Avoiding Gaps in Authorization Solutions for the Internet of Things

Stefanie Gerdes
Universität Bremen TZI
gerdes@tzi.org

Olaf Bergmann
Universität Bremen TZI
bergmann@tzi.org

Carsten Bormann
Universität Bremen TZI
cabo@tzi.org

***Abstract*—Authorization in a network of things needs to define how the involved endpoints securely obtain the authentication and authorization information that they require to setup security associations with their communication partners, and protect and validate data exchanged as defined by their owners.**

A prerequisite for the protection of the authorization process is to identify how each component involved must protect the security objectives of its owners. In this paper, we present the tasks that an endpoint must perform to validate the authorization of a communication partner. These tasks constitute a checklist to analyze authorization solutions. We summarize an analysis of a standardization proposal for the Internet of Things which indicates that essential aspects of the authorization process are not addressed by the current approach and suggests how the solution should be fixed.

I. Introduction

When physical objects are equipped with computing capabilities and interconnected with each other to form a *network of (communicating) things*, various stakeholders can be involved whose security objectives must be protected throughout the devices' life cycle. The stakeholders who are entitled to set rules that specify what happens to their assets (e.g., by accessing data on a device or changing the device state in some way) constitute the root of any authorization decision regarding their devices and thus need to be involved whenever data enters or leaves a device.

Obviously, these stakeholders cannot always be present when things communicate autonomously. The endpoints involved in the communication therefore must protect their stakeholders' security objectives. We call these stakeholders who control the rules for protecting and validating the data exchanged between endpoints their *overseeing principals*.

Authorization in this case must define how the required information to authenticate and authorize involved endpoints can be obtained in order to setup security associations with their respective communication partners. The tasks that are required if claims are received that influence the authentication and authorization are always the same and thus can be investigated without restricting the discussion to a specific solution. Many existing proposals do not cover the whole authorization process and leave aspects such as the initial provisioning of the necessary keying material to other protocols. Gaps in the authorization process occur when requirements concerning obtained data, existing relationships with other entities, and capabilities of underlying protocols and security mechanisms exist which are not explicitly defined in the specification. Without a clear guideline, implementers must provide their own interpretation. If they do not fill in these gaps correctly, vulnerabilities occur. We say that the protection is *continuous* when no gaps and vulnerabilities exist and endpoints exchange data only as defined by all overseeing principals.

Anderson and Needham demand that designers of cryptographic protocols must make the necessary naming, typing and freshness information explicit, and must state their starting assumptions and goals [1, p. 438]. Solution designers must therefore define which data the endpoints require, how this data must be obtained and validated, and how the communication between the involved endpoints must be secured.

Moreover, authorization solutions must explicitly define the security associations that are expected to already exist between endpoints, and specify which pieces of information must already have been provided to the respective endpoints. If the necessary information flows are not protected in each step of the communication, the whole authorization process may be compromised.

Identifying all relevant details and considering all assumptions and requirements that are necessary for the security of the authorization process is difficult for solution designers. Important facts may easily be overlooked. In this paper, we present a generalized checklist of tasks that must be performed to securely send and receive authentication and authorization claims. This checklist helps solution designers to discover gaps and vulnerabilities in their security specification. As an example how the checklist is used, we provide the analysis of an IoT authorization solution that is proposed for standardization.

The list of tasks is introduced in section II. The IoT authorization solution is analyzed in section III. The analysis shows that this standardization proposal does not consider essential parts of the authorization process. Necessary improvements to fix this solution derive directly from the results of the analysis. The discussion in section IV details a set of issues that are inherent to this approach and cannot be fixed easily. Our findings are summarized in section V.

Workshop on Decentralized IoT Security and Standards (DISS) 2018
18 February 2018, San Diego, CA, USA
ISBN 1-891562-51-7
https://dx.doi.org/10.14722/diss.2018.23006
www.ndss-symposium.org

II. Delegation Tasks

As stated above, endpoints are deployed by human beings, their overseeing principals, to perform certain tasks for them. If the endpoints handle sensitive data, they must act in their overseeing principals' interest and protect it. To do so, the endpoints must have a security association with their overseeing principals and be informed about their authorization decisions. The process of providing these decisions in the form of permissions to the respective endpoints must be continuously secure.

Endpoints must be able to determine if a certain peer is the holder of a certain permission. The authorization process therefore includes an authentication process. We call this synthesis of mechanism *authenticated authorization*. An endpoint usually cannot perform all aspects of authenticated authorization on its own and therefore delegates certain authentication and authorization tasks to other entities (e.g., its overseeing principal), that then provide the required input. We will call these entities *claimants* and the input that they provide *claims*. The delegating endpoint is called the *delegator*. Claims always comprise a statement which is the purpose of the claim; in this work, we distinguish *attribute claims* that inform about attributes of a certain entity such as its affiliation, and *authorization claims* that contain an entity's permissions. The entity that the claim refers to is called the *holder* of the claim.

In the following, we will describe the tasks that the involved actors must perform for each claim. A claim statement such as a certain attribute is not useful on its own. The claim must directly or indirectly comprise all relevant relationships, i.e. the holder, delegator and claimant. The endpoint must check that the claim fits into the current communication context and must ascertain that the claimant was authorized by its own overseeing principal. Claims can only be secure if all relationships are bound to the claim and checked by the endpoint. Omitting tasks for a claim may compromise the whole authorization process. An overview of the delegation tasks is presented in figure 1. We can see that the tasks cover the relevant relationships of the communication. Additionally, it must be assured that the endpoint does not use outdated information (task Dg2) and that all current information was considered (task Dg7).

The claimant must ascertain that the delegator is authorized by the claimant's overseeing principal to provide potential input for a claim, and to then receive the output of the claim. Claimants therefore must check the authorization of the communication partner before sending or receiving a message concerning a claim. We will call this task *Dg0*. If task Dg0 is omitted, claimants may accept unauthorized data as input which may compromise the claim. Also, the confidentiality of a claim may be breached if it is sent to unauthorized entities.

The claimant must then bind all required claim information together: a) the claim statement, b) the intended destination of the claim, i.e., the delegator that is intended to make use of the claim, and c) the holder of the claim which is represented by a verifier, i.e., an attribute or cryptographic keying material that can be used to identify the claim holder. Claimants must endorse that they issued the whole claim. Attackers must be prevented from cutting off or changing any part of the claim information. We will call the claim binding task *Dg1*.

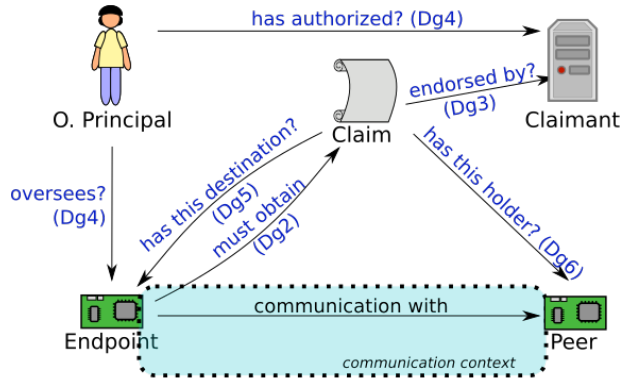

Fig. 1. Overview of the Delegation Tasks

The constructed claims and the verifiers must be securely provided to the parties that require them. Also, the delegators must ascertain that they have the most recent claims. This task is not easy to perform: an endpoint will have difficulties to determine if a claim got lost. But an authorization solution must ascertain that all necessary authentication and authorization claims are updated regularly. Even though the frequency in which updates must occur depends on the application scenario, a claim must not be infinitely valid in order to ensure a basic level of security. The task of obtaining the most recent claim is called *Dg2*.

The delegator must validate the claims it received. It must check if all the required information is bound together in the claim. We will call the task where the claim binding is checked task *Dg3*.

As the delegator's overseeing principal is the authority for its data and devices, claimants must be authorized by the respective overseeing principal to provide claims that influence the authorization. The delegator must therefore validate that the claim was issued by an authorized claimant (task *Dg4*). This task requires an own authorization process.

The delegator must also validate that it actually is the intended destination of a claim. Otherwise an attacker might trick the delegator to accept a claim that was issued for a different entity. We will call this task *Dg5*.

Due to the binding performed as part of task Dg1, attribute and authorization claims refer to a certain holder. The delegator must validate if the claim actually belongs to an entity that it currently interacts with. E.g., if the delegator currently communicates with a certain peer, it must check if the claim refers to that peer (task *Dg6*). This may be done using cryptographic methods, e.g., the claim could be bound directly to cryptographic keying material by the claimant. The delegator then may perform this task by checking if the entity, e.g., the peer, is able to use this keying material. If the claim is bound to an attribute, e.g., a hostname, the delegator requires an additional attribute claim that binds this attribute to certain cryptographic keying material, e.g., an X.509 certificate. The binding between the claim and the respective cryptographic keying material must provide continuity: there must be no missing link if multiple claims are used.

When multiple claims for the same issue exist, e.g., because a claimant provides several claims, or a delegator has several claimants, the delegator must evaluate all claims (task *Dg7*). This might, e.g., be the case if an endpoint has multiple overseeing principals and must consider the decisions of every one of them. For this paper, we will assume that every endpoint only has a single overseeing principal and a single claimant for each fact, and that therefore no additional effort is necessary to perform task Dg7.

For an effective authorization process, all tasks must be performed correctly. The enumeration below shows what happens if tasks are omitted.

- Dg0 Delegator Authorization: disclosure of confidential information.
- Dg1 Claim Configuration: claim contains wrong information.
- Dg2 Obtainment: critical information is missing.
- Dg3 Binding & Endorsement: attackers can manipulate or forge claims.
- Dg4 Authorization: unauthorized entities can issue claims.
- Dg5 Destination: man-in-the-middle attacks.
- Dg6 Holder: claim may be associated with the wrong holder.
- Dg7 Evaluation: critical information is not considered.

III. IoT Authorization Solution Analysis

In the Internet Engineering Task Force (IETF), the working group *Authentication and Authorization for Constrained Environments* (ACE) is in the process of standardizing an authentication and authorization framework for the IoT [9]. The framework is loosely based on OAuth [6] and addresses scenarios where a client (C) contacts an authorization server (AS) to obtain an access token that it then can use to prove its authorization to a resource server (RS). The overseeing principal for the RS and AS is called *resource owner* (RO) in this architecture. The client's overseeing principal is the *requesting party* (RqP).

Targeting the Internet of Things, the ACE framework allows C and RS to be constrained devices (c.f. [2]), i.e. both may be very limited regarding their processing power, energy consumption, and storage.

In the ACE architecture [5] each constrained device is supported by a less-constrained device that helps with difficult authentication and authorization tasks. The authors of the ACE framework decided to only implement the server side AS. On the client side, AS helps only with authenticating RS. The ACE framework architecture is depicted in figure 2.

The ACE access token comprises an OAuth scope parameter that indicates the authorization rules, and a confirmation (cnf) structure that contains information about the associated keying material [9, p. 32]. C must present the access token to RS and show that it is able to use the keying material specified in the token to prove that it is authorized as specified in the scope. Both symmetric and asymmetric keys may be used. The ACE framework is not designed for direct use but instead requires so-called *profiles* to specify certain details of the protocol flow. One example is the Datagram Transport Layer Security (DTLS)

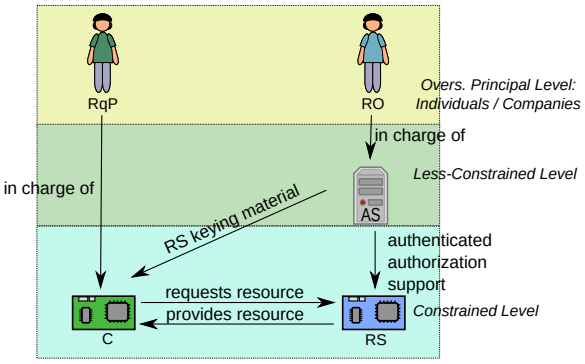

Fig. 2. ACE Framework Architecture

profile [4] that specifies how authorization information can be associated with a DTLS session between C and RS.

In this section we will analyze if the components that use the DTLS profile perform the necessary delegation tasks we introduced above. In cases where information is not provided in the DTLS profile, we also consider the ACE framework specification. We do not provide a complete analysis of the ACE framework in this work.

AS acts as a claimant for RS and provides it with the necessary authorization information and keying material for the communication with C. We will analyze the tasks for this delegation in section III-A. Additionally, AS acts in most cases as a claimant for C since it provides C with the necessary keying material for RS. The analysis for C's delegation to AS is shown in section III-B.

A. AS's Claim for RS

To perform task Dg0, AS must validate that RS is authorized to receive the claim output, i.e., the access token. The token is confidential since it may contain a symmetric key. The AS therefore must encrypt the token with the keying material that the resource owner provided for RS. Currently, confidentiality-protecting the access token is only demanded in the security considerations section of the ACE framework [9, p. 35]. The DTLS profile should explicitly mandate the confidentiality-protection of access tokens that contain confidential keying material to ensure that task Dg0 is performed.

For task Dg1, AS must bind all necessary information, i.e., the claim statement, the intended destination and holder, to the claim. The holder of the claim is defined in the cnf field, which contains a key or a reference to a key. This field must either be part of the access token or be retrieved using token introspection. If performed correctly, the holder is thereby bound to the claim.

Determining the claim statement of the ACE access token is difficult. AS may use the token as an attribute claim to inform RS that a client with certain keying material has certain attributes. But although ACE access tokens contain C's keying material, they do not necessarily contain additional information about their holder. If the access token is supposed to be an authorization claim, it must contain a scope field with authorization information. The claim then states that the entity with certain keying material has this authorization. But the ACE framework does not clearly define if the token must contain

a scope. It is therefore possible that ACE access tokens do not contain a claim statement at all. Without a statement, AS would inform RS that a not further specified client has certain keying material, which is useless. Even worse, a claim without a statement leaves room for misinterpretation. How an RS implementation must react to a missing statement is not clear (see also task Dg3 below).

The intended destination of the access token can be specified using the audience (aud) parameter. The ACE framework points out that it is important for the AS to specify the audience in the token. We therefore assume that the intended destination is specified correctly. AS must bind the information together and endorse the token. The ACE framework demands that the token must be integrity-protected, but the DTLS profile does not yet specify how this protection is accomplished. In summary, some clarifications should be made by the ACE framework and the DTLS profile to ascertain that AS performs task Dg1 when issuing the access token.

RS must obtain the most recent claim from AS for task Dg2. To determine the validity of an access token, the ACE framework introduces three mechanisms depending on the capabilities of the RS. If RS has a real-time clock and is time-synchronized with AS, the token includes an expiration time. This helps RS to detect if a token is expired. However, it is not clear how RS determines which token is newer if several tokens exist, e.g., if a token is updated.

For the second approach, RS sends an introspection request to AS. AS then sends an introspection message to help RS with this task. The introspection message is an additional claim for which the delegation tasks must be performed. The ACE framework does not specify how RS determines that the introspection message is fresh which is required for task Dg2. Also, if RS has to send an introspection message to determine the validity of each received access token, RS is vulnerable to Denial of Service attacks.

For the third approach, RS and AS store a sequence number. AS increments the sequence number for each token that it issues and includes the number in the access token. RS may have a set of valid tokens with sequence numbers that are in range of the most recently received sequence number. It is unclear how RS is supposed to resolve potentially contradicting permissions in this case. But even if RS would only accept tokens that have a higher sequence number then the token it already has, this approach is still unusable: RS receives access tokens from C. If C does not relay newly generated access tokens to RS, the existing token may be infinitely valid. Since C may have a strong interest to use an existing authorization forever, RS cannot rely on C for task Dg2.

Task Dg3 requires RS to check if all necessary information is bound to the access token and endorsed by AS. How RS reacts if required fields are missing from the access token is not specified. The ACE framework should define more clearly how RS must react to malformed tokens; missing information may be incorrectly filled in by implementations, which may lead to vulnerabilities. RS must also check if the information is endorsed by AS. The ACE framework states that access tokens must be integrity-protected by AS. It should additionally specify that RS must not accept unprotected tokens.

To perform task Dg4, RS must check if AS is authorized to provide claims. How RS obtains this knowledge is out of scope, but the ACE framework assumes that the RS has been registered with the AS, and that keying material was established between these entities. But the framework does not mention if this keying material enables RS to validate if an access token actually stems from an AS that is authorized by RO. It is therefore not clear if RS can perform task Dg4.

For task Dg5, RS must ascertain that it is the intended recipient of the token. The ACE framework specifies that the RS must reject tokens for which it is not the intended recipient [9, p. 33]. RS thereby performs task Dg5.

RS must check that the requesting C actually is the holder of the access token to perform task Dg6. To do so, RS must ascertain that C is able to use the keying material that is specified in the token's cnf field. Otherwise, an unauthorized C may access resources on RS. The ACE framework does not specify that RS must perform this check, which should be fixed. The DTLS profile specifies how C and RS must use the information in the cnf field [4, p. 11]; without the correct keying material, no secure connection can be established. Since C's requests are only authorized if they are received on a secure connection [4, p. 8], RS performs task Dg6 in the DTLS profile.

B. AS's Claim for C

In AS's response to C, AS may provide additional information to C, which is called RS information. If symmetric keys are used between C and RS, the RS information must provide C with the keying material for the requested RS. If asymmetric keys are used, AS provides C with RS's public key, unless C already knows it. It is not clear how AS determines if C already knows RS's credentials. Since AS provides C with the required keying material for RS, the RS information is an attribute claim. We will now analyze how AS and C perform the necessary delegation tasks for this claim.

To perform task Dg0, AS must ascertain that C is an authorized delegator. The ACE framework assumes that C has been registered with the AS and that C and AS obtained keying material for each other during this process. But, the framework does not mention if AS obtained authorization information for C from RO. It is therefore not clear if AS is able to perform task Dg0 for C. Assuming that C is authorized, AS must validate that input to the claim stem from the authorized C. The ACE framework allows C to specify the raw public key (RPK, [8]) that the token will be bound to in the cnf field of the C-to-AS request. If C does not use the same RPK for the integrity-protection of this message, AS is not able to determine if the message actually stems from the C with this key. C can thereby obtain valid access tokens for other clients; this allows C to hand over its permissions to other clients without disclosing its own keying material. The RS information must be confidentiality-protected if it is sensitive, e.g., if it contains a symmetric key. The ACE framework specifies that AS must confidentiality-protect messages to the client [9, p. 35].

For task Dg1, AS must bind the necessary claim information to the RS information. The claim statement must contain attributes of the resource server. C specifies the resource server it wants to communicate with in the Client-to-AS request, but the ACE framework does not define how the resource server is

represented. C and AS must have a common understanding how RS is identified, because otherwise AS may issue RS information for the wrong RS, and C will not be able to detect this mistake. How RS's identity must be represented is a difficult problem. IP addresses are not suitable for this purpose since they may change between a request and its corresponding response. Identifiers must be known to C which is difficult to accomplish without manual configuration; but manually provisioning C with all potential resource servers may not be feasible for typical IoT scenarios. A scalable solution requires descriptive attributes that are commonly understood and unique, e.g., a fully qualified domain name. *The ACE framework must define how C must specify RS because otherwise C might communicate with the wrong RS.*

The ACE framework defines a field for RS's attributes in the Client-to-AS request but not in the RS information. Without the information for which RS the access token was issued, C might be lead to communicate with the wrong RS. One solution would be if C can determine that the RS information was the response to certain request. The underlying security solution might enable C to do so, but the DTLS profile and the ACE framework do not currently specify this requirement and do also not prescribe which underlying security solution C and AS must use. If C and AS use TLS or DTLS to secure the communication, C is able to determine if the RS information belongs into the current communication context.

AS can specify the intended destination of the claim by encrypting it, using the keying material of the respective C. Since only C is then able to decrypt the message, it can assume that it is the intended destination of the message. AS must first encrypt the RS information before the integrity-protection is applied, or use a combined algorithm. Otherwise, the intended recipient is not integrity-protected and may be modified, which allows for man-in-the-middle attacks. In the ACE framework, the communication between AS and C must be confidentiality-protected. The ACE framework prefers the use of an Authenticated Encryption with Associated Data (AEAD, [7]) algorithm for the access token, but not for the RS information. For the latter, the claim destination therefore is not necessarily set.

AS defines the holder by binding its keying material to the RS information. The ACE framework specifies that AS must define the symmetric communication key for proof-of-possession tokens in the RS information. For asymmetric communication between C and RS, AS may provide C with RS's public key. We assume that if the RS information does not contain RS's keying material, it is not an attribute or authorization claim. AS therefore specifies the holder of the claim if it actually issues a claim to C.

C must ascertain that the claims that it uses for the secure communication with RS are up to date (task Dg2). The ACE framework does not specify how this is accomplished. The RS information does not necessarily contain information that helps C to determine if the keying material for authenticating RS is still valid. The RS information may (but does not have to) contain an expires_in field that specifies when the token expires. A constrained client without a synchronized clock will not be able to interpret this value. The keying material therefore may be infinitely valid. Also, the framework does not define how C distinguishes old from new RS information; an attacker might trick the client to accept outdated information. In summary, C may not be able to perform task Dg2.

Task Dg3 requires C to ascertain that the claim information is bound together and endorsed by AS. To do so, C must only accept RS information that is integrity-protected. This is defined in the framework. Since the claim statement is specified in C's request, C must be able to validate if the response belongs to the request. The ACE framework does not explicitly state this requirement. It is therefore not clear if C is able to perform task Dg3.

For task Dg4, C must check AS's authorization to provide the RS information. The ACE framework suggests that C has a list of authorized authorization servers that may be hard-coded. If the list is never updated, authorization claims are infinitely valid, which is not advisable. Also, the framework does not specify that this list must be specified by C's RqP. If these problems are solved, C might be able to perform task Dg4.

C must check if it is the intended destination of the RS information (task Dg5). As described for task Dg1, the intended destination of the RS information is only set if the claim is first encrypted and then integrity-protected or if a combined algorithm is used. C must only accept messages from AS which are thus protected, which is not clearly defined in the framework. C may therefore not be able to perform task Dg5.

To perform task Dg6, C must validate that RS can use the keying material that AS specified in the RS information. The ACE framework states that RS and C mutually authenticate each other, but it does not state that C must communicate only with authenticated resource servers. If C does not ascertain that the transmitted data is protected, the authorization rules of both RqP and RO are not considered, and C does not participate in the protection of their security objectives. It is not clear how C determines if RS is authorized.

IV. Discussion

If the identified gaps are filled in, the ACE framework helps RS to protect RO's security objectives. On the client side, important changes are required before the RS information is a sufficiently secure attribute claim that C can use to authenticate RS. How RqP's authorization rules are enforced is not addressed at all: it is not clear how C can participate in the protection.

If RqP directly controls the client, it can decide which servers the client accesses by manual intervention. But in the Internet of Things, clients will often have to act autonomously and cannot rely on their overseeing principal at the time of access. Autonomous clients are not able to make authorization decisions on their own. Constrained clients may have difficulties to store vast amounts of authorization rules, and manually provisioning clients with the required authentication and authorization information is often not feasible. Also, such a solution is very inflexible and does not allow for quick adaptions of authentication and authorization data if unexpected changes are required, e.g., when a communication partner was compromised.

The Internet of Things requires an authenticated authorization solution that fully supports autonomous clients and provides them with the necessary authentication and authorization claims. Installing an additional authorization solution requires extra

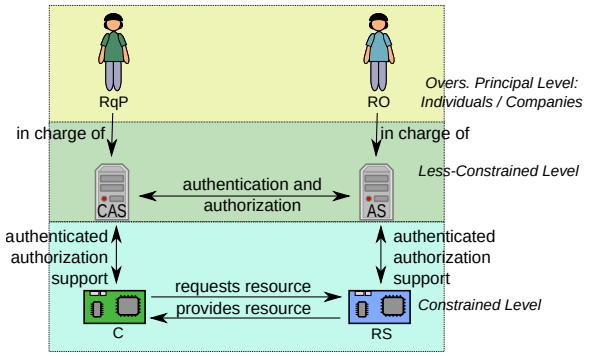

Fig. 3. Complete ACE Architecture

message exchanges, more code and potentially additional keying material. It puts more strain on the client and may, depending on the solution, also require action from the resource server. A combined authenticated authorization solution for both client and resource server is a much more suitable solution for the IoT since it reduces the overall effort.

According to Fielding and Taylor, "the Internet is about interconnecting information networks across multiple organizational boundaries" [3, p. 119]. A centralized authorization server must have a trust relationship with the client and the resource server; in the ACE framework, clients and resource servers are expected to register with the AS. A spontaneous communication between two entities that do not previously know each other is not possible. The deployment of centralized authorization servers conflicts with the Internet's scalability and multiorganizational domain requirements. The proposed ACE architecture (see figure 3) allows each constrained device to have its own less-constrained helper [5]. An authorization solution for the IoT should fully implement this architecture.

V. Conclusion

In this paper, we have introduced tasks for authenticated authorization that describe how each endpoint must act in order to communicate securely in a network of things. Any data exchanged with other entities—irrespective of the direction (send or receive)—must be authorized by the overseeing principals of the communicating devices. To protect their security objectives, it is crucial that all authorization decisions can be traced back continuously to the respective overseeing principals. We have shown that the quality of security solutions can be increased if protocol designers ascertain that the delegation tasks are performed by all involved endpoints.

Our analysis of a current proposal for standardization reveals gaps in this chain of authorization-related steps. While some of these gaps can be closed easily by revising the ACE framework and the related profile document, some design decisions were made that fundamentally limit its applicability to scenarios where clients and servers belong to the same security domain. Future standardization efforts in the IETF concerning authorization in the IoT therefore should also address global-scale, multiorganizational use-cases.

References

[1] R. J. Anderson and R. Needham, "Programming Satan's Computer," in *Computer Science Today: Recent Trends and Developments*, J. van Leeuwen, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 426–440.

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

[3] R. T. Fielding and R. N. Taylor, "Principled Design of the Modern Web Architecture," *ACM Transactions on Internet Technology (TOIT)*, vol. 2, no. 2, pp. 115–150, May 2002.

[4] S. Gerdes, O. Bergmann, C. Bormann, G. Selander, and L. Seitz, "Datagram Transport Layer Security (DTLS) Profile for Authentication and Authorization for Constrained Environments (ACE)," draft-ietf-ace-dtls-authorize-01, July 2017, Internet-Draft (Work in Progress).

[5] S. Gerdes, L. Seitz, G. Selander, and C. Bormann, "An architecture for authorization in constrained environments," draft-ietf-ace-actors-06, November 2017, Internet-Draft (Work in Progress).

[6] D. Hardt, "The OAuth 2.0 Authorization Framework," RFC 6749, October 2012, Internet Request for Comments.

[7] ISO/IEC, "Information technology – Security techniques – Authenticated encryption," February 2009, ISO/IEC 19772:2009.

[8] P. Wouters (Ed.), H. Tschofenig (Ed.), J. Gilmore, S. Weiler, and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)," RFC 7250, June 2014, Internet Request for Comments.

[9] L. Seitz, G. Selander, E. Wahlstroem, S. Erdtman, and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE)," draft-ietf-ace-oauth-authz-09, November 2017, Internet-Draft (Work in Progress).