat (inklusive öffentlichem Schlüssel des Servers, meist nach X.509v3)

ServerKeyExchange*

CertificateRequest*

ServerHelloDone

ClientCertificate*

ClientKeyExchange

Generierung und Senden von pre-master-secret (48 Byte) verschlüsselt mit öffentlichem Schlüssel des Servers (bei RSA) oder Diffie-Hellman-Verfahren

CertificateVerify*

...auslagern?

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.

```
master_secret = PRF(pre_master_secret, "master secret",
                ClientHello.random + ServerHello.random)[0..47];
```

ChangeCipherSpec**

Change Cipher Spec-Nachricht (kein Bestandteil des Handshake-Protokolls -> eigener Record).

Das Change Cipher Spec Protocol 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.

Finished

Handshakeabschluss: finished mit MAC über alle bisher ausgetauschten Handshake-Nachrichten

```
verify_data = PRF(master_secret, finished_label,
   Hash(handshake_messages)) [0..verify_data_length-1];
```

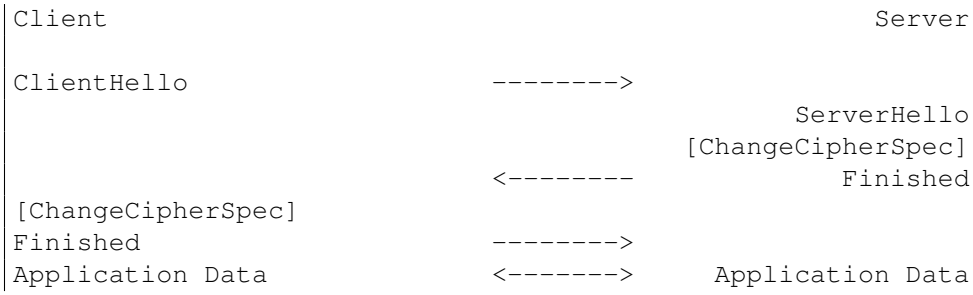

Abbildung 3.2: Nachrichtenverlauf beim abgekürzten TLS-Handshake. Entnommen aus [DR08].

3.3 Alert Protocol

Das Alert Protocol dient dazu, auftretende Fehler zu versenden, die während des Datenaustausches auftreten. Hierbei kann es sich zum Beispiel um fehlgeschlagene Überprüfung von entschlüsselten Nachrichten (bad_record_mac) oder fehlerhafte Zertifikatsüberprüfung (bad_certificate) handeln. Unterschieden wird zwischen Fehlern (fatal alert), die sofort zum Schließen der Sitzung führen, und Warnungen(warning alert). Eine Übersicht über alle Fehler findet sich in Abschnitt 7.2 von [DR08].

3.4 Application Data Protocol

Das Application Data Protocol ist zuständig für das Durchreichen von Anwendungsdaten, die von der Anwendungsschicht gesendet werden sollen.

3.5 Sitzungs- und Verbindungskonzept

Wo passt das Kapitel hier denn am Besten hin?

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-Suiteund 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, je nach verwendeten Verschlüsselungsverfahren einen Initialisierungsvektor, sowie aktuelle Sequenznummern beschrieben.

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.

3.6 Frühere SSL-/TLS-Versionen und TLS 1.3

```
struct {
    ConnectionEnd           entity;
    PRFAlgorithm            prf_algorithm;
    BulkCipherAlgorithm     bulk_cipher_algorithm;
    CipherType              cipher_type;
    uint8                   enc_key_length;
    uint8                   block_length;
    uint8                   fixed_iv_length;
    uint8                   record_iv_length;
    MACAlgorithm            mac_algorithm;
    uint8                   mac_length;
    uint8                   mac_key_length;
    CompressionMethod       compression_algorithm;
    opaque                  master_secret[48];
    opaque                  client_random[32];
    opaque                  server_random[32];
} SecurityParameters;
```

```
 struct {
    ContentType type;
    ProtocolVersion version;
    uint16 length;
    opaque fragment[TLSPlaintext.length];
} TLSPlaintext;

enum { //eher erklaeren als uebernehmen
    change_cipher_spec(20), alert(21), handshake(22),
       application_data(23),(255)
} ContentType;

struct { //eher erklaeren als uebernehmen
    uint8 major;
    uint8 minor;
} ProtocolVersion;
```

```
 struct {
    ContentType type;       /* same as TLSPlaintext.type */
    ProtocolVersion version;/* same as TLSPlaintext.version */
    uint16 length;
    opaque fragment[TLSCompressed.length];
} TLSCompressed;
```

```
struct {
    ContentType type;       /* same as TLSCompressed.type */
    ProtocolVersion version;/* same as TLSCompressed.version */
    uint16 length;
    select (SecurityParameters.cipher_type) { //eher erklaeren als
       uebernehmen -> p.22 ff
        case stream: GenericStreamCipher;
        case block:  GenericBlockCipher;
        case aead:   GenericAEADCipher;
    } fragment;
} TLSCiphertext;
```

4 Angriffe gegen 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.

4.1 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.

4.2 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.

4.3 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.

4.4 Schwache Cipher Suites

Tu es! Exportbeschränte Dinge erwähnen

4.5 Bleichenbacher-Angriff

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.

Weiter ausformulieren

4.6 Padding Oracle Angriff

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 und bei Paddingfehlern mit einem decryption_failed-error die Sitzung abbricht.

TLS 1.0 nicht? Mal gucken

nachschauen!

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).

Begirff: Padding Oracle einbauen

4.7 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.

4.8 Chosen Plaintext Angriff gegen bekannte IVs

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 Voraussetzungen 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 die 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.

4.9 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. Die Autoren erweiterten den Angriff jedoch auf einen sogenannten block-wise chosen-boundary Angriff, bei dem der Angreifer die Lage der Nachricht in den verschlüsselten Blöcken verändern kann. Die Autoren zeigten auch die praktische Umsetzbarkeit am Beispiel des Entschlüsselns einer über HTTPS gesendeten Session-ID.

4.10 CRIME

Auf der ekoparty Security Conference 2012 stellten die Entdecker des BEAST-Angriff einen weiteren Angriff vor, der die (optionale) Kompression in TLS nutzt, um beispielsweise Cookiedaten zu stehlen.

4.11 Poodle

In [MDK14] nutzen die Autoren den erneuten Verbindungsversuch mit älteren Protokollversionen wenn der Handshake fehlschlägt (SSL 3.0 Fallback), der in vielen TLS-Implementationen eingesetzt wird. Darauf aufbauend beschreiben sie einen Angriff, der bestehende Schwächen in der RC4-Chiffre bzw. in der Nicht-Prüfung von Padding im CBC-Modus in SSL 3.0 ausnutzt, um Cookiedaten zu stehlen.

RC4 noch irgendwo unterbringen? Vllt bei den Ciphersuites? http://www.isg.rhul.ac.

4.12 FREAK

Eine Gruppe von Pariser Wissenschaftlern entdeckte eine Möglichkeit, wie ein Angreifer die Kommunikationspartner während des Handshakes zur Nutzung schwacher Kryptographie (RSA export cipher suite) bringen kann. Weiterhin zeigten sie in [BDLF+] die Machbarkeit der Faktorisierung der entsprechenden RSA-Module und die Praxistauglichkeit des Angriffs.

4.13 logjam

In [ABD+] beschreiben die Autoren mehrere Angriffe gegen die Nutzung von Diffie-Hellman(DH)-Schlüsselaustausch während des TLS Handshakes. Ein Angriff richtet sich gegen kleine DH-Parameter (DHE-EXPORT), ein weiterer nutzt die weite Verbreitung von standardisierten DH-Parametern, um mittels Vorberechnung bestimmter Werte schneller diskrete Logarithmen für beim DH-Verfahren gesendete Nachrichten zu berechnen.

4.14 Zertifikate und Verwandtes

Viele Probleme, die in den letzten Jahren aufgetreten sind, betreffen nicht das TLS-Protokoll direkt, sondern die Erstellung und Validierung von (insbesondere) Server-Zertifikaten, und seien deshalb nur am Rande erwähnt. Ein guter Überblick ist in [MS13] zu finden.

Viele dieser Angriffe richteten sich gegen mangelnde Zertifikatvalidierung in TLS-Implementierungen (keine Validierung, keine Überprüfung des Servernamens, Akzeptanz unsignierter oder abgelaufener Zertifikate, ...) oder wenig Sorgfalt bei der Zertifikaterstellung durch Certificate Authorities (Nutzung von MD5, fehlerhafte Validierung von übermittelten Servernamen, fehlerhafte Ausgabe von intermediate-Zertifikaten, mangelhaft abgesicherte Server, ...).

Der Vollständigkeit halber sei hier auch noch die notwendige Sicherheit des privaten Serverschlüssels erwähnt. Gelangt ein Angreifer in seinen Besitz, so kann er den Datenverkehr problemlos mitlesen oder verändern.

Literaturverzeichnis

[ABD+] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. https://weakdh.org/imperfect-forward-secrecy.pdf. Zugriff am 22.05.2015.

[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*, pages 526–540, 2013.

[Bar04] Gregory V. Bard. The Vulnerability of SSL to Chosen Plaintext Attack, 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*, pages 7–10. INSTICC Press, 2006.

[BDLF+] Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, Santiago Zanella-Béguelin, Jean-Karim Zinzindohoué, and Benjamin Beurdouche. FREAK: Factoring RSA Export Keys. https://www.smacktls.com/#freak. Zugriff am 22.05.2015.

[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.

[DR08] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol - Version 1.2. RFC 5246, 2008.

[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.

[Kra01] Hugo Krawczyk. The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?). In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, pages 310–331, London, UK, 2001. Springer-Verlag.

[MDK14] Bodo Möller, Thai Duong, and Krzysztof Kotowicz. This POODLE Bites: Exploiting the SSL 3.0 Fallback, 2014.

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

[Res15] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3 - draft-ietf-tls-tls13-05. Technical report, 2015.

[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.

Todo list