SoK: Lessons Learned From SSL/TLS Attacks

Christopher Meyer and Jörg Schwenk

Horst Görtz Institute for IT-Security, Ruhr-University Bochum
`{christopher.meyer, joerg.schwenk}@rub.de`

Abstract. Since its introduction in 1994 the *Secure Socket Layer (SSL)* protocol (later renamed to *Transport Layer Security (TLS)*) evolved to the de facto standard for securing the transport layer. SSL/TLS can be used for ensuring data confidentiality, integrity and authenticity during transport. A main feature of the protocol is flexibility: Modes of operation and security aims can easily be configured through different cipher suites. However, during the evolutionary development several flaws were found. This paper presents an overview on theoretical and practical attacks of the last 17 years, in chronological order and four categories: Attacks on the Handshake protocol, on the Record and Application Data Protocols, on the PKI infrastructure and various other attacks.
We try to give a short "Lesson(s) Learned" at the end of each paragraph.

1 Introduction

In 1994, *Netscape*[1] addressed the problem of securing HTTP traffic by introducing the Secure Sockets Layer protocol version 2. Over the decades SSL gained improvements, security fixes and from version 3.1 on a new name - *Transport Layer Security*[2]. A key feature of SSL/TLS is the layered design with mainly two blocks:
Handshake Protocol. This is an Authenticated Key Exchange (AKE) protocol for negotiating cryptographic secrets and algorithms.
Record Protocol. This is an intermediate MAC-then-PAD-then-Encrypt layer positioned between the application and the TCP layer.

In addition, error messages are bundled in the **Alert Protocol**, and the one-message **ChangeCipherSpec Protocol** which signalizes activation of the pending state (e.g. switch from unencrypted to encrypted mode).

Due to space limitations a comprehensive introduction to SSL/TLS is skipped, but the specifications of SSL 2.0 [1], SSL 3.0 [2], TLS 1.0 [3], TLS 1.1 [4] and TLS 1.2 [5] are available online[3]. Additionally, a detailed overview on SSL/TLS is e.g. provided by Eric Rescorla in [6]. For convenience, complete communication example illustrating the handshake phase finally leading to the application data phase is given in Figure 1.

2 Attacks on the Handshake Protocol

2.1 Cipher Suite Rollback

The cipher-suite rollback attack, discussed by Wagner and Schneier in [7] aims at limiting the offered cipher-suite list provided by the client to weaker ones or

[1] `http://www.netscape.com`
[2] `http://datatracker.ietf.org/wg/tls/`
[3] SSL version 1.0 was never published.

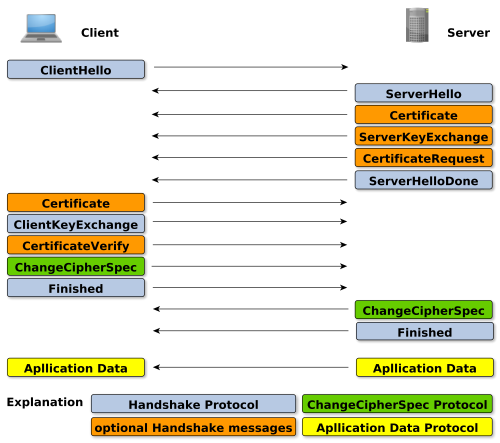

Fig. 1: SSL/TLS communication example

NULL-ciphers. A Man-in-the-middle (Mitm) attacker may alter the ClientHello message and strips of unwanted cipher-suites or replaces the whole cipher-suite list. The server has no real choice - it can either reject the connection or accept the weaker cipher-suite. An example is given in Figure 2.

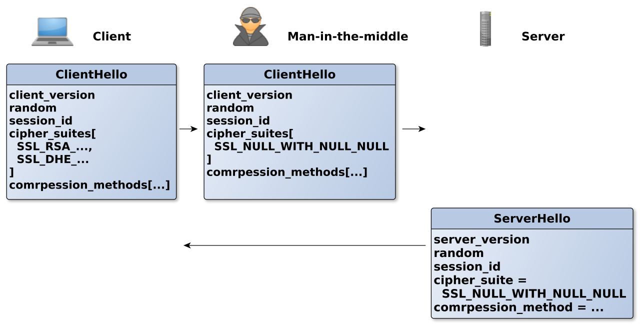

Fig. 2: Cipher-suite rollback attack - *based on:* [7]

The problem was fixed with the release of SSL 3.0 by authenticating *all messages of the Handshake Protocol*. A hash value of all handshake messages sent and received by the client (the server, resp.) was included into the computations of the Client Finished (Server Finished, resp.) message[4].

Lesson learned Authenticate what was sent and received. Theoretically, this idea was put forward in [8] with the concept of *matching conversations*.

2.2 ChangeCipherSpec Message Drop

This simple, but effective attack by Wagner and Schneier in [7] was feasible in SSL 2.0 only. During the handshake phase the cryptographic primitives are determined. For their activation it is necessary for both parties to send a ChangeCipher-

[4] However, this hash value explicitly excludes messages of the Alert and ChangeCipherSpec protocols, leaving room for future attacks.

`Spec` message. This messages informs that the following communication will be secured by the previously agreed parameters. An attacker acting as Mitm could drop the `ChangeCipherSpec` messages and causes both parties to never activate the pending state.

According to Wagner and Schneier the flaw was independently discovered by Dan Simon and addressed by Paul Kocher. The author's recommendation is to ensure that a `ChangeCipherSpec` message is received before accepting the `Finished` message. RFC 2246 [3] (TLS 1.0) enforces exactly this behaviour.

Lesson learned See Section 2.1.

2.3 Key Exchange Algorithm Confusion

Another flaw pointed out by Wagner and Schneier in [7] is related to temporary key material. SSL 3.0 supports the use of temporary key material during the handshake phase (RSA public keys or DH public parameters) signed with a long term key. A problem arises from a missing type definition of the transfered material. Each party implicitly decides, based on the context, which key material is expected and decodes accordingly. This creates the surface for a type confusion attack. This attack is strictly theoretical at time of writing.

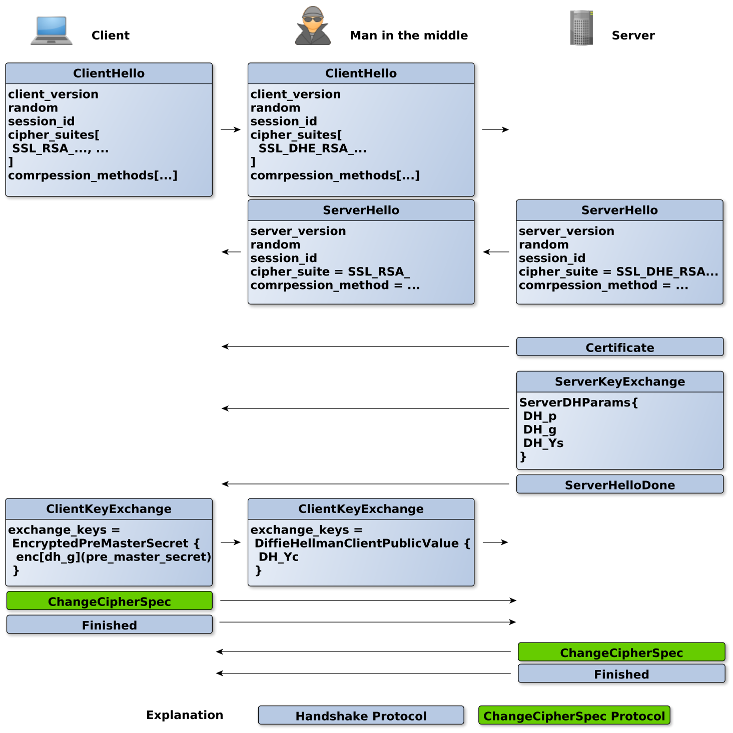

Fig. 3: Key exchange algorithm confusion attack - *based on: [7]*

Figure 3 sketches an attack where a client is fooled in establishing an RSA based key agreement while at the same time performing DHE[5] with the server.

[5] Ephemeral Diffie-Hellman Key Exchange

Lesson learned This attack highlights the need for context-free message structures: Misinterpretation of a received message should be avoided by providing explicit information on the content.

2.4 Version Rollback

Wagner and Schneier described in [7] an attack where a `ClientHello` message of SSL 3.0 is modified to look like a `ClientHello` message of SSL 2.0. This would force a server to rollback to the more vulnerable SSL 2.0.

As a countermeasure (proposed by Paul Kocher), the SSL/TLS version is also contained in the PKCS encoded `PreMasterSecret` of the `ClientKeyExchange` message (when RSA-based cipher suites are used). The countermeasure is sufficient, since SSL 2.0 only supports RSA-based key exchange.

Lesson learned Backward compatibility is a serious security threat: The countermeasure described in Section 2.1 against modification of single messages is of no help, since it was not present in Version 2.0!

2.5 Bleichenbacher's Attack on PKCS#1

In 1998, Daniel Bleichenbacher presented in [9] an attack on RSA based cipher suites. Bleichenbacher utilized the strict structure of the PKCS#1 v1.5 format and showed that it is possible to decrypt the `PreMasterSecret` in a reasonable amount of time. The `PreMasterSecret` in an RSA based cipher suite is a random value generated by the client and sent (encrypted and PKCS #1 formatted) within the `ClientKeyExchange`. An attacker eavesdropping this (encrypted) message can decrypt it later on by abusing the server as a decryption oracle.

Bleichenbacher's attack is based on a) the fixed structure of PKCS#1 and b) a known weakness of RSA to *Chosen Ciphertext Attacks* (cf. [10]). The idea is to *blind* the original ciphertext, pass it to the decrypter and finally separate the blinding value. Depending on the validness of a received PKCS structure the processing at server side differs. In particular, SSL specified different error messages for e.g invalid padding and invalid MAC. With this information one can build an oracle as given in Figure 4.

$$\mathcal{O}_{PKCS}(x) = \begin{cases} \text{true, if } x \text{ is PKCS conforming} \\ \text{false,} \qquad\qquad \text{otherwise} \end{cases}$$

Fig. 4: PKCS oracle

By the use of this oracle it is possible to decrypt the `PreMasterSecret` by continuously blinding the eavesdropped, encrypted message. Based on the oracle's responses the attacker adjusts the blinding value.

Lesson learned Apparently negligible pieces of information such as distinguishable errors, can leverage an attacker to break security. It is necessary to reveal as little information as possible on the internal processing[6].

[6] Especially error messages are a valuable source for information.

2.6 The Rise of Timing Based Attacks

Brumley and Boneh outlined in [11] a timing attack on RSA based SSL/TLS. The attack extracts the private key from a target server by observing the timing differences between sending a specially crafted `ClientKeyExchange` message and receiving an `Alert` message inducing an invalid formatted `PreMasterSecret`. Even a relatively small difference in time allows to draw conclusions on the used RSA parameters. The attack is only applicable in case of RSA based ciphersuites. Additionally, the attack requires the presence of a fine resolutive clock on the attacker's side. OpenSSL was successfully attacked - the problem was caused by performance tweaks made by the OpenSSL library that could, under special circumstances, leverage timing differences during processing. The attack was significantly improved in 2005 by Aciicmez, Schindler and Koc in [12].

As a countermeasure the authors suggest the use of RSA blinding.

Lesson learned Brumley and Boneh demonstrated that designers have to take special care on building implementations with nearly equal response times for each conditional branch of message processing.

2.7 Improvements on Bleichenbacher's Attack

Klíma, Pokorny and Rosa not only improved Bleichenbacher's attack (cf. 2.5) in [13], but were able to break a countermeasure for Bleichenbacher's attack.

Breaking the countermeasure A countermeasure against Bleichenbacher's attack is to generate a random `PreMasterSecret` in any kind of failure and continue with the handshake until the verification and decryption of the `Finished` message fails due to different key material (the `PreMasterSecret` differs at client and server side). Additionally, the implementations are encouraged to send no distinguishable error messages. These countermeasures are regarded as best-practice. Moreover, because of a different countermeasure concerning version roll-back attacks (cf. 2.4) the encrypted data includes not only the `PreMasterSecret`, but also the major and minor version number of the negotiated SSL/TLS version. Implementations should check for equality of the sent/received and negotiated protocol versions. In case of version mismatch some implementations returned distinguishable error messages to the sender (e.g. `decode_error` in case of OpenSSL). An attacker could build a new (bad version) oracle from this, as shown in Figure 5.

$$\mathcal{O}_{BadVersion}(x) = \begin{cases} \text{true,} & \text{if version number is valid} \\ \text{false,} & \text{otherwise} \end{cases}$$

Fig. 5: Bad Version Oracle

By the use $\mathcal{O}_{BadVersion}$ Klíma, Pokorny and Rosa were able to mount Bleichenbacher's attack, in spite recommended countermeasures are present.

Lesson learned Countermeasures against one vulnerability (cf. 2.4) may lead to another.

2.8 ECC Based Timing Attacks

At ESORICS 2011 Brumley and Tuveri [14] presented an attack on ECDSA based TLS connections. Only OpenSSL seemed to be vulnerable.

The problem arose from the strict implementation of an algorithm to speed up scalar multiplications[7]. From a formal point of view, the algorithm is timing resistant, but from an implementational point of view it contained a timing side-channel. Brumley and Tuveri combined this side-channel with the lattice attack of Howgrave-Graham and Smart [17] to recover secret keys.

ECDSA signatures are generated in TLS/SSL when `ECDHE_ECDSA` cipher-suites are used and rely on scalar multiplications. The authors measured the time between the `ClientHello` message and the arrival of the `ServerKeyExchange` message, which contains an ECDSA signature. As this signature can only be created on-the-fly, and not in advance, an adversary is able to measure runtime of the scalar multiplication function and draw conclusions on the input parameters.

Lesson learned Side channels may come from unexpected sources.

2.9 Even More Improvements on Bleichenbacher's Attack

In [18] Bardou, Focardi, Kawamoto, Simionato, Steel and Tsay significantly improved Bleichenbacher's attack (cf. 2.5) far beyond previous improvements (cf. 2.7). The algorithm was fine-tuned to perform faster and with lesser oracle queries. Additionally, the results were combined with previous improvements.

Lesson learned Attacks improve and adjust as time goes by. It is necessary to observe research on attacks - even if they are patched.

2.10 ECC-Based Key Exchange Algorithm Confusion Attack

In [19] Mavrogiannopoulos, Vercauteren, Velichkov and Preneel showed that the key exchange algorithm confusion attack (cf. 2.3) can be applied to ECDH. According to the authors, it is not feasible yet due to computational limitations.

Lesson learned See 2.3

3 Attacks on the Record and Application Data Protocols

3.1 MAC does not cover Padding Length

Wagner and Schneier pointed out in [7] that SSL 2.0 contained a major weakness concerning the Message Authentication Code (MAC) used for ensuring integrity. The MAC covered only data and padding, but left the padding length field unprotected. This may lead to message integrity compromise.

Lesson learned Not only since the introduction of padding oracles by Vaudenay (cf. 3.2) each single bit of information should be considered useful for an attacker. Thus, data should be integrity protected and authenticated to keep the attack vector as small as possible.

3.2 Weaknesses Through CBC Usage

Serge Vaudenay introduced a new attack class - *padding attacks* - and forced the security community to rethink on padding usage in encryption schemes (cf. [20]). These attacks rely on the fact that block encryption schemes operate on blocks of fixed length, but in practice most plaintexts have to be *padded* to fit the requested length (a multiple of the block length). After padding, the input

[7] Montgomery power ladder [15] (with improvements by López and Dahab [16])

data is passed to the encryption function, where each plaintext block (of length of the block size) is processed and chained[8]. This allows to directly influence the decryption process by altering the successive blocks.

In the case of SSL/TLS the receiver may send a `decryption_failure` alert, if invalid padding is encountered. The padding oracle is defined in Figure 6.

$$\mathcal{O}_{Padding}(C) = \begin{cases} \text{true, if } C \text{ is correctly padded} \\ \text{false,} \qquad \text{otherwise} \end{cases}$$

Fig. 6: Padding oracle

With such an oracle and clever changes to the ciphertext an attacker is able to decrypt a ciphertext without knowledge of the key. The optional MAC of SSL/TLS, ensuring message integrity, does not hinder this attack, since padding is not covered by the `MAC`.

As a solution, SSL/TLS defines equal error messages for padding and decryption errors. But there still remains room for timing attacks (cf. 3.13).

Lesson learned Although the attack is not directly applicable to standard SSL/TLS (since *Fatal* errors immediately invalidate the session and accordingly the key material), it is applicable to DTLS (cf. 3.11).

3.3 Information Leakage by the Use of Compression

In [21] Kelsey described a side-channel based on compression. SSL/TLS offers message compression as an optional feature. If compression is used it is possible to correlate output bytes of the compression to (guessed) input bytes. This uses the fact that compression algorithms, when applied to plaintext, reduce the size of the input data - if the guess for a plaintext is right the message size should decrease. With this side-channel, it is possible to draw conclusions on the plaintext. Rizzo and Duong used this observation to attack SSL/TLS (cf. 3.12). Kelsey advices that compression may also cause timing side-channels.

Lesson learned Performance optimizations can lead to side-channels.

3.4 Intercepting SSL/TLS Protected Traffic

In [22] Canvel, Hiltgen, Vaudenay and Vuagnoux extended Vaudenay's attack (cf. 3.2) to decrypt a password from an SSL/TLS secured IMAP session. They suggested three additional attack types:

Timing Attacks The authors concluded that a successful MAC verification needs significantly more time compared to an abortion caused by invalid padding.
Multi-Session Attacks This type requires a critical plaintext to be present in each TLS session (e.g. a password). The attacker checks if a given ciphertext ends with a specific byte sequence instead of trying to guess the whole plaintext.
Dictionary attacks Leveraged by the previous type this attack checks for byte sequences included in a dictionary.

As a recommendation the MAC should also cover the padding, which implies the order PAD-then-MAC-then-Encrypt.

Lesson learned The order of processing makes a big difference.

[8] Mostly according to the Cipher Block Chaining Mode (CBC) scheme which chains consecutive blocks so that a subsequent block is influenced by the output of its predecessor.

3.5 Chosen-Plain-Text Attacks on SSL

Gregory Bard observed in [23] that in a CBC Mode secured connection only the Initialization Vector (IV) of the first plaintext is chosen randomly. All subsequent IVs are simply the last block of the previous encrypted plaintext. This is contrary to cryptography best-practice. An attacker can easily verify if a particular block has a guesed value. Bard recommended as a fix the use of pseudo random IVs or compeltely dropping CBC[9]. The practicability of the attack was proven by Bard two years later (cf. 3.6).

Bodo Möller discovered this vulnerability at the same time[10]. Möller described a fix which was later used by the OpenSSL project: Prepending a single record with empty content, padding and MAC, to each message.

Lesson learned Ignoring security best-practices for the sake of simplicity may lead to vulnerabilities.

3.6 Chosen-Plain-Text Attacks on SSL Reloaded

Bard revisited the attack of Section 3.5 in 2006 [24]. He addressed the same topics as before, but provided an attack sketch how to exploit this problem by the use of a Java applet executed on the victim's machine. As already discussed, the vulnerability was fixed with TLS 1.1, since it dictates the use of explicit IVs.

Rizzo and Duong proved Bard's attack scenario to be applicable, but in a slightly different implementation (by using JavaScript instead of Java applets). The described attack was adopted in their B.E.A.S.T. tool (cf. 3.8).

Lesson learned Not only the protocol has to be considered when evaluating security - the interplay between different layers and applications is relevant, too.

3.7 Traffic Analysis of TLS

George Danezis highlighted in an unpublished manuscript [25] ways how an attacker may use the obvious fact that minimal information, despite the connection is TLS protected, remain unencrypted to analyze and track traffic.

The fields `type`, `version` and `length` of each TLS Record always remain unencrypted - even in an encrypted record. In [7] the authors already criticized the presence of such unauthenticated and unencrypted fields. RFC 2246 [3] is also aware of this information leak and advices to take care of this. Danezis identified the following information leaks:

- Requests to different URLs may differ in length which results in SSL/TLS records of different size.
- Responses to requests may also differ in length, which again yields to SSL/-TLS records of different size.
- Different structured documents may lead to a predictable behavior of the client's application (e.g. a browser normally gathers all images of a website - causing different requests and different responses).
- Content on public sites is visible to everyone, thus linking (e.g. by size) is possible.

[9] TLS 1.1 follows the first recommendation by introducing an explicit `IV` field.

[10] `http://www.openssl.org/~bodo/tls-cbc.txt`

Moreover, an attacker could also actively influence the victim's behavior and gain information by providing specially crafted documents with particular and distinguishable content lengths, structures, URLs or external resources.

The author provides some hints on how the surface of the attack can be limited, but the practicability of the recommended measures remains questionable.

- URL padding - all URLs are of equal length
- Content padding - all content is of equal size
- Contribution padding - all send data is of equal size
- Structure padding - all sites have an equal structure

This flaw was also discussed by Wagner and Schneier in [7] who credited Bennet Yee as the first one describing traffic analysis on SSL. As a countermeasure Wagner and Schneier suggested random length padding not only for block cipher mode, but for all cipher modes. The attack feasibility was proven by Chen, Wang, Wang and Zhang in [26].

Lesson learned Attackers may find ways to use every obtainable part of information for further attacks. More sophisticated attacks are possible if fields are left unauthenticated. Protocol designers and developers should be aware of this fact and sparely disclose *any* information.

3.8 Practical IV Chaining Vulnerability

Rizzo and Duong presented in [27] a tool called B.E.A.S.T. that is able to decrypt HTTPS traffic (e.g. cookies). The authors implemented and extended ideas of Bard [23], [24], Möller and Dai[11]. Rizzo and Duong created a decryption oracle based on the precondition that the IVs used by CBC (the last encryption block of the preceding encryption) are known to the attacker.

To decrypt ciphertexts byte-wise, the authors propose a new kind of attack named *block-wise chosen-boundary attack*. It requires an attacker who is able to move a message before encryption in its block boundaries. This means an attacker may prepend a message with arbitrary data in such a way that it is split into multiple blocks of block-size of the cipher. Based on this, it is possible to split a message of full block-size into two blocks: the first one consisting of arbitrary data and the first byte of the original block and the second block consisting of the remaining bytes and a single free byte. So prefixing a message with an attacker defined amount of data shifts the message (if necessary into a new block). An attacker is absolutely free to prepend any data of her choice and length. An example is given in Figure 7.

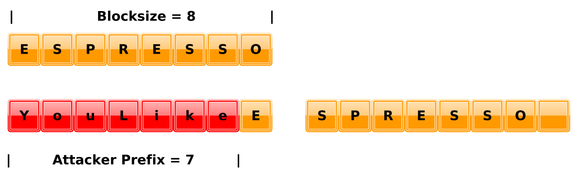

Fig. 7: Example boundary shifting

[11] http://www.weidai.com/ssh2-attack.txt

Full message decryption To decrypt a full message the attacker adjusts the amount of random prefix data so that the next *unknown* byte is always the last byte in a block. This means that the message is shifted in such a way, that the scenario illustrated in Figure 7 applies to the next unknown byte. The unknown byte becomes the last and only byte of a single block unknown to the attacker. Finally, this leads to a byte by byte message decryption.

Rizzo and Duong demonstrated how B.E.A.S.T. could be used to decrypt HTTPS secured cookies. Due to this massive vulnerability, migration to TLS Version 1.1 has been recommended since by IETF.

Lesson learned Theoretical only vulnerabilities can turn into practice.

3.9 Short Message Collisions and Busting Length Hiding

In [28] Paterson, Ristenpart and Shrimpton outlined an attack related to the MAC-then-PAD-then-Encrypt scheme in combination with short messages. In particular, their attack is applicable if all parts of a message (message, padding, MAC) fit into a single block of the cipher's block-size. Under special preconditions the authors described the creation of different ciphertexts leading to the same plaintext message.

The surface for this attack is limited, since the preconditions (message, padding and MAC have to fit into a single block) are quite strong.

Lesson learned MIN/MAX lengths for input and output data of cryptographic algorithms are beneficial.

3.10 Message Distinguishing

Paterson et al. extended in [28] the attack described in 3.9 enabling an attacker to distinguish between two messages. The attack is based on clever modification of the eavesdropped ciphertext so that it either passes the processing or leads to an error message. Based on the outcome (error/no error) it is possible to determine which content was send. The attack works only if the possible contents are of different, short length. At least, the attack remains unexploitable (until the day of writing) due to the fact that it is only possible for 80 bit truncated MACs.

Lesson learned See 3.9 and always remember 3.8.

3.11 Breaking DTLS

In [29] AlFardan and Paterson applied Vaudenay's attack (cf. 3.2) to DTLS. DTLS is a slightly different version of regular TLS adjusted to unreliable transport protocols, such as UDP. Compared to TLS, there are two major differences:

1. Complete absence of `Alert` messages
2. Messages causing protocol errors (bad padding, invalid MAC, ...) are simply dropped, instead of causing a connection abort invalidating the session keys

Vaudenay's attack works on DTLS since bad messages do not cause session invalidation. But with the lack of error messages there is no feedback whether the modified messages contained a valid padding or not. The authors adjusted Vaudenay's algorithms to use a timing oracle.

OpenSSL and GnuTLS were analyzed , both vulnerable to a timing oracle enhanced version of Vaudenay's attack. According to the authors, it was necessary to disable the protocol's *anti-replay* option, which is enabled by default.

Lesson learned The authors recommend that defining standards only by specifying differences to other standards should be avoided.

3.12 Practical Compression Based Attacks

In 2012, Rizzo and Duong presented the *C.R.I.M.E.* attack tool which targets HTTPS and is able to decrypt traffic, enabling cookie stealing and session takeover. It exploits a known vulnerability caused by the use of message compression (cf. 3.3). The attack requires compression to be enabled in an SSL/TLS session.

Basically, an attacker prefixes the secret with guessed subsequences and observes if it leads to compression (by observing the resulting ciphertext length). A decreased ciphertext length implies redundancy, so it is very likely that the guessed, prefixed subsequence caused redundancy in the plaintext. This implies that a guess, having something in common with the secret, will have a higher compression rate leading to a shorter output. When such an output is detected the attacker knows that the guess has something in common with the secret.

Lesson learned See 3.8.

3.13 Timing Based Side-Channels Strike Back

In February 2013, AlFardan and Paterson introduced the *Lucky Thirteen* attack [30]. The attack enables plaintext recovery against TLS and DTLS by exploiting the already discussed MAC-then-PAD-then-Encrypt design of the protocols. The author's tamper with the padding data and measure the time needed for MAC calculation on server side. By cleverly choosing the padding values it is possible to distinguish valid from invalid paddings, leaking information about the plaintext. Basically, the time required for MAC computation is of significance - the decryption oracle is based on these timing differences. The attack can be enhanced when combined with techniques of the B.E.A.S.T. attack (cf. 3.8).

Lesson learned Weaknesses may turn into practice and get even worse when combined with known attacks.

3.14 RC4: A Vulnerable Alternative

The stream cipher RC4 is often proposed as a countermeasure to Padding Oracle Attacks (cf. 3.2). Unfortunately, RC4 is known to have vulnerabilities and weaknesses[12]. In 2013, Isobe, Ohigashi, Watanabe and Morrii identified in [32] biases in the initial bytes of RC4 keystreams that can be used to perform plaintext recovery of encrypted ciphertexts (similar results have been discussed independently[13], but are not yet published) and thus break SSL/TLS encryption.

Lesson learned Take known weaknesses seriously!

4 Attacks on the PKI

4.1 Weak Cryptographic Primitives Lead to Colliding Certificates

Lenstra, Wang and de Weger described in 2005 how an attacker could create valid X.509 certificates with collinding MD5 hash values [33]. With that it is possible to impersonate clients or servers - this enables hard to detect attacks.

The practicality of the attack was demonstrated in 2008 by Sotirov, Stevens, Appelbaum, Lenstra, Molnar, Osvik and de Weger[14] who were able, through

[12] That, in the past, lead to the decline of e.g. WEP [31]

[13] `http://www.isg.rhul.ac.uk/tls/`

[14] `http://www.win.tue.nl/hashclash/rogue-ca/`

clever interaction between certificate requests from a legal CA and a massively parallel search for MD5 collisions, to create a valid CA certificate for TLS.

Lesson learned As long as user agents accept MD5 certificates, the surface still exists. Weak algorithms may lead to complete breach of the security.

4.2 Weaknesses in X.509 Certificate Constraint Checking

In 2008, US hacker Matthew Rosenfeld, better known as Moxie Marlinspike, published a vulnerability report [34] affecting the certificate basic constraint validation of Microsoft's Internet Explorer (IE). IE did not check if certificates were allowed to sign sub-certificates. Any valid certificate, signed by a trusted CA, was allowed to issue sub-certificates for *any* domain.

The tool `sslsniff`[15] provides a proof of concept implementation with the attacker acting as Mitm, issuing certificates for a requested domain on the fly.

Lesson learned The attack relies on a specific implementation bug and has been fixed. However, certificate validation is a critical step. This again stresses the need for well-written specifications sketching all security related processing steps in detail and, in turn, obligates developers to implement exactly as outlined.

4.3 Attacks on Certificate Issuer Application Logic

Attacks on the PKI by exploiting implementational bugs on CA side were demonstrated by Marlinspike in [35], who was able to trick the CA's issuance logic by using specially crafted domain strings. Marlinspike gained certificates for arbitrary domains, issued by trusted CAs.

Marlinspike made use of the encoding of X.509 - ASN1. ASN1 supports multiple String formats, all leading to slightly different PASCAL String representation conventions. PASCAL and C store strings differently, the first: length prefixed, and the other: `NULL` terminated.

This prepares the way for the `NULL`-Prefix attack: A sample domain name which could be used in a Certificate Signing Request (CSR) is the following `www.targetToAttack.com\0.example.com`, assuming that the attacker is the owner of `example.com`. The attack works, because the CA logic only checks the TLD (`example.com`). The leading `NULL`-byte (`\0`) is valid because of ASN1's length-prefixed representation (where `NULL`-bytes within the payload String are valid). When the prepared domain String is presented to common application logic (mostly written in languages representing Strings `NULL`-terminated), such as e.g. most browsers, the String is prematurely terminated. As a result only the String afore the `NULL` byte (`www.targetToAttack.com`) is being validated.

A specialization of the attack are wild-card certificates. The asterisk (*) can be used to create certificates, valid - if successfully signed by a trusted CA - for **any** domain (e.g., `*\0.example.com`).

Lesson learned Certification authorities should be prepared to deal with different encodings and security issues related to this.

4.4 Attacking the PKI

Marlinspike described in [36] an attack that aims at interfering the infrastructure to revoke certificates. By the use of the Online Certificate Status Protocol

[15] `http://www.thoughtcrime.org/software/sslsniff/`

(OCSP) a client application can check the revocation status of a certificate. The repsonse contains a field `responseStatus` which is not protected by a signature.

An attacker acting as Mitm could respond to every query with `tryLater`. Due to lack for a signature the client has no chance to detect the spoofed response. Thereby, a victim is not able to query the revocation status of a certificate.

Lesson learned Every sensitive message parts should be integrity protected and authenticated. If necessary, encryption should additionally be used for confidential data. If real-time checks on a PKI are required, unsigned responses should lead to a halt in protocol execution.

4.5 Wildcard Certificate Validation Weakness

Moore and Ward published a Security Advisory [37] concerning wildcard (*) usage when IP addresses are used as CN URI in X.509 certificates. According to RFC 2818 [38] wildcards are not allowed for IP addresses. The authors found multiple browsers treating IP addresses including wildcard characters as certificate CN as valid and matching. The authors could fool browsers to accept issued certificates with CN="*.168.3.48". This certificate was treated as valid for any server with a ".168.3.48" postfix.

Lesson learned Certificate validation is challenging.

4.6 Conquest of a Certification Authority

In March 2011 the *Comodo CA Ltd.* Certification Authority (CA) was successfully compromised [39]. An attacker used a reseller account to issue 9 certificates for popular domains. Except rumors, the purpose of the attack remains unclear.

Lesson learned Certification authorities have to protect their critical infrastructure with strong security mechanisms.

4.7 Conquest of Another CA

Soon after the attack on Comodo, a Dutch Certification Authority - *DigiNotar* - was completely compromised by an attacker [40]. In contrast to the Comodo impact, the attacker was able to gain control over the DigiNotar infrastructure. The attack discovery was eased by Google's Chrome web browser who complained about mismatching certificates for Google-owned domains. The browser stores hard coded copies of the genuine certificates for Google and thus was able to detect bogus certificates.

Lesson learned Beside the lesson learned from 4.6, it can be seen that mechanisms like malware and intrusion detection must be present in CA systems.

4.8 Risks of Unqualified Domain Names

The risks of unqulified domain names such as e.g. `mail`, `exchange` or `wiki` were discussed in a blog entry by Chris Palmer [41].The author used the EFF SSL Observatory[16] to identify certificates with unqulified domain names, issued by trusted CAs. This could leverage Man-in-the-middle attacks.

Lesson learned Only rely on fully qualified domain names.

[16] `https://www.eff.org/observatory`

4.9 CA's Issuing Weak Certificates

DigiCert Malaysia was blamed for issuing 22 certificates with weak 512-bit RSA keys and no certificate revocation extensions[17]. As a consequence *Entrust* revoked *DigiCert Malaysia*'s intermediate CA certificate.

Lesson learned Strong algorithms and key lengths are of major importance.

4.10 Attacks on Non-Browser Based Certificate Validation

Georgiev et al. [42] uncovered that widespread libraries for SSL/TLS suffer from vulnerable certificate validation implementations. As major causes for these problems bad and misleading API specifications, lacking interest for security concerns at all and the absence of essential validation routines were identified.

Especially, the following security tasks and robustness of the libraries' code responsible for these tasks are considered:

- Certificate chaining and verification
- Host name verification
- Certificate revocation checks
- X.509 extension handling and processing

Exploiting these vulnerabilities may lead to Mitm and impersonation attacks.

Lesson learned Clean, simple and well documented APIs are important.

4.11 Mis-Issued Certificates

A flawed business process at *TURKTRUST* accidently issued 2 intermediate CA certificates [43]. The issue was discovered by Google's Chrome Browser when it recognized bogus certificates for `*.google.com`.

Lesson learned Means for the detection of illegal certificates are needed.

5 Various Attacks

5.1 Random Number Prediction

In January 1996, Goldberg and Wagner published an article [44] on the quality of random numbers used for SSL connections by the Netscape Browser. The authors identified striking weaknesses in the algorithm responsible for random number generation. The algorithm's entropy relied on a few, predictable values.

Lesson learned Good (pseudo) random number generators (PRNGs) are essential for cryptography (cf. 5.2).

5.2 Weak Random Numbers

In 2008, Luciano Bello [45] observed during code review that the PRNG of Debian-specific OpenSSL was predictable, due to an implementation bug. A Debian-specific patch removed two very important lines in the `libssl` source code responsible for providing adequate entropy[18]. This allowed a brute force attack - the key space was significantly limited without these code lines.

Lesson learned Developers should comment security critical parts of source code, exactly explain the intention and highlight the consequences when altered. Beyond this, test cases targeting the critical code lines should be provided.

[17] `http://www.entrust.net/advisories/malaysia.htm`

[18] `http://anonscm.debian.org/viewvc/pkg-openssl/openssl/trunk/rand/md_rand.c?p2=%2Fopenssl%2Ftrunk%2Frand%2Fmd_rand.c&p1=openssl%2Ftrunk%2Frand%2Fmd_rand.c&r1=141&r2=140&view=diff&pathrev=141`

5.3 Denial of Service Enabled by Exceptions

In [46] Zhao, Vemuri, Chen, Chen, Zhou and Fu provided attacks on the TLS handshake which leads to an immediate connection shutdown.

- The first attack targets the Alert protocol of TLS and makes use of the fact that, due to yet missing cryptographic primitives during the handshake phase, all `Alert` messages remain strictly unauthenticated and thus spoofable. This enables an obvious, but effective attack: Spoofing *Fatal* `Alert` messages which cause immediate connection shutdowns.
- The second attack simply confuses a communication partner by sending either misleading, replayed or responding with wrong messages according to the expected handshake flow.

Lesson learned Even obvious and self-evident weaknesses have to be discussed and focus of research.

5.4 Renegotiation Flaw

Ray and Dispensa discovered in [47] a serious flaw induced by the renegotiation feature of TLS. The flaw enables an attacker to inject data into a running connection without destroying the session. The attacker gets no authentication cookie in plaintext, but her request is constructed to be concatenated on server side in a special way - the attacker is at no time able to decrypt traffic. *Anil Kurmus* proved the flaw to be practical[19] by stealing confidential data from Twitter sessions. The attack was slightly modified (an unfinished `POST` request was used), but the idea remained the same.

Lesson learned When switching security contexts it needs to be guaranteed that there is no pending data left.

5.5 Disabling SSL/TLS at a Higher Layer

In February 2009, Marlinspike released `sslstrip`[20] a tool which disables SSL/-TLS at a higher layer. As a precondition it is necessary for an attacker to act as Mitm. To disable SSL/TLS the tool sends `HTTP 301` - permanent redirection - responses and replaces any occurrence of `https://` with `http://`. This causes the client to move to the redirected page with SSL/TLS turned off. Finally, the attacker opens a fresh session to the (requested) server and passes-through or alters any client and server data. The attack sketch is outlined in Figure 8.

Lesson learned Proper visualization of secured connections in the user agents is necessary.

5.6 Computational Denial of Service

In 2011, the German Hacker Group *The Hackers Choice* released a tool called `THC-SSL-DoS`[21], which creates huge load on servers by overwhelming the target with SSL/TLS handshake requests. Assuming that the majority of computation

[19] `http://www.securegoose.org/2009/11/tls-renegotiation-vulnerability-cve.html`

[20] `http://www.thoughtcrime.org/software/sslstrip/`

[21] `http://www.thc.org/thc-ssl-dos/`

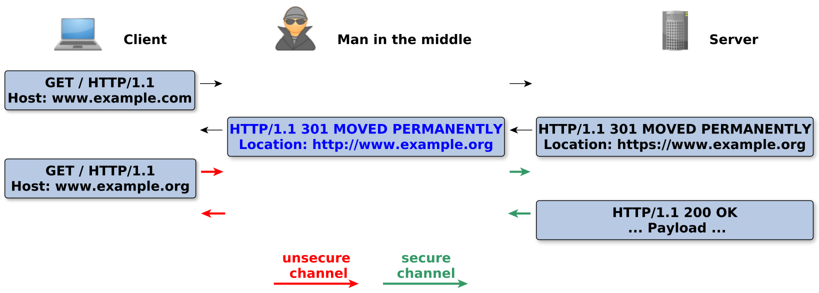

Fig. 8: Example scenario for a SSL stripping attack

during a handshake is done by the server, the attack creates more system load on the server than on the own device - leading to a Denial of Service.

Lesson learned When dealing with DoS attacks, cryptography is part of the problem, not a solution.

6 Conclusion

Summarizing the lessons learned leads to some basic hints:

1. Theoretical attacks can turn into practice
2. Side channels may appear at different layers in different situations
3. Reliable cryptographic primitives are important
4. Processes must leak as little information as possible
5. Specifications have to be implemented without own improvements
6. Critical parts in specifications and source code have to be highlighted
7. Specifications have to be verbose, unambiguous and technically detailed
8. Details on requirements and preconditions are necessary
9. Data has to be protected (authenticated, integrity ensured, encrypted, etc.)
10. The interplay between different layers must be part of the security analysis
11. Flexibility mostly means additional risks
12. Always be careful and alarmed

DoS attacks remain a future problem. Means to lower the attack surface emerged to be of increased relevance.

References

1. Hickman, K.: The SSL Protocol. Internet Draft (April 1995)
2. Freier, A., Karlton, P., Kocher, P.: The Secure Sockets Layer (SSL) Protocol Version 3.0. RFC 6101 (August 2011)
3. Dierks, T., Allen, C.: The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard) (January 1999)
4. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard) (April 2006)
5. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard) (August 2008)
6. Rescorla, E.: SSL and TLS: Designing and Building Secure Systems. Addison-Wesley (2001)
7. Wagner, D., Schneier, B.: Analysis of the SSL 3.0 protocol. The Second USENIX Workshop on Electronic Commerce Proceedings (1996)

8. Bellare, M., Rogaway, P.: Entity authentication and key distribution. (1994)
9. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Krawczyk, H., ed.: Advances in Cryptology — CRYPTO '98. Volume 1462 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (1998) 1–12
10. Davida, G.: Chosen Signature Cryptanalysis of the RSA (MIT)Public Key Cryptosystem. Technical report (1982)
11. Brumley, D., Boneh, D.: Remote timing attacks are practical. In: Proceedings of the 12th conference on USENIX Security Symposium - Volume 12. SSYM'03, Berkeley, CA, USA, USENIX Association (June 2003)
12. Aciicmez, O., Schindler, W., Koc, C.: Improving Brumley and Boneh timing attack on unprotected SSL implementations. In: Proceedings of the 12th ACM conference on Computer and communications security, ACM (November 2005)
13. Klíma, V., Pokorný, O., Rosa, T.: Attacking RSA-Based Sessions in SSL/TLS. In Walter, C., Koc, C., Paar, C., eds.: Cryptographic Hardware and Embedded Systems - CHES 2003. Volume 2779 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (September 2003)
14. Brumley, B., Tuveri, N.: Remote Timing Attacks Are Still Practical. In Atluri, V., Diaz, C., eds.: Computer Security - ESORICS 2011. Volume 6879 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (September 2011)
15. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. MC (48) (1987)
16. López, J., Dahab, R.: Fast Multiplication on Elliptic Curves Over GF(2m) without precomputation. In Koc, C., Paar, C., eds.: Cryptographic Hardware and Embedded Systems. Volume 1717 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (1999)
17. Howgrave-Graham, N.A., Smart, N.P.: Lattice Attacks on Digital Signature Schemes. Designs, Codes and Cryptography **23** (2001)
18. Bardou, R., Focardi, R., Kawamoto, Y., Simionato, L., Steel, G., Tsay, J.K.: Efficient Padding Oracle Attacks on Cryptographic Hardware. Rapport de recherche RR-7944, INRIA (April 2012)
19. Mavrogiannopoulos, N., Vercauteren, F., Velichkov, V., Preneel, B.: A Cross-Protocol Attack on the TLS Protocol. In: Proceedings of the 2012 ACM conference on Computer and communications security. CCS '12, ACM (October 2012)
20. Vaudenay, S.: Security Flaws Induced by CBC Padding — Applications to SSL, IPSEC, WTLS... In Knudsen, L., ed.: Advances in Cryptology — EUROCRYPT 2002. Volume 2332 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (April 2002)
21. Kelsey, J.: Compression and information leakage of plaintext. In: Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers. Volume 2365 of Lecture Notes in Computer Science., Springer (November 2002)
22. Canvel, B., Hiltgen, A., Vaudenay, S., Vuagnoux, M.: Password Interception in a SSL/TLS Channel. In Boneh, D., ed.: Advances in Cryptology - CRYPTO 2003. Volume 2729 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (August 2003)
23. Bard, G.V.: The Vulnerability of SSL to Chosen Plaintext Attack. IACR Cryptology ePrint Archive **2004** (May 2004)
24. Bard, G.V.: A Challenging But Feasible Blockwise-Adaptive Chosen-Plaintext Attack on SSL. In: SECRYPT 2006, Proceedings of the International Conference on Security and Cryptography, INSTICC Press (August 2006)

25. Danezis, G.: Traffic Analysis of the HTTP Protocol over TLS. Unpublished manuscript
26. Chen, S., Wang, R., Wang, X., Zhang, K.: Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow. In: Proceedings of the 2010 IEEE Symposium on Security and Privacy. SP '10, IEEE Computer Society (May 2010)
27. Rizzo, J., Duong, T.: Here Come The XOR Ninjas (May 2011)
28. Paterson, K.G., Ristenpart, T., Shrimpton, T.: Tag size does matter: attacks and proofs for the TLS record protocol. In: Proceedings of the 17th international conference on The Theory and Application of Cryptology and Information Security. ASIACRYPT'11, Berlin, Heidelberg, Springer-Verlag (December 2011)
29. AlFardan, N., Paterson, K.: Plaintext-Recovery Attacks Against Datagram TLS. In: Network and Distributed System Security Symposium (NDSS 2012). (February 2012)
30. AlFardan, N., Paterson, K.: Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. In: Proceedings of the 2013 IEEE Symposium on Security and Privacy. SP '13, IEEE Computer Society (February 2013)
31. Fluhrer, S.R., Mantin, I., Shamir, A.: Weaknesses in the Key Scheduling Algorithm of RC4. In: Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography. SAC '01, Springer-Verlag (August 2001)
32. Takanori Isobe, Toshihiro Ohigashi, Y.W., Morii, M.: Full Plaintext Recovery Attack on Broadcast RC4. In: Proc. the 20th International Workshop on Fast Software Encryption (FSE 2013). (March 2013)
33. Lenstra, A., Wang, X., de Weger, B.: Colliding X.509 Certificates. Cryptology ePrint Archive, Report 2005/067 (March 2005)
34. Rosenfeld, M.: Internet Explorer SSL Vulnerability (May 2008)
35. Rosenfeld, M.: Null Prefix Attacks Against SSL/TLS Certificates (February 2009)
36. Rosenfeld, M.: Defeating OCSP With The Character '3' (July 2009)
37. Moore, R., Ward, S.: Multiple Browser Wildcard Cerficate Validation Weakness (July 2010)
38. Rescorla, E.: HTTP Over TLS. RFC 2818 (May 2000)
39. Comodo CA Ltd.: Comodo Report of Incident - Comodo detected and thwarted an intrusion on 26-MAR-2011. Technical report (March 2011)
40. Fox-IT: Black Tulip - Report of the investigation into the DigiNotar Certificate Authority breach. Technical report (August 2012)
41. Palmer, C.: Unqualified Names in the SSL Observatory (April 2011)
42. Georgiev, M., Iyengar, S., Jana, S., Anubhai, R., Boneh, D., Shmatikov, V.: The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software. In: ACM Conference on Computer and Communications Security. (2012)
43. Langley, A.: Enhancing digital certificate security (January 2013)
44. Goldberg, Wagner: Randomness and the Netscape browser. Dr. Dobb's Journal (January 1996)
45. Weimer, F.: DSA-1571-1 openssl – predictable random number generator (May 2008)
46. Zhao, Y., Vemuri, S., Chen, J., Chen, Y., Zhou, H., Fu, Z.: Exception triggered DoS attacks on wireless networks. In: Proceedings of the 2009 IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2009. (June 2009)
47. Ray, M., Dispensa, S.: Renegotiating TLS. Technical report, PhoneFactor, Inc. (November 2009)