CoRE Working Group
Internet-Draft
Intended status: Informational
Expires: September 20, 2016

G. Selander
F. Palombini
Ericsson AB
K. Hartke
Universitaet Bremen TZI
L. Seitz
SICS Swedish ICT AB

March 19, 2016

Requirements for CoAP End-To-End Security
draft-hartke-core-e2e-security-reqs-00

Abstract

This document analyses threats to CoAP message exchanges traversing proxies and derives the security requirements for mitigating those threats.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 20, 2016.

Copyright Notice

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252] is a Web application protocol designed for constrained nodes and networks [RFC7228].

CoAP uses Datagram Transport Layer Security (DTLS) [RFC6347] for security. At the same time, CoAP relies on proxies for scalability and efficiency. These proxies are specified to perform a number of operations on CoAP messages which requires DTLS to be terminated at the proxy. The proxy therefore not only has access to the data required for performing the desired proxy functionality, but is also able to eavesdrop on or manipulate any part of the CoAP payload and metadata in transit between client and server or inject new CoAP messages without being protected or detected by DTLS.

One way to mitigate this threat is to secure CoAP communication at the application layer using an object-based security mechanism such as CBOR Encoded Message Syntax [I-D.ietf-cose-msg] instead of or in addition to the security mechanisms at the network layer or transport layer. Such a mechanism can provide "end-to-end security" at the application layer in contrast to the "hop-by-hop security" provided by DTLS.

This document analyses security requirements for CoAP requests and responses of sensor and actuator deployments involving proxies and other similar intermediaries. The analysis is based on identifying the assets associated to sensor- and actuator-based communication patterns and considering the potential threats executed through proxies to these assets. The threat analysis provides the basis for defining the security requirements that an end-to-end security mechanism for CoAP needs to meet.

1.1. Terminology

Readers are expected to be familiar with the terms and concepts described in [RFC7252].

2. Scope and Assumptions

This document presents a number of scenarios involving sensor and actuator communications over CoAP. Common to all scenarios is the presence of at least one CoAP intermediary, typically in the form of a proxy between a client requesting a resource and a server hosting a resource (see Figure 1). The proxy is responsible, for example, for reducing response time and network bandwidth use by serving responses from a cache or for enabling the client to make requests that it otherwise could not make.

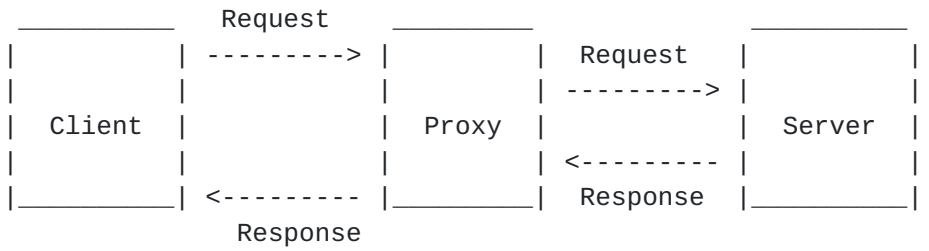

Figure 1: CoAP Message Exchanges Through A Proxy

The basic function of a proxy is to forward translated messages according to certain processing rules. For example:

- Forward a message to the next proxy when the link is up
- Only forward a request if there is no fresh cached response
- Forward a new publication to all subscribing clients

In order to perform its function, a proxy may be required to read or change certain parts of a CoAP message as defined in [RFC7252]. For example, a forward proxy is defined to transform the Proxy-Uri option to Uri-Host, Uri-Port, Uri-Path and Uri-Query options. A proxy caching responses needs to read the Cache Key and is required to change the Max-Age option in the responses.

Since a proxy might not be fully trusted, a security solution is needed that protects the client, the server and the message exchanges against certain threats while still allowing the proxy to assume its normal functionality. The client and server are assumed to have a security association, but the proxy is neither assumed to have a security association with the client nor with the server.

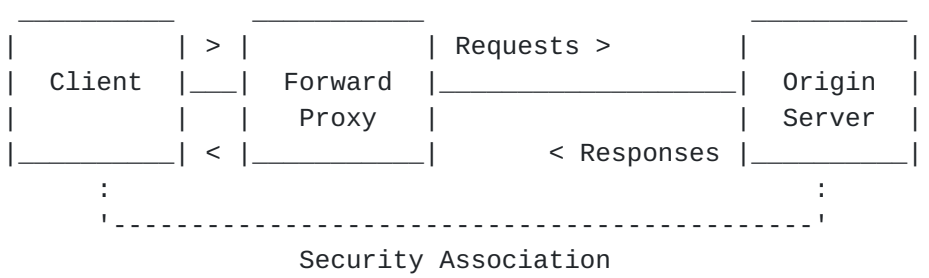

Figure 2: Security Association Between Client and Server

For a start, this document considers the following two cases: Forward proxies (as specified in [RFC7252]; Figure 2) and publish-subscribe brokers (as specified in [I-D.koster-core-coap-pubsub]; Figure 3).

The functionality assumed by these nodes is summarized in the respective scenarios analyzed in this document (Section 3).

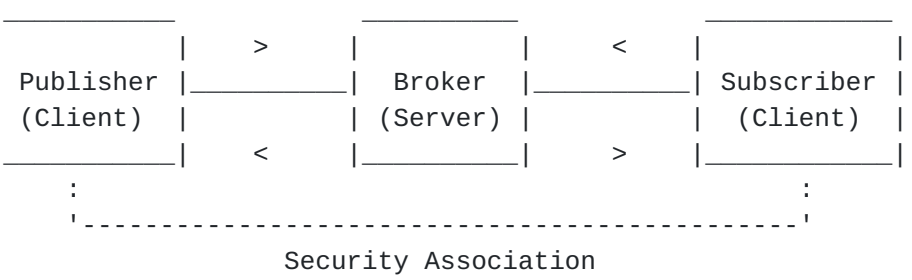

Figure 3: Security Association Between Publisher and Subscriber

[TODO: Reverse proxy and cross-protocol proxies will be added in a future version of this document.]

To identify the threats in scope, we first consider what assets need to be protected. In general, there are the following types of assets to protect:

A1: The devices at the two ends, the data generated and stored in these devices, and their (often very constrained) system resources such as available memory, storage, processing capacity, and energy.

A2: The physical environment of the devices fitted with sensors and actuators. Access to the physical environment is provided through CoAP resources that allow a remote entity to retrieve information about the physical environment (such as the current temperature) or to produce an effect on the physical environment (such as the activation of a heater).

A3: The communication infrastructure linking the two devices (which often contains some very constrained parts) and the data stored in the message processing devices.

The scope of this document is to analyze threats executed through proxies and brokers, and this is only directly affecting the assets of type A3, e.g., if a proxy is dropping all messages.

However, the intermediary node may manipulate the messages exchanged between the endpoints and thereby have an impact also on the assets A1 and A2, for example: flooding a device with messages has impact on its system resources, and successful manipulation of an actuator command, carried in a message, has an impact on the physical environment. We therefore add a fourth asset, which is the main target being evaluated in this document:

A4: The messages exchanged between a client and a server, through the proxy. This includes the CoAP header and options in request and response messages (such as the requested method or the target URI) and the CoAP resource representations, encapsulated in the message payload.

A fully trusted proxy, handling unprotected messages, is an attractive target, since proxies are aggregation points for message flows (see Section 4) and they may be an easier target from the Internet than the sensors/actuators residing behind them. A proxy may become subject to intrusion or become infected by malware and perform the attacks of a man-in-the-middle. The attack vectors for compromising a proxy and the associated risks are out of scope for this document.

The scope of the threat analysis is restricted to threats from proxies to single client to server interactions. Threats resulting from collusion between multiple proxies are also out of scope (see Section 4).

On a high level, there are the following threats from proxies to consider:

T1: The proxy illegitimately modifies a message.

T2: The proxy illegitimately sends a message, including replay, flooding, etc.

T3: The proxy illegitimately inhibits sending of a message, including delay, reordering, etc.

T4: The proxy illegitimately reads part of a message.

To assess how such threats impact the assets, we need to specify the processing rules of the intermediary nodes in different scenarios and define the associated security objectives.

3. Scenarios, Threats and Security Requirements

In this section we consider a set of scenarios involving proxies and brokers, with different processing rules and security objectives. We study the associated threats and derive the security requirements for message transfer between client and server, in the different scenarios.

Note that, since CoAP was not designed for end-to-end security, solutions complying with these security requirements extend the applicability of CoAP beyond its original scope.

To simplify the analysis, the scenarios are structured according to how requests and responses are related to each other:

One Request - One Response
: There is a one-to-one relation between request and response.

One Request - Multiple Responses
: A request may have multiple responses, but each response is securely linked to a unique request.

Multiple Requests - One Response
: One response may serve multiple requests, but each request has a single response.

Multiple Requests - Multiple Responses
: One response may serve multiple requests, and each request may have multiple responses.

3.1. One Request - One Response

In this scenario we study use cases where it is important that a response sent from one endpoint is the response to a particular request to that endpoint. Many security critical use cases require that responses are in this way "securely linked" to requests, such as alarm status retrieval and actuator command confirmation.

In this scenario there must be a unique response for each request.

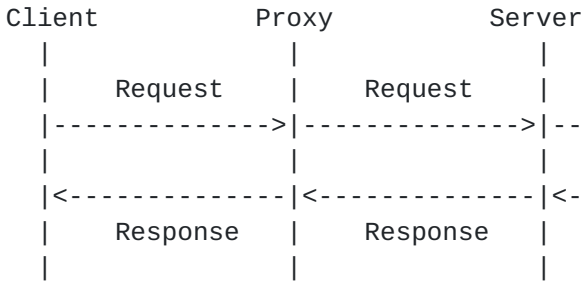

Figure 4: Message Flow with a Unique Response for Each Request

Example: Alarm status retrieval

Figure 4 can be seen as an illustration of a message exchange for a client requesting the alarm status (e.g., GET /alarm_status) from a server. Since the client wants to ensure that the alarm status received is reflecting the current alarm status and not a cached or spoofed response to the same resource, it must be able to verify that the received response is a response to this

particular request made by the client. Therefore the response must be securely linked to the request.

Example: Actuation confirmation

Another example for which Figure 4 serves as illustration is the confirmation of an actuator request. In this case a client, say in an industrial control system, requests a server that a valve should be turned to a certain level, e.g. PUT /valve_42/level with payload "3". In order for the client to correctly evaluate the result of a potential changed valve level, it is important that the client gets a confirmation how the server responded to the requested change, e.g., whether the request was performed or not. Again, the client wants to ensure that the response is reflecting the result of this particular actuation request made by the client and not a cached or spoofed response. Therefore the response must be securely linked to the request.

Functional Requirement:

- Since each response is intended to be securely linked to a particular request, the response must not be used with any other request. Hence, as much as possible of the caching functionality must be inhibited. Therefore the CoAP option Max-Age of the responses is set to 0 (see Section 5.7.1 of [RFC7252]).

3.1.1. Processing Rules

In this scenario, the desired proxy functionality is to forward a translated request to the determined destination. There are two modes of operation for requests: Either using the Proxy-Uri option (PR1.1) or using the Proxy-Scheme option together with the Uri-Host, Uri-Port, Uri-Path and Uri-Query options (PR1.2).

PR1.1 The Proxy-Uri option contains the request URI including request scheme (e.g. "coaps://"); the Proxy-Scheme and Uri-* options are not present.

If the proxy is configured to forward requests to another proxy, then it keeps the Proxy-Uri option; otherwise, it splits the option into its components, adds the corresponding Uri-* options and removes the Proxy-Uri option. Then it makes the request using the request scheme indicated in the Proxy-Uri.

PR1.2 The Proxy-Scheme option and the Uri-* options together contain the request URI; the Proxy-Uri option is not present.

If the proxy is configured to forward requests to another forwarding proxy, then it keeps the Proxy-Scheme and Uri-* options; otherwise, it removes the Proxy-Scheme option. Then it makes the request using the request scheme indicated in the removed Proxy-Scheme option.

PR1.3 Responses are forwarded by the proxy, without any modification.

3.1.2. Security Objectives

In this scenario there is a unique response for each request, so the client should be able to verify that a certain response is made in response to a specific request sent by the client.

The server should be able to verify that the proxy only has performed the message modifications intended by the client according to the processing rules.

The proxy should be prevented from reading or making modifications to messages apart from what is necessary to perform the processing rules (cf. [RFC7258]).

The security objectives are:

SO1.1 The server is able to verify that a received request originates from a client with which it has a security association, and that the request has not been received before.

SO1.2 The server is able to verify that the received request either has not been altered in transfer, or that the request is modified according to the processing rule PR1.1 or PR1.2 (Section 3.1.1).

SO1.3 The server is able to protect the response such that only authorized clients can read the response.

SO1.4 The client is able to verify that the received response originates from the requested server and resource, that it has not been altered in transfer, and that it was generated as the unique response to the request.

SO1.5 The proxy is only able to read data needed to perform the processing rules.

3.1.3. Threat Analysis and Mitigation

We now list potential threats and discuss candidate mitigation mechanisms.

3.1.3.1. T1:The proxy illegitimately modifies a message

T1:1.1 The proxy forwards a request with modified payload

This threat can be mitigated with integrity protection of payload.

T1:1.2 The proxy forwards a response with modified payload

This threat can be mitigated with integrity protection of payload.

T1:1.3 The proxy forwards a request with modified CoAP option

Note that the proxy is entitled to change certain options by processing rules PR1.1 and PR1.2. Since the change is predictable, the effective request URI can be integrity protected by the client and verified by the server. The other CoAP options in the request can be integrity protected.

T1:1.4 The proxy forwards a response with modified CoAP option

This threat can be mitigated with integrity protection of CoAP options. Since Max-Age is set to 0 the proxy is not entitled to change any options in the response so they can all be integrity protected.

T1:1.5 The proxy forwards a request with changed CoAP header fields

The proxy is entitled to change certain header fields (e.g., the token) as part of its normal operations. Malicious changes to message layer parameters may cause a denial-of-service, equivalent of dropping a message or sending spoofed messages. This is difficult to mitigate. However, changing the CoAP header Code (e.g., from GET to DELETE) may result in an error or wrong interpretation of the request which can have other security implications. A change to the Version header field may result in security errors in the interaction between different versions of CoAP. These threats can be mitigated by integrity protecting the Code and Version header fields.

T1:1.6 The proxy forwards a response with changed CoAP header fields

Similar to previous threat. Some aspects of this threat can be mitigated by integrity protecting the Code and Version header fields.

T1:1.7 The proxy forwards a different request

If the forwarded request is from another client it can be mitigated by having different security associations with different clients. If the forwarded request is from the same client but with differences in payload, options or header, then this coincides with previously listed threats. A proxy sending old requests (or reordering requests) from the same client to the same server resource can be mitigated by integrity protecting a freshness parameter (timestamp, counter, etc.) from which the order of requests can be deduced (replay/reordering protection).

T1:1.8 The proxy forwards a different response

By integrity protecting uniquely identifying information of the request in the response, the client can verify that the response was generated in reply to a particular request.

3.1.3.2. T2:The proxy illegitimately sends a message

T2:1.1 The proxy sends a request to the server without a previous request from the client

This threat may be mitigated with integrity- and replay protection.

T2:1.2 The proxy sends a response to the client without a previous response from the server

Error messages from the proxy such as 5.02 (Bad Gateway) originate from the proxy. A proxy maliciously sending error messages is a denial-of-service attack similar to not forwarding a message (T3:1.1) and is difficult to mitigate. However, responses claiming to be from the server may be mitigated with integrity protection uniquely identifying information of the request.

T2:1.3 A proxy sends a number of messages for the purpose of flooding client or server

By verifying the integrity, the client and server may mitigate certain flooding attacks. The server can use the replay/reordering protection to verify which messages are

legitimate and the client can verify if a message is a response to a previously sent request.

3.1.3.3. T3:The proxy illegitimately inhibits sending of a message

T3:1.1 The proxy does not forward a message

This is a denial-of-service attack. While these kind of threats may be difficult to mitigate, applications should have a readiness for this kind of issues and a client is able to detect a missing response.

T3:1.2 The proxy delays forwarding of a received message

Delayed forwarding may be a denial-of-service attack, similar to not forwarding. Certain delays may be legitimate, so they may be difficult to detect and mitigate. However, delayed requests and responses can also be used in attacks against actuators; see [I-D.mattsson-core-coap-actuators]. These attacks can be performed by an on-path attacker and are not restricted to proxies. The proposed mitigation is based on verifying the timeliness of the request, for example, by using time stamps or with an additional round-trip. These mitigations can be supported by a new CoAP option containing time stamp or binding the response in a first round-trip to a request of the second, as specified in [I-D.mattsson-core-coap-actuators]. By integrity protecting that new CoAP option, the threat can be mitigated.

T3:1.3 The proxy reorders the requests

This threat may be mitigated with the server integrity protecting a freshness parameter from which the order of requests can be deduced.

T3:1.4 The proxy reorders the responses

This threat may be mitigated with the server integrity protecting information specifying to which request a response belongs.

3.1.3.4. T4:The proxy illegitimately reads part of a message

T4:1.1 The proxy reads a representation/payload

This threat can be mitigated with encryption of the payload.

T4:1.2 The proxy infers information about the nature and state of the resource request/response from CoAP options

The proxy only needs to read the Uri-Host/Uri-Port and Proxy-Uri/Proxy-Scheme options of a request. The information revealed by these parameters is public on network layer. The proxy only needs to read Max-Age of the response, which is set to 0 as indicated in the functional requirements. This threat can be mitigated by encrypting all other options.

T4:1.3 The proxy infers information about the nature and state of the resource request/response from CoAP header fields

The header fields needs to be transferred in plain text to allow normal CoAP operations. The Code parameter reveals information about what RESTful action is requested. This information leakage is difficult to mitigate.

T4:1.4 The proxy reads and stores all message exchanges and can deduce information about the entire history of the corresponding interactions

This threat can be mitigated with encrypting as much as possible of the data transferred between client and server. The case of long term key compromise can be mitigated with forward secrecy.

3.1.4. Security Requirements

This section contains the security requirements and non-requirements for this scenario. For each requirement and non-requirement the associated threats are listed. The security requirements are:

R1.1 The server must authenticate a message coming from a requesting client (T1:1.1, T1:1.3, T1:1.5, T2:1.1).

R1.2 The server must verify that it has not received this request previously (T1:1.7, T3:1.3).

R1.3 The client must verify that the received response originates from the requested server (T1:1.2, T1:1.4, T1:1.6, T2:1.2).

R1.4 The client must verify that a response corresponds uniquely to a previous request that the client has made (T1:1.8, T3:1.4).

R1.5 The payload must be integrity protected and encrypted between client and server (T1:1.1-6,T4:1.1, T2:1.3, T4:1.1, [RFC7258]).

R1.6 The CoAP options except Uri-* and Proxy-* must be integrity protected in the request. The effective request URI must be integrity protected in the request (T1:1.3).

R1.7 All CoAP options in the response must be integrity protected. Max-Age must be set to 0 (T1:1.4).

R1.8 The CoAP options Uri-Host/Port and Proxy-Uri/Scheme of the request must not be encrypted. The Max-Age option of the response must not be encrypted. All other options must be encrypted (T4:1.2).

R1.9 The CoAP header fields Version and Code must be integrity protected in requests and responses. All other header fields must not be integrity protected. The header fields must not be encrypted (T1:1.5, T4:1.3).

R1.10 The communication protocol must provide forward secrecy (T4:1.4).

The security non-requirements of this scenario are:

NR1.1 The proxy may drop messages without the endpoint being able to infer that the message is lost due to the proxy (T3:1.1).

NR1.2 The proxy may delay messages without being detected (T3:1.1, T3:1.2).

NR1.3 The proxy may read the CoAP header including message layer parameters and Code, revealing the kind of RESTful action being requested and the response code (T4:1.3).

3.2. One Request - Multiple Responses

In this scenario we study use cases where it is important that a response is securely linked to a request as in the previous scenario, but where there may be multiple responses for each request. This functionality protects communication-constrained servers from repeated requests from the same client and thus saves system resources and bandwidth. This is useful in security critical monitoring scenarios where time synchronization cannot be guaranteed.

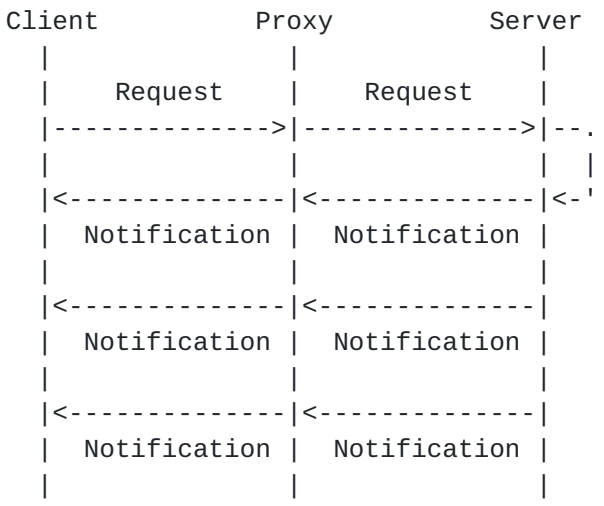

Figure 5: Message Flow of a Notification

Example: Secure parameter monitoring

Figure 5 can be seen as an illustration of a message exchange for a client monitoring an important parameter measured by the server, e.g., in a medical or process industry application. The client makes a subscription request for a resource and the server responds with notifications, thereby providing updates to the parameter in regular time intervals.

The client wants to ensure that first received notification reflects the current parameter value and that subsequent notifications are timely updates of the initial request. Since notifications may be lost or reordered, the client needs to be able to verify the order of the messages, as sent by the server. By monitoring the received messages and the time they are received, the client can detect missing notifications and take appropriate action.

Functional Requirement:

- The same functional requirement apply as in the previous scenario (Section 3.1).

3.2.1. Processing Rules

The processing rules are identical to PR 1.1 - 1.3 of the previous scenario (Section 3.1.1).

3.2.2. Security Objectives

The security objectives are similar to the previous scenario. Each response maps to a unique request, but there may be multiple responses to one request. By ordering the responses, each message in this exchange can be made unique.

The security objectives of the previous scenario (Section 3.1.2) are valid except for SO1.4 which is replaced by the following objectives:

SO2.1 The client is able to verify that the received response originates from the requested server and resource, that it has not been altered in transfer, and that it was generated as one in a sequence of responses to the request.

SO2.2 The client is able to verify the order of the responses and if a response is missing.

3.2.3. Threat Analysis and Mitigation

The threat analysis from the previous scenario carries over with a few exceptions.

3.2.3.1. T1:The proxy illegitimately modifies a message

Similar conclusions apply as in the previous scenario (Section 3.1.3.1). However, note that in T1:1.8, a proxy may maliciously reorder the responses to the same request without being detected. The mitigation specified in the previous scenario (that the client verifies the response is linked to the request) is not sufficient since there may be multiple responses.

However, analogous to how requests are protected against replay/reordering in the previous scenario, by additionally integrity protecting a parameter from which the order of responses can be deduced, this threat can be mitigated.

3.2.3.2. T2:The proxy illegitimately sends a message

Similar conclusions apply as in the previous scenario (Section 3.1.3.2). T2:1.3 can be mitigated with the additional replay/reordering protection of responses as mentioned in Section 3.2.3.1.

3.2.3.3. T3:The proxy illegitimately inhibits sending of a message

Similar conclusions apply as in the previous scenario (Section 3.1.3.3). T3:1.4 can be mitigated with the additional replay/reordering protection of responses as mentioned in Section 3.2.3.1.

3.2.3.4. T4:The proxy illegitimately reads part of a message

The same conclusions apply as in the previous scenario (Section 3.1.3.4).

3.2.4. Security Requirements

The security requirements of the previous scenario (Section 3.1.4) are valid except for R1.4 which is replaced by the following requirements:

R2.1 The client must verify that a response corresponds to a previous request that the client has made (T1:1.8, T3:1.4).

R2.2 The client must verify that it has not received this response previously and whether responses for the same request are received in the wrong order (T1:1.8, T3:1.3).

3.3. Multiple Requests - One Response

In this scenario we study caching: how a proxy may serve the same cached response to multiple clients requesting the same resource.

The caching functionality protects communication-constrained servers from repeated requests for the same resources, possibly originating from different clients. This saves system resources, bandwidth, and round-trip time.

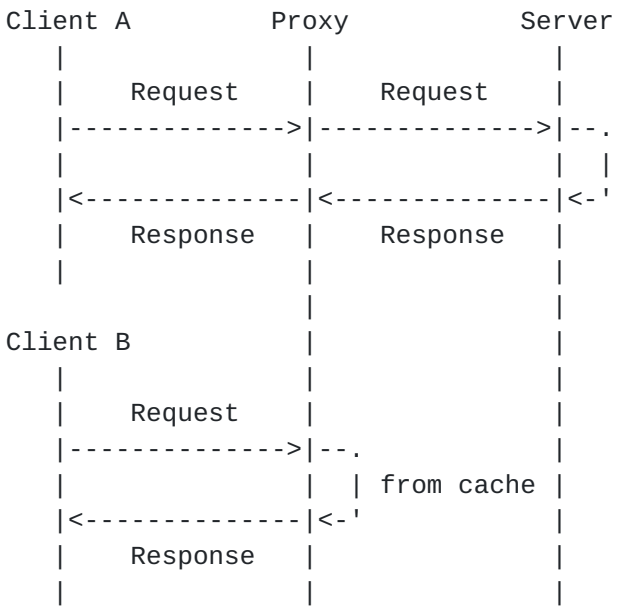

Figure 6: Message Flow for Cached Responses

In Figure 6, Client A requests the proxy to make a certain request to the server and to return the server's response. The proxy services the request by making a request message to the server according to the processing rules. If the server returns a cacheable response, then the proxy stores the response in its cache, performs any necessary translations, and forwards it to the client. Later, client B makes an equivalent request to the proxy that the proxy services by returning the response from its cache.

Cacheable responses are 2.05 (Content) responses and all error responses.

Functional Requirements:

- The proxy must be able to store cacheable responses in a cache. This requires the proxy to read the CoAP header, options, and payload and to compute the cache key for a request.

- The proxy must be able to return a fresh response from its cache without contacting the server.

- The proxy must be able to perform validation on a request by a client and a request validation to the server (see Section 5.6.2 of [RFC7252]).

3.3.1. Processing Rules

The proxy complies with the forwarding rules PR1.1 - 1.3 (Section 3.1.1) and the rules below. The rules below have priority.

PR3.1 If the proxy receives a request where the cache key matches that of a cached fresh response, then the proxy discards the request and replies with that response, else it makes a translated request.

PR3.2 The proxy caches and forwards cacheable responses. If there is already a response in the cache with the cache key of the corresponding request, then the old response in the cache is marked as stale.

PR3.3 If the proxy receives a request that contains an ETag option and the proxy has a fresh response with the same cache key and ETag, then the proxy replies to the request with a 2.03 (Valid) response without payload, else it forwards a translated request.

PR3.4 The proxy updates the Max-Age option according to the Max-Age associated with the resource representation it receives, decreasing its value to reflect the time spent in the cache.

PR3.5 If the request contains an Accept option and if there is a fresh response that matches the cache key for the corresponding request except for the Accept option, and if the Content-Format of the response matches that of the Accept option, then the proxy forwards the cached response to the requesting client.

3.3.2. Security Objectives

A caching proxy has an active role in the resource request/response procedure, so it is not surprising that it is necessary to make a trade-off between caching functionality and the protection of client-server interaction. Comparing with the scenario in Section 3.1, most of the security objectives in Section 3.1.2 cannot be met:

- The caching functionality decouples responses from requests. This implies that a client is not able to verify that a received response is generated by the server in response to a specific request.

- A client may receive a response without the server being aware that the client has made a request. A proxy could proactively generate requests or observe resources in order to keep the cache

up-to-date. Thus the server cannot in general verify that a request originates from a client as a precondition to provide a response.

Since a proxy can autonomously make requests for resource representations and there is no security association between proxy and server, the server cannot verify those requests. If a request needs to be verified then the solution to the scenario in Section 3.1 can be re-used. Therefore we do not consider the protection of requests and focus here on enabling the caching functionality and providing security to cacheable resource representations.

The security objectives for this scenario are:

SO3.1 The client is able to verify that a received response contains a resource representation to a requested server and resource, and that it has not been altered between server and client.

SO3.2 The client is able to verify that a received resource representation is fresh.

SO3.3 The server is able to protect a resource representation such that only authorized clients can read the representation.

3.3.3. Threat Analysis and Mitigation

We now list potential threats and discuss candidate mitigation mechanisms.

3.3.3.1. T1:The proxy illegitimately modifies a message

T1:3.1 The proxy forwards a request with modified payload

Out of scope of the security objectives.

T1:3.2 The proxy forwards a response with modified payload

This threat that may be mitigated with integrity protection of resource representation.

T1:3.3 The proxy forwards a request with modified CoAP options

Out of scope of the security objectives.

T1:3.4 The proxy forwards a response with modified CoAP options

This is not necessarily a threat. For example, a proxy is entitled to change Max-Age. However, changing Content-

Format may result in an error or the wrong interpretation of a representation. That kind of threat may be mitigated by securely associating resource information (such as Content-Format) to the representation in the response.

T1:3.5 The proxy forwards a request with changed CoAP header fields

As mentioned in Section 3.1, this is not necessarily a threat and it is not in scope of the security objectives to mitigate.

T1:3.6 The proxy forwards a response with changed CoAP header fields

This is not necessarily a threat, since message layer parameters may be changed by a proxy. A change of Code in the response may be misinterpreted. But as long as the responses allow verification of resource information, such a change will be detected. Thus this threat is mainly a denial-of-service. Threats arising from modification of Version are difficult to predict. A future version of CoAP must consider security implications of a proxy manipulating the version number.

T1:3.7 The proxy forwards a request different from the translated request

Out of scope of the security objectives.

T1:3.8 The proxy forwards a response to a non-equivalent request

If the response is from another server, then it can be mitigated by having different security associations with different servers. If the response is that of another resource of the same server, it can be mitigated by having different security associations of different resources, or by securely associating a resource identifier to the representation in the response. If the response is from the right server and resource, then the modifications of payload, options and header are considered previously.

T1:3.9 The proxy forwards an old response to the same resource

This is not necessarily a threat. The proxy is supposed to send a cached response, if fresh. However, if the proxy serves a stale response and manipulates the Max-Age option, then it may trick the client into believing that this is a fresh response. Since the proxy is entitled to make such

changes, this is not possible to prevent. The server may however provide other freshness information (timestamp, counter, etc.) integrity protected together with the resource representation and associated resource information from which the client may infer that Max-Age is not correct. Note that in case time synchronization cannot be assumed the information about age is limited to the order of the responses.

T1:3.10 The proxy maliciously serves a 2.03 (Valid) response to a request with an ETag option

This is not possible to prevent, since the proxy is entitled to perform such operation without involving the server. [TODO: Since the response must not include a payload (see Section 5.9.1.3 of RFC 7252), it is not clear how a server could enforce the proxy to include any integrity protected freshness information unless we define new proxy processing rules.]

T1:3.11 The proxy colludes with a legitimate client having access to the key used to generate and verify Message Authentication Codes (MAC) of responses/resource representations to generate a valid MAC.

This threat applies to responses containing a message authentication code (MAC) for integrity protecting the resource representation. The threat may be mitigated by the server digitally signing the representation with its private key instead of using a MAC.

3.3.3.2. T2:The proxy illegitimately sends a message

T2:3.1 The proxy sends a request to the server without a previous request from the client

This is not necessarily a threat, since the proxy may want to keep the cache updated with fresh representations to allow short round-trip time. A proxy maliciously making requests for the purpose of gaining information about the resources may to some extent be mitigated by encryption, but encrypting data in the cache key has an impact on how the cache can perform its legitimate operation. This is out of scope for the security objectives.

T2:3.2 The proxy sends a response to the client without a previous response from the server

This is not necessarily a threat, since the proxy is allowed to respond with a fresh, cached response. Other cases of responding inappropriately to a client request are covered in the previous section. The client can detect the case of receiving a response without having sent a request.

T2:3.3 A proxy sends a number of messages for the purpose of flooding client or server

Considering that a proxy is entitled to make resource requests, it may be difficult to protect the server against this kind of denial-of-service attacks. As for responses, by verifying the integrity and freshness of requested information, the client may mitigate certain flooding attacks.

3.3.3.3. T3:The proxy illegitimately inhibits sending of a message

T3:3.1 The proxy does not forward a message

This is not necessarily a threat. According to the processing rule, the proxy must not forward a request if there is a fresh cached response. If the proxy does not forward a request although there is no valid cache response or if the proxy does not propagate a response, then this is a denial-of-service attack. While these threats may be difficult to mitigate, missing messages are common in lossy environments so applications should be prepared for this kind of issue.

T3:3.2 The proxy delays forwarding of a received message

Delayed forwarding may be a denial-of-service attack, similar to not forwarding. Certain delays may be legitimate, so it is difficult to detect and mitigate this. Delayed requests and responses can also be used in attacks against actuators as is discussed in Section 3.1, but that is out of scope for this scenario.

T3:3.3 The proxy reorders the requests

Out of scope of the security objectives.

T3:3.4 The proxy reorders the responses

This threat may be mitigated with the server integrity protecting a freshness parameter together with the response.

3.3.3.4. T4:The proxy illegitimately reads part of a message

T4:3.1 The proxy reads a representation/payload

This threat can be mitigated with encryption of the representation, and other potential payload data.

T4:3.2 The proxy infers information about the nature and state of the resource request/response from CoAP options and header fields.

The proxy needs to read the cache key for performing caching operations. Information leaking that can be inferred from such data cannot be prevented.

T4:3.3 The proxy reads and stores all message exchanges and can deduce information about the entire history of resource access.

Since the cache key and other metadata is not in scope of the security objectives, the mitigation is restricted to encrypting the resource representations. The case of long term key compromise would nevertheless reveal the history of the resource, but this can be mitigated with forward secrecy.

3.3.4. Security Requirements

This section contains the security requirements and non-requirements for the caching scenario. For each requirement and non-requirement the associated threats are listed. The security requirements are:

R3.1 The client must be able to verify that a received resource representation originates from the requested server (T1:3.2, T1:3.8).

R3.2 The client must be able to verify that a received representation is a representation of the resource requested by the client (T1:3.2, T1:3.4, T1:3.8).

R3.3 The client must be able to verify the content format of the representation (T1:3.4).

R3.4 The client must be able to detect that a received representation is fresh (T1:3.9, T3:3.4).

R3.5 The representation must be integrity protected and encrypted from the server to the client (T1:3.2, T1:3.11, T2:3.3, T4:3.1).

R3.6 To protect against the proxy colluding with an authorized client, asymmetric cryptography must be used (T1:3.11).

R3.7 The communication protocol must provide forward secrecy (T4:3.3).

The security non-requirements of the caching scenario are:

NR3.1 The request is not protected (see Security Objectives).

NR3.2 The header and options of the response are not protected (see Security Objectives, compare R3.3).

NR3.3 The proxy may eavesdrop on metadata (including the cache key) or by making requests on behalf of alleged clients (T2:3.1, T4:3.2).

NR3.4 The proxy may drop messages without the endpoint being able to infer that the message is lost due to the proxy (T3:3.1).

NR3.5 The proxy may delay messages without being detected (T3:3.2).

NR3.6 The client may not be able to verify validity information provided by proxy when using ETag (T1:3.10).

3.4. Multiple Requests - Multiple Responses: Observe

This scenario is about replicating a resource state from a server to a client. The client observes a resource and receives notifications which may be cached. The difference compared to the previous scenario (Section 3.3) is the capability to send multiple responses in reply to a single request. The difference compared to Section 3.2 is that in this scenario multiple clients may be served with the same response.

This functionality protects communication-constrained servers from repeated requests, which may come from different clients, when the resource is unchanged. This saves system resources and bandwidth.

In addition to multiple clients' requests being served by one response, each request may result in multiple responses.

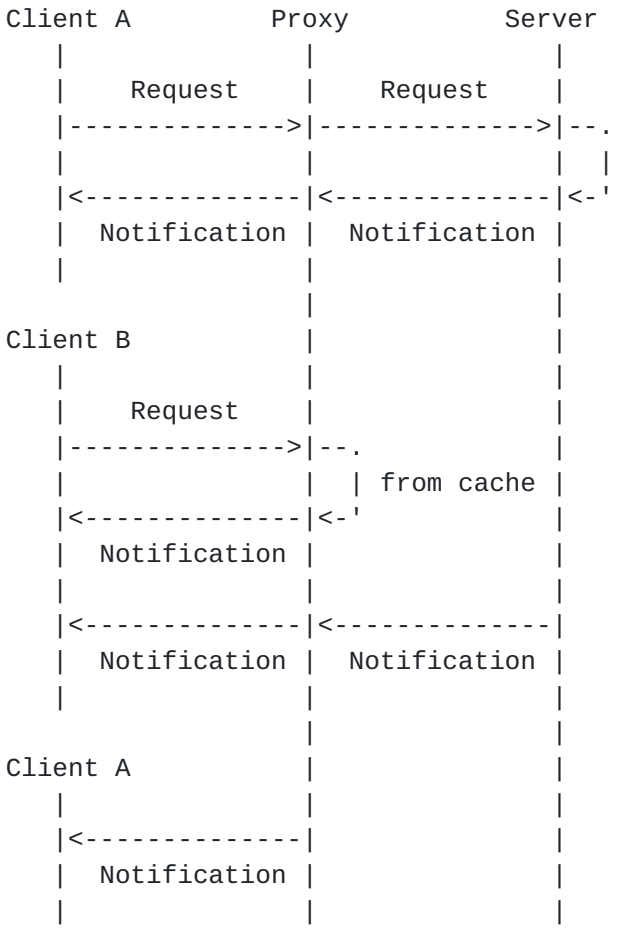

Figure 7: Message Flow for Observe with Multiple Observers

The server exposes an observable resource (e.g., the current reading of a temperature sensor). Multiple clients are interested in the current state of the resource and observe it using the CoAP resource observation mechanism [RFC7641]. The goal is to keep the state observed by the clients closely in sync with the actual state of the resource at the server. Another goal is to minimize the burden on the server by moving the task to fan out notifications to multiple clients from the server to the proxy.

Functional Requirements:

The functional requirements of the previous scenario (Section 3.3) apply, and additionally:

- The proxy must be able to observe a resource on behalf of one or more clients.

- When a client registers interest in a resource with the proxy, the proxy must be able to return a response containing the current state of the resource without contacting the server.

3.4.1. Processing Rules

The proxy complies with the processing rules PR3.1 - 3.5 of the previous scenario (Section 3.3.1). In addition, the following processing rules apply:

PR4.1 If the proxy receives a notification from the server that is out of sequence (as indicated by the Observe option), then the proxy discards the notification. Otherwise, the proxy proceeds to notify the registered observers.

PR4.2 When notifying an observer, the proxy modifies the Observe option to indicate the sequence of notifications from the proxy to the observer.

3.4.2. Security Objectives

The security objectives are identical to the previous scenario.

3.4.3. Threat Analysis and Mitigation

The threat analysis from the previous scenario carries over to this scenario.

3.4.3.1. T1:The proxy illegitimately modifies a message

The same conclusions apply as in the previous scenario (Section 3.3.3.1). For example in T1:3.4, a proxy may maliciously modify the Observe option to indicate a different order of notifications without being detected. However, the mitigation specified in the previous scenario applies: the server integrity protects a freshness parameter with the response.

3.4.3.2. T2:The proxy illegitimately sends a message

The same conclusions apply as in the previous scenario (Section 3.3.3.2).

3.4.3.3. T3:The proxy illegitimately inhibits sending of a message

The same conclusions apply as in the previous scenario (Section 3.3.3.3). The threat in T3:3.4 may be combined with manipulation of the Observe option, but the same mitigation as mentioned in (Section 3.4.3.1) applies.

3.4.3.4. T4:The proxy illegitimately reads part of a message

The same conclusions apply as in the previous scenario (Section 3.3.3.4).

3.4.4. Security Requirements

Since the security objectives and threat mitigations carry over from the previous scenario (Section 3.3), the same security requirements are valid (Section 3.3.4).

3.5. Multiple Requests - Multiple Responses: Publish-Subscribe

The intermediary node in the publish-subscribe scenario is a broker for messages from a publisher to subscriber. A subscriber subscribes to a "topic" and receives a publication. The broker fans out subsequent publications on that topic to all subscribers.

In this scenario a single request may result in multiple responses and a single response may reach multiple clients.

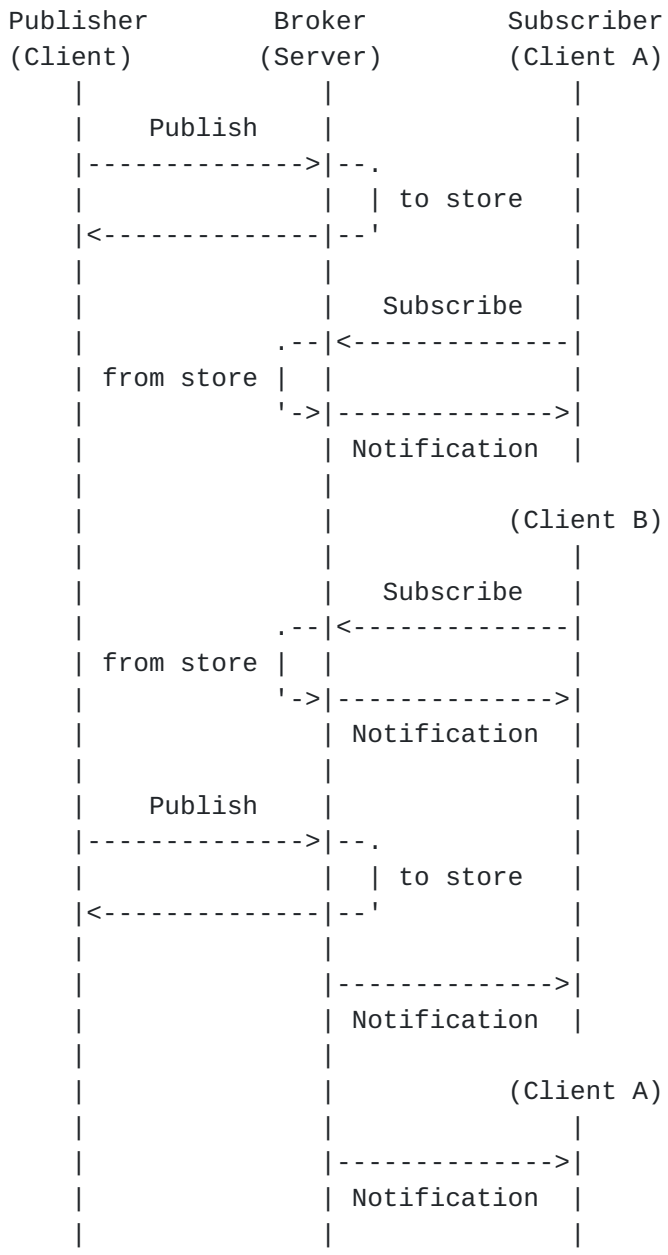

Figure 8: Message Flow for Publish-Subscribe

The broker maintains a number of topics that a publisher can publish to and a subscriber subscribe to. Topics are represented as URIs at the broker. Figure 8 illustrates the publication to a topic, implemented as a PUT request of a representation to a resource at the broker.

Subscribers can make a GET request with the Observe option to the topic URI at the broker in order to initiate the subscription on the topic. The broker provides a notification in the form of a stored representation as response to the request. Further publications of

representations to this URI are provided as notification responses to the subscription request.

Functional Requirement:

- The publication must be able to be transferred in a PUT request from the publisher and in a GET response to the subscriber.

3.5.1. Processing Rules

PR5.1 If the broker receives a subscription request to one of its resources, then the broker associates the requesting subscriber to the topic and responds with the current representation.

PR5.2 If the broker receives a publication request to one of its resources, then the broker stores the received representation on the topic and responds with the representation to the associated subscribers of that topic.

Since we are focusing on end-to-end security between publisher and subscriber, the creation and deletion of topics and other endpoint-to-broker operations are out of scope.

3.5.2. Security Objectives

The security objectives for this scenario are:

SO5.1 A subscriber is able to verify that a received response contains a resource representation of a requested topic, that the publisher is authorized to publish to that topic, and that it has not been altered between publisher and subscriber.

SO5.2 A subscriber is able to verify that a received resource representation is fresh.

SO5.3 The publisher is able to protect a resource representation such that only authorized subscribers can read the representation.

3.5.3. Threat Analysis and Mitigation

We now analyze the potential threats relevant to this scenario.

3.5.3.1. T1:The broker illegitimately modifies a message

T1:5.1 The broker responds to a subscriber request with a publication containing a modified payload

This threat may be mitigated with integrity protection of payload.

T1:5.2 The broker responds to a subscriber request with a publication containing a modified CoAP option

Since the security objective is to protect the resource representation, only options in the GET response that influence the interpretation of the resource representations have an impact. A broker is entitled to change Max-Age and may do so maliciously. The broker is not entitled to change Content-Format, but may anyway do so maliciously. To mitigate these, the subscriber needs to be able to verify information about freshness and content format provided by the publisher.

T1:5.3 The broker responds to a subscriber request with modified CoAP header fields

Since the security objective is to protect the resource representation, only header fields in the GET response that influence the interpretation of the resource representations have an impact. Changing of Code such as e.g. 2.05 (Content) to some error code is a denial-of-service.

T1:5.4 The broker modifies the publication before or during storage

This threat is analogous to the previous threats and is mitigated in the same way.

T1:5.5 The broker responds to a subscriber request with the wrong message

Modifications of payload, options, and header are considered previously. To mitigate wrong a interpretation of a response resulting from a broker sending old messages or reordering messages from the same publisher to the same subscriber, the message may integrity protect a freshness parameter (timestamp, counter, etc.) from which the age/order can be deduced (replay/reordering protection).

T1:5.6 The broker colludes with a legitimate subscriber having access to the key used to create Message Authentication Codes

(MAC) of publications in order to generate a valid MAC of a modified publication

This threat applies to publications containing a message authentication code (MAC) for integrity protecting the resource representation. The threat may be mitigated by the publisher digitally signing the representation with a private key instead of using a MAC.

3.5.3.2. T2:The broker illegitimately sends a message

T2:5.1 The broker sends a response to a subscriber request without a previous publication from the publisher

Most cases of responding inappropriately to a subscriber request are covered in the previous section. In general, authentication of publisher in combination with replay/ reordering protection will mitigate this threat.

T2:5.2 A broker sends a number of messages for the purpose of flooding the subscriber

By verifying the integrity and freshness information, the subscriber may mitigate certain flooding attacks.

3.5.3.3. T3:The broker illegitimately inhibits sending of a message

T3:5.1 The broker does not store or forward a publication

This is a denial-of-service attack. While these threats may be difficult to mitigate, missing messages are common in lossy environments so applications should have a readiness for this kind of issue.

T3:5.2 The broker does not respond to a publication request

This may be a denial-of-service attack on the publisher. While such a threat may be difficult to mitigate, missing messages are common in lossy environments so applications should have a readiness for this kind of issue.

T3:5.3 The broker delays forwarding of a received publication

Delayed forwarding may be a denial-of-service attack, similar to not forwarding. Certain delays may be legitimate, so it may be difficult to detect and mitigate.

T3:5.4 The broker reorders the publications

This threat may be mitigated by the publisher integrity protecting the message and including a freshness parameter.

3.5.3.4. T4:The broker illegitimately reads part of a message

T4:5.1 The broker reads a representation/payload

This threat can be mitigated with encryption of the representation and other potential payload data

T4:5.2 The broker infers information about the nature and state of the publication from CoAP options and header fields.

This metadata is not in scope of confidentiality. Information leaking that can be inferred from such data cannot be prevented.

T4:5.3 The broker reads and stores all publications and can deduce information about the entire history of the publications and subscriptions

Since the protection of metadata related to subscription and publication is not in scope of the security objectives, the mitigation is restricted to encrypting the resource representations. The case of long term key compromise would nevertheless reveal the history of a publication, but this can be mitigated with forward secrecy.

3.5.4. Security Requirements

This section contains the security requirements and non-requirements for the publish-subscribe scenario. For each requirement and non-requirement the associated threats are listed. The security requirements are:

R5.1 The subscriber must be able to verify that a received resource representation originates from an authorized publisher (T1:5.1, T2:5.1).

R5.2 The subscriber must be able to verify that a received representation is a representation of the resource requested by the subscriber (T1:5.1, T1:5.4, T1:5.5, T1:5.6).

R5.3 The subscriber must be able to verify the content format of the representation (T1:5.2)

R5.4 The subscriber must be able to detect that the received resource representation is older than a previously received representation of this resource (T1:5.5, T2:5.1, T3:5.4).

R5.5 The representation must be integrity protected and encrypted from publisher to subscriber (T1:5.1, T1:5.5, T2:5.2, T4:5.1).

R5.6 To protect against the proxy colluding with an authorized subscriber, asymmetric cryptography must be used (T1:5.6).

R5.7 The communication protocol must provide forward secrecy (T4:5.3).

The security non-requirements of the pub-sub scenario are:

NR5.1 The subscription request is not protected (see Security Objectives).

NR5.2 The header and options of the notification response are not protected (see Security Objectives, compare R5.3).

NR5.3 The broker may change and eavesdrop on certain metadata without being detected (T1:5.2, T1:5.3, T4:5.2).

NR5.4 The broker may drop messages without being detected (T3:5.1, T3:5.2).

NR5.5 The broker may delay messages without being detected (T3:5.3).

4. Security Considerations

A proxy or intermediary may be an aggregation point for message flows. Therefore it is an attractive target, both from a security and privacy point of view.

Unless the security mechanisms provide forward secrecy, a compromise of long term keying material means that an attacker can decrypt all previously sent information and can be directly used for any kind of manipulation of the cyber-physical system.

Therefore the key exchange mechanism used for establish keys to use with application layer security must provide forward secrecy.

Intermediary nodes are aggregation points also for metadata and therefore valuable targets for signal intelligence agencies. Pervasive monitoring is an attack [RFC7258] and the effect of collecting and correlating information from multitude of proxies must be mitigated.

Related to this, it is needed to delete all historical information from all nodes handling the plaintext data and metadata, in order to avoid information leakage. The impact of this on the intermediary nodes can be limited by confidentiality protecting as much as possible between the endpoints.

5. IANA Considerations

This document includes no request to IANA.

6. References

6.1. Normative References

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <http://www.rfc-editor.org/info/rfc7258>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <http://www.rfc-editor.org/info/rfc7641>.

6.2. Informative References

[I-D.ietf-cose-msg] Schaad, J., "CBOR Encoded Message Syntax", draft-ietf-cose-msg-10 (work in progress), February 2016.

[I-D.koster-core-coap-pubsub] Koster, M., Keranen, A., and J. Jimenez, "Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)", draft-koster-core-coap-pubsub-04 (work in progress), November 2015.

[I-D.mattsson-core-coap-actuators] Mattsson, J., Fornehed, J., Selander, G., and F. Palombini, "Controlling Actuators with CoAP", draft-mattsson-core-coap-actuators-00 (work in progress), October 2015.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <http://www.rfc-editor.org/info/rfc7228>.

Acknowledgments

The authors wish to thank John Mattsson and Ari Keranen for reviewing early versions of the draft.

Authors' Addresses

Goeran Selander
Ericsson AB
SE-164 80 Stockholm
Sweden

Email: goran.selander@ericsson.com

Francesca Palombini
Ericsson AB
SE-164 80 Stockholm
Sweden

Email: francesca.palombini@ericsson.com

Klaus Hartke
Universitaet Bremen TZI
Postfach 330440
Bremen 28359
Germany

Phone: +49-421-218-63905
Email: hartke@tzi.org

Ludwig Seitz
SICS Swedish ICT AB
Scheelevaegen 17
Lund 223 70
Sweden

Email: ludwig@sics.se