IPWAVE
Internet-Draft
Intended status: Informational
Expires: September 29, 2019

M. Msahli, Ed.
Telecom ParisTech
P. Kampanakis, Ed.
Cisco

March 28, 2019

TLS Authentication using IEEE 1609.2 certificates
draft-msahli-ipwave-extension-ieee1609-03.txt

Abstract

This document specifies the use of a new certificate type to authenticate TLS entities. The first type enables the use of a certificate specified by the IEEE and the European Telecommunications Standards Institute (ETSI).

Table of Contents

1. Introduction

The TLS protocol [RFC8446] [RFC5246] uses X509 and Raw Public Key in order to authenticate servers and clients. This document describes the use of the certificate specified by the IEEE in [IEEE1609.2] and profiled by the European Telecommunications Standards Institute (ETSI) in [TS103097]. These standards specify secure communications in vehicular environments. The certificate types are optimized for bandwidth and processing time to support delay-sensitive applications, and also provide both authentication and authorization information to enable fast access control decisions in ad hoc networks such as are found in Intelligent Transportation System (ITS). The extension is following the [RFC6066].

2. Requirements Terminology

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].

3. Extension Overview

This specification extends the Client Hello and Server Hello messages, by using the "extension_data" field of the ClientCertType Extension and the ServerCertType Extension structures defined in

RFC7250. In order to negotiate the support of IEEE 1609.2 or ETSI TS 103097 certificate-based authentication, the clients and the servers MAY include the extension of type "client_certificate_type" and "server_certificate_type" in the extended Client Hello and "EncryptedExtensions". The "extension_data" field of this extension SHALL contain a list of supported certificate types proposed by the client as provided in the figure below:

```
  /* Managed by IANA */
   enum {
       X509(0),
       RawPublicKey(2),
       1609Dot2(3),
       (255)
   } CertificateType;

   struct {
       select (certificate_type) {

           /* certificate type defined in this document.*/
            case 1609Dot2:
            opaque cert_data<1..2^24-1>;

            /* RawPublicKey defined in RFC 7250*/
           case RawPublicKey:
           opaque ASN.1_subjectPublicKeyInfo<1..2^24-1>;

           /* X.509 certificate defined in RFC 5246*/
           case X.509:
           opaque cert_data<1..2^24-1>;

            };

          Extension extensions<0..2^16-1>;
      } CertificateEntry;
```

In case where the TLS server accepts the described extension, it selects one of the certificate types in the extension described above. Note that a server MAY authenticate the client using other authentication methods. The end-entity certificate's public key has to be compatible with one of the certificate types listed in the extension described above.

4. TLS Client and Server Handshake

The "client_certificate_type" and "server_certificate_type" extensions MUST be sent in handshake phase as illustrated in Figure 1 below. The same extension shall be sent in Server Hello for TLS 1.2.

```
   Client                                                Server

 Key  ^ ClientHello
 Exch | + server_certificate_type*
      | + client_certificate_type*
      | + key_share*
      v + signature_algorithms*       -------->
                                                  ServerHello  ^ Key
                                                 + key_share*  v Exch
                                        {EncryptedExtensions}  ^ Server
                                   {+ server_certificate_type*}| Params
                                   {+ client_certificate_type*}|
                                        {CertificateRequest*}  v
                                               {Certificate*}  ^
                                         {CertificateVerify*}  | Auth
                                                   {Finished}  v
                              <-------   [Application Data*]
      ^ {Certificate*}
 Auth | {CertificateVerify*}
      v {Finished}            -------->
        [Application Data]    <------->   [Application Data]

                +  Indicates noteworthy extensions sent in the
                   previously noted message.

                *  Indicates optional or situation-dependent
                   messages/extensions that are not always sent.

                {} Indicates messages protected using keys
                   derived from a [sender]_handshake_traffic_secret.

                [] Indicates messages protected using keys
                   derived from [sender]_application_traffic_secret_N.
```

Figure 1: Message Flow with certificate type extension for Full TLS 1.3 Handshake

4.1. Client Hello

In order to indicate the support of IEEE 1609.2 or ETSI TS 103097 certificates, client MUST include an extension of type "client_certificate_type" and "server_certificate_type" in the extended Client Hello message. The Hello extension is described in Section 4.1.2 of TLS 1.3 [RFC8446].

The extension 'client_certificate_type' sent in the client hello MAY carry a list of supported certificate types, sorted by client preference. It is a list in the case where the client supports multiple certificate types.

Client MAY respond along with supported certificates by sending a "Certificate" message immediately followed by the "CetificateVerify" message. These specifications are valid for TLS 1.2 and TLS 1.3.

All implementations SHOULD be prepared to handle extraneous certificates and arbitrary orderings from any TLS version, with the exception of the end-entity certificate which MUST be first.

4.2. Server Hello

When the server receives the Client Hello containing the client_certificate_type extension and/or the server_certificate_type extension, the following options are possible:

- The server supports the extension described in this document. It selects a certificate type from the client_certificate_type field in the extended Client Hello and must take into account the client authentication list priority.

- The server does not support the proposed certificate type and terminates the session with a fatal alert of type "unsupported_certificate".

- The server does not support the extension defined in this document. In this case, the server returns the server hello without the extensions defined in this document.

- The server supports the extension defined in this document, but it does not have any certificate type in common with the client. Then, the server terminates the session with a fatal alert of type "unsupported_certificate".

- The server supports the extensions defined in this document and has at least one certificate type in common with the client. In this case, the server MAY include the client_certificate_type

extension in the Server Hello for TLS 1.2 or in Encrypted Extension for TLS 1.3. Then, the server requests a certificate from the client (via the certificate_request message)

It is worth to mention that the TLS client or server public keys are obtained from a certificate chain from a web page.

5. Certificate Verification

Verification of an IEEE 1609.2/ ETSI TS 103097 certificates or certificate chain is described in section 5.5.2 of [IEEE1609.2]. In the case where the certificate_type is 1609Dot2, the CertificateVerify message does not contain a raw signature but instead contains a Canonical Octet Encoding Rules (COER)-encoded Ieee1609Dot2Data of type signed as specified in [1609.2b], with the pduFunctionalType field present and set to tlsHandshake. A full specification of the contents of this Ieee1609Dot2Data, including optional fields, is given in [1609.2b]. The message input to the signature calculation is the usual message input for TLS 1.3, as specified in [RFC8446] section 4.4.3, consisting of pad, context string, separator and content, where content is Transcript-Hash(Handshake Context, Certificate).

6. Examples

Some of exchanged messages examples are illustrated in Figures 2 and 3.

6.1. TLS Server and TLS Client use the 1609Dot2 Certificate

This section shows an example where the TLS client as well as the TLS server use the IEEE 1609.2 certificate. In consequence, both the server and the client populate the client_certificate_type and server_certificate_type with extension IEEE 1609.2 certificates as mentioned in figure 2.

```
      Client                                                Server

   ClientHello,
   client_certificate_type*=1609Dot2,
   server_certificate_type*=1609Dot2,   -------->     ServerHello,
                                             {EncryptedExtensions}
                               {client_certificate_type*=1609Dot2}
                               {server_certificate_type*=1609Dot2}
                                             {CertificateRequest*}
                                                    {Certificate*}
                                              {CertificateVerify*}
                                                        {Finished}
     {Certificate*}            <-------         [Application Data*]
     {CertificateVerify*}
     {Finished}               -------->
     [Application Data]       <------->          [Application Data]
```

Figure 2: TLS Client and TLS Server use the IEEE 1609.2 certificate

6.2. TLS Client uses the IEEE 1609.2 certificate and TLS Server uses the X 509 certificate

This example shows the TLS authentication, where the TLS Client populates the server_certificate_type extension with the X509 certificate and Raw Public Key type as presented in figure 3. the client indicates its ability to receive and to validate an X509 certificate from the server. The server chooses the X509 certificateto make its authentication with the Client.

```
     Client                                              Server

   ClientHello,
   client_certificate_type*=(1609Dot2),
   server_certificate_type*=(1609.9Dot,
   X509,RawPublicKey),         ----------->         ServerHello,
                                          {EncryptedExtensions}
                            {client_certificate_type*=1609Dot2}
                                {server_certificate_type*=X509}
                                                 {Certificate*}
                                           {CertificateVerify*}
                                                     {Finished}
                               <---------    [Application Data*]
   {Finished}                  --------->
   [Application Data]          <-------->     [Application Data]
```

Figure 3: TLS Client uses the IEEE 1609.2 certificate and TLS Server uses the X 509 certificate

7. Security Considerations

This section provides an overview of the basic security considerations which need to be taken into account before implementing the necessary security mechanisms. The security considerations described throughout [RFC8446] and [RFC5246] apply here as well.

For security considerations in a vehicular environment, the minimal use of any TLS extensions is recommended such as :

The "client_certificate_type" [IANA value 19] extension who's purpose was previously described in [RFC7250].

The "server_certificate_type" [IANA value 20] extension who's purpose was previously described in [RFC7250].

The "SessionTicket" [IANA value 35] extension for session resumption.

In addition, servers SHOULD not support renegotiation [RFC5746] which presented Man-In-The-Middle (MITM) type attacks over the past years for TLS 1.2.

8. Privacy Considerations

For privacy considerations in a vehicular environment the use of EEE 1609.2/ETSI TS 103097 certificate is recommended for many reasons:

In order to address the risk of a personal data leakage, messages exchanged for V2V communications are signed using IEEE 1609.2/ETSI TS 103097 pseudonym certificates

The purpose of these certificates is to provide privacy relying on geographical and/or temporal validity criteria, and minimizing the exchange of private data

9. IANA Considerations

Existing IANA references have not been updated yet to point to this document.

10. Acknowledgements

The authors wish to thank Eric Rescola and Ilari Liusvaara for their feedback and suggestions on improving this document. Thanks are due to Sean Turner for his valuable and detailed comments. Special thanks to Maik Seewald for their guidance and support in the early stages of the draft.

11. References

11.1. Normative References

[IEEE1609.2] "IEEE Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages", 2016.

[IEEE1609.2b] "Draft Standard for Wireless Access in Vehicular Environments Security Services for Applications and for Management Messages", 2018.

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

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", May 2006.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", August 2008.

[RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension"", February 2010.

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", January 2011.

[RFC7250] Wouters, P., Tschofenig, H., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", June 2014.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", August 2018.

[TS103097]
"ETSI TS 103 097 v1.3.1 (2017-10): Intelligent Transport Systems (ITS); Security; Security header and certificate formats", October 2017.

11.2. Informative References

[draft-serhrouchni-tls-certieee1609-00]
KAISER, A., LABIOD, H., LONC, B., MSAHLI, M., and A. SERHROUCHNI, "Transport Layer Security (TLS) Authentication using ITS ETSI and IEEE certificates", august 2017.

Appendix A. Co-Authors

- Houda Labiod
  Telecom ParisTech
  houda.labiod@telecom-paristech.fr

- Nancy Cam-Winget
  CISCO, USA
  ncamwing@cisco.com

- Ahmed Serhrouchni
  Telecom ParisTech
  ahmed.serhrouchni@telecom-paristech.fr

- William Whyte
  Onboard Security
  wwhyte@onboardsecurity.com

Authors' Addresses

Mounira Msahli (editor)
Telecom ParisTech
France

EMail: mounira.msahli@telecom-paristech.fr

Panos Kampanakis (editor)
Cisco
USA

EMail: pkampana@cisco.com