Network Requirements for Descriptions of Things
And Some Additional Thoughts

Eliot Lear
Cisco Systems

Abstract

Device data models are employed not just by devices themselves, and not just by "controllers", but by management systems that configure device access to the network and never touch the end nodes themselves. The amount of knowledge required by the network, however, may be considerably less than what an application layer controller may need. Furthermore, it is not sufficient for the network to simply know what sort of standard function a device is intended to do, but rather the particulars of what exactly the device is, and what other functions it may support. This paper considers data models in the light of security requirements of Things, and we will approach data models from that real world concern.

Introduction

Cisco Systems predicts that by 2020 there will be some 50 billion networked devices, many of which will be constrained to a limited number of purposes. As the uses of TCP/IP grow, so do the risk of their use. With baby monitors[1], security systems[2], and refrigerators[3] being broken into, the ability of a home owner or even a savvy firewall developer to protect networks against unwanted intrusion has diminished.

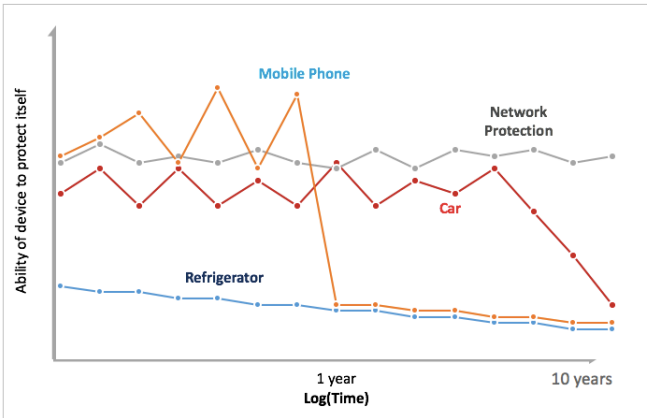

Figure 1: Security of Things over time

It must also further be recognized that the ability of manufacturers to maintain protection of their devices may be limited to nonexistent.

Figure 1 shows a conceptual diagram that demonstrates that over varying periods of time, depending on the class of device, it may (a) not be able to ever sufficiently protect itself alone or (b) only be able to protect itself, while it is supported.[4]

Even when a device is supported, additional firewall rules can reduce the risk of intrusion, between the time when a vulnerability is discovered and when it can be expected to be fixed.

Manufacturer Usage Descriptions (MUD)

Because so many devices will be hitting the network, the sheer number of types of devices mandates some means to communicate what sort of network protection is desired by the manufacturer. The approach to facilitate this information exchange is known as **manufacturer usage descriptions** or **MUD**.[5] MUD makes use of well established existing mechanisms, such as IEEE 802.1X, IEEE 802.1AR certificates, DHCP, and HTTPS. The key area for this workshop to consider, however, is the language that manufacturers use to describe their needs of the network. The primary function of MUD, at the moment is to

provide a means for access points, switches and firewalls to instantiate access lists (ACLs) that block or permit communications at layers 3 and 4. While syntax may vary, the underlying semantics of these descriptions has been stable for over a generation.

It is readily apparent that for IoT-like devices, however, mere IP address-based ACLs are not sufficient to allow necessary access. A common design pattern, for instance, is that a device is managed by one or more controllers, and perhaps nothing else. As such, MUD introduces an extension on the ACL model that contains meta-information such as "controller" or "manufacturer".[6]

Through MUD we are able to reduce the threat surface of device to those that are likely to have business talking to it. However, there is only limited (if any) semantic understanding. In contrast, controllers have a requirement of deep semantic understanding of devices. Roughly speaking this follows the nature of the end-to-end model. When a middlebox does have deep knowledge of certain devices, it is acting as much as a controller or an auditing function of that controller with more complete application knowledge than were it a classic network component. Such an auditing function might validate the appropriateness of a parameter in a given circumstance (e.g., "issue an alarm if the vat gets about 52°C").

How Many Description Languages Do We Need?

Figure 2: John Romkey and the Internet Toaster

At Interop 1989, John Romkey and Simon Hackett demonstrated that it was possible to control a toaster with SNMP. It did this with one variable that controlled power to the device. Since that time, control of networked devices has ranged from the simple to the extremely complex. Actuators of various sorts have but a few configurable or reportable states and alarms, whereas the Thing **using** the actuator or several thousands of them, is considerably more complex. Conversely, in the example of MUD, the device being configured is a network element, where the communication pattern does not rise above the transport layer (L4). As the author pens this paper, a YANG-based ACL schema is nearing completion at the IETF.[7] It is important to note that this YANG-based approach primarily requests a particular network element behavior in order to service the device, but **not** the end node's behavior. A YANG-based approach is sufficiently rich to direct network configuration. Whether YANG schema can address the broader needs of IoT depends very much on two factors: availability of tooling, and the ability of the standardization ecosystem to drive models to completion in a timely fashion. Both of these factors leave open the possibility that YANG-based schema may not be the right choice for Things, in particular.

Your Class Ain't Nothin' But Trash: Security Considerations[8]

In terms of properties of the entire system, a strictly object-oriented class-based approach is useful for both code reuse and interoperability purposes. While it is the case that vulnerabilities often manifest themselves in commonly used libraries that implement classes, bugs occur in specific versions of software on specific of devices. The cross-product of the different characteristics that could be instantiated in any particular device is already so enormous as to not be able to not be possible to enumerate. Even if a device communicated every last version of a library it used, and even if the network knew all vulnerabilities associated with all libraries, it would **still** not be possible to describe those vulnerabilities that

are **not** in libraries. A security system can provide timely protection based on each individual model from individual manufacturers at the cost of the device emitting a URI with that information.

What Needs to Be Standardized?

Previous experience demonstrates that grand unified models don't succeed. However, modularity and reuse do. Access to models also helps. A clearinghouse of usable models is more important than a single grand-unified model that is stretched beyond the vision of the original designers (or those who follow after them). The important metric is deployment, something that can be measured only once models have had a chance to succeed or fail. Model diversity presents a natural tension with interoperability, a goal to be sure, which plays out particularly with MUD because, as envisioned, there can be only limited negotiation. The cautionary note here is that those who are model experts should take care to guide – but not block – work from subject matter experts.

Although standardization of models ought not be a prerequisite for their use, it is important that the various models implemented be uniquely identified, and that any addressable identifiers they use either be well scoped or unique. Scoping can reduce the size of any registries that are required. Uniqueness can be achieved through the use of existing name spaces, such as the DNS-based URIs or URNs, or through new registries.

Conclusion

There are multiple device data model requirements. In the case of the network, a description must express the set of classes of systems that are intended to be communicated with, such as controllers, the various TCP/UDP services that are intended to be available, and not much more. In the case of a controller, the description must be considerably richer. There is no evidence that a new descriptive approach is necessary. Rather the opposite: fewer descriptive approaches that allow for more models will allow for a more organic approach that is based on the work of subject matter experts. Ambiguity of the namespace and delay in development should be avoided.

---

[1] Stanislav, M., Beardsley, T., "HACKING IoT: A Case Study on Baby Monitor Exposures and Vulnerabilities", Rapid7, September, 2015.

[2] Zetter, K., "How Thieves Can Hack and Disable Your Home Alarm System", July, 2014. http://www.wired.com/2014/07/hacking-home-alarms/

[3] Leyden, J., "Samsung smart fridge leaves Gmail logins open to attack", The Register, August, 2015.

[4] There is history and logic behind each of these example curves. In one case, a manufacturer announced its intent not to support a particular operating system at almost the same time that a new vulnerability was announced. In the case of vehicles, this curve is based on recall requirements in certain jurisdictions.

[5] Lear., E., "Manufacturer Usage Description Framework", draft-lear-mud-framework-00.txt (work in progress), January 2016.

[6] Lear, E., Manufacturer Usage Description YANG Model, draft-ietf-lear-netmod-mud-00, January 2016.

[7] Bogdanovic, D., "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-06 (work in progress), December 2016.

[8] With apologies to The Clovers, "Your Cash Ain't Nothin' But Trash", 1954; and to Peter Honeyman, "Your Cache Ain't Nothing But Trash", USENIX 1992.