ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2019

S. Gerdes
O. Bergmann
C. Bormann
Universitaet Bremen TZI
G. Selander
Ericsson AB
L. Seitz
RISE SICS

March 11, 2019

Datagram Transport Layer Security (DTLS) Profile for Authentication and Authorization for Constrained Environments (ACE)

draft-ietf-ace-dtls-authorize-07

Abstract

This specification defines a profile of the ACE framework that allows constrained servers to delegate client authentication and authorization. The protocol relies on DTLS for communication security between entities in a constrained network using either raw public keys or pre-shared keys. A resource-constrained server can use this protocol to delegate management of authorization information to a trusted host with less severe limitations regarding processing power and memory.

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 https://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 September 12, 2019.

Copyright Notice

Table of Contents

1. Introduction

This specification defines a profile of the ACE framework [I-D.ietf-ace-oauth-authz]. In this profile, a client and a resource server use CoAP [RFC7252] over DTLS [RFC6347] to communicate. The client obtains an access token, bound to a key (the proof-of-possession key), from an authorization server to prove its authorization to access protected resources hosted by the resource server. Also, the client and the resource server are provided by the authorization server with the necessary keying material to establish a DTLS session. The communication between client and authorization server may also be secured with DTLS. This specification supports DTLS with Raw Public Keys (RPK) [RFC7250] and with Pre-Shared Keys (PSK) [RFC4279].

The DTLS handshake requires the client and server to prove that they can use certain keying material. In the RPK mode, the client proves with the DTLS handshake that it can use the RPK bound to the token and the server shows that it can use a certain RPK. The access token must be presented to the resource server. For the RPK mode, the access token needs to be uploaded to the resource server before the handshake is initiated, as described in Section 5.8.1 of the ACE framework [I-D.ietf-ace-oauth-authz].

In the PSK mode, client and server show with the DTLS handshake that they can use the keying material that is bound to the access token. To transfer the access token from the client to the resource server, the "psk_identity" parameter in the DTLS PSK handshake may be used instead of uploading the token prior to the handshake.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Readers are expected to be familiar with the terms and concepts described in [I-D.ietf-ace-oauth-authz] and in [I-D.ietf-ace-oauth-params].

The authorization information (authz-info) resource refers to the authorization information endpoint as specified in [I-D.ietf-ace-oauth-authz].

2. Protocol Overview

The CoAP-DTLS profile for ACE specifies the transfer of authentication information and, if necessary, authorization information between the client (C) and the resource server (RS) during setup of a DTLS session for CoAP messaging. It also specifies how C can use CoAP over DTLS to retrieve an access token from the authorization server (AS) for a protected resource hosted on the resource server.

This profile requires the client to retrieve an access token for protected resource(s) it wants to access on RS as specified in [I-D.ietf-ace-oauth-authz]. Figure 1 shows the typical message flow in this scenario (messages in square brackets are optional):

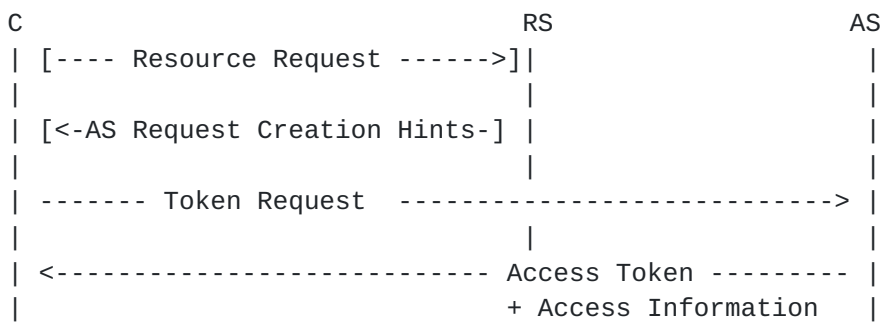

Figure 1: Retrieving an Access Token

To determine the AS in charge of a resource hosted at the RS, C MAY send an initial Unauthorized Resource Request message to the RS. The RS then denies the request and sends an AS Request Creation Hints message containing the address of its AS back to the client as specified in Section 5.1.2 of [I-D.ietf-ace-oauth-authz].

Once the client knows the authorization server's address, it can send an access token request to the token endpoint at the AS as specified in [I-D.ietf-ace-oauth-authz]. As the access token request as well as the response may contain confidential data, the communication between the client and the authorization server MUST be confidentiality-protected and ensure authenticity. C may have been registered at the AS via the OAuth 2.0 client registration mechanism as outlined in Section 5.3 of [I-D.ietf-ace-oauth-authz].

The access token returned by the authorization server can then be used by the client to establish a new DTLS session with the resource server. When the client intends to use an asymmetric proof-of-possession key in the DTLS handshake with the resource server, the client MUST upload the access token to the authz-info resource, i.e. the authz-info endpoint, on the resource server before starting the DTLS handshake, as described in Section 5.8.1 of [I-D.ietf-ace-oauth-authz]. In case the client uses a symmetric proof-of-possession key in the DTLS handshake, the procedure as above MAY be used, or alternatively, the access token MAY instead be transferred in the DTLS ClientKeyExchange message (see Section 3.3.1).

Figure 2 depicts the common protocol flow for the DTLS profile after the client C has retrieved the access token from the authorization server AS.

Figure 2: Protocol overview

3. Protocol Flow

The following sections specify how CoAP is used to interchange access-related data between the resource server, the client and the authorization server so that the authorization server can provide the client and the resource server with sufficient information to establish a secure channel, and convey authorization information specific for this communication relationship to the resource server.

Section 3.1 describes how the communication between C and AS must be secured. Depending on the used CoAP security mode (see also Section 9 of [RFC7252], the Client-to-AS request, AS-to-Client response and DTLS session establishment carry slightly different information. Section 3.2 addresses the use of raw public keys while Section 3.3 defines how pre-shared keys are used in this profile.

3.1. Communication between C and AS

To retrieve an access token for the resource that the client wants to access, the client requests an access token from the authorization server. Before C can request the access token, C and AS MUST establish a secure communication channel. C MUST securely have obtained keying material to communicate with AS. Furthermore, C MUST verify that AS is authorized to provide access tokens (including authorization information) about RS to C. Also, AS MUST securely have obtained keying material for C, and obtained authorization rules approved by the resource owner (RO) concerning C and RS that relate to this keying material. C and AS MUST use their respective keying material for all exchanged messages. How the security association between C and AS is bootstrapped is not part of this document. C and AS MUST ensure the confidentiality, integrity and authenticity of all exchanged messages.

If C is constrained, C and AS should use DTLS to communicate with each other. But C and AS may also use other means to secure their communication, e.g., TLS. The used security protocol MUST fulfill

the communication security requirements in Section 6.2 of [I-D.ietf-ace-oauth-authz].

3.2. RawPublicKey Mode

After C and AS mutually authenticated each other and validated each other's authorization, C sends a token request to AS's token endpoint. The client MUST add a "req_cnf" object carrying either its raw public key or a unique identifier for a public key that it has previously made known to the authorization server. To prove that the client is in possession of this key, C MUST use the same keying material that it uses to secure the communication with AS, e.g., the DTLS session.

An example access token request from the client to the AS is depicted in Figure 3.

```
POST coaps://as.example.com/token
Content-Format: application/ace+cbor
Payload:
{
  "grant_type" : "client_credentials",
  "req_aud"    : "tempSensor4711",
  "req_cnf"    : {
    "COSE_Key" : {
      "kty" : "EC2",
      "crv" : "P-256",
      "x"   : h'e866c35f4c3c81bb96a1...',
      "y"   : h'2e25556be097c8778a20...'
    }
  }
}
```

Figure 3: Access Token Request Example for RPK Mode

The example shows an access token request for the resource identified by the string "tempSensor4711" on the authorization server using a raw public key.

AS MUST check if the client that it communicates with is associated with the RPK in the cnf object before issuing an access token to it. If AS determines that the request is to be authorized according to the respective authorization rules, it generates an access token response for C. The access token MUST be bound to the RPK of the client by means of the cnf claim. The response MAY contain a "profile" parameter with the value "coap_dtls" to indicate that this profile MUST be used for communication between the client C and the resource server. The "profile" may be specified out-of-band, in

which case it does not have to be sent. The response also contains an access token and an "rs_cnf" parameter containing information about the public key that is used by the resource server. AS MUST ascertain that the RPK specified in "rs_cnf" belongs to the resource server that C wants to communicate with. AS MUST protect the integrity of the token. If the access token contains confidential data, AS MUST also protect the confidentiality of the access token.

C MUST ascertain that the access token response belongs to a certain previously sent access token request, as the request may specify the resource server with which C wants to communicate.

An example access token response from the AS to the client is depicted in Figure 4.

```
2.01 Created
Content-Format: application/ace+cbor
Max-Age: 3600
Payload:
{
  "access_token" : "b64'SlAV32hkKG ...
   (remainder of CWT omitted for brevity;
   CWT contains clients RPK in the "cnf" claim)',
  "expires_in" : "3600",
  "rs_cnf"     : {
    "COSE_Key"  : {
      "kty" : "EC2",
      "crv" : "P-256",
      "x"   : h'd7cc072de2205bdc1537...',
      "y"   : h'f95e1d4b851a2cc80fff...'
    }
  }
}
```

Figure 4: Access Token Response Example for RPK Mode

3.2.1. DTLS Channel Setup Between C and RS

Before the client initiates the DTLS handshake with the resource server, C MUST send a "POST" request containing the new access token to the authz-info resource hosted by the resource server. After the client
receives a confirmation that the RS has accepted the access token, it SHOULD proceed to establish a new DTLS channel with the resource server. To use the RawPublicKey mode, the client MUST specify the public key that AS defined in the "cnf" field of the access token response in the SubjectPublicKeyInfo structure in the DTLS handshake as specified in [RFC7250].

An implementation that supports the RPK mode of this profile MUST at least support the ciphersuite TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 [RFC7251] with the ed25519 curve (cf. [RFC8032], [RFC8422]).

Note: According to [RFC7252], CoAP implementations MUST support the ciphersuite TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 [RFC7251] and the NIST P-256 curve. As discussed in [RFC7748], new ECC curves have been defined recently that are considered superior to the so-called NIST curves. The curve that is mandatory to implement in this specification is said to be efficient and less dangerous regarding implementation errors than the secp256r1 curve mandated in [RFC7252].

RS MUST check if the access token is still valid, if RS is the intended destination, i.e., the audience, of the token, and if the token was issued by an authorized AS. The access token is constructed by the authorization server such that the resource server can associate the access token with the Client's public key. The "cnf" claim MUST contain either C's RPK or, if the key is already known by the resource server (e.g., from previous communication), a reference to this key. If the authorization server has no certain knowledge that the Client's key is already known to the resource server, the Client's public key MUST be included in the access token's "cnf" parameter. If CBOR web tokens [RFC8392] are used as recommended in [I-D.ietf-ace-oauth-authz], keys MUST be encoded as specified in [I-D.ietf-ace-cwt-proof-of-possession]. RS MUST use the keying material in the handshake that AS specified in the rs_cnf parameter in the access token. Thus, the handshake only finishes if C and RS are able to use their respective keying material.

3.3. PreSharedKey Mode

To retrieve an access token for the resource that the client wants to access, the client MAY include a "cnf" object carrying an identifier for a symmetric key in its access token request to the authorization server. This identifier can be used by the authorization server to determine the shared secret to construct the proof-of-possession token. AS MUST check if the identifier refers to a symmetric key that was previously generated by AS as a shared secret for the communication between this client and the resource server.

The authorization server MUST determine the authorization rules for the C it communicates with as defined by RO and generate the access token accordingly. If the authorization server authorizes the client, it returns an AS-to-Client response. If the profile parameter is present, it is set to "coap_dtls". AS MUST ascertain that the access token is generated for the resource server that C wants to communicate with. Also, AS MUST protect the integrity of

the access token. If the token contains confidential data such as the symmetric key, the confidentiality of the token MUST also be protected. Depending on the requested token type and algorithm in the access token request, the authorization server adds access Information to the response that provides the client with sufficient information to setup a DTLS channel with the resource server. AS adds a "cnf" parameter to the access information carrying a "COSE_Key" object that informs the client about the symmetric key that is to be used between C and the resource server. The access token MUST be bound to the same symmetric key by means of the cnf claim.

An example access token request for an access token with a symmetric proof-of-possession key is illustrated in Figure 5.

```
POST coaps://as.example.com/token
Content-Format: application/ace+cbor
Payload:
{
  "audience"    : "smokeSensor1807",
}
```

Figure 5: Example Access Token Request, symmetric PoP-key

An example access token response is illustrated in Figure 6. In this example, the authorization server returns a 2.01 response containing a new access token and information for the client, including the symmetric key in the cnf claim. The information is transferred as a CBOR data structure as specified in [I-D.ietf-ace-oauth-authz].

```
   2.01 Created
   Content-Format: application/ace+cbor
   Max-Age: 86400
   Payload:
   {
      "access_token" : h'd08343a10...
      (remainder of CWT omitted for brevity)
      "token_type" : "pop",
      "expires_in" : 86400,
      "profile"    : "coap_dtls",
      "cnf"        : {
        "COSE_Key" : {
          "kty" : "symmetric",
          "kid" : h'3d027833fc6267ce',
          "k"   : h'73657373696f6e6b6579'
        }
      }
   }
```

Figure 6: Example Access Token Response, symmetric PoP-key

The access token also comprises a "cnf" claim. This claim usually contains a "COSE_Key" object that carries either the symmetric key itself or a key identifier that can be used by the resource server to determine the secret key shared with the client. If the access token carries a symmetric key, the access token MUST be encrypted using a "COSE_Encrypt0" structure. The AS MUST use the keying material shared with the RS to encrypt the token.

The "cnf" structure in the access token is provided in Figure 7.

```
"cnf" : {
  "COSE_Key" : {
    "kty" : "symmetric",
    "kid" : h'eIiOFCa9lObw'
  }
}
```

Figure 7: Access Token without Keying Material

A response that declines any operation on the requested resource is constructed according to Section 5.2 of [RFC6749], (cf. Section 5.6.3. of [I-D.ietf-ace-oauth-authz]).

```
4.00 Bad Request
Content-Format: application/ace+cbor
Payload:
{
  "error" : "invalid_request"
}
```

Figure 8: Example Access Token Response With Reject

The method for how the resource server determines the symmetric key from an access token containing only a key identifier is application specific, the remainder of this section provides one example.

The AS and the resource server are assumed to share a key derivation key used to derive the symmetric key shared with the client from the key identifier in the access token. The key derivation key may be derived
from some other secret key shared between the AS and the resource server. This key needs to be securely stored and processed in the same way as the key used to protect the communication between AS and RS.

Knowledge of the symmetric key shared with the client must not reveal any information about the key derivation key or other secret keys shared between AS and resource server.

In order to generate a new symmetric key to be used by client and resource server, the AS generates a key identifier and uses the key derivation key shared with the resource server to derive the symmetric key as specified below. Instead of providing the keying material in the access token, the AS includes the key identifier in the "kid" parameter, see Figure 7. This key identifier enables the resource server to calculate the keying material for the communication with the client from the access token using the key derivation key and following Section 11 of [RFC8152] with parameters as specified here. The KDF to be used needs to be defined by the application, for example HKDF-SHA-256. The key identifier picked by the AS needs to be unique for each access token where a unique symmetric key is required.

The fields in the context information "COSE_KDF_Context" (Section 11.2 of [RFC8152]) have the following values:

- AlgorithmID = "ACE-CoAP-DTLS-key-derivation"
- PartyUInfo = PartyVInfo = (null, null, null)
- keyDataLength needs to be defined by the application

- protected MUST be a zero length bstr
- other is a zero length bstr
- SuppPrivInfo is omitted

3.3.1. DTLS Channel Setup Between C and RS

When a client receives an access token response from an authorization server, C MUST ascertain that the access token response belongs to a certain previously sent access token request, as the request may specify the resource server with which C wants to communicate.

C checks if the payload of the access token response contains an "access_token" parameter and a "cnf" parameter. With this information the client can initiate the establishment of a new DTLS channel with a resource server. To use DTLS with pre-shared keys, the client follows the PSK key exchange algorithm specified in Section 2 of [RFC4279] using the key conveyed in the "cnf" parameter of the AS response as PSK when constructing the premaster secret.

In PreSharedKey mode, the knowledge of the shared secret by the client and the resource server is used for mutual authentication between both peers. Therefore, the resource server must be able to determine the shared secret from the access token. Following the general ACE authorization framework, the client can upload the access token to the resource server's authz-info resource before starting the DTLS handshake. Alternatively, the client MAY provide the most recent access token in the "psk_identity" field of the ClientKeyExchange message. To do so, the client MUST treat the contents of the "access_token" field from the AS-to-Client response as opaque data and not perform any re-coding.

Note: As stated in Section 4.2 of [RFC7925], the PSK identity should be treated as binary data in the Internet of Things space and not assumed to have a human-readable form of any sort.

If a resource server receives a ClientKeyExchange message that contains a "psk_identity" with a length greater zero, it uses the contents as index for its key store (i.e., treat the contents as key identifier). The resource server MUST check if it has one or more access tokens that are associated with the specified key.

If no key with a matching identifier is found, the resource server MAY process the contents of the "psk_identity" field as access token that is stored with the authorization information endpoint, before continuing the DTLS handshake. If the contents of the "psk_identity" do not yield a valid access token for the requesting client, the DTLS

session setup is terminated with an "illegal_parameter" DTLS alert message.

Note1: As a resource server cannot provide a client with a meaningful PSK identity hint in response to the client's ClientHello message, the resource server SHOULD NOT send a ServerKeyExchange message.

Note2: According to [RFC7252], CoAP implementations MUST support the ciphersuite TLS_PSK_WITH_AES_128_CCM_8 [RFC6655]. A client is therefore expected to offer at least this ciphersuite to the resource server.

When RS receives an access token, RS MUST check if the access token is still valid, if RS is the intended destination, i.e., the audience of the token, and if the token was issued by an authorized AS. This specification assumes that the access token is a PoP token as described in [I-D.ietf-ace-oauth-authz] unless specifically stated otherwise. Therefore, the access token is bound to a symmetric PoP key that is used as shared secret between the client and the resource server.

While the client can retrieve the shared secret from the contents of the "cnf" parameter in the AS-to-Client response, the resource server uses the information contained in the "cnf" claim of the access token to determine the actual secret when no explicit "kid" was provided in the "psk_identity" field. If key derivation is used, the RS uses the "COSE_KDF_Context" information as described above.

3.4. Resource Access

Once a DTLS channel has been established as described in Section 3.2 and Section 3.3, respectively, the client is authorized to access resources covered by the access token it has uploaded to the authz-info resource hosted by the resource server.

With the successful establishment of the DTLS channel, C and RS have proven that they can use their respective keying material. An access token that is bound to the client's keying material is associated with the channel. Any request that the resource server receives on this channel MUST be checked against these authorization rules. RS MUST check for every request if the access token is still valid. Incoming CoAP requests that are not authorized with respect to any access token that is associated with the client MUST be rejected by the resource server with 4.01 response as described in Section 5.1.1 of [I-D.ietf-ace-oauth-authz].

The resource server SHOULD treat an incoming CoAP request as authorized if the following holds:

1. The message was received on a secure channel that has been established using the procedure defined in this document.

2. The authorization information tied to the sending client is valid.

3. The request is destined for the resource server.

4. The resource URI specified in the request is covered by the authorization information.

5. The request method is an authorized action on the resource with respect to the authorization information.

Incoming CoAP requests received on a secure DTLS channel that are not thus authorized MUST be rejected according to Section 5.8.2 of [I-D.ietf-ace-oauth-authz]

1. with response code 4.03 (Forbidden) when the resource URI specified in the request is not covered by the authorization information, and

2. with response code 4.05 (Method Not Allowed) when the resource URI specified in the request covered by the authorization information but not the requested action.

The client cannot always know a priori if an Authorized Resource Request will succeed. It MUST check the validity of its keying material before sending a request or processing a response. If the client repeatedly gets error responses containing AS Creation Hints (cf. Section 5.1.2 of [I-D.ietf-ace-oauth-authz] as response to its requests, it SHOULD request a new access token from the authorization server in order to continue communication with the resource server.

Unauthorized requests that have been received over a DTLS session SHOULD be treated as non-fatal by the RS, i.e., the DTLS session SHOULD be kept alive until the associated access token has expired.

4. Dynamic Update of Authorization Information

The client can update the authorization information stored at the resource server at any time without changing an established DTLS session. To do so, the Client requests a new access token from the authorization server for the intended action on the respective

resource and uploads this access token to the authz-info resource on the resource server.

Figure 9 depicts the message flow where the C requests a new access token after a security association between the client and the resource server has been established using this protocol. If the client wants to update the authorization information, the token request MUST specify the key identifier of the proof-of-possession key used for the existing DTLS channel between the client and the resource server in the "kid" parameter of the Client-to-AS request. The authorization server MUST verify that the specified "kid" denotes a valid verifier for a proof-of-possession token that has previously been issued to the requesting client. Otherwise, the Client-to-AS request MUST be declined with the error code "unsupported_pop_key" as defined in Section 5.6.3 of [I-D.ietf-ace-oauth-authz].

When the authorization server issues a new access token to update existing authorization information, it MUST include the specified "kid" parameter in this access token. A resource server MUST replace the authorization information of any existing DTLS session that is identified by this key identifier with the updated authorization information.

Note: By associating the access tokens with the identifier of an existing DTLS session, the authorization information can be updated without changing the cryptographic keys for the DTLS communication between the client and the resource server, i.e. an existing session can be used with updated permissions.

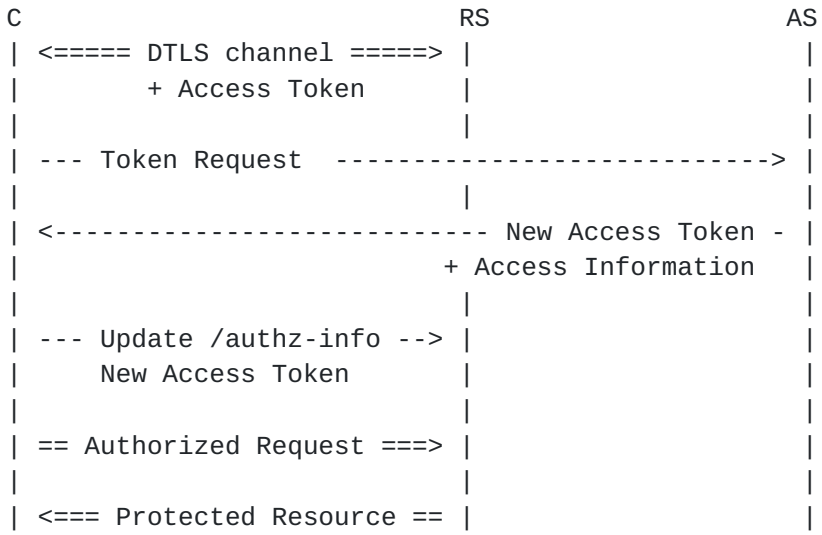

Figure 9: Overview of Dynamic Update Operation

5. Token Expiration

DTLS sessions that have been established in accordance with this profile are always tied to a specific access token. As this token may become invalid at any time (e.g. because it has expired), the session may become useless at some point. A resource server therefore MUST terminate existing DTLS sessions after the access token for this session has been deleted.

As specified in Section 5.8.3 of [I-D.ietf-ace-oauth-authz], the resource server MUST notify the client with an error response with code 4.01 (Unauthorized) for any long running request before terminating the session.

6. Security Considerations

This document specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework [I-D.ietf-ace-oauth-authz]. As it follows this framework's general approach, the general security considerations from section 6 also apply to this profile.

When using pre-shared keys provisioned by the AS, the security level depends on the randomness of PSK, and the security of the TLS cipher suite and key exchange algorithm.

Constrained devices that use DTLS [RFC6347] are inherently vulnerable to Denial of Service (DoS) attacks as the handshake protocol requires creation of internal state within the device. This is specifically of concern where an adversary is able to intercept the initial cookie exchange and interject forged messages with a valid cookie to continue with the handshake. A similar issue exists with the authorization information endpoint where the resource server needs to keep valid access tokens until their expiry. Adversaries can fill up the constrained resource server's internal storage for a very long time with interjected or otherwise retrieved valid access tokens.

The use of multiple access tokens for a single client increases the strain on the resource server as it must consider every access token and calculate the actual permissions of the client. Also, tokens may contradict each other which may lead the server to enforce wrong permissions. If one of the access tokens expires earlier than others, the resulting permissions may offer insufficient protection. Developers SHOULD avoid using multiple access tokens for a client.

7. Privacy Considerations

This privacy considerations from section 7 of the [I-D.ietf-ace-oauth-authz] apply also to this profile.

An unprotected response to an unauthorized request may disclose information about the resource server and/or its existing relationship with the client. It is advisable to include as little information as possible in an unencrypted response. When a DTLS session between the client and the resource server already exists, more detailed information MAY be included with an error response to provide the client with sufficient information to react on that particular error.

Also, unprotected requests to the resource server may reveal information about the client, e.g., which resources the client attempts to request or the data that the client wants to provide to the resource server. The client SHOULD NOT send confidential data in an unprotected request.

Note that some information might still leak after DTLS session is established, due to observable message sizes, the source, and the destination addresses.

8. IANA Considerations

The following registrations are done for the ACE OAuth Profile Registry following the procedure specified in [I-D.ietf-ace-oauth-authz].

Note to RFC Editor: Please replace all occurrences of "[RFC-XXXX]" with the RFC number of this specification and delete this paragraph.

Profile name: coap_dtls

Profile Description: Profile for delegating client authentication and authorization in a constrained environment by establishing a Datagram Transport Layer Security (DTLS) channel between resource-constrained nodes.

Profile ID: 1

Change Controller: IESG

Reference: [RFC-XXXX]

9. References

9.1. Normative References

[I-D.ietf-ace-cwt-proof-of-possession]
Jones, M., Seitz, L., Selander, G., Erdtman, S., and H. Tschofenig, "Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs)", draft-ietf-ace-cwt-proof-of-possession-06 (work in progress), February 2019.

[I-D.ietf-ace-oauth-authz]
Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth)", draft-ietf-ace-oauth-authz-22 (work in progress), March 2019.

[I-D.ietf-ace-oauth-params]
Seitz, L., "Additional OAuth Parameters for Authorization in Constrained Environments (ACE)", draft-ietf-ace-oauth-params-04 (work in progress), February 2019.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[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, <https://www.rfc-editor.org/info/rfc4279>.

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

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC7925] Tschofenig, H., Ed. and T. Fossati, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", RFC 7925, DOI 10.17487/RFC7925, July 2016, <https://www.rfc-editor.org/info/rfc7925>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Informative References

[RFC6655] McGrew, D. and D. Bailey, "AES-CCM Cipher Suites for Transport Layer Security (TLS)", RFC 6655, DOI 10.17487/RFC6655, July 2012, <https://www.rfc-editor.org/info/rfc6655>.

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <https://www.rfc-editor.org/info/rfc7250>.

[RFC7251] McGrew, D., Bailey, D., Campagna, M., and R. Dugal, "AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS", RFC 7251, DOI 10.17487/RFC7251, June 2014, <https://www.rfc-editor.org/info/rfc7251>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/info/rfc8392>.

[RFC8422] Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier", RFC 8422, DOI 10.17487/RFC8422, August 2018, <https://www.rfc-editor.org/info/rfc8422>.

Authors' Addresses

Stefanie Gerdes
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63906
Email: gerdes@tzi.org

Olaf Bergmann
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63904
Email: bergmann@tzi.org

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org

Goeran Selander
Ericsson AB

Email: goran.selander@ericsson.com

Ludwig Seitz
RISE SICS
Scheelevaegen 17
Lund 223 70
Sweden

Email: ludwig.seitz@ri.se