Distributed Security Risks and Opportunities in the W3C Web of Things

Michael McCool
Intel Corporation
michael.mccool@intel.com

Elena Reshetova
Intel Corporation
elena.reshetova@intel.com

***Abstract*—The W3C Web of Things (WoT) WG has been developing an interoperability standard for IoT devices that includes as its main deliverable a "Thing Description": a standardized representation for the metadata of an IoT device, including in particular a description of its network interface, but also allowing for multiple levels of semantic annotation. The WoT Thing Description supports a descriptive (as opposed to prescriptive) approach to interoperability. The provision of rich descriptive metadata has at least four major implications for security. First, it allows for system-wide vulnerability analysis, which can be both a risk and an opportunity. Second, metadata can enable end-to-end security in multistandards networks, avoiding exposing data within bridges otherwise needed for connecting standards pairwise. Third, metadata supports service and device discovery, which raises the question of how to limit discovery to authorized agents. Fourth, metadata can enable distributed security mechanisms for access control and micropayments. To the extent that metadata access can be decentralized, decentralized mechanisms for security can be supported, although several practical issues currently make this difficult to fully support.**

I. Introduction

The economic impact of the IoT will be strongly dependent on how well devices from different manufacturers can interoperate. Very often interoperability is taken for granted when estimating the business benefit of IoT. However, if devices do *not* interoperate, a recent study [5] concluded that 40% to 60% of the benefit of IoT will be unattainable, due to the inability to address use cases that cannot be satisfied by a single manufacturer.

Unfortunately full interoperability is hard to achieve. There are currently many competing IoT standards under development, each of which is attempting to address this problem. Most of these standards are prescriptive. In a prescriptive standard, devices are validated against specific requirements. Typically the goal is that devices validated against particular standard will interoperate with other devices also validated against that standard. In addition, it is possible to bridge multiple standards so that devices validated against one standard can communicate with devices using another standard by translating communication protocols and payloads. Of course if one standard comes to dominate bridging will be unnecessary. However, so far such unification seems to be elusive and may be impossible due to divergent requirements in different but overlapping IoT subdomains.

Unfortunately the prescriptive approach has some weaknesses. In particular, there are always going to be devices that follow older standards. There are decades-old devices in particular domains, such as building and factory automation, that are just now being connected to the IoT. These devices often represent major investments and cannot be economically replaced with newer, standards-conforming devices. This is the "brownfield" problem. In addition, new devices are being deployed today that have not been validated against any particular IoT standard, even if they use other standards such as JSON and HTTP.

As an alternative to the prescriptive approach, the W3C Web of Things (WoT) Working Group has been developing a *descriptive* approach to IoT interoperability. In the descriptive approach, metadata is provided that describes how to communicate with each particular device. The metadata itself is standardized but flexible enough to describe a wide variety of IoT network interfaces. With this approach, devices can but do not have to be prevalidated against a particular standard before being deployed. They can be described after the fact, and do not need any modification to be used as part of a Web of Things system. This solves the brownfield problem and allows older devices as well as devices satisfying different IoT standards to be integrated into a unified system.

This approach has both risks and opportunities from a security point of view.

Most obviously, IoT devices, even those conforming to a prescriptive IoT standard, may vary widely in their support for security. Therefore a Web of Things system needs to manage different levels of trust for different devices. Devices from different ecosystems or manufacturers may also take different approaches to security and may use different security mechanisms. This may cause integration challenges, even if the necessary information is provided in the metadata.

Beyond this basic concern, the availability of pervasive metadata raises several other issues from a security perspective. In this paper we discuss four major issues:

Permission to freely reproduce all or part of this paper for noncommercial purposes is granted provided that copies bear this notice and the full citation on the first page. Reproduction for commercial purposes is strictly prohibited without the prior written consent of the Internet Society, the first-named author (for reproduction of an entire paper only), and the author's employer if the paper was prepared within the scope of employment.
NDSS '18, 18-21 February 2018, San Diego, CA, USA
Copyright 2018 Internet Society, ISBN 1-1891562-49-5
http://dx.doi.org/10.14722/ndss.2018.23xxx

1) **Vulnerability analysis:** Providing information about what devices can do makes it easier to automatically scan for devices with vulnerabilities. An attacker may also use this information to plan attacks that take advantage of vulnerabilities in multiple devices. However, for the system manager, scanning can also be an opportunity to identify devices whose vulnerabilities need to be mitigated.
2) **End-to-end security:** Metadata enables end-to-end security in networks of IoT devices using multiple standards. If metadata is used to push payload adaptation to endpoints then communication payloads can be encrypted end-to-end. This contrasts with systems that use local bridging to connect devices from multiple IoT standards. Local bridges require opening (and usually re-encrypting) data in potentially-vulnerable gateways.
3) **Secure discovery:** Information about how to use a service, and ideally even its existence, should not be disclosed to agents without the authorization to use it. The WoT approach allows powerful semantic searches to be used for discovery. How can this capability be made available while still securing the metadata?
4) **Security mechanism enabling:** Metadata may be provided to enable specific security mechanisms, as well as features with security implications such as payment or scripting. What mechanisms are needed and what data needs to be provided? Also, depending on how the metadata is made available, it may or may not be possible to support decentralized approaches to security.

The next few sections first introduce the W3C Web of Things draft standard, focusing on the Thing Description metadata format. Then the security model for the WoT will be introduced, which includes a model of stakeholders, assets, attackers, and threats. Once this context has been established, we will discuss in detail the above four issues.

II. Web of Things

The Web of Things (WoT) [4] aims to provide interoperability between IoT devices. It does this by defining a metadata format, the WoT *Thing Description*, that can describe a wide range of IoT network interfaces. A Thing Description can describe the network interfaces of existing devices or can be produced and consumed by devices running a WoT runtime supporting a WoT scripting API that normalizes interactions with other devices with a common abstraction layer.

A. Architecture

The WoT architecture[4] defines three basic entities that can be organized into various configurations and topologies based on a concrete deployment scenario:

- A **WoT Thing** represents a physical or virtual IoT device and exposes a network-facing API for interaction. Each WoT Thing has an associated Thing Description (TD)[3]. A TD encodes a set of metadata describing relevant information about a Thing, such as semantic categorization, available interactions, and communication and security mechanisms. A typical example of a WoT Thing might be a garage door controller. Such a controller would provide a number of actions that can be performed on a garage door, i.e. *open*, *close*, etc. and would provide network interfaces to invoke each of these. Typically a WoT Thing plays the network role of a server as it responds to but does not initiate interactions.
- A **WoT Client** is an entity that wants to perform an action on a WoT Thing. It is able to consume a TD provided by (or for) a WoT Thing and issue actions on the target's network interfaces. For example a WoT Client might be a browser or an application on a user's smartphone that allows the user to invokve one of the actions provided by the garage door controller.
- A **WoT Servient** can be viewed as a combination of a client and server: an entity that is both providing one or more WoT Thing interfaces (as a server) and at the same time is able to operate as WoT Client to invoke interactions on other WoT Things. An example of a WoT Servient would (a service running on a) home gateway device that acts as a WoT Client towards home appliance WoT Things (such as different lights and sensors) and also exposes some higher level virtual devices (such as the collection of all the lights in the living room) in form of additional WoT Things available for a WoT Client running on a user's smartphone.

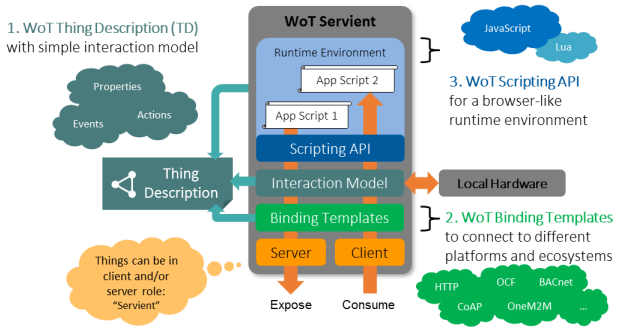

Fig. 1. WoT Servient architecture

Internally a typical architecture of a WoT Servient is shown in Figure 1. In addition to a Thing Description (TD) it also has WoT Binding Templates that can be used to instantiate a TD for a particular IoT protocol binding, such as OCF, HTTP(S), COAP etc. Internally a WoT Servient can also host a WoT Runtime and a WoT Scripting API. The WoT Scripting API is an optional component that allows implementing logic of an application Servient in a standardized way using a higher-level programming language (such as JavaScript).

B. Threat Model

Due to the large diversity of devices, use cases, deployment scenarios and requirements for WoT Things, it is impossible

to define a single WoT threat model that would fit every case. Instead we have created an overall WoT threat model [6] that can be adjusted by OEMs or other WoT system users or providers based on their concrete security requirements.

The threat model defines security-relevant WoT assets and a set of WoT threats on these assets. Threats can be in or out of scope based on the deployment scenario, security objectives, acceptable risks, and other factors. For example, in a smart home scenario involving WoT Things that record audio/video information, the privacy aspect would be very important and therefore this scenario would have high confidentiality requirements. On another hand, in an industry automation scenario involving WoT Things that control some safety-critical infrastructure, confidentiality might not be the highest priority. Instead, service availability and protection of the environment controlled by the Thing may be of topmost importance.

Being a distributed system brings additional complexity to the WoT Threat Model and the choice of relevant security mitigations. One cannot rely on standard communication infrastructure and protocols (like HTTPS) to provide an end-to-end security between all communicating parties. Instead WoT needs to support multiple security mechanisms (potentially nested or interdependent) that can be combined into single end-to-end security solution.

C. Typical Deployment Scenario

Figure 2 presents a typical WoT deployment scenario. A WoT Thing device together with a WoT Servient are placed inside the local network behind a forwarding proxy. A WoT Client that is located outside of this boundary wants to perform operations on the WoT Thing and WoT Servient. The available operations are given in corresponding Thing Descriptions. There are various ways Thing Descriptions provided by WoT Things could be made available to WoT Clients. In this case, the Thing Descriptions are uploaded by the WoT Thing and WoT Servient to a Thing Directory. A Thing Directory is a service which can be accessed by Clients to search for WoT Things it can communicate with. In order to do this the WoT Client first needs to issue a discovery query to the Thing Directory. Thing Directories can support semantic search capabilities, allowing Things to be discovered based on semantic annotations. Access to the search capabilities of a Thing Directory should only be provided to authorized clients, as with any web service. Upon obtaining a Thing Description, the WoT Client needs to make sure it has all the necessary credentials to authenticate to the forwarding proxy (if secure authentication on proxy is enabled), the WoT Servient and in some cases even to the end WoT Thing. All required information about these potentially different credentials should be provided in the obtained Thing Description.

III. Related Work

Some relevant prior work: State-of-the-Art and Challenges for the Internet of Things Security [2]. The Industrial Internet of Things Security Framework [7]. IoT Security Foundation Best Practices Guidelines [1].

IV. Risks and Opportunities

A. Local Links

Several practical pitfalls exist that can make it difficult to combine best practices for security with systems defined by the WoT in practice. Many of these issues arise when trying to access IoT devices that are located on a "local" network, for example behind a NAT in a Smart Home use case.

Naturally when one hears of the term "Web of Things" the assumption is that HTTP (or ideally, HTTPS) will be used to access IoT devices. This may or may not be the best choice for a variety of reasons. On the positive side, use of HTTP allows devices acting as HTTP servers to be accessed directly from web browsers and indeed to provide their own user interfaces via HTML. To provide better security, one would naturally want to use HTTPS rather than HTTP for this purpose. Unfortunately HTTPS and the certificate system behind it has been designed for globally accessible web sites, not devices behind NATs that may or may not have a globally accessible address and may or may not even be connected to the internet. In particular, certificate revocation checks will not work if the network both devices are on is not connected to the internet (for example, if an adhoc network is used, with the device acting as an access point) and certificates will not generally be able to tie the identity of a device to a particular URL.

Both of these problems can be avoided by using a cloud proxy or mirror (digital twin) for the device. In that case a server in the cloud is used as an intermediary. Unfortunately, this requires an active internet connection even to use local devices, and is also relatively high latency and bandwidth-inefficient.

Other secure protocols, like CoAPS, do not have the same assumptions as HTTPS and can be used more easily in an segmented network. The WoT, despite its name, can also work with these standards.

However, even with CoAP, there is an additional issue: the URLs used to access the same device can vary depending on whether the device is accessible on the local network or should be accessed via a global URL (either via NAT port forwarding, a proxy, or a digital twin). A Thing Description can include multiple links for each interaction, so in theory both local and global links can be included in a single Thing Description. However, a user of a Thing has not easy way to tell if it is on the same local network as a Thing, and if not, the "local" links won't work... or will connect to a different device. Another approach would be for the Thing Directory to return a modified Thing Description with local links to clients that it knows are on the same local network. The first approach, multiple links, has the problem of broken links (and possibly even connecting to the wrong device, if some other mechanism is not used to control access). The second approach raises the possiblity of a malicious Thing Directory, or a network attacker spoofing a Thing Directory, to redirect clients to fake devices.

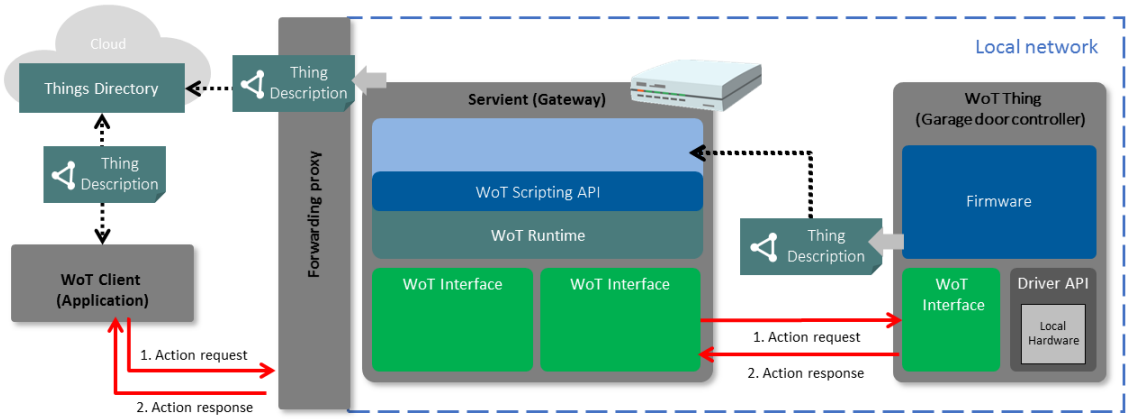

Fig. 2. Typical WoT deployment scenario

B. Vulnerability Scanning

The purpose of the Thing Descriptions is to enable easier discovery and use of devices. However, the flip side of this is that it may also become easier for attackers to discover vulnerable devices, or to infer private information simply from the type of devices available.

The first line of defense is to protect access to discovery services such as Thing Directories. If Thing Directories can only be accessed by authorized users, a number of types of attacks become more difficult. Since a Thing Directory is simply a web service, it can be protected with normal web service authorization mechanisms. However, a given system may provide other means of discovery, such as broadcast responses or extended DNS entries. These may not be as easy to protect. Note that in order to support a protected Thing Directory a protected onboarding process is needed to associate devices with a given Thing Directory and of course to provide authorized users with appropriate credentials to access the Thing Directory.

Assuming an attacker can access Thing Descriptions, however, they may be able to exploit them in various ways. First, they could try to analyze the interactions available themselves for known vulnerabilities. While the Thing Descriptions intentionally omit information about the software stack providing the service, an attacker may be able to use fingerprinting to associate a particular Thing Description to a particular device with known vulnerabilities. The vulnerabilities may not even be over the network; for instance, an attacker may search for "smart locks" that can be defeated with a known physical attack. The flip side of this, however, is that a System Maintainer can apply the same tools to scan for devices with vulnerabilities, in order to identify devices at risk so that mitigations can be put in place (such as scheduling updates to a device's firmware). Unlike a malicious attacker, the System Maintainer also has the advantage that they can legitimately access *all* Thing Descriptions in a system.

Even if an attacker cannot determine vulnerabilities, they may still be able to determine personal information about a user by the kinds of devices they have installed. For example, knowing that someone has a baby monitor lets you infer that they probably have a young child. Semantic tags make this information explicit but even without tagging, fingerprinting may be able to associate a set of interactions with a specific class of devices. The mitigation of this kind of attack is to protect the Thing Descriptions themselves and make them available *only* to authorized users.

C. Endpoint Adaptation

In a typical multistandard IoT system, bridges may be needed to connect devices conforming to different standards. For example, to connect to an AllJoyn device from a controller designed to connect to OCF devices, an OCF-to-AllJoyn bridge may be needed to translate both the protocol and the payload. In general, multiple bridges could be involved: the apparent AllJoyn device could in fact be a oneM2M device being made available over yet another bridge.

Unfortunately bridges introduce a potential security vulnerability. If the bridge devices can be compromised, they have full access to the data being carried to and from the device can can stage a variety of attacks: modifying or deleting data, injecting false data and events, or privacy invasion.

The WoT enables a way around this problem using end-to-end encryption. Basically, rather than adapting payloads in a point-to-point fashion, adaptation should take place at one

of the endpoints, ideally the one with greater capability. The endpoint doing the adaptation should look at the metadata for the target endpoint, adapt its payload for that target, and then use end-to-end encrypted communication. It may still be necessary to use bridges in between to adapt protocols (for example, bridging from HTTPS on the internet to CoAPS over a local network) but the payloads can remain encrypted.

D. Secure Discovery

One of the benefits provided by the WoT approach is to enable the use of powerful semantic searches during discovery. However, this introduces several issues related to security.

First of all, semantic searches can be abused to create DoS attacks. If arbitrary SPARQL endpoints are provided by Thing Directory services, semantic searches can be specified that can consume large amounts of resources, rendering the Thing Directories unavailable to other users. This can be mitigated by either limiting the power of searches that can be specified, or my limiting the amount of processing power that can be used in a specific search.

In the first approach, rather than providing a full SPARQL endpoint, a directory services may provide a more specialized search interface that only allows searches for a conjunction of terms and does not allow, for instance, use of arbitrary inference rules or use of other SPARQL endpoints in a federated search. The problem with this approach is that by limiting the power of the searches that may be specified, we are also limiting the potential benefit.

The second approach to mitigate DoS attacks via pathologically expensive semantic queries is to use an architecture for the search engine that can enforce resource quotas on individual queries. For example, each search could be spawned in a new process and Linux process limits could be used to enforce processing quotas. In addition, the number of searches that can be processed at the same time would have to be limited to avoid creating too many processes. If either limit is exceeded the server would return an appropriate error code. If the query processing quota is exceeded, the client would have to reformulate their query. If the server is too busy, the client would have to retry the query later.

Of course creating a new process for each search would be expensive, so ideally the search engine itself would have lighter-weight mechanisms to track resource consumption and enforce quotas.

There is another class of risks with semantic searches: inferencing. Semantic searches can infer information that is not explicitly present in the database: that is point! However, it is possible that an attacker could use this capability to invade someone's privacy. For example, they could infer personal information, such as marital status, from the ownership of certain combinations of devices. While it is difficult to prevent inference attacks in general, we should at least design semantic search engines to not use data for inference that would not be directly available to the person posing the query in the first place [8], [9]. As with the other class of risks we mentioned, mitigation requires restricting the distribution of Thing Description data to authorized users only.

E. Enabling Distributed Security

Designing the correct format for the TD security metadata field isn't a straightforward task. Many things must be taken into account: various WoT (and more generic IoT) deployment scenarios and configurations, underneath networking protocols and their security mechanisms, overall scalability etc.

A data packet in a WoT network can go over many intermediate entities, including gateways, proxies etc. Depending on the concrete setup, some of these entities might want to perform authentication and authorization of the requester and therefore the security metadata in TD should carry all needed information, i.e. what types of credentials are required by each entity, how to obtain them etc.

For example let's consider the deployment scenario in Figure 2. Suppose the forwarding proxy requires authentication of any request before passing it to the local network. Additionally let's assume that the WoT gateway performs authentication at least for some actions provided by the WoT Thing, i.e. opening a garage door (critical action) requires a certain credential. Both of these authentication methods (potentially fully different) and all associated information must be specified in the TD that the WoT Client receives from the Things Directory or otherwise the request to open the garage door cannot be performed.

Generalizing the above example, there might be N sets of fully independent security metadata that should be possible to embed into a Thing Description. Moreover, when a Thing Description is composed, these sets can be provided by separate entities: a gateway might only specify the security metadata for accessing actions defined in TD, while the forwarding proxy adds the metadata required for successful authentication of incoming requests. This means that there has to be a way to limit what security meatadata is allowed to be provided by what entity and also it must be possible for the WoT client to verify the overall integrity of resulting TD.

V. Conclusion

We have given a summary of the W3C Web of Things draft standard, with a focus on the Thing Description. The Thing Description provides a descriptive approach to interoperability, which contrasts with the prescriptive approach of most other standards. While a prescriptive approach is useful, for example to enforce minimum security requirements, a descriptive approach can support brownfield devices and can also make it easier to integrate devices that conform to different prescriptive standards.

However, beyond the basic issue of integrating devices with different levels and mechanisms for security, which will arise with any multistandard IoT system, the use of descriptive metadata raises several new security risks and opportunities. We have discussed four such points: multidevice vulnerability analysis, end-to-end security enabling, secure discovery for semantic interoperability, and (potentially decentralized) security mechanism enabling.

ACKNOWLEDGMENT

The security and threat model presented here was developed in the W3C Web of Thing WG as part of the *Web of Things (WoT) Security and Privacy Considerations* document [6]. Please see the associated github site for a list of additional contributors.

REFERENCES

[1] "IoT security foundation best practice guidelines," IoT Security Foundation, Tech. Rep., May 2017. [Online]. Available: https://iotsecurityfoundation.org/best-practice-guidelines/

[2] O. Garcia-Morchon, S. Kumar, and M. Sethi, "State-of-the-art and challenges for the internet of things security," Working Draft, IETF Secretariat, Internet-Draft draft-irtf-t2trg-iot-seccons-08, Oct. 2017. [Online]. Available: http://www.ietf.org/internet-drafts/draft-irtf-t2trg-iot-seccons-08.txt

[3] S. Käbisch and T. Kamiya, "Web of things (WoT) thing description," W3C, W3C Working Draft, Sep. 2017. [Online]. Available: https://www.w3.org/TR/2017/WD-wot-thing-description-20170914/

[4] K. Kajimoto, U. Davuluru, and M. Kovatsch, "Web of things (WoT) architecture," W3C, W3C Working Draft, Sep. 2017. [Online]. Available: https://www.w3.org/TR/2017/WD-wot-architecture-20170914/

[5] J. Manyika, M. Chui, P. Bisson, J. Woetzel, R. Dobbs, J. Bughin, and D. Aharon, "The internet of things: Mapping the value beyond the hype," McKinsey Global Institute, Tech. Rep., Jun. 2015. [Online]. Available: https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world

[6] E. Reshetova and M. McCool, "Web of things (WoT) security and privacy considerations," W3C, W3C Note, Sep. 2017. [Online]. Available: https://www.w3.org/TR/2017/WD-wot-security-20171116/

[7] S. Schrecker, H. Soroush, J. Molina, M. Buchheit, J. LeBlanc, R. Martin, F. Hirsch, A. Ginter, H. Banavara, S. Eswarahally, K. Raman, A. King, Q. Zhang, P. MacKay, and B. Witten, "The industrial internet of things security framework," Industrial Internet Consortium, Tech. Rep. IIC:PUB:G4:V1.0:PB:20160926, Sep. 2016. [Online]. Available: http://www.iiconsortium.org/IISF.htm

[8] B. Thuraisingham, "Security standards for the semantic web," *Computer Standards and Interfaces*, vol. 27, no. 3, pp. 257 – 268, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0920548904000686

[9] Z. Xia, Y. Zhu, X. Sun, and L. Chen, "Secure semantic expansion based search over encrypted cloud data supporting similarity ranking," *Journal of Cloud Computing*, vol. 3, no. 1, p. 8, Jul 2014. [Online]. Available: https://doi.org/10.1186/s13677-014-0008-2