MUD: Simple, Effective IoT Network Security

Steven Rich
Cisco Systems, Inc.
170 W. Tasman Dr.
San Jose, California 95134
Email: srich@cisco.com

Thorsten Dahm
Google Inc
1600 Amphitheatre Parkway
Mountain View, CA 94043
Email: thorstendlux@google.com

***Abstract*—Manufacturer Usage Descriptions, or MUDs, allow a manufacturer to cheaply and simply describe to the network the accesses required by an IoT device without adding any extra cost or software to the devices themselves. By doing so, the network infrastructure devices can apply access policies automatically which increase the overall security of the entire network, not just for the IoT devices themselves. This document describes the lifecycle of Manufacturer Usage Descriptions (MUDs) by describing detailed MUD scenarios from the perspective of manufacturers.**

I. MUD Motivation

The addition of IoT devices to a network can, at least theoretically, expand the attack surface of that network. Even if a device does not have exploitable vulnerabilities (in the sense of an attacker injecting and running malware on it), it may be susceptible to denial-of-service (DoS) attacks and thus could have its functionality impaired by attackers. Recent events have shown just how real, and not just theoretical, such attacks can be.

A detailed summary of the current state of understanding of the Mirai botnet's use of IoT devices can be found in this article. It is estimated that around 100,000 IoT devices generated more than a terabit per second of DDoS traffic.

Also consider the Sony Cameras IP Security article which describes a vulnerability in many camera models which could be exploited to launch attacks like those seen in the massive DDos attack on DynDNS. As both of these incidents show, more network-accessible devices which can connect to arbitrary external addresses can, if those devices permit too much access or if they have vulnerabilities which allow arbitrary code execution, be used by attackers to amplify attacks and to do so by using origin addresses spanning broad ranges of networks.

Concerns about the negative possibilities of attacks related to IoT devices is also discussed in this article in the MIT Technology Review that also discusses some of the regulatory and government angles in play. In a recent move, the U.S. Federal Government has taken the step of suing D-Link, accusing it of "poor security practices" for some of its IoT devices.

MUD provides a much more light-weight model of achieving very effective baseline security for IoT devices by simply allowing a network to automatically configure the required network access for IoT devices so that they can perform their intended functions without granting them gratuitous, unrestricted network privilege.

II. MUD High-level Introduction

Manufacturer Usage Descriptions (MUDs) provide advice to end networks on how to treat specific classes of devices. The MUD architecture is explained in [1], but we will describe it briefly here and also discuss details where necessary to understand this document. At its most basic, MUD is a system by which the IoT device itself tells the network exactly how to retrieve its network access requirements (in a "MUD File"), and network infrastructure can fetch and act upon this information. The MUD File itself is a static text file[1] which the network infrastructure element responsible for it can retrieve from the manufacturer or from whomever the manufacturer delegates the responsibility to. The MUD file may be cached, so when served, the MUD file should be returned with a "max-age" value which lets the requestor know how long it can cache it.

To "add a MUD file specification" to an IoT device is a very minimal change. To whatever dynamic network registration protocol which is currently being used by the device (e.g. DHCP, etc.), the URL for the MUD file is added as the "MUD URI". It is so simple that the device manufacturer can compile the URI into the firmware of the device, assuming that that is the most appropriate implementation.[2] The essential point is that MUD does not force a large behavioral change on the IoT device itself, and the serving up of the MUD file during the lifetime of the devices is similarly relatively low-impact. The bulk of the complexity of MUD is concentrated within the network elements which perform operations to retrieve the MUD files, possibly cache them, and then configure the network in response, but even there, the network elements effected mostly already perform all of these actions, albeit not automatically in most cases.

For this description, one can consider three general classes of actors in the MUD ecosystem:

- Device manufacturers

[1]The HTTPS server(s) which serves up the file or files can technically use whatever technology for implementation that is desired. However, each MUD file may simply be stored and served up as simple, single text files.

[2]A lightbulb, for example, may comprise a microcontroller with embedded flash and essentially no bulk storage, so the URI can be compiled directly into the data section of the software image.

- Networking equipment manufacturers
- Network operators

Note that end users are not mentioned here, as their involvement in MUD is minimal at best (and likely only present in the simplest of deployments). Note also that "Device manufacturers" are described with the assumption that they will both include MUD URIs within their devices as well as service MUD URL requests (via a cloud service or via their own web infrastructures). It is possible that a manufacturer will delegate the MUD URL retrieval function to a third party. The question of who actually services network requests for the MUD URL is an administrative one and does not affect the MUD architecture. It does give device manufactures more flexibility, though, in managing their investment into the MUD ecosystem.

This document will describe the MUD "lifecycle" from the standpoint of manufacturers, but it is also intended to be informative to persons interested in standardization, installation, or other areas where MUD may be in play. Where appropriate, suggestions of best practices will be given if there are no specific hard requirements.

III. Terminology

Before going into descriptions how MUD works, we will list terms used within the MUD ecosystem:

MUD
: Manufacturer Usage Description

MUD file
: a file containing YANG-based JSON that describes a recommended behavior

MUD file server
: an HTTPS server that hosts a MUD file

MUD controller
: the system that requests and receives the MUD file from the MUD server. After it has processed a MUD file it may direct changes to relevant network elements

URL
: Universal Resource Locator

URI
: Universal Resource Identifier. The difference between a "URI" and a "URL" is that a URI is intended to be used as an identifier in a general sense, whereas a URL is a specific use case of a URI that is used to access something at a particular network location

MUD URI
: a URI that an IoT device carries and which will be issued during operations such as DHCP requests which can be used as a URL to retrieve a MUD file

MUD URL
: the MUD URI being used as a URL

IEEE 802.1AR
: A IEEE specification for a certification-based approach for communicating device characteristics

YANG
: A data modeling language for the definition of data sent over the NETCONF network configuration protocol[2]

NETCONF
: Network Configuration Protocol[3]

JSON
: Javascript Object Notation, a human- as well as machine-readable file format containing textual representations of "objects" such as strings of characters, numbers, boolean values, and lists and dictionaries of such objects and collections of objects

Many of these terms are in common usage with the IETF or other network standards bodies and are thus used for consistency. More information about terms like "URL", "URI", "YANG", and "NETCONF" can be found in the standards and references published by the IETF and others. The value in distinguishing "URI" and "URL" will hopefully become more apparent when MUD file caching is discussed (during which time, already-retrieved MUD files will be used if the URI lookup returns a match). The actual text of a "MUD URI" and a "MUD URL" will generally be identical; the distinction lies in the use of it by various elements (IoT devices, network devices, and web services).

IV. MUD Operation

A full description of MUD is given in [1]. In short, when a device such as an IP-enabled lightbulb is connected to the network and given power, that device will perform some action to acquire a network identity, including an IP address, such as by making a DHCP request. If that request has a MUD URI in it, equipment in the network (not necessarily the DHCP server) can use that URI to retrieve the device's MUD file from the MUD file server. Some other networking component (the switch to which the bulb in connected, for example) can then act on the contents of the retrieved MUD file and apply the appropriate configurations to allow the device to function normally while restricting where it can connect.

A MUD file's contents will mostly contain descriptions of which protocols are required by the device and over what port or ports.

From the perspective of a manufacturer, the essential elements to note are the following:

1) On the device itself, the only change required to add MUD compliance/functionality is to add a field populated with a URI to whatever network access protocol is already being used (i.e., DHCP, IPv6 AD, etc.). This will be a static text string which will probably remain constant throughout the life of the product and which is identical for every instance of a product run (i.e., there is no per-serial-number version of the MUD URI)
2) The MUD file which is to be returned via an HTTPS server can be a static file and can be reused for devices which have the same network access requirements. The service which returns the MUD file will not be responsible for any security policy enforcement, as that is the job of the network which contains the devices themselves

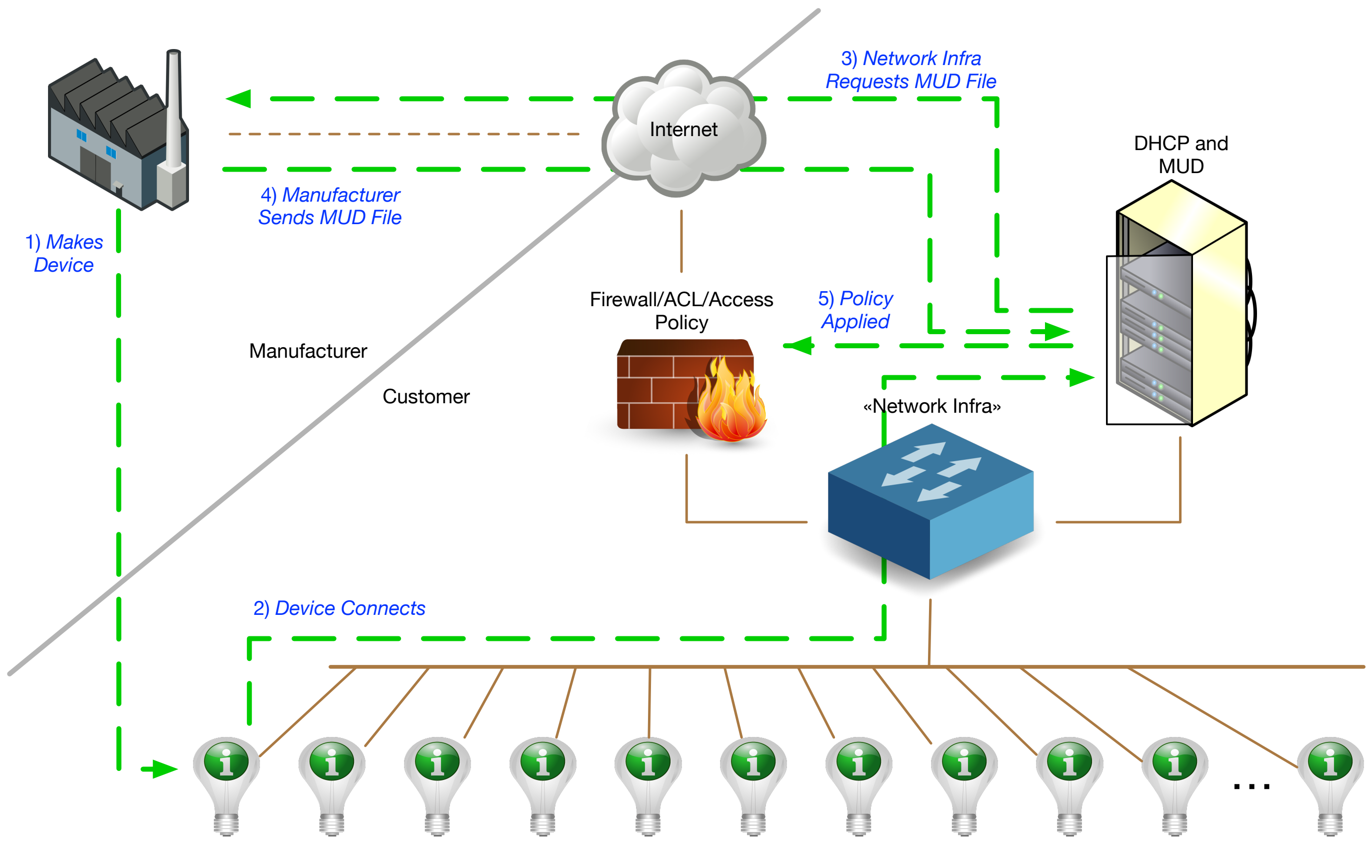

Fig. 1. MUD High-level information flow

3) MUD files are fairly short (on the order of tens of lines of text) and are thus trivial to serve either directly and are amenable to caching
4) The act of retrieving the MUD file and of acting on it is entirely up to the network infrastructure and not a responsibility of the IoT devices themselves. MUD does not impose any behavioral requirements on the IoT devices themselves other than that they must send the MUD URI during network access configuration, as mentioned earlier

How does MUD work in practice? Figure 1 on Page 3 shows a representation of the high-level MUD information flow. This document deals almost exclusively with elements in the upper left of that figure. Specifically, it describes what a manufacturer should do to put a MUD file into a device and what is required for a manufacturer (or a designee of the manufacturer) to answer requests for MUD files from network operators whose networks provide connectivity for such devices.

V. Device Manufacturer Considerations

The device manufacturers have the most insight into what resources the devices will need once they are installed in a network. They are thus best-suited to author the network profiles which will be required by the devices that they make for correct operation. Conversely, each manufacturer cannot know what each network's other requirements happen to be. As a result, the manufactures should provide configuration requirements for their devices which network operators can apply in a way best suited for their networks. The network operator can optimize operations through caching, LAN segregation, etc., and can use the MUD information to further secure the network.

If a manufacturer makes many devices which have similar network access requirements, that manufacturer may want to leverage common profiles. They should do so only when the profiles are truly close enough to be treated as the same.

Device manufacturers have three responsibilities under MUD:

- They must author a MUD profile which describes a device's requirements for network access
- They must encode a MUD URI into the device such that when the device performs DHCP or similar, the networking infrastructure is informed and can fetch and act on the MUD profile
- The manufacturer (or someone to whom the manufacturer has delegated the responsibility) must service requests to fetch the MUD profile(s) via an HTTP GET request

Since the MUD profiles can be static files, there is very little overhead required to serve these profiles. Due to their static nature, they are inherently cacheable.

Similarly, since the URI can be essentially static (the actual device configurations are easily updatable since they are contained in the MUD file, not the URI), the manufacturer can assign a name space and begin encoding the URIs into the devices relatively early in the manufacturing process. An important point is that manufacturers should adopt and follow a nomenclature that insures that they can sufficiently distinguish classes or families of devices with different requirements and assign them different URIs. From a security standpoint, it is better to have several URIs with more granular security profiles than it is to have a very few URIs with "catch-all" (and thus more open) security profiles. This ensures that a customer using a single family of devices will have the most closed network configuration possible.

If the device manufacturer decides to update the profile, then it may do so at any time, independently of updates to the firmware on the devices themselves. If it is expected that a profile may change frequently (say, for a new class of devices which aren't fully understood yet), then the MUD profile for said device should be served with a fairly short max-age (as compared to a device with a well-established network access profile).

VI. High-level MUD Lifecycle

The following lifecycle description is described considering a single device. As additional devices are added to a portfolio, the same steps are taken for each one where necessary. Each step can be isolated or coordinated with other device instances where convenient. There is little coupling inherent in the way that the various phases of MUD deployment operates to impose strict requirements in this area.

1) Based on a device's function, a MUD profile is either:
   - Chosen from a library of existing profiles for similar devices
   - Written anew to describe this device's network requirements
2) If the profile is pre-existing, the a choice is made if this device will receive a new URI or if it should be classed as identical to existing devices and use the same URI
3) The chosen URI is assigned to the device so that when the device performs network initialization, the URI is included in the request (i.e., DHCP, ANIMA, etc.)
4) In parallel or in advance (but prior to first customer shipment), the device manufacturer should allocate in an appropriate namespace and place the MUD profiles for when the URI is used as a URL.
5) The MUD profile should be made available to customers until such a time that the device is unsupported. While it is outside the scope of this document, The manufacturer should support MUD profile retrieval for each device for at least as long as the manufacturer supports the devices themselves.
6) If the profile is found to contain an error, the manufacturer should update the profile. Devices which are already deployed will continue to use the original URI (unless a firmware updates changes it), so the original profile should be corrected
7) If a device manufacturer chooses to update a MUD-enabled device's firmware, the manufacturer may update the MUD URI to a new one. The manufacturer should change the URI if the network access requirements of the new firmware are sufficiently different from those of the original firmware version.

VII. MUD URI

The MUD URI is a very visible and important part of MUD that is best done correctly from the start, for once it is embedded in an IoT device, changing it for the fielded devices will be, at best, inconvenient. Choosing a scheme for organizing the "name space" for the portion of the URI which is controlled by the device manufacturer may have knock-on effects such as the URL GET request routing behavior that must be supported during MUD file retrieval.

The format of the URI is:

`https://`*`authority`*`/.well-known/mud/`*`mud-rev`*`/`*`model`*

and may be post-fixed with "`?`*`extras`*". Referencing [4], the *authority* element is described by the "authority" type, the *model* element by the "segment" type, and *extras* by the "query" type. This gives considerable flexibility to manufacturers to structure their various namespaces to handle a huge variety of device types. However, this document will restrict itself to describing a very simple URI encoding scheme.

By far, the simplest method of assigning MUD URIs to devices is to assign each distinct model number a URI of the form

*`https://authority/.well-known/mud/mud-rev/`***model**

where the "model" element is literally the model number of the device. If a manufacturer has a model number collision problem (possibly because of acquisitions of other companies, for example), a simple scheme of a prefix or a suffix, set off with a hyphen or similar, will suffice to disambiguate them. Since the MUD files are relatively small, there is likely little value in conjuring schemes to save disk space with complicated naming conventions or structure.

VIII. MUD File Serving: Operations, Lifetypes, and Transfer

The previous section discussed how one might design the URI namespace for MUD files. Another very important consideration is the total lifecycle of the serving of MUD files via the internet for an appropriate length of time and what to do if one wants to transfer the responsibility of serving MUD files to some other entity. This section will describe several scenarios and suggest options for the transfer of responsibility of MUD files to other providers. There is no single set policy for these various activities, and organizations are free to decide how and when these transfers occur. There *are* technical considerations that must be dealt with, but this is not unlike outsourcing subsections of one's web site to payment partners or other specialists if so desired.

The single largest factor in thinking about serving MUD files throughout their lifetimes is the relative "permanence" of the URI itself (since, for some types of devices, at least, the buried-in URI will be essentially indelible).[3] Networks containing the MUD-enabled devices will make network requests to retrieve the MUD files. The MUD URIs are, quite literally, the URLs of the MUD files. There, network infrastructure devices from potentially anywhere on the internet will try to retrieve these MUD files. The volume of requests will be simple to handle (given that MUD files are static and small and that MUD servers in the network will be able to cache them and avoid redundant retrievals).

A very simple and direct way to manage MUD files and make the possible future delegation of MUD file serving to a 3^{rd}-party is to assign a URI DNS "namespace" for your company's MUD files. For example, using the fictional company "Acme Lightbulb and Sensor" and its web presence at "https://acmels.com", the DNS namespace for MUD files could be

```
mud.acmels.com
```

which can serve as the *authority* section of the MUD URI. If Acme wants to serve the MUD files themselves, then they can provision an HTTPS service that serves that address and return the requested MUD files, or they can create a CNAME to point to the actual entity who will answer the requests.

IX. Conclusion

The conclusion goes here.

References

[1] E. Lear and R. Droms and D. Romascanu, *Manufacturer Usage Description Specification*, IETF opsawg WG, 2016.

[2] M. Bjorklund, *YANG A Data Modeling Language for the Network Configuration Protocol (NETCONF)*, IETF RFC 6020, 2010.

[3] R. Enns and M. Bjorklund and J. Schoenwaelder and A. Bierman, *Network Configuration Protocol (NETCONF)*, IETF RFC 6241, 2011.

[4] T. Berners-Less and R. Fielding and L. Masinter, *Uniform Resource Identifier (URI): Generic Syntax*, IETF RFC 3986, 2005.

[3]Even if a device has a more fungible MUD URI (say, because it is easily and frequently updated), it is still wise to consider the case when a device's MUD URI cannot be easily updated since this represents the most problematic case.