TLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 25, 2007

Y. Nir
Y. Sheffer
Check Point
H. Tschofenig
Siemens

February 21, 2007

Protocol Model for TLS with EAP Authentication
draft-nir-tee-pm-00.txt

Status of this Memo

Copyright Notice

Abstract

This document describes an extension to the TLS protocol to allow TLS clients to authenticate with legacy credentials using the Extensible Authentication Protocol (EAP).

This work follows the example of IKEv2, where EAP has been added to the IKEv2 protocol to allow clients to use different credentials such as passwords, token cards, and shared secrets.

When TLS is used with EAP, additional records are sent after the ChangeCipherSpec protocol message, effectively creating an extended handshake before the application layer data can be sent. Each EapMsg handshake record contains exactly one EAP message. Using EAP for client authentication allows TLS to be used with various AAA back-end servers such as RADIUS or Diameter.

TLS with EAP may be used for securing a data connection such as HTTP or POP3, where the ability of EAP to work with backend servers can remove that burden from the application layer.

This document is a protocol model, rather than a full protocol specification.

Table of Contents

1. Introduction

This document describes a new extension to [TLS]. This extension allows a TLS client to authenticate using [EAP] instead of using a certificate, or alternatively performing the authentication at the application level. The extension follows [TLS-EXT]. For the remainder of this document we will refer to this extension as TEE (TLS with EAP Extension). The document is a protocol model as described in [RFC4101].

TEE extends the TLS handshake beyond the regular setup, to allow the EAP protocol to run between the TLS server (called an "authenticator" in EAP) and the TLS client (called a "supplicant"). This allows the TLS architecture to handle client authentication before exposing the server application software to an unauthenticated client. In doing this, we follow the approach taken for IKEv2 in [IKEv2]. However, similar to regular TLS, we protect the user identity by only sending the client identity after the server has authenticated. In this our solution defers from that of IKEv2.

Currently used applications use TLS to authenticate the server only. After that, the application takes over, and presents a login screen where the user is expected to present their credentials.

This creates several problems. It allows a client to access the application before authentication, thus creating a potential for anonymous attacks on non-hardened applications. Additionally, web pages are not particularly well suited for long shared secrets and for certain devices such as USB tokens.

TEE allows full mutual authentication to occur for all these applications within the TLS exchange. The application receives control only when the user is identified and authenticated. The authentication can be built into the server infrastructure by connecting to an AAA server. The client side can be integrated into client software such as web browsers and mail clients. An EAP infrastructure is already built-in to some operating systems providing a user interface for each authentication method within EAP.

We intend TEE to be used for various protocols that use TLS such as HTTPS, in cases where certificate based authentication is not practical. This includes web-based mail services, online banking, premium content websites and mail clients.

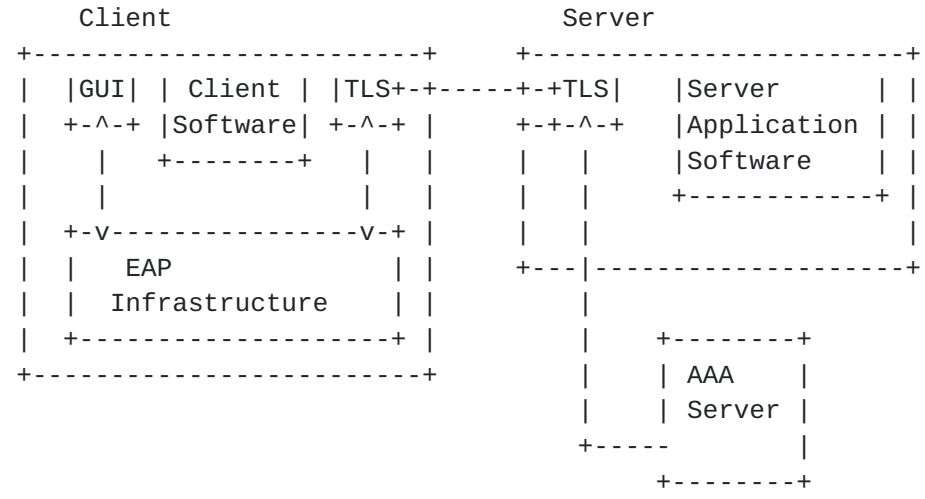
Another class of applications that may see benefit from TEE are TLS based VPN clients used as part of so-called "SSL VPN" products. No such client protocols have so far been standardized.

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. Operating Environment

TEE will work between a client application and a server application, taking care of all encryption and authentication.



The above diagram shows the typical deployment. The client has software that either includes a UI for some EAP methods, or else is able to invoke some operating system EAP infrastructure that takes care of the user interaction. The server is configured with the address and protocol of the AAA server. Typically the AAA server communicates using the RADIUS protocol with EAP ([RADIUS] and [RAD-EAP]), or the Diameter protocol ([Diameter] and [Dia-EAP]).

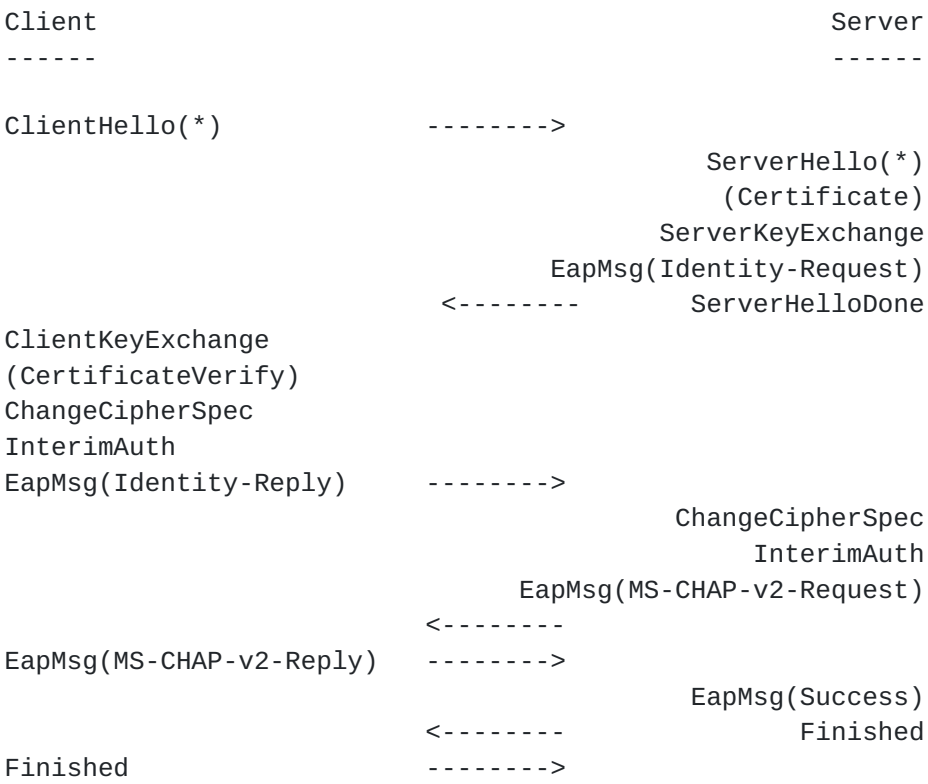
As stated in the introduction, we expect TEE to be used in both browsers and applications. Further uses may be authentication and key generation for other protocols, and tunneling clients, which so far have not been standardized.

3. Protocol Overview

The TEE extension defines the following:

- o A new extension type called tee_supported, used to indicate that the client supports this extension.
- o A new message type for the handshake protocol, called InterimAuth, which is used to sign previous messages.
- o A new message type for the handshake protocol, called EapMsg, which is used to carry a single EAP message.

The diagram below outlines the protocol structure. For illustration purposes only, we use the [I-D.dpotter-pppext-eap-mschap] EAP method .



(*) The ClientHello and ServerHello include the tee_supported extension to indicate support for TEE

The client indicates in the first message its support for TEE. The server sends an EAP identity request in the reply. The client sends the identity reply after the handshake completion. The EAP request-response sequence continues until the client is either authenticated or rejected.

3.1. The tee_supported Extension

The tee_supported extension is a ClientHello and ServerHello extension as defined in section 2.3 of [TLS-EXT]. The extension_type field is TBA by IANA. The extension_data is zero-length.

3.2. The InterimAuth Handshake Message

The InterimAuth message is identical in syntax to the Finished message described in section 7.4.9 of [TLS]. It is calculated in exactly the same way.

The semantics, however, are somewhat different. The "Finished" message indicates that application data may now be sent. The "InterimAuth" message does not indicate this. Instead, further handshake messages are needed.

Depending on the EAP method used, the Finished message may be calculated differently. See Section 3.4 for details.

The HandshakeType value for the InterimAuth handshake message is TBA by IANA.

3.3. The EapMsg Handshake Message

The EapMsg handshake message carries exactly one EAP message as defined in [EAP].

The HandshakeType value for the EapMsg handshake message is TBA by IANA.

The EapMsg message is used to tunnel EAP messages between the authentication server, which may be the co-located with the TLS server, or may be a separate AAA server, and the supplicant, which is co-located with the TLS client. TLS on either side receives the EAP data from the EAP infrastructure, and treats it as opaque. TLS does not make any changes to the EAP payload or make any decisions based on the contents of an EapMsg handshake message.

3.4. Calculating the Finished message

If the EAP method is key-generating, the Finished message is calculated as follows:

```
struct {
    opaque verify_data[12];
} Finished;

verify_data
    PRF(MSK, finished_label, MD5(handshake_messages) +
    SHA-1(handshake_messages)) [0..11];
```

The finished_label is defined exactly as in section 7.4.9 of [TLS].

The handshake_messages, similar to regular TLS is all of the data from all messages in this handshake, including any EapMsg and InterimAuth messages, up to but not including this Finished message. This is the concatenation of all the Handshake structures, as defined in section 7.4 of [TLS] and here, exchanged thus far.

The MSK is typically received from the AAA server over the RADIUS or Diameter protocol.

If the EAP method is not key-generating, then the Finished message is calculated exactly as described in [TLS]. Such methods however, are NOT RECOMMENDED. See Section 4.1 for details.

4. Security Considerations

4.1. InterimAuth vs. Finished

In regular TLS, the Finished message provides two functions: it signs all previous messages, and it signals that application data can now be used. In TEE, we sign the previous messages twice.

Some EAP methods, such as EAP-TLS, EAP-IKEv2 and EAP-SIM generate keys in addition to authenticating clients. Such methods are said to be resistant to MITM attacks as discussed in [MITM]. Such methods are called key-generating methods.

To realize the benefit of such methods, we need to verify the key that was generated within the EAP method. This is referred to as the MSK in EAP. In TEE, the InterimAuth message signs all previous messages with the master_secret, just like the Finished message in regular TLS. The Finished message signs all previous messages using the MSK if such exists. If not, then the messages are signed with the master_secret as in regular TLS.

The need for signing twice arises from the fact that we need to use both the master_secret and the MSK. It was possible to use just one Finished record and blend the MSK into the master_secret. However, this would needlessly complicate the protocol and make security analysis more difficult. Instead, we have decided to follow the example of IKEv2, where two AUTH payloads are exchanged.

It should be noted that using non-key-generating methods may expose the client to a MITM attack if the same MITM method is used in some other situation, in which the EAP is done outside of a protected tunnel with an authenticated server. Unless it can be determined that the EAP method is never used in such a situation, non-key-generating methods SHOULD NOT be used.

4.2. Identity Protection

Unlike [TLS-PSK], TEE provides identity protection for the client. The client's identity is hidden from a passive eavesdropper using TLS encryption, and it is not sent to the server until after the server's identity has been authenticated by verifying the certificate.

Active attacks are thwarted by the server authentication using a certificate or by using a suitable EAP method.

We could save one round-trip by having the client send its identity within the Client Hello message. This is similar to TLS-PSK. However, we believe that identity protection is a worthy enough goal,

so as to justify the extra round-trip.

5. Performance Considerations

Regular TLS adds two round-trips to a TCP connection. However, because of the stream nature of TCP, the client does not really need to wait for the server's Finished message, and can begin sending application data immediately after its own Finished message. In practice, many clients do so, and TLS only adds one round-trip of delay.

TEE adds as many round-trips as the EAP method requires. For example, EAP-MD5 requires 1 round-trip, while EAP-SIM requires 2 round-trips. Additionally, the client MUST wait for the EAP-Success message before sending its own Finished message, so we need at least 3 round-trips for the entire handshake. The best a client can do is two round-trips plus however many round-trips the EAP method requires.

It should be noted, though, that these extra round-trips save processing time at the application level. Two extra round-trips take a lot less time than presenting a log-in web page and processing the user's input.

It should also be noted, that TEE reverses the order of the Finished messages. In regular TLS the client sends the Finished message first. In TEE it is the server that sends the Finished message first. This should not affect performance, and it is clear that the client may send application data immediately after the Finished message.

6. IANA Considerations

IANA is asked to assign an extension type value from the "ExtensionType Values" registry for the tee_supported extension.

IANA is asked to assign two handshake message types from the "TLS HandshakeType Registry", one for "EapMsg" and one for "InterimAuth".

7. Acknowledgments

The TLS Innel Application Extension work ([TLS/IA]) has inspired the authors to create this simplified work. TLS/IA provides a somewhat different approach to integrating non-certificate credentials into the TLS protocol, in addition to several other features available from the RADIUS namespace.

The authors would also like to thanks the various contributors to [IKEv2] whose work inspired this one.

8. References

8.1. Normative References

[EAP] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[TLS] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

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

8.2. Informative References

[Dia-EAP] Eronen, P., Hiller, T., and G. Zorn, "Diameter Extensible Authentication Protocol (EAP) Application", RFC 4072, August 2005.

[Diameter] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.

[I-D.dpotter-pppext-eap-mschap] Potter, D. and J. Zamick, "PPP EAP MS-CHAP-V2 Authentication Protocol", draft-dpotter-pppext-eap-mschap-01 (work in progress), January 2002.

[IKEv2] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[MITM] Asokan, N., Niemi, V., and K. Nyberg, "Man-in-the-Middle in Tunneled Authentication Protocols", October 2002.

[RAD-EAP] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, September 2003.

[RADIUS] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC4101] Rescorla, E. and IAB, "Writing Protocol Models", RFC 4101, June 2005.

[TLS-PSK] Eronen, P. and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005.

[TLS/IA] Funk, P., Blake-Wilson, S., Smith, H., Tschofenig, N., and T. Hardjono, "TLS Inner Application Extension (TLS/IA)", draft-funk-tls-inner-application-extension-03 (work in progress), June 2006.

Authors' Addresses

Yoav Nir
Check Point Software Technologies Ltd.
3A Jabotinsky St.
Ramat Gan 52520
Israel

Email: ynir@checkpoint.com

Yaron Sheffer
Check Point Software Technologies Ltd.
3A Jabotinsky St.
Ramat Gan 52520
Israel

Email: yaronf at checkpoint dot com

Hannes Tschofenig
Siemens
Otto-Hahn-Ring 6
Munich, Bavaria 81739
Germany

Email: Hannes.Tschofenig@siemens.com
URI: http://www.tschofenig.com

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).