Notification of Revoked Access Tokens in the ACE Framework

draft-ietf-ace-revoked-tokens-notification-01

Marco Tiloca, RISE
Ludwig Seitz, Combitech
Francesca Palombini, Ericsson
Sebastian Echeverria, CMU SEI
Grace Lewis, CMU SEI

IETF 113, ACE WG, March 22nd, 2022
Recap

› An Access Token may be revoked, before expiration
  – Client/RS has been compromised, or decommissioned
  – Changed access policies or outcome of their evaluation
  – Changed ACE profile to use

› Token introspection at the AS is available only for the RS
  – Validate one Access Token at the time

› Contribution: new interface at the Authorization Server (AS)
  – The AS maintains one Token Revocation List (TRL) resource
  – The TRL contains the hashes of revoked, not-yet-expired tokens
  – C/RS can GET or GET-Observable from the TRL
  – C/RS retrieve only their own pertaining portion of the TRL

› Benefits
  – Complement token introspection
  – No need for new endpoints at C or RS
Modes of operation

› **Common features**
  – Response limited to the portion of the TRL pertaining the requester
  – TRL filtering based on authenticated identity of the requester (secure association)

› **Full Query - GET [Observe: 0] coaps://example.as.com/revoke/trl**
  – Get all the pertaining token hashes in the TRL
  – The AS MUST support it

› **Diff Query - GET [Observe: 0] coaps://example.as.com/revoke/trl?diff=3**
  – Get the N most recent, pertaining updates to the TRL
  – The AS MAY support it

› **Diff Query using the “Cursor” pattern** – Appendix B
  – Enables trasferring of TRL updates in chunks, from a “resumption point”
  – Affects also the content of responses to Full Query and simple Diff Query requests
  – The AS MAY support it
Updates since IETF 112

› Adopted as WG document in November 2021

› Received a review [1] on version -00 from Marco Rasori – Thanks!

› New version -01 available by the cut-off
  – Addressed review comments (one point is still open)
  – More on error handling and on token processing at the RS
  – Section restructuring and editorial improvements

[1] https://mailarchive.ietf.org/arch/msg/ace/XufwPd8bv1aMTzw1Hp5bENaqn_o/
Selected updates in -01 (1/2)

- Defined explicit actions on tokens for the Client/RS
  - Expunge a stored token when learning of its revocation or expiration
  - Do not accept a posted revoked token (i.e., if storing its token hash)

- More detailed actions on token hashes for the RS
  - Store the obtained token hash of a revoked token ...
  - ... until learning that the token has expired
  - This makes the RS able to:
    - Reject a token (re-)posted between its revocation and expiration
    - Reject a token belatedly posted for the first time after its revocation
Selected updates in -01 (2/2)

› Improved error handling at the TRL resource on the AS

› Fixed off-by-one errors when using the “Cursor” pattern
  – Two corner cases when preparing the response to a C/RS

› Error responses sent in a number of cases
  – The content-format is application/ace-trl+cbor
  – The payload is a CBOR map

› Defined parameters for error responses
  – ’error’ (int) and ’error_description’ (tstr, optional)

› Registration of ’error’ values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Invalid parameter value</td>
</tr>
<tr>
<td>1</td>
<td>Invalid set of parameters</td>
</tr>
<tr>
<td>2</td>
<td>Out of bound cursor value</td>
</tr>
</tbody>
</table>
Open points (1/2)

› The use of the “Cursor” pattern is still in Appendix B
  – Its mechanics and error handling are stable now
  – It is in fact the Diff Query mode, enhanced with the “Cursor” pattern

› Planned actions, if no objection
  – Bring the content from Appendix B to the document body
    › Mostly affect Section 7 (Diff Query mode) but also Section 6 (Full Query mode)
  – Add examples with the Diff Query mode using the “Cursor” pattern
Open points (2/2)

Current definition of responses

› Error responses
  – CBOR map as payload

› 2.05 responses (Full Query)
  – CBOR array as payload (Section 6)

› 2.05 responses (Diff Query)
  – CBOR array as payload (Section 7)

› 2.05 responses (Diff Query + Cursor)
  – CBOR map as payload (Appendix B)
Open points (2/2)

Current definition of responses

› Error responses
  – CBOR map as payload

› 2.05 responses (Full Query)
  – CBOR array as payload (Section 6)

› 2.05 responses (Diff Query)
  – CBOR array as payload (Section 7)

› 2.05 responses (Diff Query + Cursor)
  – CBOR map as payload (Appendix B)

If (Diff Query + Cursor) is used by the AS, CBOR maps are also used in Full Query and Diff Query responses, to specify additional information, e.g., the cursor value.

Proposal from the review of -00: just have a CBOR map in all responses.

There is a more efficient compromise …
Open points (2/2)

Current definition of responses | New definition of responses

› Error responses
  – CBOR map as payload

› 2.05 responses (Full Query)
  – CBOR array as payload (Section 6)

› 2.05 responses (Diff Query)
  – CBOR array as payload (Section 7)

› 2.05 responses (Diff Query + Cursor)
  – CBOR map as payload (Appendix B)

› Error responses
  – CBOR map as payload

› 2.05 responses (any mode)
  – The payload is a CBOR map if the AS supports the Diff Query mode and the “Cursor” pattern
  – The payload is a CBOR array otherwise
  – Clients/RSs are ok to receive either

Ok with this change? Alternatives?
Summary and next steps

- **Notification of revoked Access Token**
  - GET or GET-Observe at the AS; for both Client and RS
  - (i) Full Query; (ii) Diff Query; (iii) Diff Query with “Cursor” pattern

- **Ongoing implementation from CNR**
  - Building on the RISE implementation of ACE for the Californium library

- **Main planned next steps**
  - Integrate the “Cursor” pattern in the document body, as extension of the Diff Query mode
  - Define the response format as depending on the AS supporting the “Cursor” pattern

- **More comments are welcome!**
Thank you!

Comments/questions?

https://github.com/ace-wg/ace-revoked-token-notification
Backup
How it works

› Token hashes computed as per RFC 6920 (binary format)
  – Hash input: what is in ‘access_token’ of the AS response from /token

› TRL resource at the AS
  – CBOR array of Token hashes
  – Add token hashes when Tokens are revoked
  – Remove token hashes when revoked Tokens expire

› Interaction
  – C and RS get the URL to the TRL endpoint upon registration
  – C and RS obtain only hashes of their own pertaining Tokens
  – A registered Administrator gets all Token hashes in the TRL
Protocol overview

```
+-----------------------------+    +-------------------+    +-------------------+    +-------------------+
| Authorization Server        |    | revoke/trl         |    | TRL: {th1, th2, th3} |
+-----------------------------+    +-------------------+    +-------------------+
| th1, th2, th3               |    | th1, th2           |    | th1                |
| v                            |    | v                  |    | v                  |
| Administrator               |    | Client 1           |    | Resource Server 1  |
| th1, th2, th3               |    | th3                |
| v                            |    | v                  |
| th1, th2, th3               |    | th2, th3           |
| v                            |    | v                  |
| Client 1                     |    | Resource Server 2  |
| th1, th2                     |    | th3                |
| Resource Server 2            |
| :                             |    | :                  |
| :                             |    | : t1               |
| :                             |    | : .........        |
| :                             |    | t2                 |
| ...                           |
```

Example with Full Query

RS

Registration: POST

AS

<------------------------>

2.01 CREATED
Payload: {
  ...
  "trl_path" = "revoke/trl",
  "trl_hash" = "sha-256",
  "n_max" = 10
}

GET Observe: 0
coap://example.as.com/revoke/trl/

<------------------------>

2.05 CONTENT Observe: 42
Payload: []

...(Access Tokens t1 and t2 issued and successfully submitted to RS)
Example with Full Query (ctd.)

RS

(Access Token t1 is revoked)

<-----------------------------
2.05 CONTENT Observe: 53
Payload: [bstr.h(t1)]

(Access Token t2 is revoked)

<-----------------------------
2.05 CONTENT Observe: 64
Payload: [bstr.h(t1),
bstr.h(t2)]

(Access Token t1 expires)

<-----------------------------
2.05 CONTENT Observe: 75
Payload: [bstr.h(t2)]

(Access Token t2 expires)

<-----------------------------
2.05 CONTENT Observe: 86
Payload: []
Query modes

› Common features
  – Limited to the portion of the TRL pertaining the requester
  – TRL filtering based on authenticated identity of the requester (e.g., secure communication session)

› Full Query (Section 6) – GET [Observe: 0] coaps://example.as.com/revoke/trl
  – Request for all pertaining token hashes in the TRL
  – Return a CBOR array, with the Token hashes as elements

› Diff Query (Section 7) – GET [Observe: 0] coaps://example.as.com/revoke/trl?diff=3
  – Request for the latest N updates to the pertaining portion of the TRL list
  – Build N entries as CBOR arrays. Each entry refers to an update and has:
    › An element “deleted”, with a CBOR array of Token hashes.
    › An element “added”, with a CBOR array of Token hashes.
  – Return a CBOR array with the N arrays as element, in reverse chronological order
  – Example of usage of the Series Transfer Pattern (STP)

› Diff Query using the “Cursor” pattern (Appendix B)
  – Still Diff Query mode, but also enabling transfers in chunks from a “resumption point”
  – This results in extended responses also when the Full Query mode is used
Diff Query using the "Cursor" pattern

› Rather than the N most recent TRL updates ...
  – Get N updates from “where we previous query stopped”
  – Revert to Full Query if not possible, e.g., information loss/removal at the AS

› Use the Series Transfer Pattern (STP) and its "Cursor" pattern
  – Both (a) Full Query and (b) Diff Query responses specify also a cursor value
  – In (a), it is a pointer to the most recent, pertaining TRL update
  – In (b) it is a pointer to the most recent TRL update included in the response

› What becomes possible due to the “Cursor” pattern
  – A follow-up request may resume fetching TRL updates from after the cursor
  – Adjacent batches of TRL updates are possible to be sent, thus limiting excessive latencies

› Handled corner cases
  – No TRL updates have occurred yet, either at all or after the cursor
  – Requested updates have been deleted as too old