Object Security in Web of Things

John Mattsson, Göran Selander, Göran AP Eriksson, Ericsson Research

Abstract. The Web of Things brings new challenges that cannot be solved in a satisfactory way with only transport layer security. A more flexible solution is required, both to protect sensitive data and user privacy but also to distribute policies in a secure and standardized way. In this position paper we propose a base architecture and examine candidate open standards to provide security and privacy on the application layer. While this paper has an IoT focus, the same privacy problems arise in the general web setting with processing and storage more and more moving into the cloud.

1. Introduction

It is common to think about the Internet of Things from the perspective of sensors and transport protocols, but you can also think about it from the point of view of applications and services, where most of the impact on people and companies will be.

Continuing advances in electronics have dramatically reduced the cost for devices functioning as tags, sensors, and actuators for the physical environment, i.e. the Internet of Things (IoT). The market potential for the IoT is currently held back by fragmentation due to a plethora of communication technologies and the lack of a common approach to enabling services.

In this position paper we examine candidate open standards to provide security and privacy on the application layer to services, between devices, and at the network edge, e.g. in home hubs, or in the cloud. We demonstrate the use of web protocols for implementing services, the need for APIs for implementing IoT technologies, a shared approach to describing services as a basis for interoperability, and the underlying use of HTTP/CoAP and JSON for efficient RESTful services.

2. Background

On a high level the problem addressed is about authorization of access to data, in particular data processed in an IoT setting, e.g. sensor and actuator data or information derived from such data. Authorization and access control in IoT are highly relevant e.g. by providing the means to establish trust and is thus a business enabler. One particular important aspect of authorization is privacy. In IoT the data is closely related to the physical world and may reveal information about the people in the vicinity of the sensor, especially when combined with other information. Moreover, an actuator by definition impacts the physical world and could have a very concrete effect on people and things in its surrounding.

Although various techniques are available, the most common approach for data protection in the Internet is based on transport layer security and trust in the information storage. When talking about IoT, it is important to understand that there are many different IoT settings. Use cases such as the smart grid, connected car etc. target relatively powerful devices directly connected to the Internet and/or the power grid. Another category of devices such as building and home automation, industrial control systems etc. may be more constrained in nature and of higher cardinality. Constrained devices are not expected to disappear as Moore's law is used to drive down cost instead of improving performance, thereby enabling new business cases. Moreover, some of these devices have long life times and may not be replaced e.g. due to being built-in.

We should therefore have constrained devices in scope although we are designing services potentially to be run on non-constrained nodes. This also has the benefit that the mechanisms we design must be efficient; leaving more resources to non-security related processing.

3. Base architecture

The IETF has done work on authorization such as the web authorization protocol, OAuth 2.0 [3], but these protocols are not designed with IoT in mind. A recent IETF initiative, the Authentication and authorization in Constrained Environments (ACE) [1] is specifically addressing authorization in IoT. Although the work in ACE is still in an early phase some initial architecture discussion have proposed to reuse parts of the architecture, terminology and the RESTful paradigm from OAuth. The roles in this setup are:

- resource server (RS), hosting the resources, such as sensors and actuators of the IoT
- client (C), requesting access to data, e.g. resource representations
- resource owner (RO), owning resources in resource servers, and
- authorization server (AS), acting on behalf of RO in managing the access policies set by RO and/or making authorization decisions based on these policies.

What do not carry over from existing web authorization are in particular the actual protocols. In the target setting C and RS may be constrained, in which case HTTP/TCP is too heavyweight and protocols like CoAP/UDP are better suited. The RO does not need to be active in an M2M setting, so interactions with the RO needs to be replaced with interactions via policies used by the AS. Also, for constrained devices the message exchange should be reduced to a minimum to cut down on energy consumption in battery powered devices.

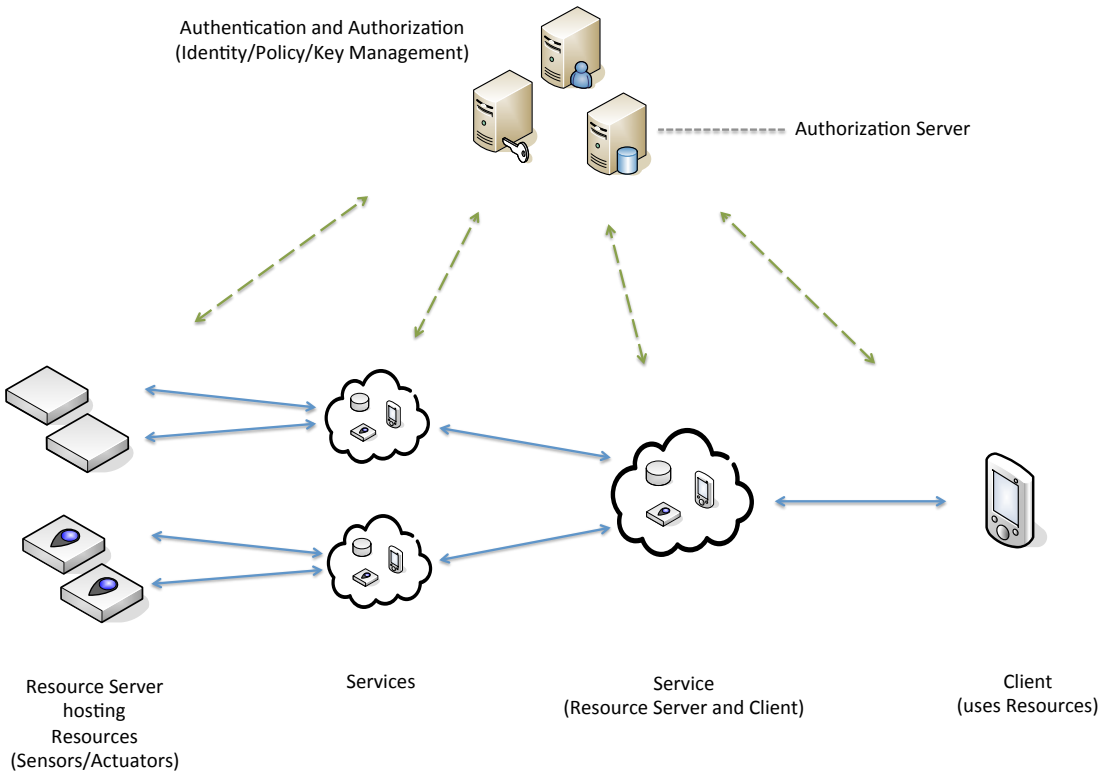

Figure 1: Base Architecture

Another important component in addressing the expected deployment and potential services in this area is to allow services between the resources and the clients performing operations on data, metadata, or the data representation. Other types of services may filter, store, or cache data as discussed in the next section. Services are clients from the point of view of the resource server and resource servers from the point of view of the client. There is also a need for more management nodes than AS, handling identity management, key management, and privacy policies. The architecture is illustrated in Figure 1.

4. Use cases and problem description

When looking at the information flow in different IoT services, there are several cases where multiple clients need access to the same resources, potentially with different access rights. There are also cases where a client and server cannot set up secure transport end to end, or where there is a need to cache data in a protected environment [5] while maintaining the access restrictions. We now look at some use cases to illustrate this.

One case deals with multiple clients with different privileges. For example, a smart meter providing data about energy usage, line quality aspects (meta-data), etc. Since energy consumption is privacy sensitive, access should be restricted on a need-to-know basis. The local energy provider and the tenant should be able to get information about energy usage, but at different granularity. The local energy provider only need information to allow accurate charging, whereas the home owner could benefit from higher precision. Assuming an unbundled setting, where the energy provider may be different from the distribution network operator, the latter should ideally only have access to metadata and total consumption. All others parties should not have any access at all.

Another case deals with multiple endpoints for the same information, but which is accessed potentially at different times and/or with different privileges, e.g. a publish-subscribe setting. This is supported in CoAP by the Observe notification architecture [4] and in Information Centric Networking[1]. In this case caching must be supported in nodes not authorized to access the content. A reverse example is configuration of large sets of devices with intermittent connectivity, where each device could request protected configuration data from one of multiple caches when it has connectivity, download, verify the data and perform the configuration.

We conclude from the use cases the following requirements:

- different rights for different clients to access resources
- confidentiality of the information objects should be guaranteed from the sensor/actuator to the different clients without having to put trust in services.
- compact representation of data and lightweight protocols
- support for caching
- based on open standards

5. Object security

As should be clear from the requirements, transport layer security is not sufficient. TLS/DTLS only supports trust models with fully trusted endpoints. Multiple end-to-end transport security sessions for the different clients and multiple handshakes would be heavy for constrained resource servers and is a very inflexible solution. HTTP supports TLS tunneled through a proxy, but CoAP does not, so DTLS must be applied hop-by-hop breaking the end-to-end security. Finally, caching to handle intermittent connectivity or time separated requests of the same data is not compatible with the transport security paradigm.

As a complementary approach we now illustrate some of the benefits with security at the application layer, the most natural candidate in this respect is some form of object security.

With “secure object” we refer to a self-contained information container with protected content that need not be associated to a session, containing for example sensor data (e.g. resource representations), policy data (e.g. authorization information), or meta-data (e.g. context information). A secure object consists essentially of a header (metadata), a payload (potentially encrypted) and an integrity verification tag, calculated on a certain part of the payload and header. Compare with S/MIME or PGP for secure email, but in our case both symmetric and asymmetric algorithms need to be supported as well as a more flexible wrapping for different clients and a more compact representation supporting constrained environments. There may be

[1] Examples of ICN approaches include CCN/NDN (ccnx.org) and NetInf (netinf.org)

multiple objects sent in one message or alternatively, different parts of the message can be protected for different clients.

One advantage of using object security is to be able to serve multiple clients with the same object. Resource data is protected at the source (resource, client, service or management server) and wrapped into a secure object, which can be published for any authorized users to access. Another advantage is that this concept clearly supports caching.

Considering different privileges of different clients, data need to be wrapped at the resource in accordance with the policies. Authorization information (either policy or policy decision) applicable to the clients is generated by the authorization server and thus needs to be transferred to the resource server (cf. OAuth Access Token [3]). If the secure object format is well designed, the same secure object format is applicable, which means that the constrained device can compactly implement parsing of data and policies.

A third and important advantage is that the trust model can be implemented on the application layer which provides a much more flexible solution and shorter time to market.

How data is represented and protected depends on the type of data and the use case. For text sent as JSON it is natural to use JOSE [2], while binary data such as real-time media may be best served by other formats. The use case determines whether to use symmetric or asymmetric keys, if confidentiality, integrity and/or replay protection is needed and which parties get access to different keys.

6. Example

We now give an example of security at the application layer (object security), e.g. using JOSE combined with channel security e.g. (D)TLS. This example is applicable to text-based data, which can be formatted as JSON and protected with JOSE.

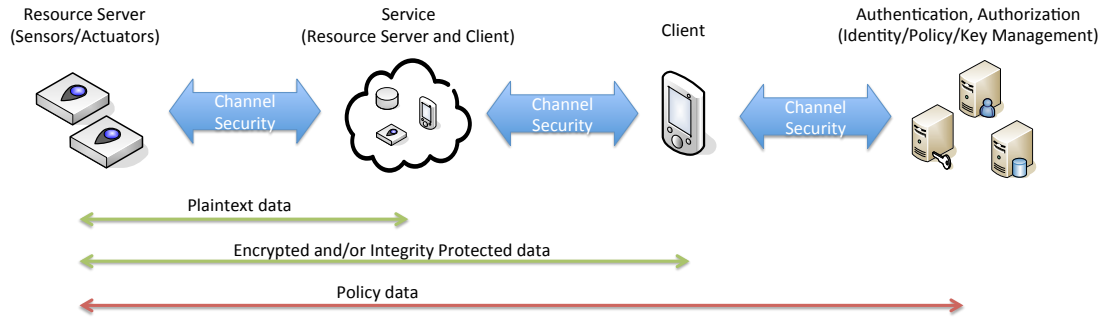

Figure 2: Object Security Example

We assume that all data is protected by hop-by-hop transport security (e.g. using DTLS). This level of protection is enough for some types of data (e.g. metadata intended for the other DTLS endpoint).

More sensitive data such as sensor data, actuator commands, or policy data are protected on the application layer by JOSE. Integrity protected but unencrypted data can be read but not changed by services. This enables services collecting and storing data to process the data and to calculate averages or accumulated change while still ensuring end-to-end integrity.

The most sensitive data can be both integrity protected and encrypted. By doing this on the application layer, services can store, cache, and filter (based on unencrypted metadata) without having access to the key, something that is not possible with transport layer protection.

In more complicated use cases with several service intermediaries, the endpoints for the JOSE containers might differ between objects. A combination of transport layer security such as DTLS and object security such as JOSE gives a very flexible framework that can be adapted to a large

number of possible use cases and trust models. In most cases the architecture and protocols used in the service could be used in the Resource Server as well, but for legacy or proprietary sensor technologies the service needs to act as the Resource Server and format everything into compatible objects.

7. Position and further work

The web of things brings new challenges that cannot be solved in a satisfactory way with only transport layer security. We strongly believe that an architecture with services (operated in the cloud or by third party provides) requires a more flexible solution, both to protect sensitive data and user privacy but also to distribute policies in a secure and standardized way.

Many of the needed pieces are already available but some are missing and W3C should in collaboration with other standardization bodies (e.g. those specifying interfaces to devices) take action to drive standardized ways for secure handling of data and policies in the web of things. The standards should be flexible as to support the many ways the web is used, e.g. by supporting both text and binary, asymmetric and symmetric crypto, media and data objects. W3C should:

- Develop standards and best practices for object security including
    - JSON text data, e.g. using the whole or a profile of JOSE
    - Binary data and other compact and legacy data formats
    - How to identify various encrypted information objects and bind metadata to them, e.g. using JSON-LD.
- Multiparty protocol for secure exchange of information objects, metadata, identities of the information objects and endpoints, key management, etc.
- Interoperable scalable formats for policies syntax, semantics
- Management and debugging of large sets of policy information
- Access control in general, privacy more specifically
- Browsers need APIs for key management, object encryption and decryption, object manipulation etc.

While this paper has an IoT focus, the same problems arise in the general web setting with processing and storage more and more moving into the cloud. While object security has mostly been used to protect streaming media, protecting sensitive user data uploaded and stored in the cloud posses the same problems and could potentially use the same APIs [6]. To protect against pervasive monitoring in a cloud setting, the browser must provide keys and do the encryption. To protect user privacy also metadata currently send in headers such as the service URL needs to be protected in a secure object.

References

[1] IETF Authentication and Authorization for Constrained Environments (ACE)
https://datatracker.ietf.org/doc/charter-ietf-ace/

[2] IETF Javascript Object Signing and Encryption (JOSE)
https://datatracker.ietf.org/wg/jose/charter/

[3] The OAuth 2.0 Authorization Framework
http://tools.ietf.org/html/rfc6749

[4] Constrained Application Protocol (CoAP)
http://tools.ietf.org/html/draft-ietf-core-coap

[5] CoRE Mirror Server
http://tools.ietf.org/html/draft-vial-core-mirror-proxy

[6] Encrypted Media Extensions
http://www.w3.org/TR/encrypted-media/