Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 22, 2016

D. Migault
Ericsson

February 19, 2016

TLS/DTLS Content Provider Edge Server Abstract API
draft-mglt-lurk-tls-abstract-api-00

Abstract

This document describes the interactions between the Edge Server and the Content Provider in a split authentication scenario.

This document provides an abstract description of the information exchanged between an Edge Server and a Content Provider.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on August 22, 2016.

Table of Contents

1. Introduction 2
2. Terminology 3
3. Protocol Overview 3
3.1. Premaster Computation 5
3.2. Master Computation 6
3.3. Signature Computation 8
4. Input Parameters Description 9
4.1. Static parameters 10
4.1.1. API Version 10
4.1.2. TLSVersion 10
4.1.3. ObjectRequest 10
4.1.4. AuthenticationMethod 10
4.1.5. MasterMethod 10
4.1.6. SignatureMethod 10
4.1.7. PRF 11
4.2. TLS Hanshake parameters 11
4.2.1. ClientHello.random, ClientServer.random 11
4.2.2. session_hash, DataHash 11
4.2.3. handshake_messages 11
4.2.4. PSK_ID 11
4.3. Cryptographic Parameters 11
4.3.1. RSAEncryptedPreMaster 11
4.3.2. DHECDHTLSClientPublicDHECDHKey 13
4.3.3. TLSClientEdgeServerDHSecret 13
4.3.4. TLSClientEdgeServerDHECDHSecret 13
5. Output Parameters Description 13
5.1. Premaster 13
5.2. Signature 13
5.3. Error 13
6. Security Considerations 14
7. Acknowledgements 14
8. References 14
Author's Address 14

1. Introduction

This document assumes a TLS session is established between a TLS Client authenticates a Content Provider while being connected with an TLS Edge Server.

The architecture is defined more in details in [draft-cairns-tls-session-key-interface], and the split authentication models are described in [draft-mglt-tls-lurk-requirements].

Motivation for providing an abstract API is to to provide a reference that may be implemented in various ways. As a result, this document

does not consider the names of the procedures, their implementations, nor how the informations are exchanged between the Edge Server and the Content Provider. This document only describes the information exchanged between the Edge Server and the Content Provider and the associated treatment or verifications that may apply.

2. Terminology

3. Protocol Overview

In a split authentication scenario, the Edge Server and the Content Provider needs to interact in order to set up the TLS/DTLS session between the TLS Client and the Edge Server while the TLS Client authenticates the Content Provider.

The current document considers the following authentication methods: dhe_dss, dhe_rsa, dh_anon, rsa, dh_dss and dh_rsa described in [RFC6347], ecdh_ecdsa, ecdhe_ecdsa, ecdh_rsa, ecdhe_rsa and ecdh_anon described in [RFC4492] as well as psk, dhe_psk and rsa_psk that are described in [RFC4279]. These authentication methods are designated in this document by the AuthenticationMethod variable.

```
AuthenticationMethod = [rsa, dh_dss, dh_rsa, dh_dss, dh_rsa,
                        ecdh_rsa, dh_anon, ecdh_anon, dhe_dss,
                        dhe_rsa, ecdhe_ecdsa, ecdhe_rsa, psk,
                        dhe_psk, rsa_psk]

                        AuthenticationMethod
```

The interactions between the Edge Server and the Content Provider depend on the authentication method. First of all, the object request, designated in this document as ObjectRequest, depends on the authentication method. When the authentication methods is one of rsa, dh_dss, dh_rsa rsa, dh_dss, dh_rsa, ecdh_rsa, the Content Provider is likely to be requested by the Edge Server a premaster secret or a master secret. When the authentication method is an ephemeral Diffie Hellman or ephemeral Elliptic Diffie Hellman authentication methods like dhe_dss, dhe_rsa, ecdhe_ecdsa, ecdhe_rsa, the Content Provider is likely to be requested a signature. When the authentication method is an anonymous authentication method like dh_anon, ecdh_anon no interactions are expected from the Content Provider and the Edge Server. Finally, when the authentication method is PSK based, the Content provider is likely to be requested a master secret but not the premaster secret as the premaster contains the PSK in clear text.

Second, the object of the request like a master secret or the signature may be generated with different inputs. For example, a signature or a master secret may use the hash of some clear text. The different inputs provided by the Edge Server are designated as InputParameters.

Finally, the Content provider may have its own policies regarding the TLS version, the authentication methods, the object request as well as the associated inputs. This means that the Content Provider may refuse some request defined in this document based on its own policies. This should be indicated in an error message by the Content provider in its response.

As a result, the basic scheme considered for this API adopts a Query / Response pattern. The Query message provides all necessary information for the Content Provider, to proceed to the Query, and send the output result in the Response.

```
Query:
APIVersion = 1.0
TLSVersion = [1.2, 1.3]
ObjectRequest = [premaster, master, signature]
InputParameters

Response:
Output [PreMaster, Master, Signature, Error]
```

API Query / Response

The Query is associated with the following parameters:

(a) APIVersion: which designates the version of the API used. The version is expected to be useful when the authentication methods or TLS version will be added.
(b) TLSVersion: the associated version of the TLS.
(c) ObjectRequest: the requested output. Currently the requested output can be one of the following: pre_master, master, signature.
(d) InputParameters: the parameters provided by the Edge Server in order to compute the ObjectRequest. These InputParameters will be defined in the remaining of the document, and are very specific to each Query.

The Response returns the ObjectRequest or an Error. When an Error is returned it SHOULD provide some indication why the request have not been proceeded. Note that the types of errors are:

(a) Input Format Parameter Errors: that is errors associated to the format of a given parameter. For example a TLS version that is out of bound, or a Client,random that does not have the expected size would trigger such an error.
(b) Input Parameters Incompatibility Error: that is a combination of input parameters that is not acceptable on a standard point of view. This could be for example requesting a signature with an authentication method that is expected to provide a master secret.
(c) Not Permitted Input Parameters Error: that is a combination of parameters that is either not accepted by the Content Provider, or not implemented by the API. For example, the Content Provider may only generate signature from the complete clear text and refuse to generate it simply based on the hash of the content.

InpuParameters depends on the ObjectRequest value, so the InputParameters can be:

(a) PremasterInputParams: when the query requests a premaster, that is ObjectRequest is set to premaster.
(b) MasterInputParams: when the query requests a master secret, that is ObjectRequest is set to master.
(c) SignatureInputParams: when the query requests a signature, that is the ObjectRequest is set to signature.

```
InputParameters:
    select(ObjectRequest)
    case pre_master:
        PremasterInputParams
    case master:
        MasterInputParams
    case signature:
        SignatureInputParams

   InpuParameters
```

3.1. Premaster Computation

A premaster may be requested only for a subset of authentication methods, that is RSA and non anonymous Diffie Helmman based methods. In the case of RSA, the encrypted premaster is provided whereas Diffie Hellman based authentication provides the TLS Client public key.

When another authentication method is associated to the premaster request an Input Parameters Incompatibility Error must be sent

indicating that the authentication method is not compatible with the premaster request.

(a) INCOMPATIBLE_INPUT_PARAMETERS_ERROR

In order to indicate the error is associated to the authentication method the two following error message are added. Note that an unacceptable error may occur when there is a incompatibility between the parameters. This can occur due to local policies or due to an incompatibility with the specification. It is the responsibility of the Edge Server to implement the specification properly, and thus to receive the error only when it results from a local policy.

AUTHENTICATION_METHOD_UNACCEPTABLE_ERROR

The description of the input parameters are provide by the figure below:

```
PreMasterInputParams
    - AuthenticationMethod
    - PreMasterInputData

PreMasterInputData:
    select(AuthenticationMethod)
    case rsa:
        RSAEncryptedPreMaster
    case dh_dss, dh_rsa, dh_dss, dh_rsa, ecdh_rsa:
        DHECDHTLSClientPublicDHECDHKey
    case dhe_dss, dhe_rsa, ecdhe_ecdsa,
         ecdhe_rsa, psk, dhe_psk and rsa_psk:
        /* Input Parameters Incompatibility Error
           INCOMPATIBLE_INPUT_PARAMETERS_ERROR
           AUTHENTICATION_METHOD_UNACCEPTABLE_ERROR
           is generated
        */
```

PreMasterInputParams

3.2. Master Computation

The computation of a master secret needs more information than the computation of the premaster. Similarly to the premaster generation, generation of a master secret is reserved to a subset of authentication methods. Note that the PSK based authentication methods are able to generate master secret but not to generate premaster. The reason is that the premaster embeds the PSK in clear text.

Similarly to the premaster generation, when an authentication method is not permitted an Input Parameters Incompatibility Error INCOMPATIBLE_INPUT_PARAMETERS_ERROR with and AUTHENTICATION_METHOD_UNACCEPTABLE_ERROR are returned.

As premaster are used to compute the master, when a authentication method enable the generation of the premaster the same input parameters must be also provided to the Content Provider.

Master secret may be generated using different methods and using different inputs. More specifically, the extended master secret may be generated from the hash of the exchange messages, or the exchange message themselves. The figure below represent a method by the type of the master secret (that is the standard master secret or the extended master secret) as well as the expected inputs. Such representation is not normative and provided for illustration. As a result, alternative representation may be used.

```
MasterInputParams
    AuthenticationMethod
    MasterMethod = [standard_master,
                    extended_master_from_session_hash,
                    extended_master_method_from_handshake_messages]
    MasterInputData

MasterInputData
    select(AuthenticationMethod)
    case rsa:
        MasterMethodParams
        RSAEncryptedPreMaster
    case dh-scdh:
        MasterMethodParams
        DHECDHTLSClientPublicDHECDHKey
    case dhe_dss, dhe_rsa, ecdhe_ecdsa, dh_anon, ecdh_anon
         ecdhe_rsa, psk
        /* Input Parameters Incompatibility Error
           INCOMPATIBLE_INPUT_PARAMETERS_ERROR
           AUTHENTICATION_METHOD_UNACCEPTABLE_ERROR
           is generated
        */
    case psk, dhe_psk, rsa_psk:
        PSKInputData


MasterMethodParams
    select(MasterMethod)
    case standard_master:
        - ClientHello.random
```

```
            - ClientServer.random
            - PRF
        case extended_master_from_session_hash
            - session_hash
            - PRF
        case extended_master_method_from_handshake_messages
            - handshake_messages
            - PRF

    PSKInputData
       select(PSKSubAuthenticationMethod)
        case psk:
            - PSK_ID
        case PSK_rsa:
            - RSAEncryptedPreMaster
        case psk_dhe:
            - TLSClientEdgeServerDHSecret
            - PSK_ID
```

MasterInputParams

3.3. Signature Computation

Similarly to the generation of the premaster and the generation of the master, the signature is associated to restricted subset of authentication methods, i.e. dhe_dss, dhe_rsa, ecdhe_ecdsa ecdhe_rsa. When another authentication method is chosen an Input Parameters Incompatibility Error INCOMPATIBLE_INPUT_PARAMETERS_ERROR with an AUTHENTICATION_METHOD_UNACCEPTABLE_ERROR are returned.

Similarly to the generation of the master, different method may be used to generate the signature and different parameters may be needed to generate the signature. In some case, the hash is provided whereas in some other case, the whole content to sign is provided.

The figure below provides a representation of the possible parameters provided by the Edge Server.

```
SignatureInputParams
     AuthenticationMethod
     SignatureMethod = (CryptoAlgo = [dss, rsa, ecdsa],
                        DataType = [hash, content])
     SignatureInputData

SignatureInputData
    select(AuthenticationMethod)
        case dhe_dss, dhe_rsa, ecdhe_ecdsa ecdhe_rsa:
            SignatureInputData
        case rsa, dh_dss, dh_rsa, dh_dss, dh_rsa, ecdh_rsa, dh_anon,
             ecdh_anon, psk, dhe_psk, rsa_psk:
        /* Input Parameters Incompatibility Error
           INCOMPATIBLE_INPUT_PARAMETERS_ERROR
           AUTHENTICATION_METHOD_UNACCEPTABLE_ERROR
           is generated
        */

SignatureInput
    select(SignatureMethod.DataType)
    case hash:
        - DataHash
    case content:
        - ClientHello.random
        - ClientServer.random
        - TLSClientEdgeServerDHECDHSecret

                          MasterInputParams
```

4. Input Parameters Description

This section lists the possible parameters exchanged between the Edge Server and the Content Provider. For each parameters, this section determine what checks and error may be associated to them and communicated to the Edge Server.

In order to address the different errors, for each input, there is an associated Input Format Parameter Errors that indicates the input does not follow the format expected in this document. In addition, there is an Not Permitted Input Parameters Error that indicates the API does not support the provided input. This is mostly due to local policies. In addition, the local policies may also prevent a given input value in combination with other input values. In this case, the error returned by the Content Provider is an Input Parameters Incompatibility Error INCOMPATIBLE_INPUT_PARAMETERS_ERROR with the Not Permitted Input Parameters Error associated to the conflicting parameters.

4.1. Static parameters

4.1.1. API Version

The Input Format Parameter Errors associated to the API version output are:

API_VERSION_FORMAT_ERROR

4.1.2. TLSVersion

The Input Format Parameter Errors and Not Permitted Input Parameters Error associated to the TLS version output are:

TLS_VERSION_FORMAT_ERROR
TLS_VERSION_UNACCEPTABLE_ERROR

4.1.3. ObjectRequest

The Input Format Parameter Errors and Not Permitted Input Parameters Error associated to the requested output are:

REQUEST_OUTPUT_FORMAT_ERROR
REQUEST_OUTPUT_UNACCEPTABLE_ERROR

4.1.4. AuthenticationMethod

The Input Format Parameter Errors and Not Permitted Input Parameters Error associated to the authentication method are:

AUTHENTICATION_METHOD_FORMAT_ERROR
AUTHENTICATION_METHOD_UNACCEPTABLE_ERROR

4.1.5. MasterMethod

The Input Format Parameter Errors and Not Permitted Input Parameters Error associated to the master methods are:

MASTER_METHOD_FORMAT_ERROR
MASTER_METHOD_UNACCEPTABLE_ERROR

4.1.6. SignatureMethod

The Input Format Parameter Errors and Not Permitted Input Parameters Error associated to the signature method are:

SIGNATURE_METHOD_FORMAT_ERROR
SIGNATURE_METHOD_UNACCEPTABLE_ERROR

4.1.7. PRF

The Input Format Parameter Errors and Not Permitted Input Parameters Error associated to the pseudo random function are:

PRF_FUNCTION_FORMAT_ERROR
PRF_FUNCTION_UNACCEPTABLE_ERROR

4.2. TLS Hanshake parameters

4.2.1. ClientHello.random, ClientServer.random

The Input Format Parameter Error associated to the ClientHello randoms is:

CLIENT_RANDOM_FORMAT_ERROR (most likely a length different from 32 bytes)

4.2.2. session_hash, DataHash

The Input Format Parameter Error associated to the hash is:

HASH_FORMAT_ERROR (most likely an unexpected length)

4.2.3. handshake_messages

The Input Format Parameter Error associated to the hash is:

HANDSHAKE_MESAGE_FORMAT_ERROR

4.2.4. PSK_ID

The Not Input Parameter Error associated to the hash is:

PSK_ID_UNACCEPTABLE_ERROR (unkown PSK id)

4.3. Cryptographic Parameters

4.3.1. RSAEncryptedPreMaster

Encrypted RSA values are expected to follow the PKCS#1 format. Upon receipt of the parameter, the Content Provider checks the format of encrypted parameters. If an error is detected, the Content Provider can either respond with a randomly generated pre_master or master or return a Input Format Parameter Error. The random value will result in an aborted session, and so motivation for providing a random value is to prevent notifying the Edge Server a format error has been detected. This behavior is recommended when the Content provider

does not trust the Edge Server or suspect it is under attack. On the other hand, sending a error may also notify the Edge Server an attack is being suspected, so mitigating mechanisms may be activated by the Edge Server. In any case, if an format error is detected the error returned by the Content Provider may be:

RSA_ENCRYPTED_FORMAT_ERROR

If the parameters ar enot expected, the Content Provider may provide in addition to an INCOMPATIBLE_INPUT_PARAMETERS_ERROR the following message:

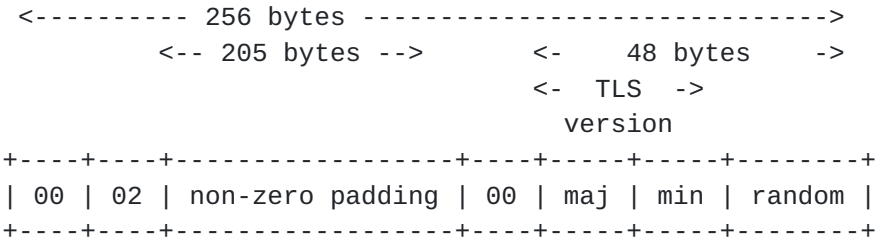
RSA_ENCRYPTED_UNACCEPTABLE_ERROR

4.3.1.1. Checking RSA Encrypted Premaster Format

The first two bytes must be 00 02 followed by non zero padding until a 00 byte is found, followed by the two byte for the TLS version and finally the 46 bytes of the pre master secret. If the format is appropriated the premaster is returned, otherwise an ERROR_UNVALID_ENCRYPTED_MASTER is returned, or a randomly generated Premaster Secret which will silently discard the TLS session - see Section 7.4.7.1 of [RFC5246].

As defined in section 8.1.1 [RFC2546], the pre_master is 48-byte generated by the TLS Client. The two first bytes indicates the TLS version and MUST be the same value as the one provided by the ClientHello.client_version, and the remaining 46 bytes are expected to be random.

The pre_master is encrypted with the public key of the TLS Server as a EncryptedPreMasterSecret structure sent in the Client Key Exchange Message as described in section 7.4.7.1 [RFC5246]. The encryption follows for compatibility with previous TLS version RSAES-PKCS1-v1_5 scheme described in [RFC3447], which results in a 256 byte encrypted message for a 2048-bit RSA key or 128 byte encrypted message for a 1024 bit RSA key.



PKCS#1 padding for pre_master secret encrypted with 2048-bit RSA key

4.3.2. DHECDHTLSClientPublicDHECDHKey

TBD

4.3.3. TLSClientEdgeServerDHSecret

TBD

4.3.4. TLSClientEdgeServerDHECDHSecret

TBD

5. Output Parameters Description

5.1. Premaster

The premaster and master are an opaque number bytes. premaster are 46 byte length and master are 48 byte length.

Note that the PreMasterSecret structure of [RFC5246] includes the protocol version.

```
struct {
    ProtocolVersion client_version;
    opaque random[46];
} PreMasterSecret;
```

PreMasterSecret Structure

5.2. Signature

In order to identify the signature, the signature structure should have the signature algorithm, the hash algorithm and the value of the signature.

5.3. Error

In this document, the Error message can carry various messages. More specifically, when a query is not accepted because of incompatibility of parameters provided, the Content provider returns INCOMPATIBLE_INPUT_PARAMETERS_ERROR. In order to provide more indication to the Edge Server additional message as the convicting parameters may be added.

6. Security Considerations

7. Acknowledgements

8. References

[RFC2546] Durand, A. and B. Buclin, "6Bone Routing Practice", RFC 2546, DOI 10.17487/RFC2546, March 1999, <http://www.rfc-editor.org/info/rfc2546>.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, DOI 10.17487/RFC3447, February 2003, <http://www.rfc-editor.org/info/rfc3447>.

[RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, DOI 10.17487/RFC4279, December 2005, <http://www.rfc-editor.org/info/rfc4279>.

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, DOI 10.17487/RFC4492, May 2006, <http://www.rfc-editor.org/info/rfc4492>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

Author's Address

Daniel Migault
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada

Phone: +1 514-452-2160
Email: daniel.migault@ericsson.com