OAuth for Operational Technology?

Abstract

This document considers following question: can OAuth present a match for securing Operational Technology? This consideration does focus on these aspects of Operational Technology which do not rely on Web technologies. Some parts of Operational Technology use them e.g. the RESTful Web services in BACnet cf. [1]. They are not in the main focus of this document.

What Is Operational Technology?

According Gartner[1], Operational Technology (short: OT) comprises "*hardware and software that detects or causes a change through the direct monitoring and/or control of physical devices, processes and events in the enterprise*". OT bears critical processes and infrastructure in sectors such as industry, utilities, building/traffic management and transportation. OT exists since decades.

How Does It Differ?

OT is related to IT and IoT: they all utilize software modules and communication technologies to implement distributed systems. At the same time it is important to note that OT is unequal to IT and IoT:

- OT vs. IT:
  - *OT characteristics*: physical objects, constrained computing devices, domain-specific protocol stacks e.g. BACnet, CAN, Modbus
  - *IT characteristics*: digital objects, less/unconstrained computing devices, Internet stack i.e. TCP/IP and HTTP
- OT vs. IoT:
  - *OT characteristics:* traditional, brown-field (existing components/ metadata/products etc.), focus on capital goods owned by legal entities
  - *IoT characteristics*: emerging, green-field (new components/metadata/ products etc.), inclusion of consumer goods owned by individuals

How About Security?

Figure 1 shows a solution pattern that is common in OT. Following properties hold:

[1] http://www.gartner.com/it-glossary/operational-technology-ot/

- Operators usually need to authenticate against engineering management systems which authorize the instructions provided to them. Engineering management systems are often accessed by means of HTTP-over-SSL/TLS.
- Callers of controllers do often not need to authenticate and there is no dedicated authorization of instructions sent to controllers. Controllers are usually accessed by means of domain-specific protocols.
- Callers of devices do often not need to authenticate. There is no authorization of instructions sent to devices. Devices also are accessed by means of domain-specific protocols.

This means: syntactically correct instructions that can be sent to OT components (controllers or devices) will usually be executed without further checks.

Since OT systems operate sensitive resources the common approach does only work when accompanied by means such as components/network isolation, physical security.

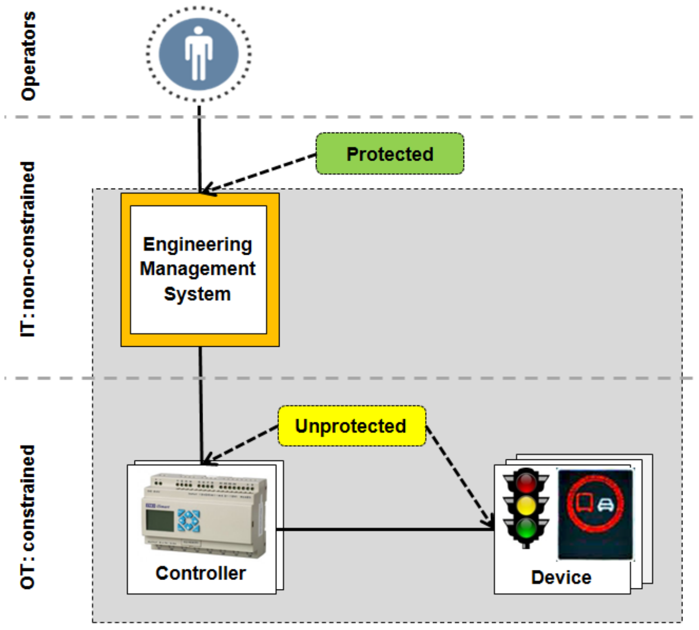

Figure 1: Traditional solution pattern in OT

Why Worry?

The traditional OT solution pattern suggests: *every production is an island*. It imposes limitations with respect to requirements such as:

- Usage of wireless networks (making physical isolation difficult)
- Inclusion of components developed by 3rd parties (without a priori knowledge of the domain-specific stacks)
- Interactions across domain boundaries
- Ad-hoc composition of systems

These requirements are fundamental for Industrial Internet and Industry 4.0. Their proposition is: *no production is an island*.

Assuming that OT is subject to a change process from *every* to *no production is an island*, then OT components (controllers and devices) have to become security-enabled.

What Must Be Achieved?

The primary requirements for OT security-enabling are:

- Authorization of instructions (e.g. requests) sent to callees: OT components essentially are daemon processes that handle private resources. When they start to expose endpoints in internal, partner or even public-facing networks then callees need to authorize instructions before processing them.
- Entity authentication: authorization in distributed systems implies a need to authenticate properties[2] claimed by the caller. Moreover properties claimed by the callee need to be authenticated by the caller before sending sensitive information to them.
- Message authentication for such instructions and corresponding receipts (e.g. responses): authorization and entity authentication depend on the authentication of data objects passed between callers and callees.

This perception of OT security requirements matches that of RFC 7744 (cf. [2])[3].

How To Do The Trick?

Corresponding security requirements hold for mail or bank accounts and other private resources on the Internet. Consider GMail as an example:

- Message authentication for requests and responses is done on transport-level by means of SSL/TLS.
- Entity authentication of callees (*GMail*) is done by means of SSL/TLS server authentication utilizing X.509 certificates. Entity authentication of callers (*you*) is done by means of username/password or one-time-password credentials.
- Authorization is done by making sure that the authenticated identifier of the caller matches the mail account owner.

But the well-known Internet security tools cannot be used in OT, at least not with copy&paste:

- The Internet security Swiss-knife SSL/TLS resp. DTLS is often unavailable and possibly even out-of-reach[4].

[2] They can be: identifiers, attributes, affiliations, assignments, memberships etc.
[3] RFC 7744 does not distinguish between IoT (e.g. Home Automation) and OT (e.g. Building Automation).
[4] Due to aspects such as: different heritage than Web (no TCP/IP, HTTP) and applicable constraints.

- Callers are no human beings who are equipped with less-constrained computing devices: the well-known (human) user authentication solutions don't do it.
- Callees are no (Web) applications operated on non-constrained computing devices: the well-known (Web) application authentication solutions don't do it.
- Resources are owned by legal entities, not individual users: solutions that assume to check with a resource owner as an attentive/responsive individual don't do it.

This leaves two viable approaches for doing the trick:

- Create OT security mechanisms from scratch building upon primitive means e.g. cryptographic algorithms.
- Borrow and transform security mechanisms from other domains.

What Can Be Borrowed?

Obviously the Web resp. its security presents a prime candidate for the second approach. Web security falls into two disciplines:

- Security for presentation-oriented Web applications aka Web UIs.
- Security for service-oriented Web applications aka Web APIs, RESTful services.

The former is less relevant for the OT domain[5], the latter is. The best-current-practices for securing service-oriented Web applications are characterized by:

- OAuth for request authorization and client authentication
- SSL/TLS for server authentication, request/response authentication and encryption

This boils down to OAuth for reasons already given above.

Why Is OAuth Appealing?

OAuth has a number of properties that make it a candidate for adoption in OT security:

- *Feature match*: OAuth addresses request authorization, caller authentication and (in case of PoP) also HTTP request authentication.
    - This is a good match with the primary security requirements for OT.
- *Software availability*: mature, production-grade and extensible software components are available for OAuth.
    - This facilitates re-use for (a subset of) the required security components. It can help to kick-start OT security.
- *Production use*: OAuth already is in large-scale production use in IT and protects mission-critical assets.
    - This provides some reassurance concerning the potential of OAuth.

[5] Interactions with and among controllers and devices

- *Divide-and-conquer*: OAuth is built on the principle of offloading complex security processing tasks from resource servers (delivering business functionality) to authorization servers (delivering security functionality).
    - This matches the desired work-split for securing OT.
- *Uneven distribution of complexity*: on side of clients and resource servers OAuth is lightweight/in-complex.
    - OT controllers and devices are (severely) constrained and can bear only lightweight security mechanisms.
- *Variety of security models*: OAuth supports protected and unprotected registration as well as bearer and PoP tokens in addition to further options (e.g. revocation). Moreover, the PoP mechanisms support application-level request signing.
    - There will be no one-size-fits-all OT security solution i.e. OT security should be assumed to incarnate according various models/profiles.
- *Versatile security tokens*: OAuth facilitates security tokens with arbitrary contents. OAuth specifies an optional token form-factor (JWT). JWT payload is versatile. JWTs can be signed and/or encrypted with symmetric or asymmetric schemes.
    - Security tokens in OT will be domain-specific: the need to express something about an OT component is different in different sectors e.g. industry, utilities, building/traffic management, transportation.
- *Extensibility*: OAuth supports additional endpoints, additional grants for defined endpoint, additional parameters for defined request/response messages, and additional values for defined keywords.
    - OT is different from IT as already indicated above. Hence an adoption of OAuth will depend on modifications and extensions.

Which Gymnastics Are Needed?

A number of modifications and extensions need to be considered in an adoption of OAuth for OT security. This includes:

- *Protocol bindings*:
    - The supply of security token (bearer) resp. authenticator (PoP) objects needs to be realized for other protocols than HTTP (cf. [3], [4]). For CoAP this is addressed in [5]. But CoAP is no common protocol in OT.
    - The acquisition of security tokens may have to be realized by other means than HTTP (cf. [6]) – depending on the availability of HTTP stacks on OT components. The same holds for registration/management and other exchanges e.g. token introspection and revocation.
- *Response signing*: an absence of SSL/TLS resp. DTLS demands PoP tokens and also suggests the addition of response signing on application-level. This allows asserting the authenticity of response messages and establishing the authentication of callees resp. responders.

- *Stateful engines/protocols*: OT often uses state in exchanges with and among controllers and devices. This has an impact on the supply of security tokens (by-value or by-reference) and authenticators esp. the handling of failure cases.
- *On-behalf-of*: OT components supposed to acquire security tokens are more constrained than in IT. They cannot be expected to be capable of acquiring tokens from arbitrary token endpoints which includes concerns such as discovery and registration. They should be able to outsource the acquisition of (cross-domain) security tokens to (own-domain) helper services. For CoAP this is architecturally addressed in [7]. But [5] does not underpin this concept with protocol elements[6].
- *Legal entity-owned resources*: the main use case that drove the evolution of OAuth was authorization for individually-owned resources[7] utilizing a discretionary model. But OT is concerned with legal entity-owned resources and their authorization is different. For instance there is a resource owner but that (being a legal entity, not an individual) is no actor who could give consent during a synchronous exchange. This has several implications:
  - User resp. user agent-centric OAuth grant types are not suitable for OT.
  - The syntax for values[8] of the (optional) `scope` abstraction (section 3.3 in [6]) is limiting and may require workarounds.
- *Metadata re-use*: rich metadata about OT components does already exist and is managed by engineering systems. OT security can only bring not yet existing information[9] and needs to integrate this delta with the existing metadata and its management.
- *Relationship management integration*: comprehensive information about the designated relations between OT components does also exist and is managed by engineering systems. OT security needs to honor existing relationship data when deciding about issuing requested security tokens.
- *Profiling*: security-enabling options (e.g. protected/unprotected registration, bearer/PoP, token signing/encryption schemes) as well as security token payload needs to be profiled in a way that is specific to the considered OT stack (e.g. BACnet, CAN, Modbus) and application domain (e.g. industry, utilities, building/traffic management, transportation).

[6] „*Since this draft focuses on the problem of access control to resources, we simplify the actors by assuming that the client authorization server (CAS) functionality is not stand-alone but subsumed by either the authorization server or the client (see section 2.2 in [I-D.ietf-ace-actors]).*"

[7] Example: *I want office24.com to print my photos stored at Google Drive.*

[8] *"…list of space-delimited, case-sensitive strings"*

[9] Credentials for the authentication of OT components.

Conclusion

OAuth already is in large-scale production use on the one hand. On the other hand it is still evolving as a building block for Web security. At the same time, the security-enabling of OT is in its incubation.

The adoption of OAuth presents a viable option for securing exchanges in OT. It appears to provide a shortcut solution towards OT security.

An OAuth adoption for OT security approach requires a number of modifications and extensions. The specification of OAuth allows such modifications and extensions. The best-in-class OAuth software components allow implementing them in an efficient way.

References

[1] N.N.: Proposed Addendum am to Standard 135-2012, BACnet - A Data Communication Protocol for Building Automation and Control Networks. ASHRAE Public Review Draft (http://www.bacnet.org/Addenda/Add-135-2012am-PPR1-draft-26_chair_approved.pdf). April 2014

[2] L. Seitz et al.: Use Cases for Authentication and Authorization in Constrained Environments. IETF RFC 7744 (https://datatracker.ietf.org/doc/rfc7744). Jan. 2016

[3] M. Jones: The OAuth 2.0 Authorization Framework: Bearer Token Usage. IETF RFC 6750 (https://datatracker.ietf.org/doc/rfc6750). Oct. 2012

[4] J. Richer et al.: A Method for Signing HTTP Requests for OAuth. IETF Internet Draft (https://datatracker.ietf.org/doc/draft-ietf-oauth-signed-http-request, work-in-progress). Feb. 2016

[5] L. Seitz et al.: Authorization for the Internet of Things using OAuth 2.0. IETF Internet Draft (https://datatracker.ietf.org/doc/draft-ietf-ace-oauth-authz work-in-progress). Feb. 2016

[6] D. Hardt: The OAuth 2.0 Authorization Framework. IETF RFC 6749 (https://tools.ietf.org/html/rfc6749). Oct. 2012

[7] S. Gerdes et al.: An architecture for authorization in constrained environments. IETF Internet Draft (https://datatracker.ietf.org/doc/draft-ietf-ace-actors, work-in-progress). Mar. 2016

Author

Oliver Pfaff, Siemens AG, CT RDA IT-Security
Otto-Hahn-Ring 6, D-81739 Munich
oliver.pfaff@siemens.com