Delegated Authenticated Authorization Framework (DCAF)

draft-gerdes-ace-dcaf-authorize

Stefanie Gerdes, Olaf Bergmann, Carsten Bormann
{gerdes | bergmann | cabo} @tzi.org

IETF-94, ACE Meeting, 2015-11-02
Review Comments

- Renzo: included in 04-version of DCAF:
  - Improved readability.
  - Removed inconsistencies.
  - Clarified definitions of CBOR keys.
  - Clarified handling of Ticket Request Messages.
  - Improved description of Nonces.

- Ludwig: addressed with 04-version of DCAF and DCAF-COSE
  - Also support COSE.
  - Address Server-Initiated Token Request (“Pull”).
  - Address piggy-backed protected content in SAM Information Message (“client-pull”).
  - Use a resource to store tokens (DCAF-COSE).
  - Bind an authorization token to the security context between C and RS using COSE.
Features of DCAF

- Secure exchange of authorization information.
- Establish security association between constrained nodes (secure distribution of session keys).
- Establish security association between a constrained and a less-constrained nodes.
- Support of class-1 devices (RFC 7228).
- Requires only symmetric key cryptography on the constrained nodes.
- DCAF-DTLS supports CoAP Observe (RFC 7641) and blockwise transfer without additional overhead.
- Relieve constrained nodes from managing complex authentication and authorization tasks.
Features of DCAF (2)

- Supports multiple owners.
- Defines cross-domain constrained to constrained communication (Required for constrained environments -> t2trg Meeting Prague).
- Relay security associations of less-constrained devices to constrained devices: Constrained devices only need the security association with their less-constrained device.
- Protects both sides of the communication (not only access to resources).
- Privacy: no device identifiers required on the constrained level.
- Provides a high level of implementation details.
- Explicit transfer of authorization information to the constrained devices possible: no additional knowledge required by the constrained nodes.
- Other formats for transmission of authorization information possible.
- Supports DTLS and Object Security (COSE).
The DCAF universe

- Communication Security using DTLS (draft-gerdes-ace-dcaf-authorize)
- Server-Initiated Ticket Request (draft-gerdes-ace-dcaf-sitr)
- Application Level Security using COSE (draft-bergmann-ace-dcaf-cose)

related:

- Examples for using DCAF with less-constrained devices (draft-gerdes-ace-dcaf-examples)
- Authorization Transitions in the lifecycle of constrained devices (draft-gerdes-ace-a2a)
Contact S’s Less Constrained Device for Authorization

[Diagram showing the flow of Access Request and Access Ticket between CAM, SAM, C, and S]
Access Ticket

Access Request

CAM

SAM

Access Ticket

Face:
[server authorization info]
nonce
[lifetime]

Client Information:
verifier (session key)
Access Ticket: Adding Client Information

Access Ticket

Face:
[server authorization info]
nonce
[lifetime]

Client Information:
verifier (session key)
[client authorization info, nonce]
[lifetime]
Use Access Ticket to Establish Security Context

Security Association
(e.g. using DTLS or COSE)
Key Derivation

Security Association
- transfer Ticket Face during setup

C
- use session key from Verifier (direct or derived)
S
- derive session key from Ticket Face and $K_{S,SAM}$
Access Ticket Parts

Access Request

CAM

SAM

Access Ticket

Face:
[server authorization info]
nonce
[lifetime]

Client Information:
verifier (session key)
[client authorization info, nonce]
[lifetime]
RS Permits Authorized Requests Over Secure Channel

Security Association

C <-> S

CoAP traffic

use Client Info for authorization

use Ticket Face for authorization
Combined Actors

Security Association (SA)

- CAM
- SAM

<table>
<thead>
<tr>
<th>Constrained nodes need an AM</th>
</tr>
</thead>
</table>

CAM = SAM

CAM and SAM can be combined in single-domain scenarios

C = CAM

not-so-constrained nodes can act on their own

S = SAM

not-so-constrained nodes can act on their own
Flexibility

- DCAF can be used as a simple protocol for secure transmission of dynamically created session keys (implicit authorization).
- DCAF can additionally securely transmit authorization information to the server and/or the client.
- DCAF defines how combinations of actors work together.
- DCAF can be used as needed.
Reference implementation of DCAF-DTLS adds

- about 440 Bytes Code
- 54 Bytes data for ticket face
- 722 Bytes parser for CBOR payload

to existing CoAP/DTLS server (ARM Cortex M3).
Evaluation: DCAF Memory Usage (ROM, RAM)

Numbers from Tobias Hartwich’s C implementation for Wismote using Contiki, libcoap, tinydtls, cn-cbor
Server-Initiated Ticket Request (SITR)

draft-gerdes-ace-dcaf-sitr

- In some scenarios, C might not be able to reach CAM or SAM
- S requests ticket for C
- C sends CAM information message to S to initiate SITR
CAM Information Message

CAM

SAM

C

this is my Request, this is my CAM

RS
SI Access Ticket

Server-Initiated Access Request

CAM

Access Ticket

SAM

Face:
 verifier (session key)

Client Information:
 [client authorization info]
 nonce
 [lifetime]

s
SI Access Ticket: Adding Server Information

Server-Initiated Access Request

**CAM**

**SAM**

**Face:**
- verifier (session key)
- [server authorization info, nonce]
- [lifetime]

**Client Information:**
- [client authorization info]
- nonce
- [lifetime]

**Access Ticket**
SIT Key Derivation

Security Association

- CAM
- SAM

C -> S
- transfer Client Info during setup
- derive session key from Client Info and $K_{C,CAM}$
- use session key from Verifier
Problem with Server-Initiated Solutions

- All solutions where the server requests a ticket for the client ("Pull Model") are prone to DOS attacks.
- Use solutions where the Client request the ticket whenever possible
Summary

- mutual authentication client-server, with symmetric keys (no need to separately obtain RPK to authenticate server)
- can make good use of DTLS-PSK
- can also use COSE with MAC, for transition of untrusted proxies
## DCAF-COSE vs. OSCOAP

<table>
<thead>
<tr>
<th></th>
<th><strong>DCAF-COSE</strong></th>
<th><strong>OSCOAP</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Changes to COSE</strong></td>
<td>use COSE as is (-06) no changes required</td>
<td>invent &quot;Secure Message format&quot; (COSE-profile in Appendix A)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>invent &quot;COSE Optimizations&quot; that are not COSE-compatible (new message types, remove unprotected header, alg ...)</td>
</tr>
<tr>
<td><strong>Security Context</strong></td>
<td>use parameter kid (identifies auth info and session key)</td>
<td>invent new parameter cid (identifies cipher suite, keys, alg-specific parameters, different for client and server: &quot;typically identifies the sending party&quot;)</td>
</tr>
<tr>
<td><strong>Replay protection</strong></td>
<td>use parameter nonce (-&gt; local time)</td>
<td>invent new parameter seq (-&gt; sequence number, no freshness information)</td>
</tr>
<tr>
<td><strong>Re-key</strong></td>
<td>Server sends SAM Information Message</td>
<td>&quot;out of scope&quot; (Section 7.1)</td>
</tr>
<tr>
<td><strong>Signaling</strong></td>
<td>use existing payload types two new options (not critical due to usual content-format handling)</td>
<td>implicit, new payload type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>new critical option</td>
</tr>
<tr>
<td><strong>Handling of unknown options</strong></td>
<td>COSE extension parameter to signal required options</td>
<td>not supported</td>
</tr>
<tr>
<td>RFC 7252, 7641 options block-wise</td>
<td>needs more work in CoRE WG</td>
<td></td>
</tr>
</tbody>
</table>
## DCAF-COSE vs. OAuth Profiling

<table>
<thead>
<tr>
<th></th>
<th>DCAF</th>
<th>OAuth Profiling</th>
</tr>
</thead>
<tbody>
<tr>
<td>C may be class 1</td>
<td>yes</td>
<td>only in single domain</td>
</tr>
<tr>
<td>cross-domain</td>
<td>yes</td>
<td>not for constrained-to-constrained communication</td>
</tr>
<tr>
<td>multi-owner</td>
<td>yes</td>
<td>?</td>
</tr>
<tr>
<td>PoP tokens</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Authn support</td>
<td>for C and RS</td>
<td>for RS; for C only in single domain</td>
</tr>
<tr>
<td>Authz support</td>
<td>for C and RS</td>
<td>for RS; for C only in single domain</td>
</tr>
<tr>
<td>/token</td>
<td>no</td>
<td>only in single domain</td>
</tr>
<tr>
<td>csp signaling</td>
<td>by RS or resource description</td>
<td>by AS</td>
</tr>
<tr>
<td>token introspection</td>
<td>optional</td>
<td>optional</td>
</tr>
<tr>
<td>dynamic session keys</td>
<td>(D)TLS-PSK COSE</td>
<td>(D)TLS OSCOAP</td>
</tr>
<tr>
<td>CWT</td>
<td>possible</td>
<td>possible</td>
</tr>
<tr>
<td>Privacy</td>
<td>no endpoint identifiers required</td>
<td>?</td>
</tr>
</tbody>
</table>
Discussion

Transport of Ticket Face for DTLS-PSK:

- **psk_identity**
  - Opaque for the client, no semantic restrictions
  - mandatory -> good interoperability
  - All known DTLS libraries pass it to the application to determine the PSK

- **supplemental data (RFC 4680)**
  - Client and server must support this extension.
  - Needs to define a new SupplementalDataType or a new AuthzDataFormat for client_authz (cf. RFC 5878)
  - Derivation of master-secret from supplemental data is not allowed ("Information provided in a supplemental data object [...] MUST NOT need to be processed by the TLS protocol.", RFC 4680)
How to proceed

- Accept DCAF as one of the building blocks that ACE is working on