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 five major implications for security. First, the need for local links and, more generally, disconnected and segmented networks in IoT raises several practical considerations regarding what metadata should be provided. Second, metadata allows for system-wide vulnerability analysis, which can be both a risk and an opportunity. Third, metadata can enable end-to-end security in multistandards networks, avoiding exposing unencrypted data within bridges otherwise needed for adapting standards pairwise. Fourth, metadata supports service and device discovery, which raises the question of how to limit discovery to authorized agents. Fifth, 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.**

I. Introduction

The economic impact of the IoT will be strongly dependent on how well devices from different manufacturers can interoperate. Very often wide interoperability is taken for granted when estimating the business benefit of IoT. Conversely, if IoT devices do *not* interoperate, a recent study [11] 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.

Full interoperability is hard to achieve. There are currently many competing IoT standards under development, each of which is attempting to address interoperability. Most of these standards are prescriptive. In a prescriptive standard, devices are validated against specific requirements. Typically the goal is that devices validated against a particular standard will interoperate with other devices also validated against that standard. It is also 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. If one standard comes to dominate bridging will be unnecessary, but so far such unification seems to be elusive. Unification is complicated by the divergent requirements in different but overlapping IoT subdomains. As an example of divergent requirements, some application areas require real-time response (bounded low-latency responses) and others require low power (requiring long sleep times) or wireless access (in which it is very difficult to acheive real-time guarantees due to the possibility of interference or collisions). Basic interaction patterns can also vary: some standards use a web-inspired resource-centric (RESTful) client/server interaction model, while others use a message-centric publish-subscribe model.

In addition to the issue of interoperability between modern prescriptive IoT standards, there are always going to exist devices that follow older standards or specifications. 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 devices conforming to the latest standard. This is the "brownfield" problem. Moreover new devices are being deployed today and for the forseeable future that have not been validated against any particular IoT standard, even if they use other *de jure* or *de facto* standards such as JSON, HTTP, or MQTT.

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, using a set of interaction patterns that includes as a subset both resource-centric and message-centric interaction models. 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.

NDSS '18, 18-21 February 2018, San Diego, CA, USA

http://dx.doi.org/10.14722/ndss.2018.23xxx

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, have different trust models, have different levels of acceptable risk, and may use different security mechanisms. This may cause integration challenges, even if the necessary information is provided in the metadata.

It is vital that new standards are analyzed for security risks before, not after, deployment. The goal of this paper is to start the discussion of the risks and opportunities presented by WoT metadata and, more generally, the descriptive approach to interoperability. ***We do not present solutions, but rather a set of problems.*** We have identified at least five such problems:

1. Local Links: End-point IoT devices can be deployed in many ways: starting with deployments in closed, only locally accessible, networks, extending to systems on local networks but behind a proxy or a firewall providing access to the global internet, and ending with deployments on a globally addressed network. In fact, it may be possible to reach the same device via multiple paths. As a result the metadata provided by IoT devices should allow for expression of different ways (links) to reach each device and a way to securely update this information when the deployment setup changes. Security mechanisms with assumptions about global connectivity also may not operate correctly in disconnected networks. In IoT deployments, even nominally fully connected systems may have to deal with frequent loss of full connectivity.

2. 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.

3. Endpoint Adaptation: Metadata enables end-to-end security in networks of IoT devices using multiple standards. Pushing payload adaptation to endpoints is an enabler for end-to-end encryption of payloads via object security or tunnelled transports (or both). 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.

4. Secure Discovery: Information about how to use a service, and ideally even its existence, should not be disclosed to entities 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?

5. Enabling Distributed Security: Metadata may be provided to enable specific security mechanisms, as well as to support features with security implications such as payment or scripting. What specific mechanisms are needed and what data needs to be provided in order to satisfy the overall goal of interoperability? Depending on how the metadata is made available, it may or may not be possible to support decentralized approaches to security. In general, the mechanism used to provide the metadata is an essential component of the security architecture.

The next few sections first introduce the W3C Web of Things draft standard, focusing on the Thing Description metadata format. Then the high-level WoT Threat model will be introduced, which includes a model of stakeholders, assets, attackers, and threats, as well as a typical WoT deployment scenario. Once this context has been established, we will discuss in detail the above five issues.

II. Web of Things

The Web of Things (WoT) [10] 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 Web of Things (WoT) architecture [10] defines three basic entities that can be organized into various configurations and topologies based on a concrete deployment scenario:

• **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) [9]. A TD encodes a set of metadata describing relevant information about a Thing, such as semantic categorization, available interactions, and communication and security mechanisms. Typically a WoT Thing plays the network role of a server as it responds to but does not initiate interactions. *For example:* A WoT Thing might be a garage door controller. Such a controller would provide a number of interactions that can be performed on a garage door, i.e. *open*, *close*, etc. and would provide network interfaces to invoke each of these.

• **WoT Client:** An entity that wants to perform an action on a WoT Thing. It is able to consume a TD provided by (or for) another WoT Thing and invoke interactions 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 invoke one of the interactions provided by the above garage door controller.

• **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. *For example:* A WoT Servient might be a service running on a home gateway device that provides a generalized "lock up" service (including closing the garage door, but also arming the home alarm, securing door locks, turning off lights, etc.) with its own network interface.

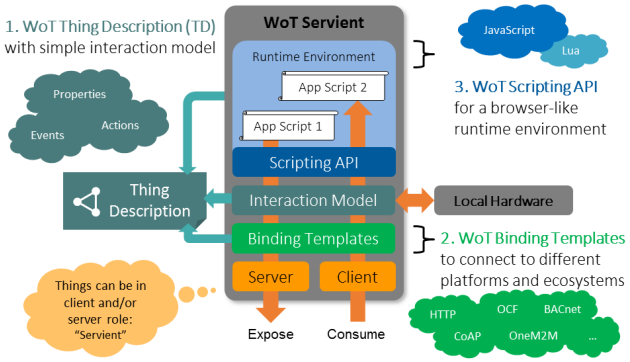

Fig. 1. WoT Servient architecture

Internally a typical architecture of a WoT Servient is shown in Figure 1. In addition to a Thing Description (TD), a full WoT Servient also has WoT Binding Template metadata that can be used to instantiate a TD for a particular IoT protocol binding, such as OCF, HTTP(S), COAP etc.

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 the logic of an application Servient in a standardized way using a higher-level programming language (the current WoT draft focuses on JavaScript). In this document, however, we focus on the security implications of the metadata alone, and do not consider further the security implications of the Scripting API (which raises many issues such as application isolation, prevention of DDoS attacks, and so on).

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 *framework* [14] 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 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, privacy would be considered important and therefore this scenario would have high confidentiality requirements. On the other hand, in an industry automation scenario involving WoT Things that control some safety-critical infrastructure, service availability and protection of the environment controlled by the Thing may be of the most importance, not privacy.

Deploying the WoT in a distributed system brings additional complexity to the WoT Threat Model and the choice of relevant security mitigations. In particular we cannot rely on single standard communication infrastructure and protocol (like HTTPS) to provide end-to-end security between all communicating parties. Instead we need to support multiple security mechanisms (potentially nested or interdependent) that can be combined into a 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 interactions on the WoT Thing and WoT Servient. The available interactions 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

The WoT approach applies to a wide diversity of IoT devices and use cases. It has to take be able to support best practices in IoT security which are however still rapidly evolving. Recently some attempts have been made to define best practices for IoT security. The IETF Thing-to-Thing Research group has a RFP under development, *State-of-the-Art and Challenges for the Internet of Things Security* [8], which includes a threat model similar to what we have defined for the WoT. However, this threat model does not consider the importance of protecting access to descriptive metadata. The Industrial Internet of Things has published a comprehensive *Security Framework* [15]. This is useful, but focuses on industrial use cases. The IoT Security Foundation has published a *Best Practices Guidelines* [5] document as well. All three documents consider various additional factors we do not have space to go into here, such as trust management over the lifecycle of the device.

There are several survey papers [4], [18], [7] that attempt to describe the current IoT security challenges and provide suggestions for the future work in the area.

IV. Risks and Opportunities

A. Local Links

Many issues arise when trying to access IoT devices that are located on a "local" network, for example behind a NAT in a

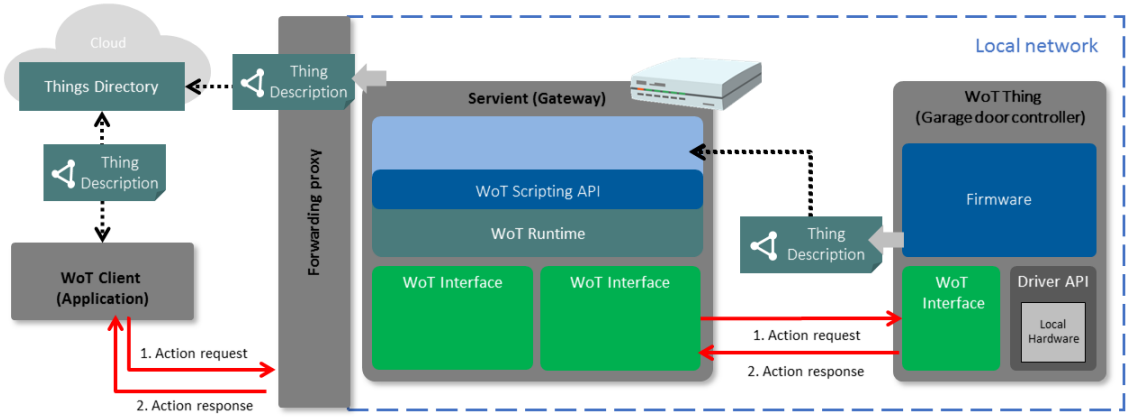

Fig. 2. Typical WoT deployment scenario

Smart Home use case. In addition, devices may be accessible over multiple routes and protocols.

When one hears the term "Web of Things" the assumption is that HTTP will be used to access IoT devices. While leveraging web standards was one of the original goals of the Web of Things concept [13], the current WoT draft supports other choices as well, such as CoAP and MQTT. However, use of HTTP conveniently allows Things acting as web servers to be accessed directly from web browsers to provide human user interfaces via HTML. To provide better security, one would naturally want to use HTTPS rather than HTTP. Unfortunately the way HTTPS is supported in web browsers has been designed for globally accessible web sites, not devices behind NATs that may or may not have a globally accessible address and may not even be continously connected to the internet. In particular, certificate revocation checks as currently implemented in web browsers will not work if the network both devices are on is not connected to the internet (for example, if an *ad hoc* 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, has relatively high latency, and is bandwidth-inefficient. Various mechanisms have been explored to deal with this issue [2] but no generally accepted solution has been adopted. Therefore a metadata standard needs to be able to deal with a variety of approaches.

The issues noted above are not directly relevant if humans do not need to communicate directly with a Thing using a web browser. In machine-to-machine (M2M) use cases we have more flexibility in how we establish trust and validate connections.

Even for M2M use cases there is another 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 no easy way to tell if it is on the same local network as another target Thing, and if not, the "local" links won't work. Another approach would be for the (trusted and authenticated) Thing Directory to return a modified Thing Description with local links to clients that it knows are on the same local network.

B. *Vulnerability Scanning*

The purpose of the Thing Descriptions is to enable easier discovery and use of devices. 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. In order to support a protected Thing Directory a protected onboarding process is needed to associate devices with a given Thing Directory. A mechanism is also needed to provide authorized entities 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 a bridge is compromised, an attacker would have full access to the data being carried to and from the bridge and 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 endpoint end-to-end payload adaption. This is complementary to but distinct from object security [12].

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: either end-to-end (tunnelled) transport security, object security, or both.

If multiple transport protocols are used, such as a combination of HTTP/TLS and CoAP/DTLS, bridging those protocols may create another compromise possibility. A mechanism supporting object security such as JOSE [1] should then be used in combination with endpoint adaptation, but for this to work the target (typically constrained) endpoint needs to support it. Object security also supports secure state caching in the cloud or gateways, an important consideration for devices that may need a digital twin to handle transactions for them while they are in standby, conserving battery, or otherwise offline.

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 by limiting the processing power that can be used in a specific search.

In the first approach, rather than providing a full SPARQL endpoint, a directory service 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. Unfortunately, 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 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. 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. By their nature, semantic searches can infer information that is not explicitly present in the database. It is possible that an attacker could use this capability to invade someone's privacy. For example, they could infer personal information (e.g. relationship status) from the ownership of certain combinations of devices (e.g. multiple toothbrushes). 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 entity posing the query in the first place [16], [17]. 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

Choosing what security metadata to include in a TD is not a straightforward task. Many things must be taken into account: various deployment scenarios and configurations, underlying networking protocols and their security mechanisms, overall scalability, system security priorities, and so on.

A data packet in a WoT network can go over many intermediate entities, including gateways and proxies. Some of these entities might need to perform authentication and authorization of the requester. The security metadata in the

TD should indicate what types of credentials are required by each entity and how to obtain them.

For example, consider the deployment scenario in Figure 2. Suppose the forwarding proxy requires authentication of any request before passing it to the local network. Additionally assume that the WoT gateway performs authentication at least for certain critical interactions provided by the WoT Thing (for example, unlocking a door). Both of these authentication methods (potentially different) and associated information must be specified in the TD that the WoT Client receives from the Thing Directory. The WoT gateway can perform the authentication of the WoT Client in a number of ways depending on what standards the target devices support. For example, the OCF Security Specification [6] describes a number of ways IoT devices can perform authentication based on either symmetrical or asymmetrical credentials (including certificates). However, provisioning credentials in advance is not ideal for a distributed network such as the WoT is intended to support. Another possibility is to use a token-based authentication mechanism, such as OAuth 2.0 or Proof-of-Possession tokens [3]. Token-based authentication allows a WoT Client to dynamically request a token from a remotely located authorization server and present it for authentication to a WoT gateway or Thing. This method has its own risks, as tokens need to be protected from interception.

Generalizing the above example, there might be N sets of fully independent security metadata necessary in 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 interactions defined in a 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 metadata is allowed to be provided by what entity. Also, it must be possible for the WoT client to verify the overall integrity of the 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.

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.

With the goal of starting discussion, we have presented five such problems: security over local and multiple links, 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 [14]. Please see the associated github site for a list of additional contributors. Barry Leiba provided feedback on an early draft.

References

[1] "JOSE: JSON Object Signing and Encryption," 2014. [Online]. Available: https://datatracker.ietf.org/wg/jose/charter/

[2] "HTTPS in Local Network," W3C, Community Group, 2017. [Online]. Available: https://www.w3.org/community/httpslocal/

[3] "IETF Authentication and Authorization for Constrained Environments (ACE)," Nov. 2017. [Online]. Available: https://tools.ietf.org/pdf/draft-ietf-ace-oauth-authz-09.pdf

[4] "IoT 2020: Smart and secure IoT platform," 2017. [Online]. Available: http://www.iec.ch/whitepaper/pdf/iecWP-loT2020-LR.pdf

[5] "IoT Security Foundation Best Practice Guidelines," IoT Security Foundation, Tech. Rep., May 2017. [Online]. Available: https://iotsecurityfoundation.org/best-practice-guidelines/

[6] "The OCF Security Specification," Jun. 2017. [Online]. Available: https://openconnectivity.org/specs/OCF_Security_Specification_v1.0.0.pdf

[7] E. Fernandes, A. Rahmati, K. Eykholt, and A. Prakash, "Internet of Things Security Research: A Rehash of Old Ideas or New Intellectual Challenges?" *IEEE Security and Privacy*, vol. 15, no. 4, pp. 79–84, 2017.

[8] O. Garcia-Morchon, S. Kumar, and M. Sethi, "State-of-the-Art and Challenges for the Internet of Things Security," IETF Secretariat, Internet-Draft, Oct. 2017. [Online]. Available: http://www.ietf.org/internet-drafts/draft-irtf-t2trg-iot-seccons-08.txt

[9] 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/

[10] 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/

[11] 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

[12] J. Mattsson, G. Selander, and G. A. Eriksson, "Object Security in Web of Things," in *W3C Workshop on the Web of Things: Enablers and services for an open Web of Devices*. W3C, Jun. 2014. [Online]. Available: https://www.w3.org/2014/02/wot/papers/mattsson.pdf

[13] B. Ostermaier, F. Schlup, and M. Kovatsch, "Leveraging the Web of Things for Rapid Prototyping of UbiComp Applications," in *UbiComp 2010: Ubiquitous Computing, 12th International Conference*, Nov. 2010, pp. 375–376. [Online]. Available: http://doi.acm.org/10.1145/1864431.1864443

[14] 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/

[15] 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., Sep. 2016. [Online]. Available: http://www.iiconsortium.org/IISF.htm

[16] 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

[17] 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

[18] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of IoT Systems: Design Challenges and Opportunities," in *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '14. IEEE Press, 2014, pp. 417–423. [Online]. Available: http://dl.acm.org/citation.cfm?id=2691365.2691450