Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 24, 2016

E. Lear
Cisco Systems

January 21, 2016

Manufacturer Usage Description Framework
draft-lear-mud-framework-00

Abstract

A key presumption of the Internet architecture has been that devices are general purpose computers. By constraining the set of devices that connect to the Internet to non-general purpose devices, we can introduce a set of network capabilities that provides an additional layer of protection to those devices. One such capability is the Manufacturer Usage Description (MUD). This work builds on many existing network capabilities so as to be easily deployable by all involved. The focus of this work is primarily, but not exclusively, in the realm of security; and again primarily, but not exclusively, relating to smart objects.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 24, 2016.

Table of Contents

1. Introduction

The Internet has largely been constructed on general purpose computers; those devices that may be used for a purpose that is specified by those who buy the device. [RFC1984] presumed that an end device would be most capable of protecting itself. This made sense when the typical device was a workstation or a mainframe, and it continues to make sense for general purpose computing devices today, including laptops, smart phones, and tablets.

[RFC7452] discusses design patterns for, and poses questions about, smart objects. Let us then posit a group of objects that are specifically not general purpose computers. These devices therefore have a purpose to their use. By definition, therefore, all other purposes are NOT intended. The combination of these two statements can be restated as a manufacturer usage description (MUD) that can be applied at various points within a network. Although this memo may seem to stress access requirements, usage intent also consists of quality of service needs a device may have.

We use the notion of "manufacturer" loosely in this context, to simply mean the entity or organization that will state how a device

is intended to be used. In the context of a lightbulb, this might indeed be the lightbulb manufacturer. In the context of a smarter device that has a built in Linux stack, it might be integrator of that device. The key points are that the device itself is expected to serve a limited purpose, and that there may exist an organization in the supply chain of that device that will take responsibility for informing the network about that purpose.

The converse statement holds that general computing systems will benefit very little from MUD, as their manufacturers cannot envision a specific communication pattern to describe.

The intent of MUD is to therefore solve for the following problems:

- Substantially reduce the threat surface on a device entering a network to those communications intended by the manufacturer.

- Provide for a means to scale network policies to the ever-increasing number types of devices in the network.

- Provide a means to address at least some vulnerabilities in a way that is faster than it might take to update systems. This will be particularly true for systems that are no longer supported by their manufacturer.

- Keep the cost of implementation of such a system to the bare minimum.

No matter how good a MUD-enabled network is, it will never replace the need for manufacturers to patch vulnerabilities. It may, however, provide network administrators with some additional protection when those vulnerabilities exist.

1.1. A Simple Example

A light bulb is intended to light a room. It may be remotely controlled through the network; and it may make use of a rendezvous service of some form that an app on smart phone accesses. What we can say about that light bulb, then, is that all other network access is unwanted. It will not contact a news service, nor speak to the refrigerator, and it has no need of a printer or other devices. It has no Facebook friends. Therefore, an access list applied to it that states that it will only connect to the single rendezvous service will not impede the light bulb in performing its function, while at the same time allowing the network to provide both it and other devices an additional layer of protection.

1.2. Determining Intended Use

The notion of intended use is in itself not new. Network administrators apply access lists every day to allow for only such use. This notion of white listing was well described by Chapman and Zwicky in [FW95]. Programmatically profiling systems have existed for years as well. These systems make use of heuristics that take at least some time to assert what a system is.

A system could just as easily tell the network what sort of protection it requires without going into what sort of system it is. This would, in effect, be the converse of [RFC7488]. In seeking a general purpose solution, however, we assume that a device has so few capabilities that it will implement the least necessary capabilities to function properly. This is a basic economic constraint. Unless the network would refuse access to such a device, its developers would have no reason to implement such an approach. To date, such an assertion has held true.

If the network does not apply heuristics and a device is not capable of articulating what it needs from the network, perhaps there is a third approach that builds on capabilities already in both. There are four such potential capabilities for the network to determine what sort of client it has:

1. For those devices that are meant to operate in a secure environment [IEEE8021X] and [IEEE8021AR] provides a means for certificate-based device identification.

2. In the absense of DHCP in IPv6 (e.g., stateless address selection), [IEEE8021AB] can be used to learn the same information.

3. In the IP network context, every device needs an IP address. [RFC2131] specifies the dynamic host configuraiton protocol, necessary for all IPv4 and IPv6 implementations. Client use of a DHCP option would inform the network of what the device thinks it is, and provide a pointer to additional policy information.

4. Finally, for equipment that does not emit any information, it is possible for the access switch to proxy the information into the system.

With these capabilities, a device may impart some piece of information to the network. In the immortal words of David John Wheeler, "All problems in computer science can be solved by another level of indirection, except of course for the problem of too many indirections." Our means of providing this level of indirection is a

Universal Resource Identifier (URI) [RFC3986] that references a file put in place by someone who knows something about the device - the manufacturer. As we will later discuss, we can later relax whether it is indeed the manufacturer who is specifying the URI.

With a simple resolution of a URI, a file is retrieved. We are now to the point in the discussion where we have to decide how the manufacturer expresses intent. We have already stated that Things themselves have limited capabilities. Let us also assume that we in the networking business wish to stand on the shoulders of giants and also not reinvent the wheel. While such a wheel is not _perfectly_ rounded for our purposes, YANG models [RFC6020] and their derivative XML provide sufficient richness for the manufacturer to clearly state at least simple intent. They are thus our starting point.

1.3. Types of Policies

Once we know how to determine intended use and who can determine it, there is still the question of what that sort of policies can in fact be intended. At least initially, we envision that as a beginning host-level access policies. The manufacturer may specify either specific hosts or certain classes. An example of a class might be "devices of a specified manufacturer type", where the manufacturer type itself is indicated simply by the authority of the MUD-URI. Another example might to allow or disallow local access. Just like other policies, these may be combined. For example:

```
Allow access to host controller.example.com with QoS AF11
Allow access to devices of the same manufacturer
Allow access to and from controllers who need to speak COAP
Allow access to local DNS/DHCP
Deny all other access
```

To add a bit more depth that should not be a stretch of anyone's imagination, one could also make use of port-based access lists. Thus a printer might have a description that states:

```
Allow access for port IPP or port LPD
Allow local access for port HTTP
Deny all other access
```

In this way anyone can print to the printer, but local access would be required for the management interface.

Other non-access policies may be possible as well. For instance, suppose a manufacturer is able to make use of an authentication infrastructure. That could be specified in the usage description such that the details could be filled in by the controller. In

addition, QoS policies are sufficiently mature and ubiquitous as to be valuable in this context as well. And so for instance, for voice/video services:

Set QoS AF13 to SIP-GW.EXAMPLE.COM

The converse highlights a design consideration: policies that are articulated by the manufacturer must be ubiquitously understood, or they may not be applied. That is- applying half a policy is not safe.

2. The Manufacturer Usage Description Architecture

With these components laid out we now have the basis for an archicture. This leads us to ASCII art.

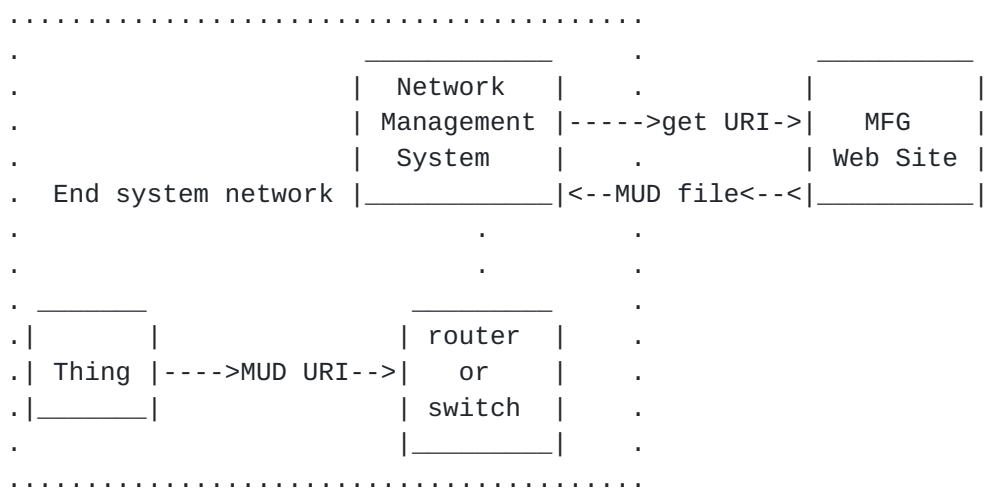

Figure 1: MUD Architecture

In the above diagram, the switch or router collects MUD URIs and forwards them to the network management system for processing. This happens in different ways, depending on how the URI is communicated. For instance, in the case of DHCP, the DHCP server might receive the URI and then process it. In the case of IEEE 802.1X, the switch would tunnel the URI to the authentication server, who would then process it.

The information returned by the web site is valid for the duration of the device's connection, or as specified in the description. Thus if the device is mobile, when it moves on, any configuration in the switch is removed. Similarly, from time to time the description may be refreshed, based on new capabilities or communication patterns or vulnerabilities.

The web site is run by or on behalf of the manufacturer. Its domain name is that of the authority found in the MUD URI. For legacy cases where Things cannot emit a URI, if the switch is able to determine the appropriate URI, it may proxy it, the trivial cases being a map between some registered device or port and a URI.

2.1. What does a MUD URI look like?

To begin with, MUD takes full advantage of both the https: scheme and the use of .well-known. HTTPS is important in this case because men in the middle could otherwise harm the operation of a class of devices. .well-known is used because we wish to add additional structure to the URI. And so the URI is specified in draft-lear-netmod-mud-pre0. It looks like this:

```
https://manufacturer.example.com/.well-known/mud/v1/model/version#extra
```

"model" represents a device model as the manufacturer wishes to represent it. It could be a brand name or something more specific. "version" provides a means to indicate what version the product is. Specifically if it has been updated in the field, this is the place where evidence of that update would appear. Once again, the field is opaque. From a controller standpoint, therefore, only comparison and matching operations are safe.

2.2. Communicating to the Manufacturer

We assume that the the manufacturer has at its disposal a web service running atop port 443 with standard HTTPS semantics, and that its capabilities are at par with today's web servers. We further assume that this web server has no semantic understanding itself of MUD. This poses us a particular challenge: either we are to cast in stone the model that is put in place, or we must find a mechanism by which the switch or its controller can choose an appropriate set of capabilities.

2.3. Using YANG-based XML

Because NETCONF is well distributed within network infrastructure and YANG has become the accepted way to generate schema for NETCONF, these we attempt to adapt the protocol and the modeling language, respectively. At some point in the near future, it will likely be the case that XML gives way to JSON[RFC7159]. YANG can be used for either, and so it seems even more appropriate to make good use of it. This work makes use of XML because of the breadth of toolsets available, and not for any love of angle brackets. That is subject to change.

The descriptions specified in MUD files should be based on relatively ubiquitous network capabilities. Access lists are such an example, and QoS policies follow closely behind. For security purposes, these policies must only apply to the device that is connecting, and should not modify other parts of a network element's configuration. The key scaling properties here are as follows:

- A manufacturer should only have to maintain and distributer one file per device model.
- A network management system need not retrieve that same file when the same model appears in multiple places in its network.
- Updates should occur at periods specified by the manufacturer to manage load.

2.4. Instantiating Policy

The network management system receiving the MUD file must convert it into an access list that a network element understands, and apply it to an appropriate interface, limiting its applicability only to the device in question. In some cases, the policies will be abstract. For example, "local" would be translated to the set of networks that are within the same administrative domain. It is the network management system's responsibility to see that the configuration is removed when the device detaches, and that the configuration is consistent with other policies that might apply to that device. Importantly, network management systems should always defer to the network administrator's wishes. As such, a conflicting policy should not be deployed, but rather logged.

Human interaction may be required in some cases. In the home, one could imagine description simply being instantiated, whereas in the enterprise, someone may need to review the description before it is applied.

It is distinctly possible that a highly advanced enterprise would ignore any manufacturer recommendations altogether but still use the URI received from devices as a classifier.

2.5. When Configuration Can't Change

In some environments it may not be possible for policy reasons to make changes to network elements to instantiate usage descriptions as a means of enforcement. These very same descriptions may be used as a means to audit activity of a device to determine whether or not it is acting in accordance with the the manufacturer's intent.

3. Related Work

3.1. Relationship to ANIMA

[I-D.ietf-anima-bootstrapping-keyinfra] specifies a means by which a device is configured with appropriate credentials for a given network. This work specifies a means to configure the network rather than the device. In fact, one key assumption of MUD is that it will be extremely painful to make any end system changes.

4. Security Considerations

The three mentioned means for a device to emit a MUD URI each have their own security properties, and will be discussed in separate drafts. A risk they share in common, however, is that the URI could point to to a site that contains malware. To avoid such problems, several countermeasures are suggested:

- All XML should be well formed and validated against appropriate schema.

- Only XML whose capability name spaces are known should be processed at all.

- Any names within the XML (such as access-list or ACE names) should be replaced with local instances, so as to avoid overwriting existing configuration.

- Controllers are encouraged to validate the reputation of the authority of the web site.

By emitting a URI the device may identify itself to an interloper. As it happens, most devices can be relatively easily fingerprinted based on their communications patterns. However, if this is of concern, devices should emit the URI to network controllers over secure channels.

Use of certain operations, such as SameManufacturer scale less well than others. Frequent connects and disconnects could cause configuration storms. To address this risk, as the number of changes increase, modifications to devices other than the one connecting should decrease or simply be scheduled. In as much as this is an attack, it can also be mitigated through device authorization mechanisms such as 802.1X.

5. IANA Considerations

The IANA is requested to enjoy a coffee or tea, as there is nothing in this document that otherwise requires their attention.

6. Acknowledgments

The author thanks Bernie Volz, Eric Vyncke, and Cullen Jennings for their helpful suggestions.

7. Informative References

[FW95] Chapman, D. and E. Zwicky, "Building Internet Firewalls", January 1995.

[I-D.ietf-anima-bootstrapping-keyinfra] Pritikin, M., Richardson, M., Behringer, M., and S. Bjarnason, "Bootstrapping Key Infrastructures", draft-ietf-anima-bootstrapping-keyinfra-01 (work in progress), October 2015.

[IEEE8021AB] Institute for Electrical and Electronics Engineers, "Link Layer Discovery Protocol", 2005.

[IEEE8021AR] Institute for Electrical and Electronics Engineers, "Secure Device Identity", 1998.

[IEEE8021X] Institute for Electrical and Electronics Engineers, "Port Based Network Access Control", 1998.

[RFC1984] IAB and , "IAB and IESG Statement on Cryptographic Technology and the Internet", BCP 200, RFC 1984, DOI 10.17487/RFC1984, August 1996, <http://www.rfc-editor.org/info/rfc1984>.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <http://www.rfc-editor.org/info/rfc2131>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7452] Tschofenig, H., Arkko, J., Thaler, D., and D. McPherson, "Architectural Considerations in Smart Object Networking", RFC 7452, DOI 10.17487/RFC7452, March 2015, <http://www.rfc-editor.org/info/rfc7452>.

[RFC7488] Boucadair, M., Penno, R., Wing, D., Patil, P., and T. Reddy, "Port Control Protocol (PCP) Server Selection", RFC 7488, DOI 10.17487/RFC7488, March 2015, <http://www.rfc-editor.org/info/rfc7488>.

Author's Address

Eliot Lear
Cisco Systems
Richtistrasse 7
Wallisellen CH-8304
Switzerland

Phone: +41 44 878 9200
Email: lear@cisco.com