Identity Protection within EAP-TLS
<draft-urien-badra-eap-tls-identity-protection-01.txt>

Status

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of 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 http://www.ietf.org/shadow.html

This Internet-Draft will expire on April 2007.

Copyright Notice

Copyright (C) The Internet Society (2006). All Rights Reserved.
Abstract

This document defines a mechanism that ensures EAP-TLS identity protection. The main idea is to encrypt the client’s certificate.

Three procedures are proposed in order to determine the certificate encryption mechanism,

- Implicit, the client’s certificate is encrypted according to a pre-defined algorithm, deduced from the server’s certificate.

- Notified, the EAP-identity response message, delivered by the client includes information that precise the encryption algorithm to be used.

- Negotiated, the client indicates a list of encryption algorithm, the server chooses one of them, and indicates its choice.

Table of Contents

Abstract..........................2
1 Introduction........................3
1.2 Requirements language...........4
2 EAP-TLS Identity Protection overview........4
3 Key generation....................5
4 Certificate Encryption............5
5 Implicit mode.....................5
6 Notified mode.....................6
7 Negotiated mode...................7
8 Security Considerations............8
9 References........................8
10 Authors’ Addresses...............8
1 Introduction

According to [EAP], "EAP Methods deriving keys MUST provide for mutual authentication between the EAP peer and the EAP Server".

Consequently, in [EAP-TLS], which is a quite transparent transport of TLS [TLS] over EAP, "the mutual authentication will occur between the peer and the EAP server". For that, when the EAP server sent a certificate_request message, then the peer MUST reply with certificate and certificate_verify handshake messages.

![Mutual authentication exchanges within TLS](image)

The peer always sends its certificate in clear text, and therefore exposes its identity (e.g. an X509 certificate) to eavesdropping.

Thus, an intruder can easily get the certificate and then derive the certificate owner’s real identity.

This document describes how identity protection may be integrated as an optional service in EAP-TLS.

Identity protection is achieved by the encryption of the client’s certificate, according to a cryptographic algorithm that may be selected by different methods; its associated key (enc-key) is calculated from the master secret and the random values exchanged by TLS server and client entities.

Three procedures are proposed in order to fix the certificate encryption mechanism,

- Implicit, the client’s certificate is encrypted according to a pre-defined algorithm, deduced from the server’s certificate.
- Notified, the identity response message delivered by the client includes information that precise the encryption algorithm to be used.

- Negotiated, the client indicates a list of encryption algorithm, the server chooses one of them, and indicates its choice.

1.2 Requirements language

The key words "MUST", "SHALL", "SHOULD", and "MAY", in this document are to be interpreted as described in RFC-2119.

2 EAP-TLS Identity Protection overview

Client                                               Server

ClientHello                    -------->  ServerHello
Certificate
ServerKeyExchange
CertificateRequest
<--------      ServerHelloDone
(Certificate)enc-key // peer identity is sent encrypted
ClientKeyExchange
CertificateVerify
[ChangeCipherSpec]
Finished                     -------->
[ChangeCipherSpec]
<--------             Finished

Figure 2. Mutual authentication exchanges with identity protection

At the end of the hello phase, the client generates the pre_master_secret, encrypts it under the server’s public key, and sends the result to the server.

According to [TLS] a shared secret, called master_secret is computed from, among others, the pre_master_secret.

Hence, the client and the server have all the security parameters to generate the session keys.

Before sending its certificate message, the client encrypts its certificate using the negotiated symmetric algorithm through the anonymity extension and a key (enc-key) derived as follows.

enc-key = PRF(SecurityParameters.master_secret, "client_certificate", SecurityParameters.client_random + SecurityParameters.server_random);
The server repeats the same operation by decrypting the certificate message using the appropriate symmetric algorithm and its key derived in the same manner.

3 Key generation.

The encryption key is obtained according to the following formula:

\[
enc\text{-}key = PRF(SecurityParameters.master\text{-}secret, \\
"client\text{-}certificate", \\
\text{SecurityParameters.client\text{-}random + SecurityParameters.server\text{-}random});
\]

Where
- PRF is the Pseudo Random Function defined in [TLS].
- "client\text{-}certificate" is an ASCII label
- client\text{-}random is the random number included in the client hello message
- server\text{-}random is the random number included in the server hello message.

If the encryption algorithm requires an IV parameter, then this value is equal to the first \( n \) bytes (where \( n \) is the size of the block cipher) produced by the PRF function.

4 Certificate Encryption

If a stream cipher is chosen, then the peer’s certificate is encrypted with the enc_key, without any padding byte.

If a block cipher is selected, then padding bytes are added to force the length of the certificate message to be an integral multiple of the block cipher’s length.

According to [TLS] a certificate is an opaque data, opaque ASN.1Cert<1..2^24-1>, which includes a length field, whose size is three bytes.

When a stream cipher is used, there is no modification of the length value.

If a block cipher is applied, then the length value is the sum of the certificate size plus extra padding bytes.

5 Implicit mode

A certificate is encoded according to the ASN.1 BER (Binary Encoding Rules). It always begins by the SEQUENCE tag (encoded as ‘30’ in
hexadecimal notation), followed by the byte '8x' which indicates an extended length, whose size is \( x \) bytes (typically \( x=2 \) or \( x=4 \)).

Opaque-Data-Length

30 8x Certificate-Length

Content of the SEQUENCE TAG.

The TLS server software may easily checks the coherence between the length of the opaque data, which transports the certificate, and the size associated to its first SEQUENCE TAG. If it detects incoherence, it will attempt to decrypt the client's certificate with the implicit cryptographic algorithm, and afterwards will verify the correctness of the computed result.

6 Notified mode

According to [EAP] "It is RECOMMENDED that the Identity Response be used primarily for routing purposes and selecting which EAP method to use. EAP Methods SHOULD include a method-specific mechanism for obtaining the identity, so that they do not have to rely on the Identity Response".

The value included in the EAP-Identity may be understood as a Network Access Identifier (NAI) whose left part is empty and optional right part (@realm) indicates the authentication server name.

In the negotiated mode, the client notifies the certificate encryption algorithm in the right part of its EAP identity. Therefore the EAP-Identity is expressed as

\[
\text{algorithm-name}@\text{realm},
\]

where \text{algorithm-name} is the identification of the encryption function expressed in an ASCII format

The \text{algorithm-name} comprises two parts, separated by the '.' character
- Right part indicates the algorithm name (rc4, 3des, aes)
- Left part indicates the key size

Examples

rc4.128@realm, rc4 encryption with a 128 bits key size
3des.112@realm, triple DES encryption, in CBC mode, with two 56 bits keys.
aes.128@realm, AES encryption, in CBC mode, with a 128 bits key size.
7 Negotiated mode

In order to allow an EAP-TLS peer to request identity protection exchange, a new extension type is added to the Extended Client and Server Hello messages.

TLS clients and servers include an extension of type ‘identity-protection (TBD)’ in the Extended Client and Server Hello messages.

The ‘extension_data’ field of this extension contains a list of encryption algorithms supported by the client, ordered by preference.

If the server is willing to accept using the extension, the client and the server negotiate the symmetric algorithm that will be used to encrypt/decrypt the client certificate.

The TLS extension is sent by the client to indicate to the server that the client certificate will be sent encrypted using a symmetric algorithm negotiated through that extension.

The symmetric algorithm uses a key derived from the random values and from the master_secret.

```
struct {
    SymmetricAlgorithm symmetric_alg_list<0..2^16-1>;
} IdentityProtection;
```

```
enum { rc4_128(0), (255) } SymmetricAlgorithm;
```

The "extension_data" field of this extension shall contain the symmetric algorithms (and their key length) supported by the client.
Encryption algorithms are sent in order of the client’s preference (favorite choice first).

8 Security Considerations

Security issues are discussed throughout this memo and in [EAP-TLS], [EAP-TLS] and [TLS-EXT]

9 References


10 Authors’ Addresses

Pascal Urien
ENST Paris
France
Email: Pascal.Urien@enst.fr

Mohamad Badra
ENST Paris
France
Email: Mohamad.Badra@enst.fr
Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the IETF’s procedures with respect to rights in IETF Documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2006). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.