Status

• Update -05 according to review

• WGLC review comments from Jim - PR #9 included in -05 fixes most of them:
  • Take out EDHOC appendix
  • Change term "MitM" with "on-path attacker"
  • Add section on discarding the sec ctx
  • Change uniqueness requirement on IDs
  • Define structure to transport OSCORE sec ctx input parameters
  • Remove uri path from the document
  • Motivate use of nonce in Protocol overview

• One open point discussed here
Add section on discarding the sec ctx

• The client MUST discard the current security context associated with an RS when:
  • the Sequence Number space ends.
  • the access token associated with the context expires.
  • the client receives a number of 4.01 Unauthorized responses to OSCORE requests. The exact number needs to be specified by the application.
  • creating a new security context from an old non-expired token

• The RS MUST discard the current security context associated with a client when:
  • Sequence Number space ends.
  • Access token associated with the context expires.
Define structure to transport OSCORE sec ctx input parameters

- Example of OSCORE_Security_Context using JSON:

```json
"OSCORE_Security_Context" : {
  "alg" : "AES-CCM-16-64-128",
  "clientId" : b64'qA',
  "serverId" : b64'Qg',
  "ms" : b64'+a+Dg2jjU+eIiOFCa9lObw'
}
```

- CDDL definition for OSCORE_Security_Context (CBOR):

```cddl
OSCORE_Security_Context = {
  ? 1 => bstr, ; ms
  ? 2 => bstr, ; clientId
  ? 3 => bstr, ; serverId
  ? 4 => tstr / int, ; hkdf
  ? 5 => tstr / int, ; alg
  ? 6 => bstr, ; salt
  ? 7 => bstr / tstr ; rpl }
```

- IANA considerations: registry creation (Expert Review Required), parameters registration, CWT and JWT registration, expert review guidelines
The one issue left

• Assumptions:
  
  • Client and RS can forget security contexts and do not keep track of all the tokens received.

  • Client can get an old non-expired token from AS.
Protocol Overview from v-02 (June 2018)

- Master Secret
- Master Salt
- Client ID
- Sender ID

- Sender Key
- Receiver Key
- Base IV
- Partial IV = Sequence Number (starts at 0)
Proposal

Adding random Nonces N1 and N2 in Sec Ctx derivation (Created by RS and C resp)

- Master Secret
- Master Salt
- Client ID
- Sender ID
- Sender Key
- Receiver Key
- Base IV
- Partial IV = Sequence Number (starts at 0)

This will avoid reuse of nonces and keys on RS and C for a security context derived from the same input parameter
Motivation: N1 (RS nonce)

• Issue:
  • RS looses security context and token
  • C reposts the same token, triggering security context derivation
  • Attacker replays an old OSCORE Request from C to RS

• This leads to reuse of nonces on the server side

• RS sends a random nonce N1 to avoid this.
This will cause reuse of AEAD nonces and keys on the RS for a different message for a security context derived from the same input parameter.
Solution

Adding a random Nonce N1 in Sec Ctx derivation 
(Created by RS)

This will avoid reuse of nonces and keys on RS for a security context derived from the same input parameter
Motivation: N2 (C nonce)

• Issue:
  • C looses security context and token
  • C gets a token, and posts it to RS
  • An on-path attacker replays an old message from RS to C, containing an old nonce N1 for security context derivation

• This leads to reuse of nonces on the client side
Nonce N1 is not protected so an on-path attacker can replace it, provoking an old security context to be created on the Client, and nonces reuse.

Uses Security Context created with Nonce N1

Looses Security Context

- Master Secret
- Master Salt
- Client ID
- Sender ID

Issue

POST /token
Access token + RS Information
POST /authz-info

Sec Ctx Derivation (N1)

OSCORE Request
AEAD Nonce = A

2.01 Created (Nonce N1)

Sec Ctx Derivation (N2)

OSCORE Request
AEAD Nonce = A

4.01 Unauthorized

Attacker

C

RS

AS
Conclusion

• Because of these security issues, we consider that using nonces can not be optional.

• Question to the WG: how do we transport N1 and N2 and include them in OSCORE Security Context derivation?
  • N1 || N2 as ID Context; transported as kid context (currently in the draft)
  • N1 as salt, N2 as ID Context; N1 transported as payload of 2.01 Created, N2 as kid context
  • N1 || N2 as ID Context; N2 transported at the same time of the token in the POST /authz-info (new content-format), N1 transported as payload of 2.01 Created
Proposal 1: N1 || N2 as kid context

- Master Secret
- Master Salt
- Client ID
- Sender ID

- ID Context = N1 || N2

- Sender Key
- Receiver Key
- Base IV
- Partial IV = Sequence Number (starts at 0)

• ID Context = N1 || N2 is used in Security Context derivation

• kid context to transport ID Context in the first OSCORE request

• kid context can be omitted in further OSCORE requests

• Con: RS derives a sec context when receiving an unknown kid context; we send N1 when only N2 is needed.

• Pro: we don’t use salt, leaving it to the application
Proposal 2: N1 as salt, N2 as ID Context

- Salt = N1 is used in Security Context derivation
- ID Context = N2 is used in Security Context derivation
- kid context to transport ID Context in the first OSCORE request, salt is transported as payload of 2.01 Created
- kid context can be omitted in further OSCORE requests
- Pro: we send N2 only
- Con: we use salt
Proposal 3: N1 || N2 as ID Context

- ID Context = N1 || N2 is used in Security Context derivation
- N1 transported as payload of 2.01 Created
- N2 transported together with the token

Pro: cleaner, don’t send nonces in OSCORE message

Con: Changes in Ace for POST /authz-info:
  - Allow use of Content-Format: application/ace+cbor together with CBOR map as payload (which MUST contain token)
Last Question

• Should we use Content-Format: “application/ace+cbor” for 2.01
  Created and use the registered parameter “nonce” to send N1?