ACE Working Group
Internet-Draft
Intended status: Informational
Expires: September 26, 2015

S. Gerdes
Universitaet Bremen TZI
L. Seitz
SICS Swedish ICT AB
G. Selander
Ericsson
C. Bormann, Ed.
Universitaet Bremen TZI

March 25, 2015

Actors in the ACE Architecture
draft-gerdes-ace-actors-04

Abstract

Constrained nodes are small devices which are limited in terms of processing power, memory, non-volatile storage and transmission capacity. Due to these constraints, commonly used security protocols are not easily applicable. Nevertheless, an authentication and authorization solution is needed to ensure the security of these devices.

Due to the limitations of the constrained nodes it is especially important to develop a light-weight security solution which is adjusted to the relevant security objectives of each participating party in this environment. Necessary security measures must be identified and applied where needed.

This document gives an overview of the necessary terminology and introduces the actors in an architecture as guidance for the development of authentication and authorization solutions for constrained environments. The actors represent the relationships between the logical functional entities involved.

We also present a problem description for authentication and authorization in constrained-node networks, i.e. networks where some devices have severe constraints on memory, processing, power and communication bandwidth.

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 26, 2015.

Table of Contents

1. Introduction

Constrained nodes are small devices with limited abilities which in many cases are made to fulfill a single simple task. They have limited hardware resources such as processing power, memory, non-volatile storage and transmission capacity and additionally in most cases do not have user interfaces and displays. Due to these constraints, commonly used security protocols are not always easily applicable.

Constrained nodes are expected to be integrated in all aspects of everyday life and thus will be entrusted with vast amounts of data. Without appropriate security mechanisms attackers might gain control over things relevant to our lives. Authentication and authorization mechanisms are therefore prerequisites for a secure Internet of Things.

The limitations of the constrained nodes ask for security mechanisms which take the special characteristics of constrained environments into account. Therefore, it is crucial to identify the tasks which must be performed to meet the security requirements in constrained scenarios. Moreover, these tasks need to be assigned to logical functional entities which perform the tasks: the actors in the architecture. Thus, relations between the actors and requirements for protocols can be identified.

In this document, an architecture is developed to represent the relationships between the logical functional entities involved.

1.1. Terminology

Readers are required to be familiar with the terms and concepts defined in [RFC4949].

In addition, this document uses the following terminology:

Resource (R): an item of interest which is represented through an interface. It might contain sensor or actuator values or other information.

Constrained node: a constrained device in the sense of [RFC7228].

Actor: A logical functional entity that performs one or more tasks. Depending on the tasks an actor must perform, the device that contains the actor may need to have certain system resources available. Multiple actors may share, i.e. be present within, a device or even a piece of software.

Resource Server (RS): An entity which hosts and represents a Resource.

Client (C): An entity which attempts to access a resource on a Server.

Resource Owner (RO): The principal that is in charge of the resource and controls its access permissions.

Requesting Party (RqP): The principal that is in charge of the Client and controls permissions concerning authorized representations of a Resource.

Principal: An individual that is either RqP or RO or both.

Authorization Server (AS): An entity that prepares and endorses authentication and authorization data for a Server.

Client Authorization Server (CAS): An entity that prepares and endorses authentication and authorization data for a Client.

Attribute Binding Authority: An entity that is authorized to validate claims about an entity.

2. Problem Statement

The scenario this document addresses can be summarized as follows:

- C wants to access R on a RS.

- A priori, C and RS do not necessarily know each other and have no security relationship.

- C and / or RS are constrained.

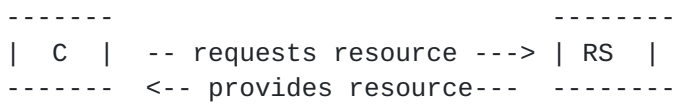

Figure 1: Basic Scenario

The security requirements of any specific version of this scenario will include one or more of:

- Rq0.1: No unauthorized entity has access to (or otherwise gains knowledge of) R.

- Rq0.2: C is exchanging status updates of a resource only with authorized resources. (When C attempts to access R, that access reaches an authorized R).

Rq0.1 requires authorization on the server side while Rq0.2 requires authorization on the client side.

3. Security Objectives

The security objectives that can be addressed by an authorization solution are confidentiality and integrity. Additionally, allowing only selected entities limits the burden on system resources, thus helping to achieve availability. Misconfigured or wrongly designed authorization solutions can result in availability breaches: Users might no longer be able to use data and services as they are supposed to.

Authentication mechanisms can achieve additional security objectives such as non-repudiation and accountability. They are not related to authorization and thus are not in scope of this draft, but still should be considered by Authenticated Authorization solutions. Non-repudiation and accountability may require authentication on device level, if it is necessary to determine which device performed an action. In other cases it may be more important to find out who is responsible for the device's actions.

The importance of a security objective depends on the application the authorization mechanism is used for. [I-D.ietf-ace-usecases] indicates that security objectives differ for the various constrained environment use cases.

In many cases, one participating party might have different security objectives than the other. However, to achieve a security objective, both parties must participate in providing a solution. E.g., if RqP requires the integrity of sensor value representations RS is hosting, Both C and RS need to integrity-protect the transmitted data. Moreover, RS needs to protect the access to the sensor representation to prevent unauthorized users to manipulate the sensor values.

4. Authentication and Authorization

Authorization solutions aim at protecting the access to items of interest, e.g. hardware or software resources or data: They enable the principal of such a resource to control who can access it and how.

To determine if an entity is authorized to access a resource, an authentication mechanism is needed. According to the Internet Security Glossary [RFC4949], authentication is "the process of verifying a claim that a system entity or system resource has a certain attribute value." Examples for attribute values are the ID of a device, the type of the device or the name of its owner.

The security objectives the authorization mechanism aims at can only be achieved if the authentication and the authorization mechanism work together correctly. We use the term _authenticated authorization_ to refer to a synthesis of mechanism for authentication and authorization.

If used for authorization, the authenticated attributes must be meaningful for the purpose of the authorization, i.e. the authorization policy grants access permissions based on these attributes. If the authorization policy assigns permissions to an individual entity, the authenticated attributes must be suitable to uniquely identify this entity.

In scenarios where devices are communicating autonomously there is less need to uniquely identify an individual device. For a principal, the fact that a device belongs to a certain company or that it has a specific type (e.g. light bulb) is likely more important than that it has a unique identifier.

Principals (RqP and RO) need to decide about the required level of granularity for the authorization, ranging from _device authorization_ over _owner authorization_ to _binary authorization_ and _unrestricted authorization_. In the first case different access permissions are granted to individual devices while in the second case individual owners are authorized. If binary authorization is used, all authenticated entities have the same access permissions.

Unrestricted authorization for an item of interest means that no authorization mechanism is used (not even by authentication) and all entities are able to access the item as they see fit. More fine-grained authorization does not necessarily provide more security. Principals need to consider that an entity should only be granted the permissions it really needs to ensure the confidentiality and integrity of resources.

For all cases where an authorization solution is needed (all but Unrestricted Authorization), the authorizing party needs to be able to authenticate the party that is to be authorized. Authentication is therefore required for messages that contain representations of an accessed item. More precisely, the authorizing party needs to make sure that the receiver of a message containing a representation, and the sender of a message containing a representation are authorized to receive and send this message, respectively. To achieve this, the integrity of these messages is required: Authenticity cannot be assured if it is possible for an attacker to modify the message during transmission.

In some cases, only one side (only the client side or only the server side) requires the integrity and / or confidentiality of a resource value. In these cases, principals may decide to use binary authorization which can be achieved by an authentication mechanism or even unrestricted authorization where no authentication mechanism is required. However, as indicated in Section 3, the security objectives of both sides must be considered. The security objectives of one side can often only be achieved with the help of the other side. E.g., if the server requires the confidentiality of a resource representation, the client must make sure that it does not send resource updates to parties other than the server. Therefore, the client must at least use binary authorization.

5. Autonomous Communication

The use cases defined in [I-D.ietf-ace-usecases] demonstrate that constrained devices are often used for scenarios where their principals are not present at the time of the communication. Moreover, these devices often do not have any user interfaces or displays. Even if the principals are present at the time of access, they may not be able to communicate directly with the device. The devices therefore need to be able to communicate autonomously. In some scenarios there is an active user at one endpoint of the communication. Other scenarios ask for true machine to machine (M2M) communication.

To achieve the principals' security objectives, the devices must be enabled to enforce the security policies of their principals.

6. Actors

This section describes the various actors in the architecture. An actor consists of a set of tasks and additionally has an security domain (client domain or server domain) and a level (constrained, principal, less-constrained). Tasks are assigned to actors according to their security domain and required level.

Note: Actors are a concept to understand the security requirements for constrained devices. The architecture of an actual solution might differ as long as the security requirements that derive from the relationship between the identified actors are considered. Several actors might share a single device or even be combined in a single piece of software. Interfaces between actors may be realized as protocols or be internal to such a piece of software.

6.1. Constrained Level Actors

As described in the problem statement (see Section 2), either C or RS or both of them may be located on a constrained node. We therefore define that C and RS must be able to perform their tasks even if they are located on a constrained node. Thus, C and RS are considered to be Constrained Level Actors.

C performs the following tasks:

- Communicate in a secure way (provide for confidentiality and integrity of messages).
- Validate that an entity is an authorized source for R.
- Securely transmit an access request.

RS performs the following tasks:

- Communicate in a secure way (provide for confidentiality and integrity of messages).
- Validate the authorization of the requester to access the requested resource as requested.
- Securely transmit a response to an access request.

R is an item of interest such as a sensor or actuator value. R is considered to be part of RS and not a separate actor. The device on which RS is located might contain several resources of different ROs. For simplicity of exposition, these resources are described as if they had separate RS.

As C and RS do not necessarily know each other they might belong to different security domains.

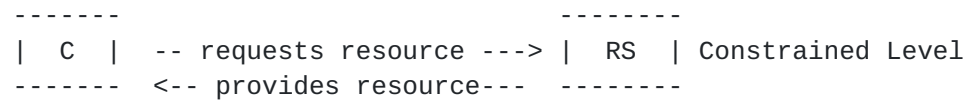

Figure 2: Constrained Level Actors

6.2. Principal Level Actors

Our objective is that C and RS are under control of principals in the physical world, the Requesting Party (RqP) and the Resource Owner (RO) respectively. The principals decide about the security policies of their respective endpoints and belong to the same security domain.

RqP is in charge of C, i.e. RqP specifies security policies for C, e.g. with whom C is allowed to communicate. By definition, C and RqP belong to the same security domain.

RqP must fulfill the following task:

o Configure for C authorization information for sources for R.

RO is in charge of R and RS. RO specifies authorization policies for R and decides with whom RS is allowed to communicate. By definition, R, RS and RO belong to the same security domain.

RO must fulfill the following task:

o Configure for RS authorization information for accessing R.

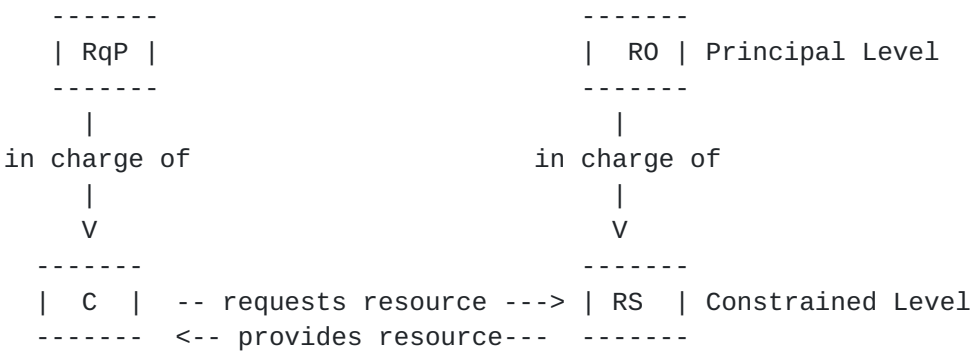

Figure 3: Constrained Level Actors and Principal Level Actors

6.3. Less-Constrained Level Actors

Constrained level actors can only fulfill a limited number of tasks and may not have network connectivity all the time. To relieve them from having to manage keys for numerous endpoints and conducting computationally intensive tasks, another complexity level for actors is introduced. An actor on the less-constrained level belongs to the same security domain as its respective constrained level actor. They also have the same principal.

The Client Authorization Server (CAS) belongs to the same security domain as C and RqP. CAS acts on behalf of RqP. It assists C in authenticating RS and determining if RS is an authorized source for R. CAS can do that because for C, CAS is the authority for claims about RS.

CAS performs the following tasks:

- Validate on the client side that an entity has certain attributes.
- Obtain authorization information about an entity from C's principal (RqP) and provide it to C.
- Negotiate means for secure communication to communicate with C.

The Authorization Server (AS) belongs to the same security domain as R, RS and RO. AS acts on behalf of RO. It supports RS by authenticating C and determining C's permissions on R. AS can do that because for RS, AS is the authority for claims about C.

AS performs the following tasks:

- Validate on the server side that an entity has certain attributes.
- Obtain authorization information about an entity from RS' principal (RO) and provide it to RS.
- Negotiate means for secure communication to communicate with RS.

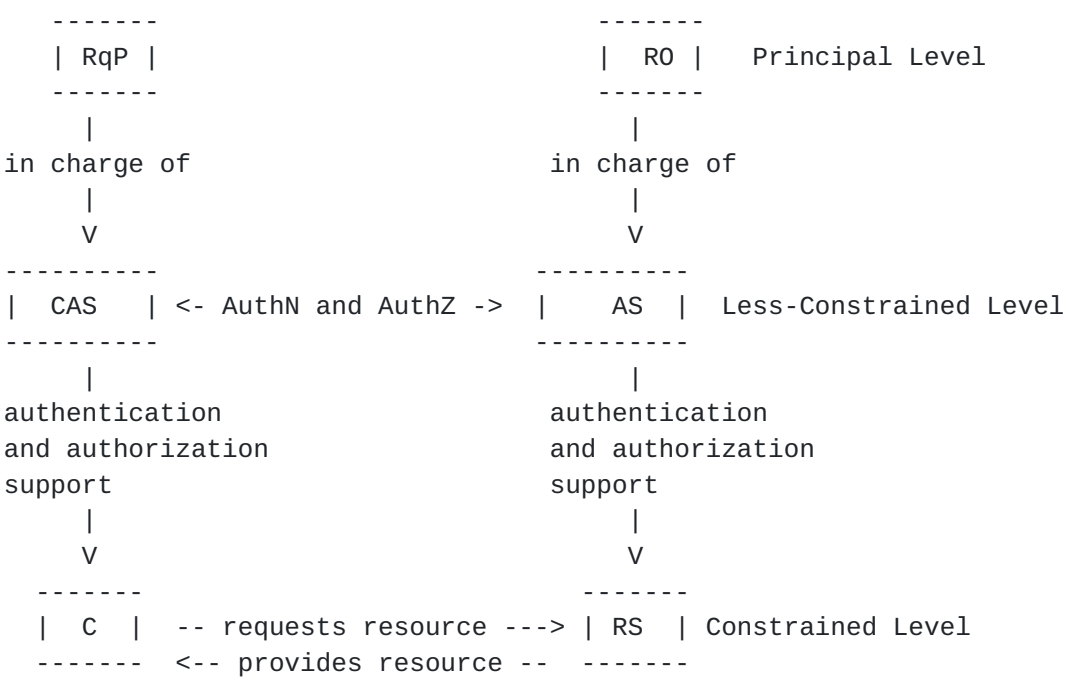

Figure 4: Overview of all Complexity Levels

For more detailed graphics please consult the PDF version.

7. Architecture Variants

As mentioned in section Section 6, actors can share a single device or even be combined in a single piece of software. If C is located on a more powerful device, it can be combined with CAS:

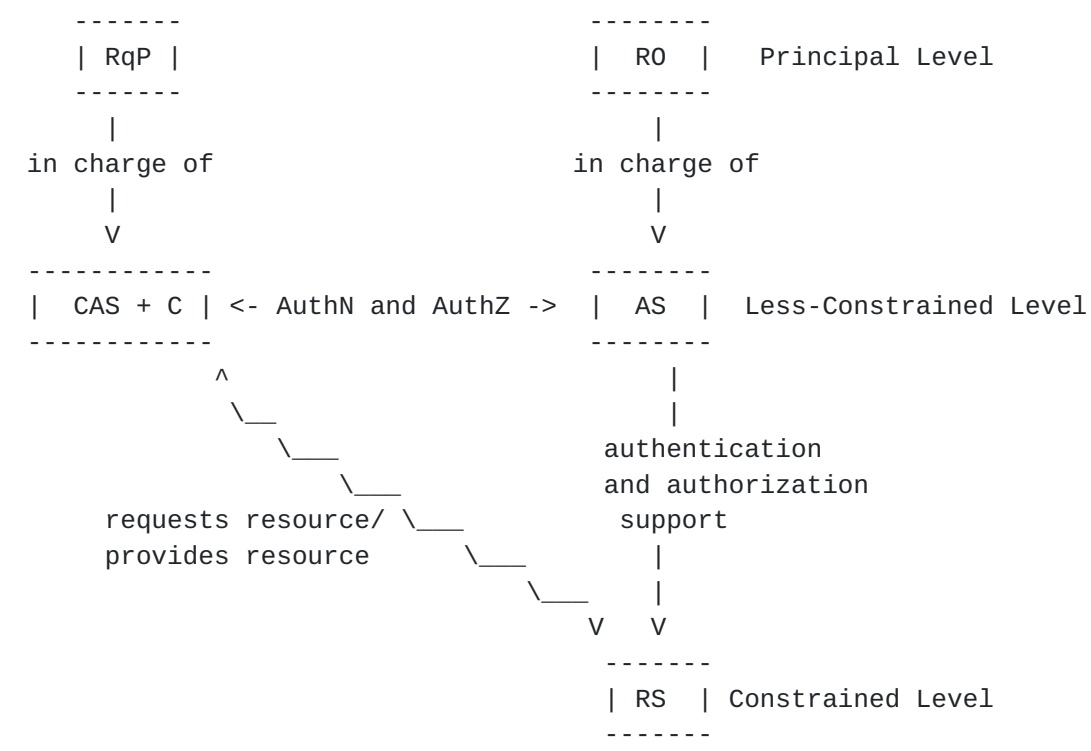

Figure 5: Combined C and CAS

If RS is located on a more powerful device, it can be combined with AS:

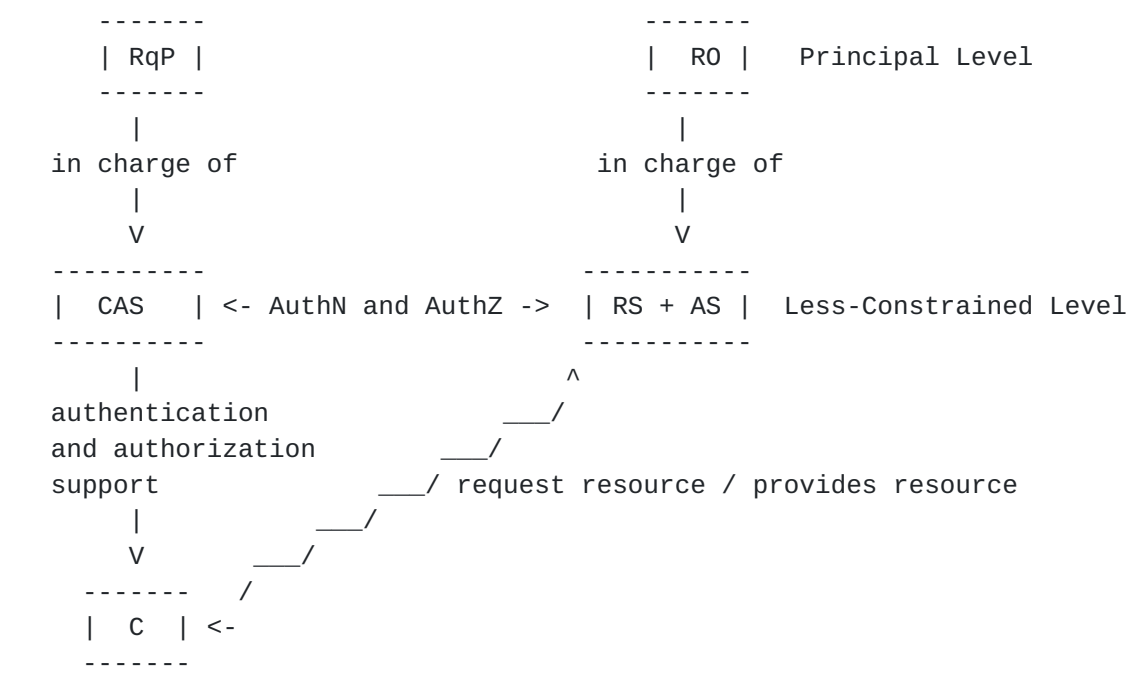

Figure 6: Combined AS and RS

If C and RS have the same principal, CAS and AS can be combined.

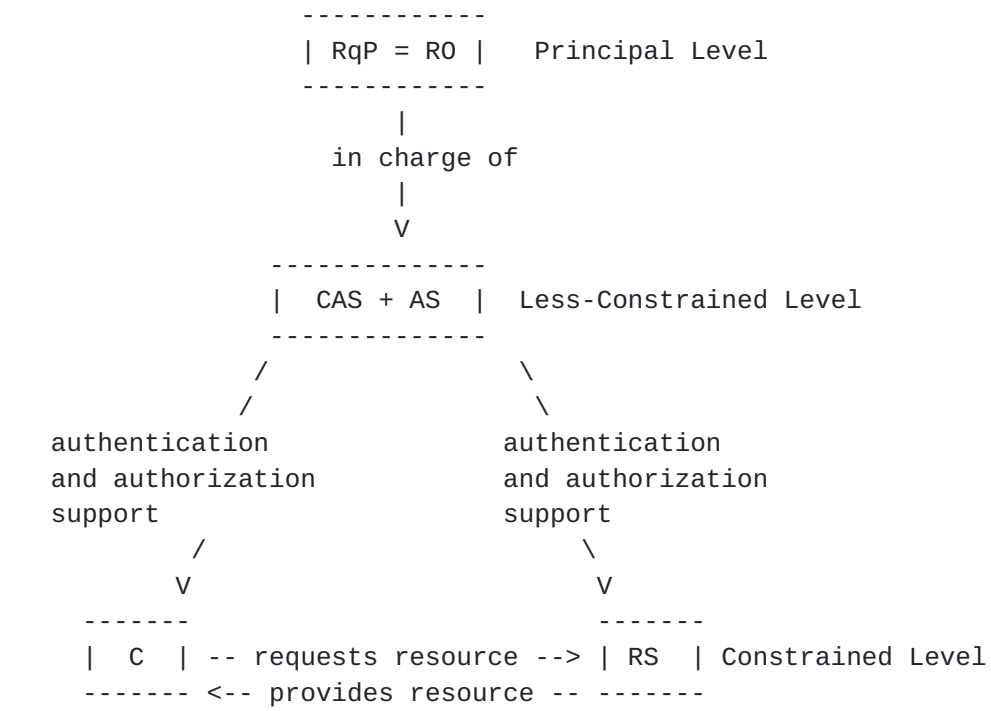

Figure 7: CAS combined with AS

8. Kinds of Protocols

Devices on the less-constrained level potentially are more powerful than constrained level devices in terms of processing power, memory, non-volatile storage. This results in different characteristics for the protocols used on these levels.

8.1. Constrained Level Protocols

A protocol is considered to be on the constrained level if it is used between the actors C and RS which are considered to be constrained (see Section 6.1). C and RS might not belong to the same security domain. Therefore, constrained level protocols need to work between different security domains.

FIXME

Figure 8: Constrained Level Tasks

Commonly used Internet protocols can not in every case be applied to constrained environments. In some cases, tweaking and profiling is required. In other cases it is beneficial to define new protocols which were designed with the special characteristics of constrained environments in mind.

On the constrained level, protocols need to address the specific requirements of constrained environments. Examples for protocols that consider these requirements is the transfer protocol CoAP (Constrained Application Protocol) [RFC7252] and the Datagram Transport Layer Security Protocol (DTLS) [RFC6347] which can be used for channel security.

Constrained devices have only limited storage space and thus cannot store large numbers of keys. This is especially important because constrained networks are expected to consist of thousands of nodes. Protocols on the constrained level should keep this limitation in mind.

8.1.1. Cross Level Support Protocols

Protocols which operate between a constrained device on one side and the corresponding less constrained device on the other are considered to be (cross level) support protocols. Protocols used between C and CAS or RS and AS are therefore support protocols.

Support protocols must consider the limitations of their constrained endpoint and therefore belong to the constrained level protocols.

8.2. Less-Constrained Level Protocols

A protocol is considered to be on the less-constrained level if it is used between the actors CAS and AS. CAS and AS might belong to different security domains.

On the less-constrained level, HTTP [RFC7230] and Transport Layer Security (TLS) [RFC5246] can be used alongside or instead of CoAP and DTLS. Moreover, existing security solutions for authentication and authorization such as the Web Authorization Protocol (OAuth) [RFC6749] and Kerberos [RFC4120] can likely be used without modifications and there are no limitations for the use of a Public Key Infrastructure (PKI).

FIXME

Figure 9: Less-constrained Level Tasks

9. Introduction to Problem Description

Authorization is the process of deciding what an entity ought to be allowed to do. This memo is about properties of security protocols to enable explicit and dynamic authorization of clients to access a resource at a server, in particular in constrained environments when the client and/or server are constrained nodes.

Relevant use cases are provided in [I-D.ietf-ace-usecases], which also lists some authorization problems derived from the use cases. In this memo we present a more specific problem description for authentication and authorization in constrained RESTful environments together with a detailed set of assumptions and requirements (cf. Section 12).

9.1. Terminology

Certain security-related terms are to be understood in the sense defined in [RFC4949]. These terms include, but are not limited to, "authentication", "authorization", "confidentiality", "(data) integrity", "message authentication code", and "verify".

RESTful terms including "resource", "representation", etc. are to be understood as used in HTTP [RFC7231] and CoAP [RFC7252].

Terminology for constrained environments including "constrained device", "constrained-node network", "class 1", etc. are defined in [RFC7228].

"Explicit" authorization is used here to describe the ability to specify in some detail which entity has access to what and under what conditions, as opposed to "implicit" authorization where an entity is either allowed to access everything or nothing.

"Dynamic" authorization means that the access control polices and the parameters on which they are evaluated may change during normal operations, as opposed to "static" authorization meaning that access control policies cannot be changed during normal operations and may require some special procedure such as out-of-band provision.

10. Background

We assume a client-server setting, where a client wishes to access some resource hosted by a server. Such resources may e.g. be sensor data, configuration data, or actuator settings. Thus access to a resource could be by different methods, some of which change the state of the resource. In this memo, we consider the REST setting i.e. GET, POST, PUT and DELETE, and application protocols in scope are HTTP [RFC7231] and CoAP [RFC7252].

We assume that the roles of client and server are not fixed, i.e. a node which is client could very well be server in some other context and vice-versa. Further we assume that in some cases, clients are not previously known to servers, thus we cannot assume that the server has access control policies specific to that client when the client initiates communication.

Finally we also assume that in a significant number of cases, the server and/or the client are too constrained to handle the evaluation of complex access control policies and related configuration on their own. Many authorization solutions involve a centralized, trusted third party, supporting the client and/or resource server. A trusted third party provides a more scalable way to centrally manage authorization policies, in order to ensure consistent authorization decisions. The physical separation of policy decision and policy enforcement is an established principle in policy based management, e.g. [RFC2748].

Borrowing from OAuth 2.0 [RFC6749] terminology we name the entities: client (C), resource server (RS), authorization server (AS - the third party), and resource owner (RO). RO is in charge of the access control policies implemented in the AS governing the actions of RS. However, the RO need not be active in a constrained device access control setting, so we cannot rely on timely interactions with the RO. In the target setting RS is typically constrained, C may be constrained, whereas AS is not assumed to be constrained.

Since RS is constrained, we assume that it needs to offload authorization policy management and/or authorization decision making to AS. This means that some authorization information needs to be transferred from AS to RS.

Protecting information carried between AS and RS, requires some a priori established cryptographic keys. How those keys are established is out of scope for this problem description.

AS may for example be implemented as a cloud service, in a home server, or in a smartphone. C and RS may or may not have connectivity to AS at the time of the access request, e.g. because they cannot handle multiple, simultaneous connections. Another reason for intermittent connectivity may be that constant connectivity is not affordable (e.g. due to limited battery power, or a sensor mobility business case for which cellular connectivity cost too much or is not available). Obviously, in order for a client request to reach RS there must be connectivity between C and RS, but that could be a short range technology such as Bluetooth, ZigBee, or NFC. Furthermore, if there is not sufficient authorization information about C in RS, and neither C nor RS can access AS, access requests will be denied. Therefore we assume that either C or RS can access AS at some point in time, prior to the client's request.

As a summary, there are potentially three information flows that needs to be protected (see Figure 10):

1. The transfer of authorization information from AS to RS

2. The transfer of cryptographic keys or credentials from AS to RS and C, respectively

3. The access request/response procedure between C and RS

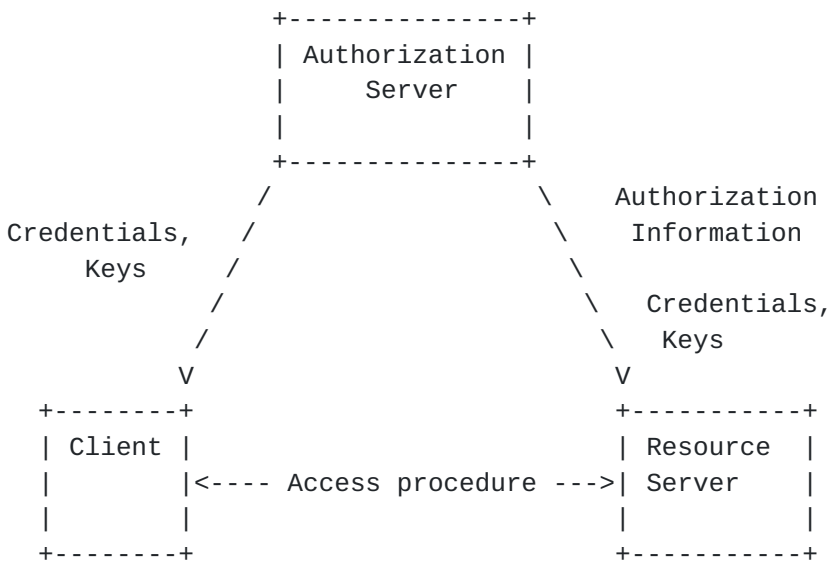

Figure. Information flows that needs to be protected. Only showing origin and destination, actual flow may pass intermediary nodes.

Figure 10

NOTE:

The information flow in Figure 10 above enables RO to control the interactions of a constrained RS by means of access control policies. There is an ongoing discussion about an analogous information flow enabling the stakeholder associated to C ("Requesting Party" in UMA terminology [I-D.hardjono-oauth-umacore]) to control the interactions of a constrained C by means of policies. While this would not be policies for access control to resources, it could be useful in certain settings which require dynamically changing interaction patterns with a constrained client without updating firmware. Such a solution could potentially reuse all security components required to protect the information flow in 1., so no additional specifications would be needed. This aspect is not discussed further in this draft.

11. Problem Description

A number of problems needs to be solved in order to achieve explicit and dynamic authorization, as is described in this section.

11.1. Authorization

The core problem we are trying to solve is authorization. The following problems related to authorization need to be addressed:

- AS needs to transfer authorization information to RS.
- The transferred authorization information needs to follow a defined format and encoding, which must be efficient for constrained devices, considering size of authorization information and parser complexity.
- The RS needs to be able to verify the authenticity of the authorization information. There is a trade-off here between processing complexity and deployment complexity.
- The RS needs to enforce the authorization decisions of the AS. The authorization information it obtained from AS might require additional policy evaluation (e.g. matching against local access control lists, evaluating local conditions). The required "policy evaluation" at the RS needs to be adapted to the capabilities of the constrained device.
- Finally, as is indicated in the previous bullet, for a particular authorization decision there may be different kinds of authorization information needed, and these pieces of information may be transferred to RS at different times and in different ways prior to or during the client request.

11.2. Authentication

The following problems need to be addressed, when considering authentication:

- RS need to authenticate AS to ensure that the authorization information and related data comes from the correct source.
- C may need to to authenticate AS to ensure that it gets security information related to the resources from the right source.
- In some use cases RS needs to authenticate some property of C, in order to bind it to the relevant authorization information. In other use cases, authentication and authorization of C may be implicit, e.g. by encrypting the resource representation the RS only providing access to those who possess the key to decrypt.

- C may need to authenticate RS, in order to ensure that it is interacting with the right resources. Alternatively C may just verify the integrity of a received resource representation.

- AS may need to authenticate its communication partner (either C or RS), in order to ensure it serves the correct device.

11.3. Communication Security

There are different alternatives to provide communication security, and the problem here is to choose the optimal one for each scenario. We list the available alternatives:

- Session-based security at transport layer such as DTLS [RFC6347] offers security, including integrity and confidentiality protection, for the whole application layer exchange. However, DTLS may not provide end-to-end security over multiple hops. Another problem with DTLS is the cost of the handshake protocol, which may be too expensive for constrained devices especially in terms of memory and power consumption for message transmissions.

- An alternative is object security at application layer, e.g. using [I-D.selander-ace-object-security]. Secure objects can be stored or cached in network nodes and provide security for a more flexible communication model such as publish/subscribe (compare e.g. CoRE Mirror Server [I-D.koster-core-coap-pubsub]). A problem with object security is that it can not provide confidentiality for the message headers.

- Hybrid solutions using both session-based and object security are also possible. An example of a hybrid is where authorization information and cryptographic keys are provided by AS in the format of secure data objects, but where the resource access is protected by session-based security.

11.4. Cryptographic Keys

With respect to cryptographic keys, we see the following problems that need to be addressed:

Symmetric vs Asymmetric Keys
: We need keys both for protection of resource access and for protection of transport of authentication and authorization information. Do we want to support solutions based on asymmetric keys or symmetric keys in both cases? There are classes of devices that can easily perform symmetric cryptography, but consume considerably more time/battery for asymmetric operations.

On the other hand asymmetric cryptography has benefits e.g. in terms of deployment.

Key Establishment
How are the corresponding cryptographic keys established? Considering Section 11.1 there must be a binding between these keys and the authorization information, at least in the sense that AS must be able to specify a unique client identifier which RS can verify (using an associated key). One of the use cases of [I-D.ietf-ace-usecases] describes spontaneous change of access policies - e.g. giving a hitherto unknown client the right to temporarily unlock your house door. In this case C is not previously known to RS and a key must be provisioned by AS.

Revocation and Expiration
How are keys replaced and how is a key that has been compromised revoked in a manner that reaches all affected parties, also keeping in mind scenarios with intermittent connectivity?

12. Assumptions and Requirements

In this section we list a set of candidate assumptions and requirements to make the problem description in the previous sections more concise and precise.

12.1. Architecture

The architecture consists of at least the following types of nodes:

- RS hosting resources, and responding to access requests
- C requesting access to resources
- AS supporting the access request/response procedure by providing authorization information to RS.
  - AS may also provide other services such as authenticating C on behalf of RS, or providing cryptographic keys or credentials to C and/or RS to secure the request/response procedure.
- The architecture may contain intermediary nodes between any pair of C, RS and AS, such as e.g. forward/reverse proxies in the CoRE architecture. The solution shall not unduly restrict the use of intermediaries.
  - The architecture shall support session based security and data object security.

12.2. Constrained Devices

- C and/or RS may be constrained in terms of power, processing, communication bandwidth, memory and storage space, and moreover
  - unable to manage complex authorization policies
  - unable to manage a large number of secure connections
  - without user interface
  - without constant network connectivity
  - unable to precisely measure time
  - required to save on wireless communication due to high power consumption
- AS is not a constrained device.
- All devices under consideration can process symmetric cryptography without incurring an excessive performance penalty.
  - We assume the use of a standardized symmetric key algorithm, such as AES.
  - Except for the most constrained devices we assume the use of a standardized cryptographic hash function such as SHA-256.
- Public key cryptography requires additional resources (e.g. RAM, ROM, power, specialized hardware).
- A DTLS handshake involves significant computation, communication, and memory overheads in the context of constrained devices.
  - The RAM requirements of DTLS handshakes with public key cryptography are prohibitive for certain constrained devices.
  - Certificate-based DTLS handshakes require significant volumes of communication, RAM (message buffers) and computation.
- The solution shall support a simple scheme for expiring authentication and authorization information on devices which are unable to measure time (cf. section Section 13.2).

12.3. Authentication

- RS need to authenticate AS to ensure that the authorization information and related data comes from the correct source.

- Depending on use case, C, RS or AS may need to authenticate each other.

12.4. Authorization

- The authorization decision is based on credentials presented by C, the requested resource, the RESTful method, and local context in RS at the time of the request, or on any subset of this information.

- The authorization decision is taken either by AS or RS.

- The authorization decision is enforced by RS.

  * RS needs to have access to authorization information in order to verify that C is allowed to access the resource as requested.

  * RS needs to make sure that it provides resource access only to authorized clients.

- Apart from authorization for access to a resource, authorization may also be required for access to information about a resource (e.g. resource descriptions).

- The solution may need to be able to support the delegation of access rights.

12.5. Authorization Information

- Authorization information is transferred from AS to RS using Agent, Push or Pull mechanisms [RFC2904].

- RS shall authenticate that the authorization information is coming from AS.

- The authorization information may also be encrypted end-to-end between AS and RS.

- RS may not be able to communicate with AS at the time of the request from C.

- RS may store or cache authorization information.

- Authorization information may be pre-configured in RS.
- Authorization information stored or cached in RS shall be possible to change. The change of such information shall be subject to authorization.
- Authorization policies stored on RS may be handled as a resource, i.e. information located at a particular URI, accessed with RESTful methods, and the access being subject to the same authorization mechanics. AS may have special privileges when requesting access to the authorization policy resources on RS.
- There may be mechanisms for C to look up the AS which provides authorization information about a particular resource.

12.6. Resource Access

- Resources are accessed in a RESTful manner using GET, PUT, POST, DELETE.
- By default, the resource request shall be integrity protected and may be encrypted end-to-end from C to RS. It shall be possible for RS to detect a replayed request.
- By default, the response to a request shall be integrity protected and encrypted end-to-end from RS to C. It shall be possible for C to detect a replayed response.
- RS shall be able to verify that the request comes from an authorized client
- C shall be able to verify that the response to a request comes from the intended RS.
- There may be resources whose access need not be protected (e.g. for discovery of the responsible AS).

12.7. Keys and Cipher Suites

- AS and RS have established cryptographic keys. Either AS and RS share a secret key or each have the other's public key.
- The transfer of authorization information is protected with symmetric and/or asymmetric keys.
- The access request/response can be protected with symmetric and/or asymmetric keys.

- There must be a mechanism for RS to establish the necessary key(s) to verify and decrypt the request.
- There must be a mechanism for C to establish the necessary key(s) to verify and decrypt the response.
- There must be a mechanism for C to look up the supported cipher suites of a RS.

12.8. Network Considerations

- The solution shall prevent network overload due to avoidable communication with AS.
- The solution shall prevent network overload by compact authorization information representation.
- The solution shall optimize the case where authorization information does not change often.
- The solution where possible shall support an efficient mechanism for providing authorization information to multiple RSs, for example when multiple entities need to be configured or change state.

12.9. Legacy Considerations

- The solution shall work with existing infrastructure.
- The solution shall support authorization of access to legacy devices.

13. Security Considerations

This document discusses authorization-related tasks for constrained environments and describes how these tasks can be mapped to actors in the architecture.

The entire document is about security. Security considerations applicable to authentication and authorization in RESTful environments are provided in e.g. OAuth 2.0 [RFC6749].

In this section we focus on specific security aspects related to authorization in constrained-node networks.

13.1. Physical Attacks on Sensor and Actuator Networks

The focus of this work is on constrained-node networks consisting of connected sensors and actuators. The main function of such devices is to interact with the physical world by gathering information or performing an action. We now discuss attacks performed with physical access to such devices.

The main threats to sensors and actuator networks are:

- Unauthorized access to data to and from sensors and actuators, including eavesdropping and manipulation of data.

- Denial-of-service making the sensor/actuator unable to perform its intended task correctly.

  A number of attacks can be made with physical access to a device including probing attacks, timing attacks, power attacks, etc. However, with physical access to a sensor or actuator device it is possible to directly perform attacks equivalent of eavesdropping, manipulating data or denial of service. For example:

- Instead of eavesdropping the sensor data or attacking the authorization system to gain access to the data, the attacker could make its own measurements on the physical object.

- Instead of manipulating the sensor data the attacker could change the physical object which the sensor is measuring, thereby changing the payload data which is being sent.

- Instead of manipulating data for an actuator or attacking the authorization system, the attacker could perform an unauthorized action directly on the physical object.

- A denial-of-service attack could be performed physically on the object or device.

  All these attacks are possible by having physical access to the device, since the assets are related to the physical world. Moreover, this kind of attacks are in many cases straightforward (requires no special competence or tools, low cost given physical access, etc.)

  As a conclusion, if an attacker has physical access to a sensor or actuator device, then much of the security functionality elaborated in this draft is not effective to protect the asset during the physical attack.

Since it does not make sense to design a solution for a situation that cannot be protected against we assume there is no need to protect assets which are exposed during a physical attack. In other words, either an attacker does not have physical access to the sensor or actuator device, or if it has, the attack shall only have effect during the period of physical attack.

13.2. Time Measurements

Measuring time with certain accuracy is important to achieve certain security properties, for example to determine whether a public key certificate, access token or some other assertion is valid.

Dynamic authorization in itself requires the ability to handle expiry or revocation of authorization decisions or to distinguish new authorization decisions from old.

For certain categories of devices we can assume that there is an internal clock which is sufficiently accurate to handle the time measurement requirements. If RS can connect directly to AS it could get updated in terms of time as well as revocation information.

If RS continuously measures time but can't connect to AS or other trusted source, time drift may have to be accepted and it may not be able to manage revocation. However, it may still be able to handle short lived access rights within some margins, by measuring the time since arrival of authorization information or request.

Some categories of devices in scope may be unable measure time with any accuracy (e.g. because of sleep cycles). This category of devices is not suitable for the use cases which require measuring validity of assertions and authorizations in terms of absolute time.

14. IANA Considerations

This document has no actions for IANA.

15. Acknowledgements

The authors would like to thank Olaf Bergmann, Robert Cragie, Klaus Hartke, Sandeep Kumar, John Mattson, Corinna Schmitt, Mohit Sethi, Hannes Tschofenig, Vlasios Tsiatsis and Erik Wahlstroem for contributing to the discussion, giving helpful input and commenting on previous forms of this draft. The authors would also like to acknowledge input provided by Hummen et al. [HUM14delegation].

16. Informative References

[HUM14delegation]
Hummen, R., Shafagh, H., Raza, S., Voigt, T., and K. Wehrle, "Delegation-based Authentication and Authorization for the IP-based Internet of Things", 11th IEEE International Conference on Sensing, Communication, and Networking (SECON'14), June 30 - July 3, 2014.

[I-D.hardjono-oauth-umacore]
Hardjono, T., Maler, E., Machulak, M., and D. Catalano, "User-Managed Access (UMA) Profile of OAuth 2.0", draft-hardjono-oauth-umacore-12 (work in progress), February 2015.

[I-D.ietf-ace-usecases]
Seitz, L., Gerdes, S., Selander, G., Mani, M., and S. Kumar, "ACE use cases", draft-ietf-ace-usecases-03 (work in progress), March 2015.

[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-01 (work in progress), March 2015.

[I-D.selander-ace-object-security]
Selander, G., Mattsson, J., and L. Seitz, "March 9, 2015", draft-selander-ace-object-security-01 (work in progress), March 2015.

[RFC2748] Durham, D., Boyle, J., Cohen, R., Herzog, S., Rajan, R., and A. Sastry, "The COPS (Common Open Policy Service) Protocol", RFC 2748, January 2000.

[RFC2904] Vollbrecht, J., Calhoun, P., Farrell, S., Gommans, L., Gross, G., de Bruijn, B., de Laat, C., Holdrege, M., and D. Spence, "AAA Authorization Framework", RFC 2904, August 2000.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, May 2014.

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

[RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

Authors' Addresses

Stefanie Gerdes
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63906
Email: gerdes@tzi.org

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

Email: ludwig@sics.se

Goeran Selander
Ericsson
Faroegatan 6
Kista 164 80
Sweden

Email: goran.selander@ericsson.com

Carsten Bormann (editor)
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org