ACE Working Group
Internet-Draft
Intended Status: Informational
Expires: April 30, 2015

L. Seitz
SICS Swedish ICT
G. Selander
Ericsson

October 27, 2014

Problem Description for Authorization in Constrained Environments
draft-seitz-ace-problem-description-02

Abstract

We 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.

Status of this Memo

Copyright and License Notice

Table of Contents

1. Introduction

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.seitz-ace-usecases], which also lists some requirements 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 more detailed set of assumptions and requirements (cf. section 4).

1.1 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.

2. 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 authorization 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 does not need to be active in an constrained device access control setting, so interactions with the RO are out of scope for this memo. 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. This information may for example be specific access control decisions such as "client C has the right to access this URI with this RESTful method, this payload value, under these local conditions", "client C has the right to access these URIs" or more indirect information "client C is in this access group". In the latter it is assumed that RS knows what rights are associated to a particular access group. As mentioned above, since RS is constrained, it needs to offload authorization policy management and/or authorization decision making to AS, and thus some authorization information needs to be transferred from AS to RS. This information may for example be specific access control decisions such as "client C has the right to access this URI with this RESTful method, this payload value, under these local conditions", "client C has the right to access these URIs", or more indirect information "client C is in this access group". In the latter it is assumed that RS knows what rights are associated to a particular access group.

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. However, cryptographic keys that are used to protect information between AS and C are in scope: The reason being that dynamic access control is one of the use cases to be supported, and this may involve granting access to a client previously unknown to the server. An RS may have multiple trusted ASs corresponding to resources of different ROs, in which case it requires a key for each AS. This is a straightforward extension and is not further elaborated in this memo.

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 (e.g. because AS is switched off), or may only have intermittent connectivity, where a connection at the time of access request cannot be guaranteed. 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, NFC, etc. 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 a number of information flows that needs to be secured:

a. The transfer of authorization information from AS to RS

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

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

3. Problem Description

From the background described in the previous section, we see the following problems that need to be solved in order to achieve explicit and dynamic authorization:

- Authorization

  RS must have access to authorization information.

  Given that the relevant information has been provided to RS, it

must be able to handle an access request from C (match request against authorization information, grant or deny the request, and in the case of grant perform what is requested).

- Authentication

  Some property of C needs to be authenticated to bind authorization information to it.

  RS needs to establish the authenticity of authorization information, and that it comes from a trusted AS.

  Finally some property of RS needs to be authenticated to C, so that C can verify that it is communicating with the intended RS.

- Communication Security

  Communication security is needed to protect the integrity, and sometimes the secrecy of information flows between the parties. This includes the flow of authentication and authorization information, but also the actual request and response upon which access control is performed.

- Key establishment

  C and RS need to establish cryptographic keys in order to set up secure communications

Clearly, these problems are interconnected and need to take into account the involved constrained devices.

3.1. Authorization

The core problem we are trying to solve is authorization.

- AS needs to transfer authorization information to RS. This can be done with or without involvement of C. In the case of C involvement there are three different message sequences: Agent, Pull or Push [RFC2904].

  (i) In the agent sequence, C submits its request to AS and AS contacts RS to execute the request on C's behalf.

  (ii) When using the pull sequence, C contacts RS and RS pulls authorization information directly from AS as a reaction to C's request (as e.g. in RADIUS [RFC2865]).

  (iii) In the push sequence, C is used as intermediary between AS

and RS, and authorization information is transferred in the form of some token (as e.g. in OAuth [RFC6749]).

- What does the transferred authorization information contain and how should it be formatted/encoded? This must be efficient for constrained devices, considering size of authorization information and parser complexity.

- How does RS verify the authenticity of the authorization information? There is a trade-off between processing complexity and deployment complexity in using digital signatures with asymmetric keys or message authentication codes with symmetric keys.

- How does RS enforce authorization decisions of AS? Does the authorization information it obtained from AS require additional policy evaluation (e.g. matching against local access control lists, evaluating local conditions)? What kind of "policy evaluation" can we assume a constrained device to be capable of?

- 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. For example, local access control lists for particular access groups may be pushed from AS to RS without involvement of C at regular intervals, whereas an assertion of group membership (client attribute) of a particular C can be pushed involving C as in (iii) above.

3.2. Authentication

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

- RS needs to authenticate some property of C, in order to bind it to the relevant authorization information. This could e.g. be a digital signature or a message authentication codes performed by C where a corresponding cryptographic key is contained in the authorization information.

- In many use cases C wants to authenticate RS, in order to ensure that it is interacting with the right resources.

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

- Since AS has a trust relation to both C and RS, it could also provide them with the means of mutual authentication (similar to a Kerberos [RFC4120] server). This would make it possible for RS to authenticate previously unknown clients.

3.3. Communication Security

There are different alternatives to provide communication security.

- One is session-based security at transport layer such as DTLS [RFC6347], which offers security, including integrity and confidentiality protection, for the whole application layer exchange. One cost of DTLS is the handshake protocol, which may be expensive for constrained devices especially in terms of memory and power consumption for message transmissions.

- An alternative is data object security at application layer, e.g. using JWE [I-D.ietf-jose-json-web-encryption]. 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.vial-core-mirror-proxy]). However, data object security only may not provide confidentiality for the message headers. For example, information such as the RESTful method, the host address, and the resource URI may be revealed.

- A solution to the overall authorization problem may be based on session-based security only, data object security only or a hybrid. 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. (For secure objects containing authorization information, compare e.g. OAuth 2.0 MAC Tokens [I-D.ietf-oauth-v2-http-mac].)

- A hybrid solution may also be useful to support a flexible trust model, e.g. a resource representation wrapped end-to-end in JWE sent over DTLS hop-by-hop in a case where an intermediary is allowed to read the header but not the payload.

- A detailed analysis how different use cases benefit from different communication security paradigms is beyond the scope of this memo. Current Internet standards support both approaches, and this should be possible to leverage also in constrained environments.

3.4. Cryptographic Keys

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

- Symmetric vs Asymmetric Keys

  Do we want to support solutions based on asymmetric keys or symmetric keys, or both? The question applies both to protection of resource access and to protection of authentication and authorization information.

  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 3.1 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.seitz-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?

4. 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.

4.1 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.

4.2 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

- C and RS may be class 1 (potentially with large effort) or more powerful devices.

- AS is not a constrained device.

- All devices 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).

- A DTLS handshake with public key cryptography involves significant computation, communication, and memory overheads in the context of constrained devices.

  - The RAM requirements of DTLS handshake with public key cryptography may be prohibitive for constrained devices.

  - Certificate-based DTLS handshake requires extensive resources e.g. in terms of ROM.

- The solution shall support a simple scheme for expiring authentication and authorization information on devices which are unable to measure time (cf. section 5.2).

4.3 Authorization

- The authorization decision may be based on credentials presented by C, resource, RESTful method and local context in RS at the time of the request.

- The authorization decision may be 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.

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

4.4 Authorization information

- Authorization information is information that allows RS to verify that a requesting C is authorized.

- Authorization information includes self-contained information such as authorization decisions or client capability lists which

allows RS to directly match against a request.

- o Authorization information includes also such information that RS may need to combine, in order to verify that a requesting C is authorized, including client attributes and authorization polices (e.g. access control lists) based on client attributes.

4.5 Access to authorization information

- o Authorization information may a priori be transferred directly between AS and RS, or via C; using Agent, Push or Pull mechanisms [RFC2904].

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

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

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

- o RS may store or cache authorization information.

- o Authorization information may be pre-configured in RS.

- o Authorization information stored or cached in RS shall be possible to change. The change of such information shall be subject to authorization.

- o 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.

- o There may be mechanisms for C to look up the AS which provides authorization information about a particular resource.

4.6 Resource access

- o Resources are accessed in a RESTful manner using GET, PUT, POST, DELETE.

- o 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. (DTLS supports this.)

- By default, the response to a request shall be integrity protected and may be encrypted end-to-end from RS to C. It shall be possible for C to detect a replayed invalid response. (DTLS supports this.)

- C shall be able to verify that the response to a request comes from the intended RS. (DTLS supports this.)

- RS shall be able to verify that the request comes from an authorized client

- There may be resources whose access need not be protected (e.g. for discovery of the responsible AS).

4.7 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 may be protected with symmetric and/or asymmetric keys. (DTLS supports this.)

- There must be a mechanism for RS to establish the necessary key(s) to verify and decrypt the request. (DTLS supports this.)

- There must be a mechanism for C to establish the necessary key(s) to verify and decrypt the response. (DTLS supports this.)

- There may be a mechanism for C to look up the supported cipher suites of a RS. (DTLS supports this.)

4.8 Communication security paradigm

- The solution shall support session based security and/or data object security.

4.9 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.

4.10 Legacy considerations

- The solution shall work with existing infrastructure.

- The solution shall support authorization of access to legacy devices.

4.11 Open issues

The section lists some known open issues

- What is the level of functionality that can be achieved with class 1 devices
  - with off-the-shelf software?
  - using heroic efforts?

- Is authorization for network access in scope?

- Should the model of draft-gerdes-ace-actors [I-D.gerdes-ace-actors] (in particular the Authorization Manager) be included in the default architecture?

- Should the requirements include cross-domain authorization?

5. Security Considerations

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.

5.1 Physical attacks on sensor and actuator networksmirror

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.

5.2 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.

However there are simpler schemes to grant certain temporary access requests in a secure manner. For example, one-time authorization grants based on some freshness maintained between AS and RS such as sequence numbers or nonces. For the convenience of the reader we now outline a very simplistic scheme. AS may keep a counter for each RS, step the counter for each time it generates new authorization information and include the counter in the authorization information. RS accepts as fresh authorization information with a higher counter compared to highest previously received counter value. If the authorization information is fresh, RS grants the associated access request and replaces the old counter value with the new. The

security considerations of this scheme is out of scope for this memo.

6. IANA Considerations

This document has no actions for IANA.

7. Acknowledgements

The authors would like to thank Carsten Bormann, Stefanie Gerdes, 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 the 00-version. The authors would also like to acknowledge input provided by draft-gerdes-ace-actors [I-D.gerdes-ace-actors] and Hummen et al [HUM14delegation].

8. References

8.1 Normative References

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, August 2007.

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

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

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

8.2 Informative References

[I-D.seitz-ace-usecases]
Seitz, L., Gerdes, S., and Selander, G., "ACE use cases", draft-seitz-ace-usecases (work in progress), February 2014.

[I-D.ietf-jose-json-web-encryption]
Jones, M., Hildebrand, J., "JSON Web Encryption (JWE)", draft-ietf-jose-json-web-encryption (work in progress), April 2014.

[I-D.vial-core-mirror-proxy]
Vial, M., "CoRE Mirror Server", draft-vial-core-mirror-

proxy (expired), July 2012.

[I-D.ietf-oauth-v2-http-mac]
Richter, J., Mills, W., Tschofenig, H. (Ed.), and P. Hunt, "OAuth 2.0 Message Authentication Code (MAC) Tokens", draft-ietf-oauth-v2-http-mac (work in progress), January 2014.

[I-D.gerdes-ace-actors]
Gerdes, S., "Actors in the ACE Architecture", draft-gerdes-ace-actors-01 (work in progress), July 2014.

[HUM14delegation] Hummen, R., Shafagh, H., Raza, S., Voigt, T., Wehrle, K., "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.

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

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 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.

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

[RFC6749] Hardt, D., Ed., "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.

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

Authors' Addresses

Ludwig Seitz
SICS Swedish ICT AB
Scheelevagen 17
22370 Lund
SWEDEN
EMail: ludwig@sics.se

Goeran Selander
Ericsson
Farogatan 6
16480 Kista
SWEDEN
EMail: goran.selander@ericsson.com