TLS Extension for Optimizing Application Protocols, Specifically SASL
draft-williams-tls-app-sasl-opt-00.txt

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the
provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering
Task Force (IETF), its areas, and its working groups. Note that
other groups may also distribute working documents as Internet-
Drafts.

Internet-Drafts are draft documents valid for a maximum of six months
and may be updated, replaced, or obsoleted by other documents at any
time. It is inappropriate to use Internet-Drafts as reference
material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at

This Internet-Draft will expire on October 12, 2009.

Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the
document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal
Provisions Relating to IETF Documents in effect on the date of
publication of this document (http://trustee.ietf.org/license-info).
Please review these documents carefully, as they describe your rights
and restrictions with respect to this document.
Abstract

This document specifies an extension to Transport Layer Security (TLS) for carrying application data which is suitable for delayed integrity protection and does not require privacy protection. In particular we describe how to use this extension to reduce the number of round-trips needed for application-layer authentication, specifically Simple Authentication (SASL), and through it, Generic Security Services (GSS-API).

Table of Contents

1. Introduction .................................................. 3
   1.1. Conventions used in this document .................... 3
2. The Extension and Optimization of SASL ........................ 4
3. IANA Considerations ........................................... 7
4. Security Considerations ......................................... 8
5. References ..................................................... 9
   5.1. Normative References .................................... 9
   5.2. Informative References .................................. 9
Author’s Address .................................................. 10
1. Introduction

Many applications use TLS [RFC5246] and then Simple Authentication and Security Layers (SASL) [RFC4422] on top of TLS. This requires at least two round-trips for TLS, then one round-trip for SASL mechanism negotiation, then as many round-trips as the negotiated SASL mechanism requires. One and a half of the TLS round-trips can carry extensions such that we could piggyback some application data on those TLS messages to save up to two round-trips. This document specifies how to take advantage of TLS extensions to reduce the number of round-trips needed altogether.

First we define a TLS extension for use in Client Hello and Handshake messages. This extension will carry typed application data. Then we describe how to reduce the number of round-trips for SASL applications. And through the new SASL/GSS-API bridge [I-D.ietf-sasl-gs2] we obtain support for use of GSS-API [RFC2743] mechanisms as well. [RFC2743] applications. We achieve a one and a half round-trip reduction for SASL applications.

In the case of SASL applications we use the first TLS round-trip to optimize the SASL mechanism negotiation. Then we use the client’s handshake message to send the first authentication message of the selected SASL mechanism. Note that the TLS channel binding [RFC5056] is available at that time, thus no special considerations apply to how channel binding is done. Use of channel binding protects against man-in-the-middle attacks, including downgrade attacks on mechanism negotiation.

1.1. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].
2. The Extension and Optimization of SASL

When a client application wishes to exchange one or more application messages prior to the conclusion of a TLS exchange it uses the TLS Client Hello message extension to a) indicate its intention to the server, and b) optionally send the first application message to the server. These messages will not have any privacy or integrity protection applied by TLS unless a ChangeCipherSpec has been done earlier (i.e., unless the application has already done one TLS handshake).

When this message is received the server MUST either ignore the extension or pass it to the application, which then MUST respond to that application data via a new handshake message (see below). If the server ignores it then the client will discover that the server does not support this extension when the client receives the server’s handshake messages. Otherwise there must be a corresponding application data handshake message in the server’s response, and that indicates that the server TLS and application implementations support this extension.

The extension contents are defined by the application. In order to save the application having to encode application data types and lengths we define two application data extension types and we allow the Client Hello to carry one of each of these extensions:

- pfapp_data (<TBD>)
- sasl_sml_req (<TBD>)

The "pf" prefix indicates "pre-Finished message exchange". It is the application’s responsibility to define the contents of the pfapp_data extension.

The sasl_sml_req (SASL server mechanism list request) message contains an empty payload.

We also define new Handshake messages that may be used after the Client Hello messages:
enum {
    finished(20), pfapp_data(<TBD>),
    sasl_sml(<TBD>), sasl_msg(<TBD>), (255)
} HandshakeType;

struct {
    HandshakeType msg_type;    /* handshake type */
    uint24 length;             /* bytes in message */
    select (HandshakeType) {
        case hello_request:                     HelloRequest;
        ...                                  
        case pfapp_data:                      PFAppData;
        case sasl_sml:                        SaslSML;
        case sasl_msg:                       SaslMsg;
    } body;
} Handshake;

opaque PFAppData<2^16-1>;
opaque SaslSML<2^16-1>;
opaque SaslMsg<2^16-1>;

A generic application protocol using these extensions might look like:

Client                                               Server

ClientHello w/ sasl_sml_req -------->

ServerHello
    SaslSML*
    Certificate*
    ServerKeyExchange*
    CertificateRequest*
        <-------- ServerHelloDone

Certificate*
SaslMsg*
ClientKeyExchange
CertificateVerify*
    [ChangeCipherSpec]
Finished            -------->
    [ChangeCipherSpec]
        <--------Finished
SASL auth messages <--------> SASL auth messages
Application Data  <--------> Application Data

The TLS channel binding types that are suitable for use with SASL in this facility are:
o tls-server-end-point

o tls-unique

See the IANA channel binding type registry for more information about these channel binding types. The channel binding type to use is to be selected as described in [I-D.ietf-sasl-channel-bindings] (namely: if there is a server certificate, then use tls-server-end-point, else use tls-unique).

Note that the application has to construct its first SASL authentication message for sending in the same half-round trip as the client’s Finished message, yet the client’s Finished message is used in the tls-unique channel binding type. This means that the Finished message MUST be constructed before the client’s SaslMsg, and the SaslMsg is not integrity protected by the client’s Finished message, though it will be integrity protected by the server’s Finished message.
3. IANA Considerations

When this document is approved for the Standards-Track the &lt;TBD&gt; values above will be filled in and the IANA TLS ExtensionType and HandshakeType registries will have to be updated to reflect these assignments. (These registries require IETF Consensus and Standards action, respectively.)
4. Security Considerations

The security considerations of [RFC4422], [I-D.ietf-sasl-channel-bindings], [RFC5246] and [RFC5056] apply, as do those of [RFC2743] when used via the SASL/GS2 bridge [I-D.ietf-sasl-gs2].

As usual with TLS there is no privacy protection of client identity unless the client first completes a handshake without authenticating itself, changes the cipher spec, then initiates a new handshake where it does authenticate itself. In this case, client authentication being done via SASL, this means not sending a SaslMsg until after the initial ChangeCipherSpec exchange.

The use of SASL mechanisms that do not provide channel binding support is NOT RECOMMENDED, but if they are used then the application MUST ensure that the server identity authenticated by TLS corresponds to the server identity authenticated by SASL if any, and to the server name provided by the user.

The initial SASL authentication message is not protected by the TLS client’s Finished message, but it is protected by the server’s Finished message. If channel binding is used, as it should be, the initial SASL authentication message will be bound to the TLS channel. Therefore the server can detect modifications to the initial SASL authentication message to the best of the selected SASL mechanism’s ability, and the client can detect modifications to its initial SASL authentication message through the server’s TLS Finished message.

The SASL mechanism negotiation is protected by the TLS Finished messages.
5. References

5.1. Normative References

[I-D.ietf-sasl-channel-bindings]
Williams, N., "SASL And Channel Binding",
draft-ietf-sasl-channel-bindings-00 (work in progress),
April 2009.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate

[RFC4422] Melnikov, A. and K. Zeilenga, "Simple Authentication and

[RFC5056] Williams, N., "On the Use of Channel Bindings to Secure


5.2. Informative References

[I-D.ietf-sasl-gs2]
Josefsson, S. and N. Williams, "Using GSS-API Mechanisms
in SASL: The GS2 Mechanism Family", draft-ietf-sasl-gs2-11
(work in progress), March 2009.

[RFC2743] Linn, J., "Generic Security Service Application Program
Author’s Address

Nicolas Williams
Sun Microsystems
5300 Riata Trace Ct
Austin, TX  78727
US

Email: Nicolas.Williams@sun.com