Observe Notifications as CoAP Multicast Responses

draft-tiloca-core-observe-multicast-notifications-05

Marco Tiloca, RISE
Rikard Höglund, RISE
Christian Amsüss
Francesca Palombini, Ericsson

IETF 110, CoRE WG, March 08th, 2021
Recap

› Observe notifications as **multicast responses**
  – Many clients observe the same resource on a server S
  – Improved performance due to multicast delivery
  – Multicast responses are not defined yet --- Token binding, security, …

› Example of relevant use case
  – Pub-Sub scenario, also discussed at IETF 104
  – Many subscribers to a same topic on the Broker
  – Better performance
  – Subscribers can remain clients only
How

› Define multicast responses, in particular Observe notifications

› Token space managed by the server
  – The Token space belongs to the group (clients)
  – The group entrusts the management to the server
  – All clients in a group observation use the same Token value

› Group OSCORE to protect multicast notifications
  – The server aligns all clients of an observation on a same external_aad
  – All notifications for a resource are protected with that external_aad
Phantom request and error response

› The server requests the observation on its own, e.g. when:
  1. A first traditional registration request comes from a first client
  2. Some threshold is crossed – clients can be shifted to a group observation

› Consensus on Token & external_aad, by using a phantom observation request
  – Generated inside the server, it does not hit the wire
  – Like if sent by the group, from the multicast IP address of the group
  – Multicast notifications are responses to this phantom request

› The server sends to clients a 5.03 error response with:
  – Transport-specific information, e.g. the IP multicast address where notifications are sent to
  – The serialization of the phantom observation request
  – The serialization of the latest multicast notification (optional)
Updates from -05

› New payload format for the informative response

```plaintext
informative_response_payload = {
  1 => array, ; 'tp_info', i.e. transport-specific information
  2 => bstr, ; 'ph_req' (transport-independent information)
  ? 3 => bstr ; 'last_notif' (transport-independent information)
}
```

› The same ‘tp_info’ content applies to both ‘ph_req’ and ‘last_notif’

› ‘ph_req’ - Serialization of the phantom request

› ‘last_notif’ - Serialization of latest sent multicast notification
  – Now only optional to include
Updates from -05

› New payload format for the informative response

```
informative_response_payload = {
  1 => array, ; 'tp_info', i.e. transport-specific information
  2 => bstr, ; 'ph_req' (transport-independent information)
  ? 3 => bstr ; 'lastnotif' (transport-independent information)
}
```

tp_info = {
  srv_addr ; Addressing information of the server
  ? req_info ; Request data extension
}

srv_addr = {
  tp_id : int, ; Identifier of the used transport protocol
  + elements ; Number, format and encoding
  ; based on the value of 'tp_id'
}

req_info = {
  + elements ; Number, format and encoding based on
  ; the value of 'tp_id' in 'srv_addr'
}

tp_info = {
  tp_id : 1, ; UDP as transport protocol
  srv_host : #6.260(bstr), ; Src. address of multicast notifications
  srv_port : uint, ; Src. port of multicast notifications
  token : bstr, ; Token of the phantom request and
  associated multicast notifications
  cli_addr : #6.260(bstr), ; Dst. address of multicast notifications
  ? cli_port : uint ; Dst. port of multicast notifications
}

Concrete encoding for this document, where ‘tp_id’ = 1 (UDP)

› Defined new IANA registry, for ‘tp_id’ values, and formats of ‘srv_addr’ and ‘req_info’

› Format reused for the Response-Forwarding option in draft-tiloca-core-groupcomm-proxy
Updates from -05

› There is no client-server negotiation of multicast notification service
  – The proposed mechanisms is used in situations where:
    › Individual notifications are not feasible; or
    › Individual notifications are not preferred beyond a certain number of observers
  – Future applications can define negotiation mechanisms if need be

› Signaling of multicast notification service
  – A web link can include the target attribute “grp_obs”, as a simple hint

› Revised processing in the presence of forward proxies
  – Improved mechanics without and with Group OSCORE (Section 9 and Section 10)
  – Updated examples in Appendix E and Appendix F
Updates from -05

› Appendix C – OSCORE group self-managed by the server
  – The client’s observation request works as a joining request
  – The informative response includes also group keying material
  – This mirrors the case where the client joins an OSCORE group only as silent server
  – Not suitable when backward security and forward security are required

› Appendix D – Phantom request as deterministic request
  – Each client builds the same phantom request, see draft-amsuess-core-cachable-oscore
  – No need to include the phantom request in the ’ph_req’ of informative responses
Summary

› Latest additions
  – New flexible and extensible encoding of the informative response
  – No negotiation with clients; just signaling of support for group observations
  – Revised processing with proxies; updated examples in Appendix E and F
  – New Appendix C: OSCORE group self-managed by the server
  – New Appendix D: phantom request as deterministic request

› Next steps
  – Case with reverse proxy – Mechanics and example
  – Case with deterministic request and proxy – Mechanics and example

› Ready for WG adoption?
Thank you!

Comments/questions?

https://gitlab.com/crimson84/draft-tiloca-core-observe-responses-multicast
Backup
Server side

1. Build a GET phantom request; Observe option set to 0

2. Choose a value T, from the Token space for messages …
   – … coming from the multicast IP address and addressed to target resource

3. Process the phantom request
   – As coming from the group and its IP multicast address
   – As addressed to the target resource

4. Hereafter, use T as token value for the group observation

5. Store the phantom request, store (not send) reply for last_notif
Interaction with clients

- The server sends to new/shifted clients an **error response** with
  - ‘tp_info’: transport-specific information
    - ‘srv_addr’ and ‘srv_port’: destination address/port of the phantom request
    - ‘token’: the selected Token value T, used for ‘ph_req’ and ‘last_notif’
    - ‘cli_addr’ and ‘cli_port’: source address/port of the phantom request
  - ‘ph_req’: serialization of the phantom request
  - ‘last_notif’: serialization of the latest sent notification for the target resource

- When the value of the target resource changes:
  - The server sends an Observe notification to the IP multicast address ‘cli_addr’
  - The notification has the Token value T of the phantom request

- When getting the error response, a client:
  - Configures an observation for an endpoint associated to the multicast IP address
  - Accepts observe notifications with Token value T, sent to that multicast IP address
C1 registration

GET
Token: 0x4a
Observe: 0 (Register)
<Other options>

(S allocates the available Token value 0x7b.)

(S sends to itself a phantom observation request PH_REQ as coming from the IP multicast address GRP_ADDR.)

/S

GET
Token: 0x7b
Observe: 0 (Register)
<Other options>

(S creates a group observation of /r.)

(S increments the observer counter for the group observation of /r.)
C1 registration

C_1 <------------------------ [ Unicast ] ------------------------ S

5.03
Token: 0x4a
Content-Format: application/informative-response+cbor

<Other options>
Payload: {
  tp_info : [1, bstr(SRV ADDR), SRV PORT, 0x7b, bstr(CRP ADDR), CRP PORT],
  ph_req : bstr(0x01 | OPT),
  last_notif : bstr(0x45 | OPT | 0xff | PAYLOAD)
}
C2 registration

```
GET
Token: 0x01
Observe: 0 (Register)
<Other options>

(S increments the observer counter
for the group observation of /r .)
```

```
5.03
Token: 0x01
Content-Format: application/informative-response+cbor
<Other options>
Payload: {
    tp_info : [1, bstr(SRV_ADDR), SRV_PORT, 0x7b],
    ph_req : bstr(0x01 | OPT),
    lastnotif : bstr(0x45 | OPT | 0xff | PAYLOAD)
}

(The value of the resource /r changes to "5678".)
```
Multicast notification

- Same Token value of the Phantom Request

- Enforce binding between
  - Every multicast notification for the target resource
  - The (group) observation that each client takes part in
Security with Group OSCORE

› The phantom request is protected with Group OSCORE
  – $x$: the Sender ID (‘kid’) of the Server in the OSCORE group
  – $y$: the current SN value (‘piv’) used by the Server in the OSCORE group
  – Note: the Server consumes the value $y$ and does not reuse it as SN in the group

› To secure/verify all multicast notifications, the OSCORE $external_aad$ is built with:
  – ‘req_kid’ = $x$
  – ‘req_piv’ = $y$

› The phantom request is still included in the informative response
  – Each client retrieves $x$ and $y$ from the OSCORE option
Security with Group OSCORE

In the error response, the server can *optionally* specify also:

- `join-uri`: link to the Group Manager to join the OSCORE group
- `sec-gp`: name of the OSCORE group
- `as-uri`: link to the ACE Authorization Server associated to the Group Manager
- `cs-alg`: countersignature algorithm
- `cs-alg-crv`: countersignature curve of the algorithm
- `cs-key-kty`: countersignature key type
- `cs-key-crv`: countersignature curve of the key
- `cs-kenc`: countersignature key encoding
- `alg`: AEAD algorithm
- `hkdf`: HKDF algorithm
C1 registration w/ security

C.1  ---------------- [ Unicast w/ OSCORE ] ------------------------> $ /r
0.05 (FETCH)
Token: 0x4a
OSCORE: {kid: 1 ; piv: 101 ; ...}
<Other class U/I options>
0xff
Encrypted_payload {
  0x01 (GET),
  Observe: 0 (Register),
  <Other class E options>
}

($ allocates the available Token value 0x7b .)

(S sends to itself a phantom observation request PH_REQ
as coming from the IP multicast address GRP_ADDR .)

---------------------------------------------------------------
\----------------------------------------------> /r
0.05 (FETCH)
Token: 0x7b
OSCORE: {kid: 5 ; piv: 501 ;
  kid context: 57ab2e; ...}
<Other class U/I options>
0xff
Encrypted_payload {
  0x01 (GET),
  Observe: 0 (Register),
  <Other class E options>
}
<Counter signature>

(S steps SN_5 in the Group OSCORE Sec. Ctx ; SN_5 <= 502)
(S creates a group observation of $ /r .)

($ increments the observer counter for the group observation of $ /r .)
C1 registration w/ security

```
C_1 <----------------- [ Unicast w/ OSCORE ] --------------------- S
  2.05 (Content)
  Token: 0x4a
  OSCORE: { piv: 301; ... }
  <Other class U/I options>
  0xff
  Encrypted_payload {
    5.03 (Service Unavailable),
    Content-Format: application/informative-response+cbor,
    <Other class E options>,
    0xff,
    CBOR_payload {
      tp_info : [1, bstr(SRV_ADDR), SRV_PORT,
                 0x7f, bstr(GRP_ADDR), GRP_PORT],
      ph_req : bstr(0x05 OPT 0xff PAYLOAD SIGN),
      last_notif : bstr(0x45 OPT 0xff PAYLOAD SIGN),
      join_uri : "coap://myGM/ace-group/myGroup",
      sec_gp : "myGroup"
    }
  }
```

5: Sender ID ('kid') of S in the OSCORE group
501: Sequence Number of S in the OSCORE group when S created the group observation
C2 registration w/ security

5: Sender ID (‘kid’) of S in the OSCORE group
501: Sequence Number of S in the OSCORE group when S created the group observation
When encrypting and signing the multicast notification:
- The OSCORE `external_aad` has `req_kid = 5` and `req_iv = 501`
- Same for all following notifications for the same resource

Enforce secure binding between
- Every multicast notification for the target resource
- The (group) observation that each client takes part in
Support for intermediary proxies

› How it works
  – The proxy (next to the server) directly listens to the IP multicast address
  – The original Token of the phantom request has to match at the proxy
  – The proxy forwards multicast notifications back to each client
    › The proxy uses the Token values offered by the clients

› Without end-to-end security (Section 9)
  – The proxy can retrieve the phantom request from the informative response
  – No need to forward the informative response back to the clients

› With end-to-end security (Section 10)
  – The informative response is also protected with OSCORE or Group OSCORE
  – The proxy cannot retrieve the phantom request from the informative response
  – Each client cannot retrieve the phantom request from the proxy