ACE
Internet-Draft
Intended status: Informational
Expires: September 9, 2015

H. Tschofenig
ARM Limited

March 8, 2015

The OAuth 2.0 Internet of Things (IoT) Client Credentials Grant
draft-tschofenig-ace-oauth-iot-01.txt

Abstract

As Internet of Things (IoT) deployments increase steadily the need for a better user experience for handling the authentication and authorization tasks in constrained environments increases.

While several technologies have been developed already that allow federated access to protected resource the nature of IoT deployments requires care with the limited resources available on many of these devices.

This document defines a new OAuth 2.0 authorization grant for the interaction between constrained clients and resource servers to obtain access tokens for access to protected resources. It does so by leveraging prior work on OAuth 2.0, CoAP, and DTLS.

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 September 9, 2015.

Table of Contents

1. Introduction

Early Internet of Things deployments used Internet connectivity to push data to a cloud service or to an application on a smart phone. While these IoT deployments offer great benefits for their users they also suffer from a usability problem that can best be demonstrated with a door lock example.

Consider an enterprise environment where access to different parts of the campus is granted to employees dynamically and on a need-by-need basis. New employees receive access rights and those who decide to leave the company get their access rights revoked.

When an employee approaches a door the door lock is supposed to check the authorization rights of that employee in a fraction of a second and to grant (or deny) access appropriately. The building managers expect a centralized management of employees and their access rights and no prior interaction of employees with any object, such as a

door, upfront (such as it would be needed with pairing mechanisms utilized by some IoT technologies).

To accomplish such a seamless user experience and offering security at the same time it is necessary to make use of an authentication and authorization server that manages policies for access to protected resources (such as door locks in the previous example). As outlined in [I-D.seitz-ace-usecases], it is assumed that resource servers do not necessarily need to contact the authorization server every time they receive an access request.

OAuth 2.0 [RFC6749] is a technology that offers such a design pattern via the use of access tokens, which are requested by clients, and subsequently presented to resource servers when demanding access to protected resources managed by those resource servers.

OAuth 2.0 was, however, design primarily for use with the HTTP-based web infrastructure and has only recently been extended for use with SASL [I-D.ietf-kitten-sasl-oauth]. This document extends the OAuth 2.0 idea one step further and defines a Constrained Application Protocol (CoAP)- based transport profile for OAuth 2.0. CoAP is specified in RFC 7252 [RFC7252].

The benefits are as follows:

- The use of the Constrained Application Protocol (CoAP) [RFC6749] (instead of HTTP) for encoding of requests and responses leads to smaller and fewer message exchanges since CoAP uses a binary format and relies on UDP rather than TCP.

- The of Datagram Transport Layer Security (DTLS) [RFC6347] (instead of Transport Layer Security (TLS) for authentication) of the authorization server to the client and for establishment of an integrity protected and confidential communication channel follows the spirit of TLS but is tailored to the unreliable transport protocol used by CoAP.

- The use of DTLS allows to re-use well-analyzed security functionality and does not require re-designing the same features at the application layer. Note that the use of DTLS also allows different credential-types to be re-used, as explained in [I-D.ietf-dice-profile].

- The use of DTLS client-side authentication instead of the previously defined application layer client authentication mechanisms (as, for example defined in Section 2.3 of RFC 6749) offers security properties that have not been exploited in the TLS usage in the Web.

- Allows to re-use the standardized access token format, namely JSON Web Token (JWT) [I-D.ietf-oauth-json-web-token], as well as proof-of-possession tokens [I-D.hunt-oauth-pop-architecture].

Intentionally left outside the scope of this document are the following items:

- Registration of the client with the authorization server. This includes the provisioning of security credentials at the client for subsequent use in case of client authentication.

- The interaction between the client and the resource server when presenting the access token since a variety of different deployment models may be envisioned. In some deployment scenarios the use of bearer tokens may be appropriate whereas in others proof-of-possession techniques may provide the desired level of security. Also the desire to use application layer security (in comparison to re-using DTLS) leads to different security designs.

- The interaction between the resource server and the authorization server for provisioning of access control policies as well as the ability to ask the authorization in real-time for access control decisions (using the token introspection endpoint [I-D.richer-oauth-introspection]). This is an optional part of the OAuth 2.0 architecture that may be used in deployments where a tied coupling between the authorization server and the resource servers exists and the real-time interaction is desired.

- Cross realm use where authorization servers from different administrative domains are involved. The use of this protocol needs to be beneficial in a single domain concept to subsequently see demand in a cross realm situation. Experience from other federated authentication and authorization protocols shows that adding the cross realm support is technically but complex from a business point of view.

- The authorization server may place authorization information inside the access token so that the resource server can enforce these access control decisions autonomously. This information may come in various flavors, such as basic JWT claims (such 'audience', 'expiration time', 'not before') or more sophisticated access control information like [I-D.bormann-core-ace-aif]. These policy languages are largely independent of the OAuth 2.0 framework.

2. Terminology

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' in this specification are to be interpreted as described in [RFC2119].

All terms are as defined in The OAuth 2.0 Authorization Framework [RFC6749].

3. Applicability Statement

The OAuth 2.0 IoT grant defined in this document is only applicable for OAuth 2.0 clients that are resource constrained. For all other clients regular OAuth 2.0 can be re-used since those clients will be able to execute the RFC 6749-defined client credential grant, which uses HTTPS as a transport.

The communication between the client and a resource constrained resource server is not described in this document and orthogonal to this document.

4. IoT Client Credentials Grant

This IoT credential grant is a variation of the client credential grant defined in RFC 6749.

The client can request an access token using only its client credentials when the client is requesting access to the protected resources under its control, or those of another resource owner that have been previously arranged with the authorization server (the method of which is beyond the scope of this specification).

The IoT client credentials grant type MUST only be used by confidential clients.

Figure 1: IoT Client Credentials Flow.

The exchange illustrated in Figure 1 includes the following steps:

(A) The client authenticates with the authorization server and requests an access token from the token endpoint. Mutual authentication between the client and then authorization server authentication takes place as part of the DTLS exchange.

(B) The authorization server authenticates the client, and if the request is valid and authorized, issues an access token.

4.1. Authorization Request and Response

Since the client authentication is used as the authorization grant, no additional authorization request is needed.

4.2. Access Token Request

The client makes a request to the token endpoint by adding the following parameters using the "application/x-www-form-urlencoded" format with a character encoding of UTF-8 in the CoAP request entity-body:

grant_type: OPTIONAL. Value MUST be set to "urn:ietf:params:oauth:grant-type:iot" for this grant type. The value is optional since the client may have been pre-provisioned by the authorization server with information about the grant type.

scope: OPTIONAL. The scope of the access request as described by Section 3.3 of RFC 6749.

audience: OPTIONAL. Value is used by the client to indicate what resource server, as the intended recipient, it wants to access. This field is defined in [I-D.tschofenig-oauth-audience]

(QUESTION: Would it be useful to also use a JSON encoding here?)

In order to prevent man-in-the-middle attacks, the client MUST require the use of DTLS with server authentication for any request sent to the authorization and token endpoints. If certificate-based server authentication is used then the client MUST validate the TLS certificate of the authorization server, as defined by [RFC6125].

4.3. Access Token Response

If the access token request is valid and authorized, the authorization server issues an access token as described in Section 5.1 of RFC 6749 but encoded in a CoAP message using the Content response code with the response encoded as a JSON structure in the payload of the message. A refresh token MUST NOT be included. If the request failed client authentication or is invalid, the

authorization server returns an error response using the CoAP 4.00 'Bad Request' response code with the error messages defined in Section 5.2 of RFC 6749.

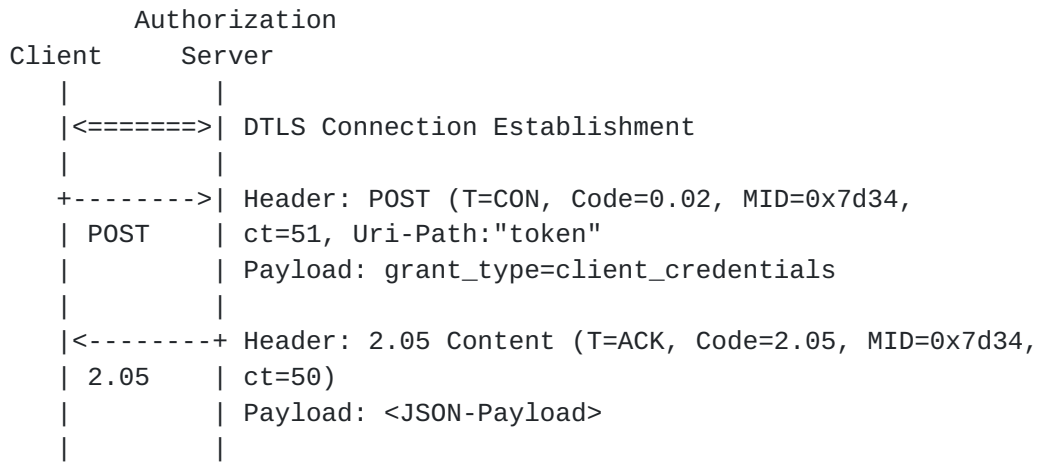
Note that the HTTP "Cache-Control" parameters are not used in the CoAP response message.

QUESTION: Would it be useful to use the CBOR encoding for the response? This could reduce the response size by a few %.

5. Example

For example, the client makes a CoAP carrying the Access Token Request protected with DTLS to the authorization server. It then receives a successful Access Token Response containing the access token.

In the example below content-type 51 corresponds to the 'application/x-www-form-urlencoded'.




```
<JSON-Payload>:=
{
    "access_token":"2YotnFZFE...jr1zCsicMWpAA",
    "token_type":"bearer",
    "expires_in":28800
}
```

Figure 2: Example CoAP POST Message Exchange.

6. Security Considerations

This document re-uses a sub-set of the OAuth 2.0 functionality specified in RFC 6749 and intentionally inherits the security properties of OAuth 2.0, and DTLS. The discussion in Section 10 of RFC 6749 and Section 4 of RFC 6819 are relevant for this document.

7. IANA Considerations

7.1. Sub-Namespace Registration of urn:ietf:params:oauth:grant-type:iot

This specification registers the value "grant-type:iot" in the IANA urn:ietf:params:oauth registry established in An IETF URN Sub-Namespace for OAuth [RFC6755].

- URN: urn:ietf:params:oauth:grant-type:iot
- Common Name: Internet of Things (IoT) Client Credentails Grant Type Profile for OAuth 2.0
- Change controller: IETF
- Specification Document: [[this document]]

8. References

8.1. Normative References

[I-D.ietf-dice-profile]
Tschofenig, H. and T. Fossati, "A TLS/DTLS 1.2 Profile for the Internet of Things", draft-ietf-dice-profile-09 (work in progress), January 2015.

[I-D.tschofenig-oauth-audience]
Tschofenig, H., "OAuth 2.0: Audience Information", draft-tschofenig-oauth-audience-00 (work in progress), February 2013.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.

[RFC6755] Campbell, B. and H. Tschofenig, "An IETF URN Sub-Namespace for OAuth", RFC 6755, October 2012.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

8.2. Informative References

[I-D.bormann-core-ace-aif]
Bormann, C., "An Authorization Information Format (AIF) for ACE", draft-bormann-core-ace-aif-01 (work in progress), July 2014.

[I-D.hunt-oauth-pop-architecture]
Hunt, P., Richer, J., Mills, W., Mishra, P., and H. Tschofenig, "OAuth 2.0 Proof-of-Possession (PoP) Security Architecture", draft-hunt-oauth-pop-architecture-02 (work in progress), June 2014.

[I-D.ietf-kitten-sasl-oauth]
Mills, W., Showalter, T., and H. Tschofenig, "A set of SASL Mechanisms for OAuth", draft-ietf-kitten-sasl-oauth-19 (work in progress), January 2015.

[I-D.ietf-oauth-json-web-token]
Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", draft-ietf-oauth-json-web-token-32 (work in progress), December 2014.

[I-D.richer-oauth-introspection]
Richer, J., "OAuth Token Introspection", draft-richer-oauth-introspection-06 (work in progress), July 2014.

[I-D.seitz-ace-usecases]
Seitz, L., Gerdes, S., Selander, G., Mani, M., and S. Kumar, "ACE use cases", draft-seitz-ace-usecases-02 (work in progress), October 2014.

Author's Address

Hannes Tschofenig
ARM Limited
Austria

Email: Hannes.Tschofenig@gmx.net
URI: http://www.tschofenig.priv.at