Using onion routing with CoAP

draft-amsuess-t2trg-onion-coap-02

Christian Amsüss
Marco Tiloca, RISE
Rikard Höglund, RISE

IETF T2TRG interim meeting – Paris – May 21st, 2024
Overview (1/2)

› General
  – CoAP was designed to function both with direct connections and via proxies
  – This document defines how to establish chains of CoAP proxies (circuits), and ...
  – … supplemental services for discovery, naming, and message forwarding logic, and ...
  – … enables the operation of clients and (hidden) services similar to TOR

› Main building blocks
  – Resource directory (RFC 9176)
    › Registration/discovery of hidden services and associated proxies
  – OSCORE (RFC 8613)
    › Securing the CoAP messages exchanged
  – EDHOC (RFC 9528)
    › Establishing an OSCORE Security Context between a client/server and a proxy
  – draft-ietf-core-oscore-capable-proxies
    › OSCORE used at proxies and nested OSCORE protection ==> Secure tunneling
Overview (2/2)

- **Enabling Tor-like functionality for CoAP**
  - Common
    - EDHOC is used to establish an OSCORE association between OSCORE peers
  - The Client uses OSCORE
    - End-to-end with the (hidden) service
    - With each of the proxies in the (client-side) circuit with the (hidden) service
  - The Server uses OSCORE
    - End-to-end with the Client
    - With each of the proxies in the (server-side) circuit with the Client
  - Resource Directory
    - Allows for hidden services and associated proxies to advertise themselves as available
    - Allows for clients to discover a hidden service and the proxy to refer to for it
  - Proxies
    - Relays messages to servers/clients or other proxies
    - Allows clients and (hidden) services to retrieve the list of available proxies, or a subset of it
Proxy Discovery

› **Rationale**
  – Clients and servers need to discovery lists of eligible proxies
    › Along with metadata such as if the proxy wishes to function as an exit node
  – A list of registered proxies is maintained
    › The list needs to be signed by independent entities
    › The list will be large so it should be split into smaller pieces
  – Devices will be provisioned with a fragment of the list at bootstrap
  – Devices can retrieve the latest list from proxies in the network

› **Proxy list contents**
  – Proxy’s cryptographic identity
  – Affiliation information (operator and location)
  – Optionally a public IP address: If available as first hop
Example Topology

Without hidden service

With hidden service

Client-side circuit: P1, P3, P6, P7
Server-side circuit: P5, P2, P4, P7

P7 acts as Introduction Point
Without Hidden Service

› Rationale
  – A client picks the proxies P1, P2, P3 to hide its position from the server S
  – Nest OSCORE-protected forwarding information
    › Provide a proxy Px with its forwarding instructions protected with OSCORE for Px
    › A CoAP request can contain one forwarding instruction per OSCORE layer (source routing)
      - One instruction per forward-proxy (Proxy-Uri; or Uri-Host + Proxy-Scheme)

› Circuit establishment procedure
  – Client runs EDHOC with the P1
  – Client attempts to run EDHOC with P2 through P1
  – P1 runs EDHOC with P2
  – Client completes EDHOC with P2 through P1
  – ...
  – OSCORE Contexts: C<->S, C<->P1, C<->P2, C<->P3, P1<->P2, P2<->P3, P3<->S

Without Hidden Service
With Hidden Service (1/2)

› **Addressing convention**
  – The advertised address of a hidden service is a "public-authentication-credential" address
  – Its building is based on draft-amsuess-t2trg-rdlink and draft-ietf-core-transport-indication

› **Example address**
  – coap://tlsa.X.pubcred.arpa
  – X is the base32url encoding of the binary representation a CBOR map
  – The CBOR map conveys the public authentication credential of the hidden service by value or by reference
    › As allowed by the COSE Header parameters (to be) registered at [https://www.iana.org/assignments/cose/cose.xhtml#header-parameters](https://www.iana.org/assignments/cose/cose.xhtml#header-parameters)
With Hidden Service (2/2)

› As a hidden service, the server hides its position – This means that:
  – The server
    › Chooses an "Introduction Point" proxy
    › Establishes a server-side circuit terminating at the Introduction Point
    › Asks the Introduction Point to make a registration at the Resource Directory
  – The client
    › Through a proxy, looks up for the hidden service by name, thus discovering the Introduction Point
    › Establishes a client-side circuit terminating at the Introduction Point
    › Establishes an end-to-end association with the hidden service

› When the client contacts the hidden service for the first time …
  – The hidden service may offer to set up and switch to a different data-circuit where to communicate
  – This takes traffic away from the "Introduction Point" to a different "Rendez-Vous Node" proxy
Circuit for Hidden Service (1/2)

› Defining 2 new CoAP options
  – Enable-Reverse-Proxy (empty)
    › When the request you are proxying for me results in role reversal, please reverse-proxy to me
  – Reverse-Uri-Host (text string HOST)
    › Your incoming reversal requests that relate to this request REQ must have Uri-Host = HOST

› Assuming P4 both as Introduction Point ...
  – The Server establishes a server-side circuit terminating at the P4
    › Reverse-forwarding rules are installed on the proxies using the CoAP options
  – P4 registers itself as Introduction Point for the Server at the Resource Directory
  – The Client looks for the Server and discovers P4 as Introduction Point
  – The Client establishes a client-side circuit terminating at the P4
  – The Client runs EDHOC with the Server through the full double-circuit
Circuit for Hidden Service (2/2)

Runtime-built Routing Tables at Proxies

› Requested Host (unique lookup key)
  – Matched against the value of the Uri-Host option of incoming requests

› Next Hop Address
  – Where to forward an incoming request to.

› Next Hop Virtual Host (it can be empty)
  – The value to be specified in the Uri-Host option of an incoming request when forwarding it.

› CTX_1 - Identifier of OSCORE Sec Ctx
  – Used for performing the innermost protection of an incoming request to forward.

› CTX_0 - Identifier of OSCORE Sec Ctx (it can be empty)
  – Used for performing the outermost protection of a request to forward.

Forwarding REQ towards a hidden service

1. Remove the Uri-Host option from REQ.

2. If the field "CTX_0" is not empty, move to step 3. Otherwise:
   – If the field "Next Hop Virtual Host" is not empty, add to REQ a Uri-Host option, with the value specified in the field "Next Hop Virtual Host".
   – Protect REQ with CTX_1, then move to step 5.

3. Protect REQ with CTX_1.

4. If the field "Next Hop Virtual Host" is not empty, add to REQ a Uri-Host option, with value the same value specified in the field "Next Hop Virtual Host". Then, protect REQ with CTX_0.

5. Send REQ to the address specified in the field "Next Hop Address".
Next Steps

› **Continue updating the Internet Draft**
  - We have separate, detailed notes on a number of points, including:
    › Conceptual comparison with TOR
    › Strawman of step-by-step circuit establishment, with a hidden service
    › Separate circuit establishment when the Rendez-Vous node is a different proxy
  - Open issues tracked at: [https://gitlab.com/chrysn/onion-coap/-/issues](https://gitlab.com/chrysn/onion-coap/-/issues)

› **Any feedback and questions are welcome**
Thank you!

Comments/questions?

https://gitlab.com/chrysn/onion-coap/
Backup
Cross section of the Client 1 --> Server 1 connection