Seminar Report

A Messy State of the Union: Taming the Composite State Machines of TLS

Michael Plattner
M.Plattner@student.uibk.ac.at

02 June 2017

Abstract

In this seminar work we talk about the paper published in the *Communications of the ACM* [1]. The article is about Transport Layer Security, especially testing some implementations of the theoretical TLS standard. Those tests are based on state machines and depending on the states, the transitions into other states were tested.

List of Figures

List of Tables

Contents

1 Introduction

The Transport Layer Security (TLS) protocol, or as it's sometimes referred to, the Secure Sockets Layer (SSL) protocol, is a stateful, connection-oriented, client-server protocol. [3]
In the OSI Reference [4] model TLS is located in the Layer 4, the Transport Layer. The main use of TLS is encrypting data for a secure communication. The current version is 1.2 and described in the RFC 5246.[1]

The protocol consists of 2 different parts. The *Handshake Protocol*, which consists of deciding the cipher suite and the authentication features. On the other hand there comes the *record protocol*, which secures the connection with the session key established. Furthermore it provides data integrity and data origin.

In this paper we focus on the Handshake Protocol. According to [1] is the connection secure, if it uses a secure key exchange and a strong record encryption scheme. So the problem can be put down to the security of these building blocks.

2 TLS State Machine

The TLS State machine is the main machine on which the article [1] operates on. The state machine is used to find security issues in the TLS connection. The states defined in the TLS state machine are shown in Table 1.

[1] https://tools.ietf.org/html/rfc5246

State
ClientHello
ServerHello
ServerCertificate
ServerKeyExchange
CertificateRequest
ServerHelloDone
ClientCertificate
ClientKeyExchange
ClientCertificateVerify
ClientCCS
ClientFinished
ServerNewSessionTicket
ServerCCS
ServerFinished
ApplicationData

Table 1: States of the TLS state machine

2.1 TLS Connection

A connection starts from the client, with a `ClientHello`. The server answers with a ServerHello, followed by a `ServerCertificate`. After the certificate there comes the `ServerKeyExchange` if it is required by the key exchange algorithm. After that, comes the `CertificateRequest` from the client, followed by a `ServerHelloDone`. If there is a CertificateRequest, the client offers a `ClientCertificate`. In addition there must be a `ClientKeyExchange` which is followed by `ClientCertificateVerify`. If the verification was successful, the `ClientCCS` (where CCS stands for change cipher spec) will be submitted. With that being done the Client has `ClientFinished` and the server creates a new `ServerNewSessionTicket`. With the ticket enhances a `ServerCCS` and with that the server has `ServerFinished`. From that Point on `ApplicationData` can be sent.

3 TLS Problems

In theory flow Graph of a TLS connection is secure, but some implementations do allow transitions from a state to another state without authentication of Server. So man-in-the-middle attacks may give a false sense of security, while retrieving all information that is being sent via the Channel.
The four main problems we want to address are: [1]

- FREAK
- SKIP Ephemeral
- SKIP Verify
- SKIP Exchange

3.1 FREAK

HOP to `RSA_EXPORT` comes from the early days of SSL and TLS 1.0. Several Cipher-suites use RSA public keys for the `ServerKeyExchange` that have a size not larger than 512 bits. By Sending the `ServerKeyExchange` during regular RSA handshakes some TLS implementations accepted this message and fallback to RSA_EXPORT with 512-bit RSA key. So now the server certificate is no longer encrypted secure enough, because 512-bit RSA keys can be factored within hours. This is problematic because most system store the RSA_EXPORT for a lifetime, so precomputing of the key can be done.

3.2 SKIP Ephemeral

SKIP Ephemeral or forward secrecy[2] downgrade is important for Browsers that support TLS false start[3]. Even though the handshake may not have completed because of high

[2] If server has been impersonated, still ensure prior connections to be secret.
[3] https://tools.ietf.org/html/rfc7918

latency times, the client starts sending encrypted application data to the server. This data may contain sensible information. Because the cipher spec may not have been completed and hence need protection of forward secrecy. The problem in some implementations is that they allow `ServerKeyExchange` to be skipped and therefore a ciphers without forward secrecy gets used.

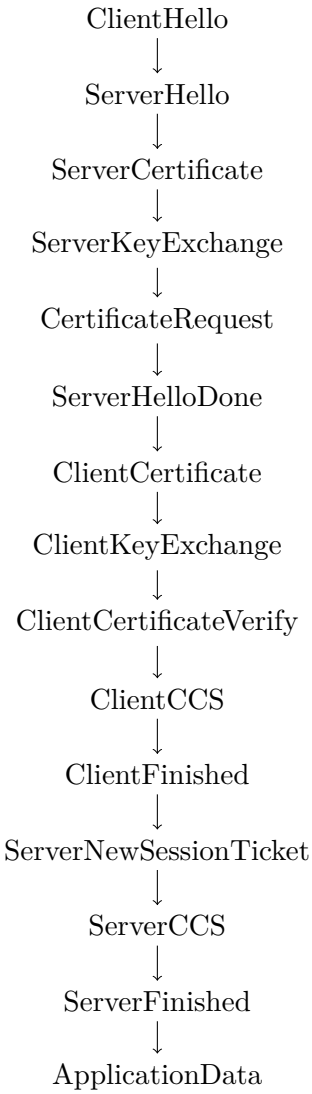

Figure 1: Expected State flow of TLS

3.3 SKIP Verify

SKIP verify allows a malicious client to skip the `ClientCertificateVerify` message. This message usually guarantees the ownership of the client, but with this bug a malicious client is being interpreted as a valid and therefore can access sensible data from the client he tries to get information from. The only thing that the attacker must know is the public certificate of the client and he than can log in as that specific user to the Website.

3.4 SKIP Exchange

SKIP exchange or server impersonation is when the whole key exchange between the server and the client is skipped and the TLS state goes from `ServerCertificate` to `ServerFinished`. The whole data sent through the connection will not be encrypted! This impersonation of the server is handled as following: A Client C wants to connect to a secured Server S. The network attacker M hijacks the TCP connection and sends the Server certificate of S to the Client C. Then he skips all messages that should be in the Key Exchange from the Server and sends data to C that is not encrypted. Now C misinterprets the connection as secure data from S. Now all sensible data that will be sent to S is actually sent to M

4 Conclusion

We just mentioned a few problems that concern security in the TLS protocol. There are many further issues that may cause problem, e.g. as mentioned in this paper about previous attacks. [2]. Hopefully we can find good ways to test security issues automatically, like with the state machine mentioned in [1]. There are still many flaws and a lot of different perspectives have to be taken into consideration, but overall the security increases over time with further research. Considering the huge impact that TLS around the globe, many security fixes will become available, but not to forget more risk and smarter attacks may be found.

References

[1] Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Jean Karim Zinzindohoue. A messy state of the union: Taming the composite state machines of tls. *Commun. ACM*, 60(2):99–107, January 2017.

[2] Christopher Meyer and Jörg Schwenk. Lessons learned from previous ssl/tls attacks-a brief chronology of attacks and weaknesses. *IACR Cryptology EPrint Archive*, 2013:49, 2013.

[3] Sean Turner. Transport layer security. *IEEE Internet Computing*, 18(6):60–63, 2014.

[4] Hubert Zimmermann. Osi reference model–the iso model of architecture for open systems interconnection. *IEEE Transactions on communications*, 28(4):425–432, 1980.