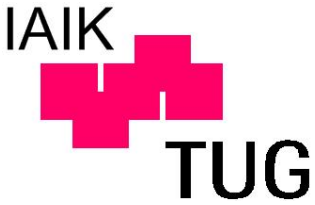

SessionTicket Extension

iSaSiLk Version 6

Contents

1 Introduction

This manual is an amendment to the iSaSiLk *TLS Extensions* [ISASILK-EXT] manual. It describes how to use the iSaSiLk TLS library with the *SessionTicket* extension as specified by [RFC 4507] and its successor draft [RFC 4507bis].

This manual is structured in the same way as the iSaSiLk *TLS Extensions* [ISASILK-EXT] manual. The second chapter gives a short theoretical introduction into the *SessionTicket* extension; for a detailed description refer to the *SessionTicket* specification ([RFC 4507]) of the TLS working group. The third and already last chapter shows how to configure and use iSaSiLk with the *SessionTicket* extension.

A client-server demo is included in the `src/demo/extensions` directory of your iSaSiLk distribution.

For installation of the iSaSiLk library and a general introduction into the iSaSiLk TLS extension framework please refer to the iSaSiLk *TLS Extensions* [ISASILK-EXT] manual.

2 The SessionTicket extension

TLS *SessionTicket* extension has been introduced by RFC 4507 – *TLS Session Resumption without Server-Side State* [RFC 4507]. It provides an alternative TLS session management strategy where the server does not have to keep any session related information.

Instead of maintaining a (likely comprehensive) session database the server packs the required session data (cipher suite, master secret, …) into a session ticket, encrypts the ticket using the AES cipher [AES] and sends it to the client. For protecting the integrity of the encrypted session ticket a SHA1-HMAC ([RFC 2104]) is calculated and sent along with the ticket (see [RFC 4507]).

The client, when receiving the session ticket, stores it together with its local session information. When later wishing to resume the session, the client sends the session ticket back to the server. The server verifies the MAC value and decrypts the ticket. If the ticket still is valid (within the session resumption period) the session can be resumed based on the session information contained in the ticket.

However, before actually enforcing session resumption without server-side state client and server have to agree on using session tickets by exchanging *SessionTicket* extensions at the beginning of the TLS handshake. First of all the client has to send an empty *SessionTicket* extension to announce that it is able to receive a session ticket. If the server wants to use session tickets, too, it responds with an empty *SessionTicket* extension. At the end of the handshake (before the *ChangeCipherSpec* message) the server then sends the encrypted ticket to the client who may sent it back when later wishing to resume the session. See [RFC 4507] for a detailed description of the handshake course.

Note that the *SessionTicket* encoding has been changed from [RFC 4507] to [RFC 4507bis] which simply puts the ticket into the `extension_data` field since done so by most applications. iSaSiLk 4.1 now also uses the 4507bis format when sending a *SessionTicket* extension, but is able to parse both, the 4507 and 4507bis format. Also according [RFC 4507bis] iSaSiLk 4.1 uses SHA-256 instead of SHA-1 as hash algorithm for HMAC based ticket protection.

3 The SessionTicket extension and iSaSiLk

In this chapter we describe how to configure iSaSiLk for using the *SessionTicket* extension for session resuming. As accustomed from the iSaSiLk extensions manual [ISASILK-EXT], we first discuss how to configure and use the *SessionTicket* extension on the client side, and then explain how it has to be used on the server side.

3.1.1.1 Client Side

The *SessionTicket* extension is implemented by class `SessionTicket` of the iSaSiLk package `iaik.security.ssl`. A client that wants to use session tickets simply includes an empty `SessionTicket` extension into the extension list of the extended *ClientHello* message:

```
// create empty SessionTicket
SessionTicket sessionTicket = new SessionTicket();
// add to ExtensionList
ExtensionList extensions = new ExtensionList();
...
extensions.addExtension(sessionTicket);
...
// set extensions for the SSLClientContext configuration:
SSLClientContext clientContext = new SSLClientContext();
...
clientContext.setExtensions(extensions);
```

Listing 3-1: Adding a SessionTicket extension to your client extension list

By default a client-side `SessionTicket` extension is considered as critical. This means that the handshake will be aborted if the client sends a `SessionTicket` extension but the server does not respond with a `SessionTicket` extension.

3.1.1.2 Server Side

On the server side we have to tell iSaSiLk the AES and MAC keys to be used for ticket encryption and MAC protection. You yourself can create and specify AES and MAC key; however, more simple you may use an empty `TicketKeyBag` to let iSaSiLk create the required keys:

```
// create empty TicketKeyBag
TicketKeyBag ticketKeys = new TicketKeyBag();
// create SessionTicket
SessionTicket sessionTicket = new SessionTicket(ticketKeyBag);
// add to ExtensionList
ExtensionList extensions = new ExtensionList();
extensions.addExtension(sessionTicket);
...
// set extensions for the SSLServerContext configuration:
SSLServerContext serverContext = new SSLServerContext();
...
serverContext.setExtensions(extensions);
```

Listing 3-2: Server-side SessionTicket extension with empty TicketKeyBag

By default iSaSiLk will use the ticket keys for the whole server lifetime (as long as the server is running). However, you also can configure iSaSiLk to periodically refresh the ticket keys by specifying a `TicketKeyBag` validity period:

```
...
TicketKeyBag ticketKeys = new TicketKeyBag();
// set TicketKeyBag validity period
long validityPeriod = ...;
ticketKeys.setValidityPeriod(validityPeriod);
...
```

Listing 3-3: Setting a TicketKeyBag validity period

The default ticket validity period is -1 meaning that the ticket keys are used as long as the iSaSiLk server is running. If you specify a positive value when calling method `setValidityPeriod` new keys will be generated after the given time period has expired. To guarantee a smooth key transition, the old keys are still valid for the period of one ticket lifetime after new keys have been activated.

Server-side extensions are *not critical* by default. Setting your `SessionTicket` extension object to *critical* would mean that iSaSiLk would abort the handshake if the client does not send a `SessionTicket` extension.

4 References

[AES] National Institute of Standards and Technology (NIST): "Advanced Encryption Standard (AES)", Federal Information Processing Standards (FIPS) Publication 197, November 2001.

[IAIK-JCE] IAIK-JCE Provider, IAIK, Stiftung SIC, 2003, http://jce.iaik.tugraz.at

[JAVA] Java™ Technology, Sun Microsystems, Inc., http://java.sun.com/

[JCA] Java™ Cryptography Architecture API Specification & Reference, SUN Microsystems, Inc.,

http://java.sun.com/j2se/1.4/docs/guide/security/CryptoSpec.html

[JCE] Java™ Cryptography Extension (JCE) API Specification & Reference, SUN Microsystems, Inc.,

http://java.sun.com/j2se/1.4.2/docs/guide/security/jce/JCERefGuide.html

[TLS] T. Dierks and C. Allen: "The TLS Protocol Version 1.0", RFC 2246, January 1999.

T. Dierks and E. Rescorla, "The TLS Protocol, Version 1.1", RFC 4346, April, 2006.

[ISASILK-EXT] iSaSiLk 4: TLS Extensions manual, IAIK, Stiftung SIC, 2007

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

[RFC 4366] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, T. Wright: "Transport Layer Security (TLS) Extensions", RFC 4366, April 2006

[RFC 4507] J. Salowey, H. Zhou, P. Eronen, H. Tschofenig: "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 4507, May 2006

[RFC 4507bis] J. Salowey, H. Zhou, P. Eronen, H. Tschofenig: "Transport Layer Security (TLS) Session Resumption without Server-Side State", draft-salowey-tls-rfc4507bis-01.txt, August 2007