ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 7, 2020

C. Sengul
Nominet
A. Kirby
Oxbotica
P. Fremantle
University of Portsmouth

October 5, 2019

MQTT-TLS profile of ACE
draft-ietf-ace-mqtt-tls-profile-01

Abstract

This document specifies a profile for the ACE (Authentication and Authorization for Constrained Environments) to enable authorization in an MQTT-based publish-subscribe messaging system. Proof-of-possession keys, bound to OAuth2.0 access tokens, are used to authenticate and authorize MQTT Clients. The protocol relies on TLS for confidentiality and server authentication.

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 April 7, 2020.

Table of Contents

1. Introduction

This document specifies a profile for the ACE framework [I-D.ietf-ace-oauth-authz]. In this profile, Clients and a Broker use MQTT to exchange Application messages. The protocol relies on TLS for communication security between entities. The MQTT protocol interactions are described based on the MQTT v5.0 - the OASIS Standard [MQTT-OASIS-Standard-v5]. It is expected that MQTT deployments will retain backward compatibility for MQTT v3.1.1 clients, and therefore, this document describes a reduced set of protocol interactions suited to MQTT v3.1.1 - the OASIS Standard [MQTT-OASIS-Standard]. However, it is RECOMMENDED to use MQTT v5.0 as it works more naturally with ACE-style authentication and authorization.

MQTT is a publish-subscribe protocol and supports two main types of Client operation: publish and subscribe. Once connected, a Client can publish to multiple topics, and subscribe to multiple topics. The MQTT Broker is responsible for distributing messages published by the publishers to the appropriate subscribers. Each publish message contains a Topic Name, which is used by the Broker to filter the subscribers for the message. Subscribers must subscribe to the topics to receive the corresponding messages.

In this document, message topics are treated as resources. Clients use an access token, bound to a key (the proof-of-possession key) to authorize with the MQTT Broker their connection and publish/subscribe permissions to topics. In the context of this ACE profile, the MQTT Broker acts as the Resource Server (RS). In the rest of the document RS and Broker are used interchangeably. To provide communication confidentiality and Resource Server authentication, TLS is used. This document makes the same assumptions as the Section 4 of the ACE framework [I-D.ietf-ace-oauth-authz] regarding Client and RS registration with the AS and establishing of keying material.

This document describes the authorization of the following exchanges between Clients and the Broker.

- Connection establishment between the Clients and the Broker
- Publish messages from the Clients to the Broker, and from the Broker to the Clients
- Subscribe messages from the Clients to the Broker

While the Client-Broker exchanges are over MQTT, the required Client-AS and RS-AS interactions are described for HTTPS-based communication, using 'application/ace+json' content type, and unless

otherwise specified, using JSON encoding. The token may be a reference, or JWT. For JWT tokens, this document follows RFC 7800 [RFC7800] for PoP semantics for JWTs. The Client-AS and RS-AS may also be based on CoAP. It is also possible to use 'application/ace+cbor' content type, and CBOR encoding, and CWT and associated PoP semantics to reduce the protocol memory and bandwidth requirements. For more information on Proof of Possession semantics for CWTs, see Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs) [I-D.ietf-ace-cwt-proof-of-possession].

1.1. Requirements Language

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.

1.2. ACE-Related Terminology

The terminology for entities in the architecture is defined in OAuth 2.0 RFC 6749 [RFC6749] such as "Client" (C), "Resource Server" (RS) and "Authorization Server" (AS).

The term "endpoint" is used following its OAuth definition, to denote resources such as /token and /introspect at the AS.

The term "Resource" is used to refer to an MQTT Topic Name, which is defined in Section 1.3. Hence, the "Resource Owner" is any entity that can authoritatively speak for the topic.

Certain security-related terms such as "authentication", "authorization", "confidentiality", "(data) integrity", "message authentication code", and "verify" are taken from RFC 4949 [RFC4949].

1.3. MQTT-Related Terminology

The document describes message exchanges as MQTT protocol interactions. The Clients are MQTT Clients, which connect to the Broker to publish and subscribe to Application Messages. For additional information, please refer to the MQTT v5.0 - the OASIS Standard [MQTT-OASIS-Standard-v5] or the MQTT v3.1.1 - the OASIS Standard [MQTT-OASIS-Standard].

MQTTS
: Secured transport profile of MQTT. MQTTS runs over TLS.

Broker

The Server in MQTT and acts as an intermediary between Clients that publish Application Messages, and the Clients that made Subscriptions. The Broker acts as the Resource Server for the Clients.

Application Message
: The data carried by the MQTT protocol. The data has an associated QoS level and a Topic Name.

Topic Name
: The label attached to an Application Message, which is matched to a Subscription.

Subscription
: A subscription comprises a Topic Filter and a maximum Quality of Service (QoS).

Topic Filter
: An expression that indicates interest in one or more Topic Names. Topic Filters may include wildcards.

MQTT sends various control messages across a network connection. The following is not an exhaustive list and the control packets that are not relevant for authorization are not explained. These include, for instance, the PUBREL and PUBCOMP packets used in the 4-step handshake required for the QoS level 2.

CONNECT
: Client request to connect to the Broker. After a network connection is established, this is the first packet sent by a Client.

CONNACK
: The Broker connection acknowledgment. The first packet sent from the Broker to a Client is a CONNACK packet. CONNACK packets contain return codes indicating either a success or an error state to a Client.

PUBLISH
: Publish packet that can be sent from a Client to the Broker, or from the Broker to a Client.

PUBACK
: Response to PUBLISH packet with QoS level 1. PUBACK can be sent from the Broker to a Client or a Client to the Broker.

PUBREC

Response to PUBLISH packet with QoS level 2. PUBREC can be sent from the Broker to a Client or a Client to the Broker.

SUBSCRIBE
: The Client subscribe request.

SUBACK
: Subscribe acknowledgment.

PINGREQ
: A ping request sent from a Client to the Broker. It signals to the Broker that the Client is alive, and is used to confirm that the Broker is still alive.

Will
: An application message published by the Server after the network connection is closed in cases where the network connection is not closed normally. If the Will Flag is set, then the payload of the CONNECT message has information about the Will. The Will consists of the Will Properties, Will Topic, and Will Payload fields in the CONNECT message.

2. Protocol Interactions

This section describes the following exchanges between Clients, the Broker, and the Authorization Server according to the MQTT v5.0.

- Authorizing connection establishment between the Clients and the Broker
- Authorizing publish messages from the Clients to the Broker, and from the Broker to the Clients
- Authorizing subscribe messages from Clients to the Broker

Section 3 describes how these exchanges can also be supported using the MQTT v3.1.1. MQTT v5.0 brokers MAY also only support the basic operation; however, this is NOT RECOMMENDED.

In this profile document, message topics are treated as resources. The Clients are assumed to have identified the publish/subscribe topics of interest out-of-band (topic discovery is not a feature of the MQTT protocol). A resource owner can pre-configure policies at the AS that give Clients publish or subscribe permissions to different topics.

2.1. Authorizing Connection Establishment

This section specifies how Clients establish an authorized connection to an MQTT Broker. Figure 1 shows the basic protocol flow during connection establishment.The token request and response use the /token endpoint of the authorization server, specified in the Section 5.6 of the ACE framework [I-D.ietf-ace-oauth-authz]. Steps (D) and (E) are optional, and use the introspection endpoint, specified in the Section 5.7 of the ACE framework. The Client and Broker use HTTPS to communicate to AS via these endpoints. If the Client is resource-constrained, a Client Authorisation Server may carry out the token request on behalf of the Client, and later, onboard the Client with the token. Also, these interfaces may be implemented using other protocols, e.g., CoAP or MQTT. The interactions between a Client and its Client Authorization Server for token onboarding, and the MQTTS support for token requests are out of scope of this document.

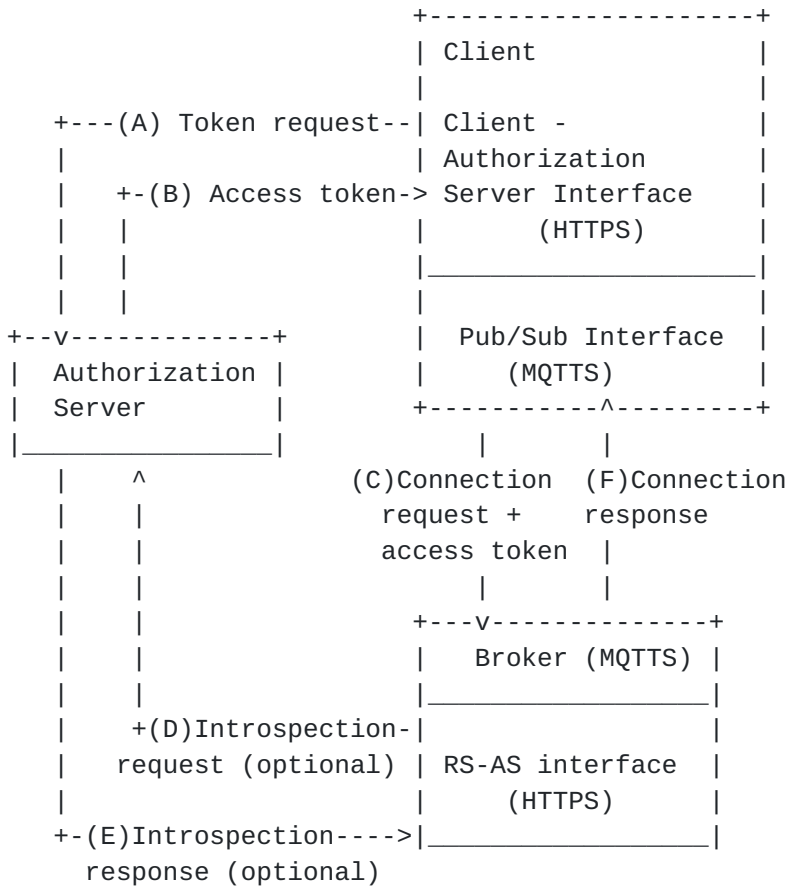

Figure 1: Connection establishment

2.1.1. Client Token Request to the Authorization Server (AS)

The first step in the protocol flow (Figure 1 (A)) is the token acquisition by the Client from the AS. When requesting an access token from the AS, the Client MAY include parameters in its request as defined in Section 5.6.1 of the ACE framework [I-D.ietf-ace-oauth-authz]. The media format is 'application/ace+json'. The profile parameter MUST be set to 'mqtt_tls'. The OAuth 2.0 AS uses a JSON structure in the payload of its responses both to client and RS.

If the AS successfully verifies the access token request and authorizes the Client for the indicated audience (e.g., RS) and scopes (e.g., publish/subscribe permissions over topics), the AS issues an access token (Figure 1 (B)). The response includes the parameters described in Section 5.6.2 of the ACE framework [I-D.ietf-ace-oauth-authz]. The included token is assumed to be Proof-of-Possession (PoP) token by default. This document follows RFC 7800 [RFC7800] for PoP semantics for JWTs. The PoP token includes a 'cnf' parameter with a symmetric or asymmetric PoP key. Note that the 'cnf' parameter in the web tokens are to be consumed by the resource server and not the Client.

In the case of an error, the AS returns error responses for HTTP-based interactions as ASCII codes in JSON content, as defined in Section 5.2 of RFC 6749 [RFC6749].

2.1.2. Client Connection Request to the Broker (C)

This section describes how the Client transports the token to the Broker (RS) via the CONNECT control message after the TLS handshake. This is similar to an earlier proposal by Fremantle et al. [fremantle14]. Alternatively, the token may be used for the TLS session establishment as described in the DTLS profile for ACE [I-D.gerdes-ace-dtls-authorize]. In this case, both the TLS PSK and RPK handshakes MAY be supported. This may additionally require that the Client transports the token to the Broker before the connection establishment. To this end, the Broker MAY support /authz-info endpoint via the "authz-info" topic. Then, to transport the token, Clients publish to "authz-info" topic unauthorized. The topic "authz-info" MUST be publish-only for Clients (i.e., the Clients are not allowed to subscribe to it). This option is described in more detail in Appendix B.

After the token acquisition, the Client connects to the RS (Broker) using the CONNECT message of MQTT over TLS. For server authentication, the client MAY either have the ability to receive and validate a certificate or a raw public key from the Broker. The

client needs to use this raw public key in the TLS handshake together with an out-of-band validation technique (see RFC 7250 [RFC7250] for details).

Figure 2 shows the structure of the MQTT CONNECT control message used in MQTT v5.0. A CONNECT message is composed of a fixed header, a variable header and a payload. The fixed header contains Control Packet Type (CPT), Reserved, and Remaining Length. The Variable Header contains the Protocol Name, Protocol Level, Connect Flags, Keep Alive, and Properties. The Connect Flags in the variable header specify the behavior of the MQTT connection. It also indicates the presence or absence of fields in the Payload. The payload contains one or more encoded fields, namely a unique Client identifier for the Client, a Will Topic, Will Payload, User Name and Password. All but the Client identifier can be omitted depending on flags in the Variable Header.

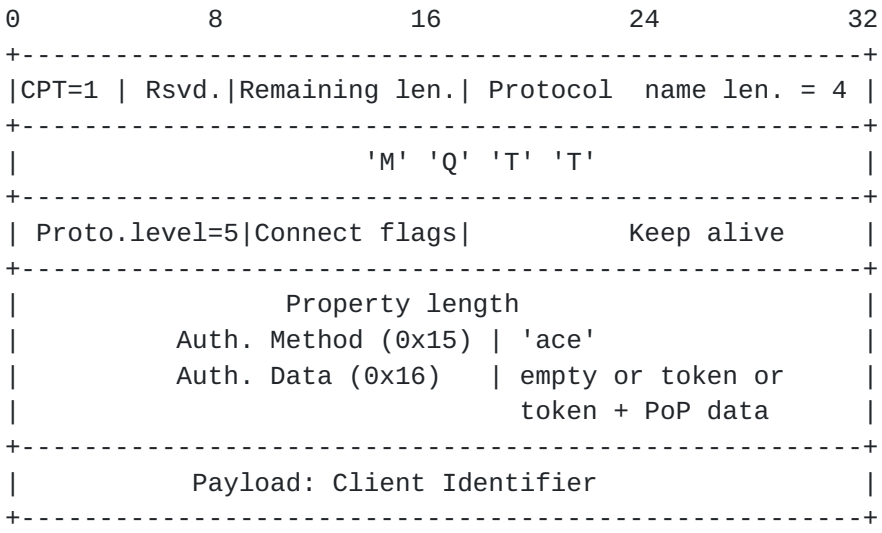

Figure 2: MQTT CONNECT control message. (CPT=Control Packet Type, Rsvd=Reserved, len.=length, Proto.=Protocol)

Connect Flags include Clean Start, Will, Will QoS, Will Retain, Password and Username flags. Figure 6 shows how the MQTT connect flags MUST be set to initiate a connection with the Broker.

User name flag	Pass. flag	Will retain	Will QoS	Will Flag	Clean	Rsvd.
0	0	X	X X	X	1	0

Figure 3: MQTT CONNECT flags. (Rsvd=Reserved)

To achieve a clean session (i.e., the session starts without an existing session), the Clean Start Flag MUST be set to 1. In addition, if the Session Expiry Interval is present in the CONNECT message, it MUST be set to 0.

The Will Flag indicates that a Will message needs to be sent if network connection is not closed normally. The situations in which the Will message is published include disconnections due to I/O or network failures, and the server closing the network connection due to a protocol error. The Client may set the Will Flag as desired (marked as 'X' in Figure 3). If the Will Flag is set to 1 and the Broker accepts the connection request, the Broker must store the Will message, and publish it when the network connection is closed according to Will QoS and Will retain parameters, and MQTT Will management rules. To avoid publishing Will Messages in the case of temporary network disconnections, the Client my specify a Will Delay Interval in Will Properties. Section 2.5 explains how the Broker deals with the retained messages in further detail.

For token transport, the RS SHOULD support AUTH (Authentication Exchange) method. The RS MAY support token transport via username and password, which is described in Section 3 for MQTT v3.1.1. The rest of this section describes the AUTH method, for which the username and password flags MUST be set to 0.

To implement the AUTH (Authentication Exchange) method, the Client MUST set the Authentication Method as a property of a CONNECT packet by using the property identifier 21 (0x15). This is followed by a UTF-8 Encoded String containing the name of the authentication method, which MUST be set to 'ace'. If the RS does not support this profile, it sends a CONNACK with a Reason Code of '0x8C (Bad authentication method)'.

The Authentication Method is followed by the Authentication Data, which has a property identifier 22 (0x16) and is binary data. Based on the Authentication Data, this profile allows:

- Proof-of-Possession over predefined field
- Proof-of-Possession via challenge/response
- Unauthorised request: Authorisation Server discovery

2.1.2.1. Proof-of-Possession over Predefined Field

For this option, the Authentication Data MUST contain the token and the keyed message digest (MAC) or the Client signature. To calculate the keyed message digest (MAC) or the Client signature, the Client

SHOULD apply the PoP key to the CONNECT payload. The CONNECT payload has at least a Client Identifier, and if the Will Flag is set to 1, may contain Will-related information. The Client Identifier is a MUST be a UTF-8 Encoded String (i.e., is prefixed with a two-byte integer length field that gives the number of bytes in a UTF-8 encoded string itself). The Client Identifier may be 1-23 UTF-8 encoded bytes, and contain only the characters "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ". However, according to MQTTv5 standard, the Broker may except longer Client Identifiers, and characters not included in the list given above. Clients MUST change their Client Identifier for each session, if the Client Identifier is the only source of randomness in the payload to defend against a replay attack. If the Client reuses its Client Identifier across different sessions, the Authentication Data MUST also contain a nonce, and the keyed message digest (MAC) or the Client signature MUST be computed over this nonce. Finally, the token is validated as described in Section 2.1.3 and the server responds with a CONNACK.

2.1.2.2. Proof-of-Possession via challenge/response

For this option, the RS follows a challenge/response protocol. The success case is illustrated in Figure 4. If the Authentication Data only includes the token, the RS MUST respond with an AUTH packet, with the Authenticate Reason Code set to '0x18 (Continue Authentication)'. This packet includes the Authentication Method, which MUST be set to 'ace' and Authentication Data. The Authentication Data MUST NOT be empty and contains a challenge for the Client. The Client responds to this with an AUTH packet with a reason code '0x18 (Continue Authentication)'. Similarly, the Client packet sets the Authentication Method to 'ace'. The Authentication Data in the Client's response contains the signature or MAC computed over the RS's challenge. Next, the token is validated as described in Section 2.1.3.

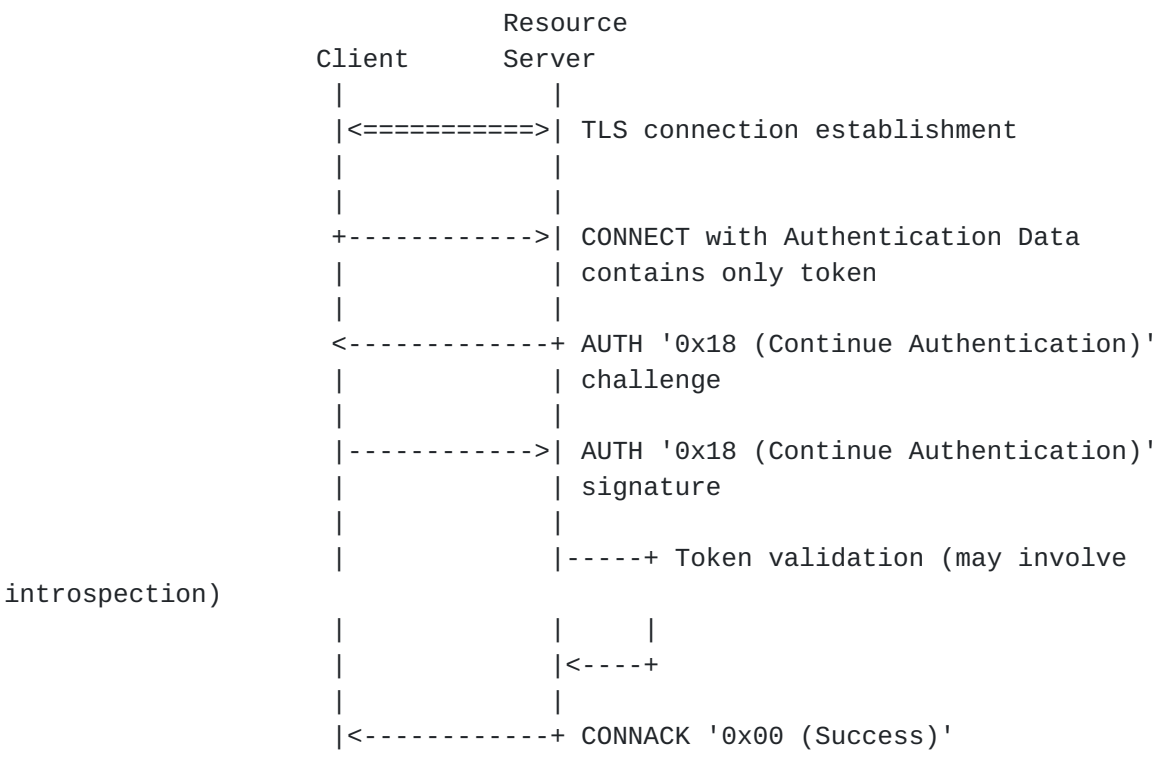

Figure 4: PoP Challenge/Response Protocol Flow - Success

2.1.2.3. Unauthorised Request: Authorisation Server Discovery

Finally, this document allows the CONNECT message to have an empty Authentication Data field. This is the AS discovery option and the RS responds with the CONNACK reason code '0x87 (Not Authorized)' and includes a User Property (identified by 38 (0x26)) for the AS creation hints as dedined in the Section 5.1.2 of the ACE framework [I-D.ietf-ace-oauth-authz].

2.1.3. Token Validation

The RS MUST verify the validity of the token. This validation MAY be done locally (e.g., in the case of a self-contained token) or the RS MAY send an introspection request to the AS. If introspection is used, this section follows similar steps to those described in Sections 5.7 of the ACE framework [I-D.ietf-ace-oauth-authz]. The communication between AS and RS MUST be confidential, mutually authenticated and integrity protected.

The Broker MUST check if the token is active either using 'exp' claim of the token or 'active' parameter of the introspection response. Also, if present in the access token, RS must check that the 'iss' corresponds to AS, the 'aud' field corresponds to RS. It also has to check whether the 'nbf' and the 'iat' claims are present and valid.

To authenticate the Client, the RS validates the signature or the MAC, depending on how the PoP protocol is implemented. To authorize the Client, the Broker uses the scope field in the token (or in the introspection result). The scope field contains the publish and subscribe permissions for the Client. If the Will Flag is set,then the Broker MUST check that the token allows the publication of the Will message.

Scope strings SHOULD be encoded as a permission, followed by an underscore, followed by a topic filter. Two permissions apply to topics: 'publish' and 'subscribe'. An example scope field may contain multiple such strings, space delimited, e.g., 'publish_topic1 subscribe_topic2/#'. Hence, this access token would give 'publish' permission to the 'topic1', 'subscribe' permission to all the subtopics of 'topic2'.

2.1.4. The Broker's Response to Client Connection Request

Based on the validation result (obtained either via local inspection or using the /introspection interface of the AS), the Broker MUST send a CONNACK message to the Client. The reason code of the CONNACK is '0x00 (Success)' if the authentication is successful. In case of an invalid PoP token, the CONNACK reason code is '0x87 (Not Authorized)'.

If the RS accepts the connection, it MUST store the token until the end of connection. On Client or RS disconnection, the token is discarded, and the Client MUST provide a token inside each CONNECT message.

If the token is not self-contained and the Broker uses token introspection, it MAY cache the validation result to authorize the subsequent PUBLISH and SUBSCRIBE messages. PUBLISH and SUBSCRIBE messages, which are sent after a connection set-up, do not contain access tokens. If the introspection result is not cached, then the RS needs to introspect the saved token for each request. The Broker SHOULD use a cache time out to introspect tokens regularly.

2.2. Authorizing PUBLISH Messages

2.2.1. PUBLISH Messages from the Publisher Client to the Broker

On receiving the PUBLISH message, the Broker MUST use the type of message (i.e., PUBLISH) and the Topic name in the message header to compare against the cached token or its introspection result.

If the Client is allowed to publish to the topic, the RS must publish the message to all valid subscribers of the topic. The Broker may

also return an acknowledgment message if the QoS level is greater than or equal to 1.

In case of an authorization failure, an error MAY be returned to the Client. For this the QoS level of the PUBLISH message, should be set to greater than or equal to 1. This guarantees that RS responds with either a PUBACK or PUBREC packet with reason code '0x87 (Not authorized)'.

On receiving a PUBACK with '0x87 (Not authorized)', the Client MAY reauthenticate as described in Section 2.4, and pass a new token following the same PoP methods as described in Figure 2.

2.2.2. PUBLISH Messages from the Broker to the Subscriber Clients

To forward PUBLISH messages to the subscribing Clients, the Broker identifies all the subscribers that have valid matching topic subscriptions (i.e., the tokens are valid, and token scopes allow a subscription to the particular topic). The Broker sends a PUBLISH message with the Topic name to all the valid subscribers.

RS MUST stop forwarding messages to the unauthorized subscribers. For Clients with invalid tokens, there is no way to inform the Client that an authorization error has occurred other than sending a DISCONNECT message. The RS SHOULD send a DISCONNECT message with the reason code '0x87 (Not authorized)'. Note that the server-side DISCONNECT is a new feature of MQTT v5.0 (in MQTT v3.1.1, the server needs to drop the connection).

2.3. Authorizing SUBSCRIBE Messages

In MQTT, a SUBSCRIBE message is sent from a Client to the Broker to create one or more subscriptions to one or more topics. The SUBSCRIBE message may contain multiple Topic Filters. The Topic Filters may include wildcard characters.

On receiving the SUBSCRIBE message, the Broker MUST use the type of message (i.e., SUBSCRIBE) and the Topic Filter in the message header to compare against the stored token or introspection result.

As a response to the SUBSCRIBE message, the Broker issues a SUBACK message. For each Topic Filter, the SUBACK packet includes a return code matching the QoS level for the corresponding Topic Filter. In the case of failure, the return code is 0x87, indicating that the Client is 'Not authorized'. A reason code is returned for each Topic Filter. Therefore, the Client may receive success codes for a subset of its Topic Filters while being unauthorized for the rest.

2.4. Token Expiration and Reauthentication

The Broker MUST check for token expiration whenever a CONNECT, PUBLISH or SUBSCRIBE message is received or sent. The Broker SHOULD check for token expiration on receiving a PINGREQUEST message. This may allow for early detection of a token expiry.

The token expiration is checked by checking the 'exp' claim of a JWT or introspection response, or via performing an introspection request with the Authorization server as described in Section 5.7 of the ACE framework [I-D.ietf-ace-oauth-authz]. Token expirations may trigger the RS to send PUBACK, SUBACK and DISCONNECT messages with return code set to 'Not authorised'. As a response, the Client MAY re-authenticate by sending an AUTH packet with a Reason Code of 0x19 (Re-authentication)

To re-authenticate, the Client sends an AUTH packet with reason code '0x19 (Re-authentication)'. The Client MUST set the authentication method as 'ace' and transport the new token in the Authentication Data. The Client and the RS go through the same steps for proof of possession validation as described in Section 2.1.2. If the re-authentication fails, the server MUST send a DISCONNECT with the reason code '0x87 (Not Authorized)'. The Clients can also proactively update their tokens before they receive a message with 'Not authorized' return code.

2.5. Handling Disconnections and Retained Messages

In the case of a Client DISCONNECT, due to the Clean Session flag, the Broker deletes all session state but MUST keep the retained messages. By setting a RETAIN flag in a PUBLISH message, the publisher indicates to the Broker that it should store the most recent message for the associated topic. Hence, the new subscribers can receive the last sent message from the publisher of that particular topic without waiting for the next PUBLISH message. The Broker MUST continue publishing the retained messages as long as the associated tokens are valid.

In case of disconnections due to network errors or server disconnection due to a protocol error (which includes authorization errors), the Will message must be sent if the Client supplied a Will in the CONNECT message. The Client's token scopes MUST include the Will Topic. The Will message MUST be published to the Will Topic when the network connection is closed regardless of whether the corresponding token has expired. In the case of a server-side DISCONNECT, the server returns the '0x87 Not Authorized' return code to the Client.

3. Reduced Protocol Interactions for MQTT v3.1.1

This section describes a reduced set of protocol interactions for the MQTT v3.1.1 Client.

3.1. Token Transport

To transport the token to the Broker, the Clients use the username and password fields of the CONNECT control message after the TLS handshake. Figure 5 shows the structure of the MQTT CONNECT message.

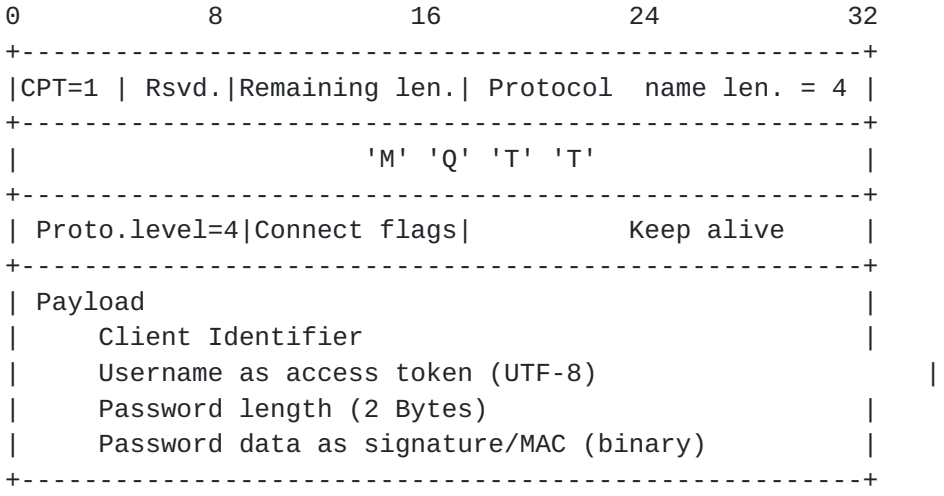

Figure 5: MQTT CONNECT control message. (CPT=Control Packet Type, Rsvd=Reserved, len.=length, Proto.=Protocol)

Figure 6 shows how the MQTT connect flags MUST be set to initiate a connection with the Broker.

User name flag	Pass. flag	Will retain	Will QoS	Will Flag	Clean	Rsvd.
1	1	X	X X	X	1	0

Figure 6: MQTT CONNECT flags. (Rsvd=Reserved)

The Clean Session Flag MUST be set to 1. The Client may set the Will Flag as desired (marked as 'X' in Figure 6). Username and Password flags MUST be set to 1 to ensure that the Payload of the CONNECT message includes both Username and Password fields.

The CONNECT message defaults to ACE for authentication and authorization. The Username field MUST be set to the access token.

The Password field MUST be set to the keyed message digest (MAC) or signature associated with the access token for proof-of-possession. The Client MUST apply the PoP key to the payload as described in Section 2.1.2.1.

In MQTT v3.1.1, the MQTT Username as a UTF-8 encoded string (i.e., is prefixed by a 2-byte length field followed by UTF-8 encoded character data) and may be up to 65535 bytes. Therefore, an access token that is not a valid UTF-8 MUST be Base64 [RFC4648] encoded. (The MQTT Password allows binary data up to 65535 bytes.)

3.2. Handling Authorization Errors

Handling errors are more primitive in MQTT v3.1.1 due to not having appropriate error fields, error codes, and server-side DISCONNECTS. In the following, we list how errors are handled without such protocol support.

- CONNECT without a token: It is not possible to support AS discovery via sending a tokenless CONNECT message to the Broker. This is because a CONNACK packet in MQTT v3.1.1 does not include a means to provide additional information to the Client. Therefore, AS discovery needs to take place out-of-band. CONNECT attempt MUSY fail.

- Client-RS PUBLISH authorization failure: In case of a failure, it is not possible to return an error in MQTT v3.1.1. Acknowledgement messages only indicate success. In the case of an authorization error, the Broker SHOULD disconnect the Client. Otherwise, it MUST ignore the PUBLISH message. Also, DISCONNECT messages are only sent from a Client to the Broker. So, server disconnection needs to take place below the application layer.

- SUBSCRIBE authorization failure: In the SUBACK packet, the return code must be 0x80 indicating 'Failure' for the unauthorized topic(s). Note that, in both MQTT versions, a reason code is returned for each Topic Filter.

- RS-Client PUBLISH authorization failure: When RS is forwarding PUBLISH messages to the subscribed Clients, it may discover that some of the subscribers are no more authorized due to expired tokens. These token expirations SHOULD lead to disconnecting the Client, rather than silently dropping messages.

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

Profile name: mqtt_tls

Profile description: Profile for delegating Client authentication and authorization using MQTT as the application protocol and TLS For transport layer security.

Profile ID:

Change controller: IESG

Reference: [RFC-XXXX]

5. Security Considerations

This document specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework [I-D.ietf-ace-oauth-authz]. Therefore, the security considerations outlined in [I-D.ietf-ace-oauth-authz] apply to this work.

In addition, the security considerations outlined in MQTT v5.0 - the OASIS Standard [MQTT-OASIS-Standard-v5] and MQTT v3.1.1 - the OASIS Standard [MQTT-OASIS-Standard] apply. Mainly, this document provides an authorization solution for MQTT, the responsibility of which is left to the specific implementation in MQTT v5.0 standard. In the following, we comment on a few relevant issues based on the current MQTT specifications.

In this document, RS uses the PoP access token to authenticate the Client. If the Client is able, TLS certificates sent from the Client can be used by the RS to authenticate the Client. The Client may authenticate the RS either using a server cerficate or the RPK method. In the case of RPK, client needs to use this raw public key in the TLS handshake together with an out-of-band validation technique (see [RFC7250] for details).

To authorize a Client's publish and subscribe requests in an ongoing session, the RS caches the access token after accepting the connection from the Client. However, if some permissions are revoked

in the meantime, the RS may still grant publish/subscribe to revoked topics. If the RS caches the token introspection responses, then the RS should use a reasonable cache timeout to introspect tokens regularly. When permissions change dynamically, it is expected that AS also follows a reasonable expiration strategy for the access tokens.

The RS may monitor Client behaviour to detect potential security problems, especially those affecting availability. These include repeated token transfer attempts to the public "authz-info" topic, repeated connection attempts, abnormal terminations, and Clients that connect but do not send any data. If the RS supports the public "authz-info" topic, described in Appendix B, then this may be vulnerable to a DDoS attack, where many Clients use the "authz-info" public topic to transport fictitious tokens, which RS may need to store indefinitely.

6. Privacy Considerations

The privacy considerations outlined in [I-D.ietf-ace-oauth-authz] apply to this work.

In MQTT, the RS is a central trusted party and may forward potentially sensitive information between Clients. Clients may choose to encrypt the payload of their messages. However, this would not provide privacy for other properties of the message such as Topic Name.

7. References

7.1. Normative References

[I-D.gerdes-ace-dtls-authorize]
Gerdes, S., Bergmann, O., Bormann, C., Selander, G., and L. Seitz, "Datagram Transport Layer Security (DTLS) Profile for Authentication and Authorization for Constrained Environments (ACE)", draft-gerdes-ace-dtls-authorize-01 (work in progress), March 2017.

[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-24 (work in progress), March 2019.

[MQTT-OASIS-Standard]
Banks, A., Ed. and R. Gupta, Ed., "OASIS Standard MQTT Version 3.1.1 Plus Errata 01", 2015, <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>.

[MQTT-OASIS-Standard-v5]
Banks, A., Ed., Briggs, E., Ed., Borgendale, K., Ed., and R. Gupta, Ed., "OASIS Standard MQTT Version 5.0", 2017, <http://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html>.

[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>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[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>.

[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>.

7.2. Informative References

[fremantle14]
Fremantle, P., Aziz, B., Kopecky, J., and P. Scott, "Federated Identity and Access Management for the Internet of Things", research International Workshop on Secure Internet of Things, September 2014, <http://dx.doi.org/10.1109/SIoT.2014.8>.

[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-08 (work in progress), October 2019.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[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>.

[RFC7800] Jones, M., Bradley, J., and H. Tschofenig, "Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs)", RFC 7800, DOI 10.17487/RFC7800, April 2016, <https://www.rfc-editor.org/info/rfc7800>.

Appendix A. Checklist for profile requirements

- AS discovery: AS discovery is possible with the MQTT v5.0 described in Section 2.1.2.
- The communication protocol between the Client and RS: MQTT
- The security protocol between the Client and RS: TLS
- Client and RS mutual authentication: RS provides a server certificate or RPK during TLS handshake. Client transports token and MAC via the MQTT CONNECT message.
- Content format: For the HTTPS interactions with AS, "application/ace+json". The MQTT payloads may be formatted in JSON.
- PoP protocols: Either symmetric or asymmetric keys can be supported.
- Unique profile identifier: mqtt_tls
- Token introspection: RS uses HTTPS /introspect interface of AS.
- Token request: CAS uses HTTPS /token interface of AS.
- /authz-info endpoint: It MAY be supported using the method described in Appendix B, but is not protected.
- Token transport: In MQTT CONNECT message for both versions of MQTT. AUTH extensions also used for authentication and re-authentication for MQTT v5.0 as described in Section 2.1.2.

Appendix B. The Authorization Information Endpoint

The main document described a method for transporting tokens inside MQTT CONNECT messages. In this section, we describe an alternative method to transport an access token.

The method consists of the MQTT Broker accepting PUBLISH messages to a public "authz-info" topic. A Client using this method MUST first connect to the Broker, and publish the access token using the "authz-info" topic. The Broker must verify the validity of the token (i.e., through local validation or introspection). After publishing the token, the Client disconnects from the Broker and is expected to try reconnecting over TLS.

In MQTT v5.0, the Broker can return 'Not authorized' error to a PUBLISH request for QoS greater or equal to 1. In MQTT v3.1.1, after the Client published to the "authz-info" topic, it is not possible for the Broker to communicate the result of the token verification. In any case, any token authorization failure affect the subsequent TLS handshake, which can prompt the Client to obtain a valid token.

Appendix C. Document Updates

Version 00 to 01:

- Present the MQTTv5 as the RECOMMENDED version, and MQTT v3.1.1 for backward compatibility.
- Clarified Will message.
- Improved consistency in the use of terminology, and upper/lower case.
- Defined Broker and MQTTS.
- Clarified HTTPS use for C-AS and RS-AS communication. Removed reference to actors document, and clarified the use of client authorization server.
- Clarified the Connect message payload and Client Identifier.
- Presented different methods for passing the token, and PoP.
- Added new figures for AUTH methods, updated CONNECT message figure.

Acknowledgements

The authors would like to thank Ludwig Seitz for his review and his input on the authorization information endpoint, presented in the appendix.

Authors' Addresses

Cigdem Sengul
Nominet
4 Kingdom Street
London W2 6BD
UK

Email: Cigdem.Sengul@nominet.uk

Anthony Kirby
Oxbotica
1a Milford House, Mayfield Road, Summertown
Oxford OX2 7EL
UK

Email: anthony@anthony.org

Paul Fremantle
University of Portsmouth
School of Computing, Buckingham House
Portsmouth PO1 3HE
UK

Email: paul.fremantle@port.ac.uk