September 2021

MANUFACTURER USADGE DESCRIPTION (MUD) LAYER 2 (L2) SUPPORT FOR ENHANCED SECURITY AND FUNCTIONALITY FOR MULTIPLE NETWORK INTERFACES

David Hanes

Sebastian Jeuk

Gonzalo Salgueiro

MANUFACTURER USADGE DESCRIPTION (MUD) LAYER 2 (L2) SUPPORT FOR ENHANCED SECURITY AND FUNCTIONALITY FOR MULTIPLE NETWORK INTERFACES

AUTHORS:
David Hanes
Sebastian Jeuk
Gonzalo Salgueiro

ABSTRACT

The growth of Internet of things (IoT) devices is increasing dramatically. IoT devices often have multiple interfaces and connectivity options, but these may not be communicated to a network in an automated fashion. Techniques are presented herein that extend the Manufacturer Usage Description (MUD) artifact so that it may be used to effectively communicate information about an IoT device's multiple connectivity options to a network in an automated manner and to notify the network of possible interface vulnerabilities so that any number of different actions (e.g., alerts, configuration checks, etc.) may be performed.

DETAILED DESCRIPTION

It is estimated that by 2023 there will be nearly 30.0 billion network connected devices and connections, up from 18.4 billion in 2018. Further, it is estimated that Internet of Things (IoT) devices will account for 50% (approximately 14.7 billion) of all network connected devices, compared to 33% (approximately 6.1 billion) in 2018.

IoT devices are almost always on the network edge and, as the edge evolves, the capabilities of these devices must also evolve, especially from a security perspective. Now, and even more so in the future, edge devices will often have more than one network interface. Connection technologies, such as Bluetooth™, Wi-Fi®, LiFi, cellular, Ethernet, Universal Serial Bus (USB), etc. are all options for medium to advanced IoT devices. The reason for this is that not every connection type is ideal for every scenario and sometimes conditions may change, especially if a sensor is not in a static environment. Flexibility and multiple connectivity options are important, but at the same time every port or interface on an IoT device represents a potential vulnerability for the entire network.

The Manufacturer Usage Description (MUD) as defined by Internet Engineering Task Force (IETF) Request for Comments (RFC) 8520 provides for the ability to onboard and securely connect IoT devices to a network at scale. Unfortunately, MUD is not aware of multiple connectivity options or network interfaces. Aspects of the techniques presented herein extend MUD to allow it to communicate to a network if there are multiple connectivity options for an IoT device, a connectivity preference, connectivity redundancy options, and other functions. This information is needed for onboarding IoT devices at scale and delivering next generation capabilities at the network edge. Of particular interest and note in aspects of the techniques presented herein is the extension of the MUD standard to provide for the notification of possible security vulnerabilities at the Layer 2 (L2) interface level.

With IoT devices increasingly having multiple modalities for connecting to a network, it is critical to ensure that these IoT devices can be securely onboarded at scale. As noted previously, MUD is an IETF specification that is designed to support this but in its current incarnation MUD is unable to communicate these connection methods (e.g., interfaces) to the network.

Traditionally, MUD operates in the narrow vein of telling a network how to configure itself in the way of secure access policies for an IoT device. Aspects of the techniques presented herein support extending MUD to not only communicate a recommended network security configuration but to also notify the network of other security vulnerabilities.

MUD is meant to help securely configure a network for IoT devices at scale, but IoT devices with multiple interfaces can be vulnerable to security breaches. According to aspects of the techniques presented herein, it is not required that, for example, a network need to jam Bluetooth or necessarily account for such interfaces directly. Rather, an IoT controller may pass along the manufacturer-sourced information and alert other systems that may be responsible for IoT device configuration and provisioning. In the future, as intent-based networks evolve, this sort of integration may become more seamless.

The main reasons that it is critical for MUD to communicate an IoT device's connectivity options include security, preference, and redundancy. If a network is not aware of an IoT device's other network interfaces then these are security vulnerabilities if they are not addressed. Every interface or port on an IoT device is a potential opening for a hacker to access the entire network.

In addition to security, IoT devices with multiple interfaces may have an interface preference (e.g., one interface may use less battery power) that needs to be understood by the network. Also, the organization deploying the sensor may have an interface preference for cost reasons. Additionally, redundancy is important. For example, if one interface is not able to connect perhaps the IoT device may try another network interface. In some cases this may be warranted but in other cases it could be a significant drain on resources.

Figure 1, below, illustrates elements of how, according to aspects of the techniques presented herein, an IoT device's connectivity options may be inserted into a MUD file (e.g., by the device manufacturer) for properly configuring a network.

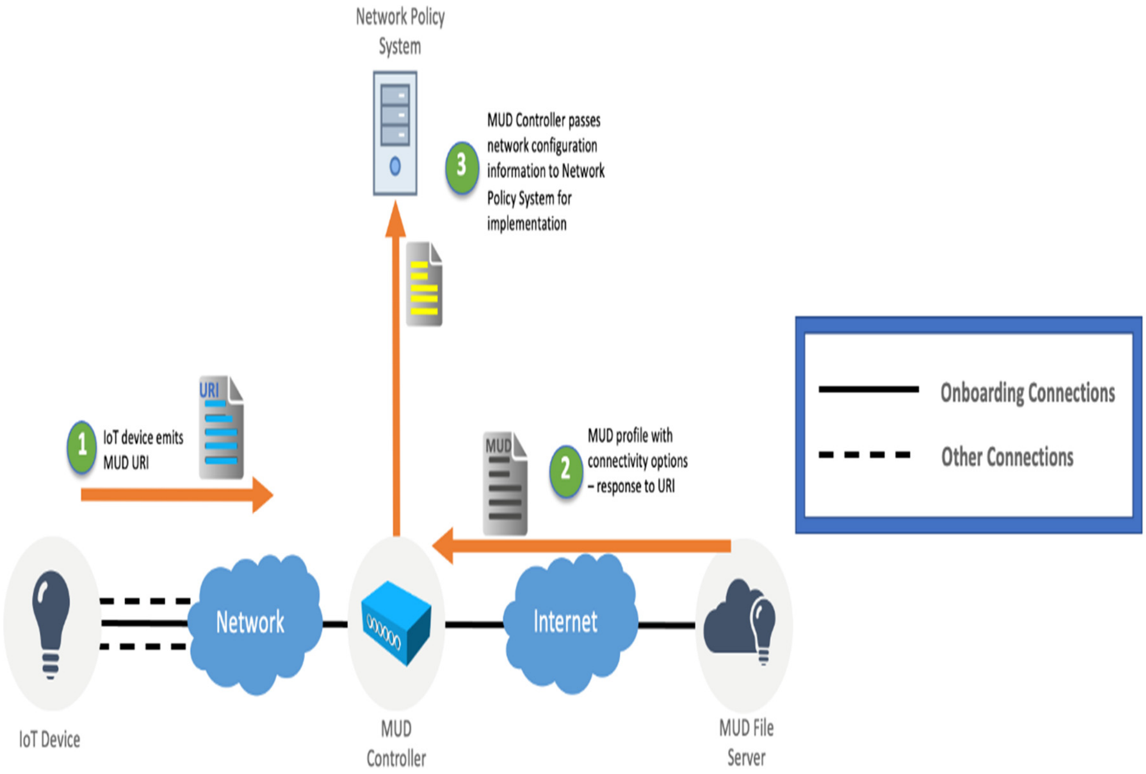
Network Policy System
3
MUD Controller passes network configuration information to Network Policy System for implementation
Onboarding Connections
Other Connections
URI
1
IoT device emits MUD URI
MUD
2
MUD profile with connectivity options – response to URI
Network
Internet
IoT Device
MUD Controller
MUD File Server

Figure 1: Illustrative Connectivity Option Inclusion

As illustrated in Figure 1, a MUD file is created and maintained by the manufacturer of an IoT device. Under aspects of the techniques presented herein, the manufacturer adds

L2 connectivity parameters to the security policies that already form the basis of MUD files. Such connectivity parameters may contain, possibly among other things, the following information:

- A listing of all of the network interfaces;
- For each interface, a listing of the supported protocols, versions, frequencies, encryption algorithms, and any other options; and/or
- A manufacturer-recommended network interface.

Consider an example encompassing an IoT camera that has a Wi-Fi and a Bluetooth network interface. In this example, consider that the IoT camera is selected by an organization that wants to deploy hundreds of the cameras for their factory. To deploy at scale the organization wants to use MUD. MUD can help with the network security and access policies for this device in its current form. However, MUD does not currently account for the Wi-Fi and Bluetooth interfaces. According to aspects of the techniques presented herein, the following information may be explicitly added to the device's MUD profile:

```
Network interfaces:
   ---> Wi-Fi
   ------> 802.11b/g/n
   --->Bluetooth
   ------> Bluetooth 4.2
   ---> USB
   ------> USB 2.1
Recommended Interface
   ---> Wi-Fi
```

While the example above is illustrated at a high-level but, nonetheless, provides a glimpse into the type of information that makes the techniques presented herein valuable to MUD. From a security perspective, the network should know about all of the interfaces to cover any vulnerabilities. For the case involving the camera device in the above example, knowing about the Bluetooth interface allows the Bluetooth interface to be disabled and perhaps password protection to be added to the USB interface. It is important to note that while MUD is able to provide the necessary details to the network, it does not trigger the configuration of the IoT device itself. Such configuration can be provided by a controller or an external entity using Network Configuration Protocol (NETCONF) processes, Representational state transfer (REST) application programming interface (API) calls, and/or any other bulk configuration option, as represented by the Network Policy System in Figure 1, above.

It is important to note that the Network Policy System that is depicted in Figure 1, above, encompasses more than just network policies and includes, for example, configuration management, IoT controllers, and so on. As noted above, while MUD is able to provide the necessary details to the network it does not trigger the configuration of the IoT Device itself. Such activity should be done by a controller or an external entity through, for example, NETCONF, REST API calls, or any other bulk configuration option as represented by the Network Policy System.

To further clarify, Figure 2, below, presents a revised version of Figure 1.

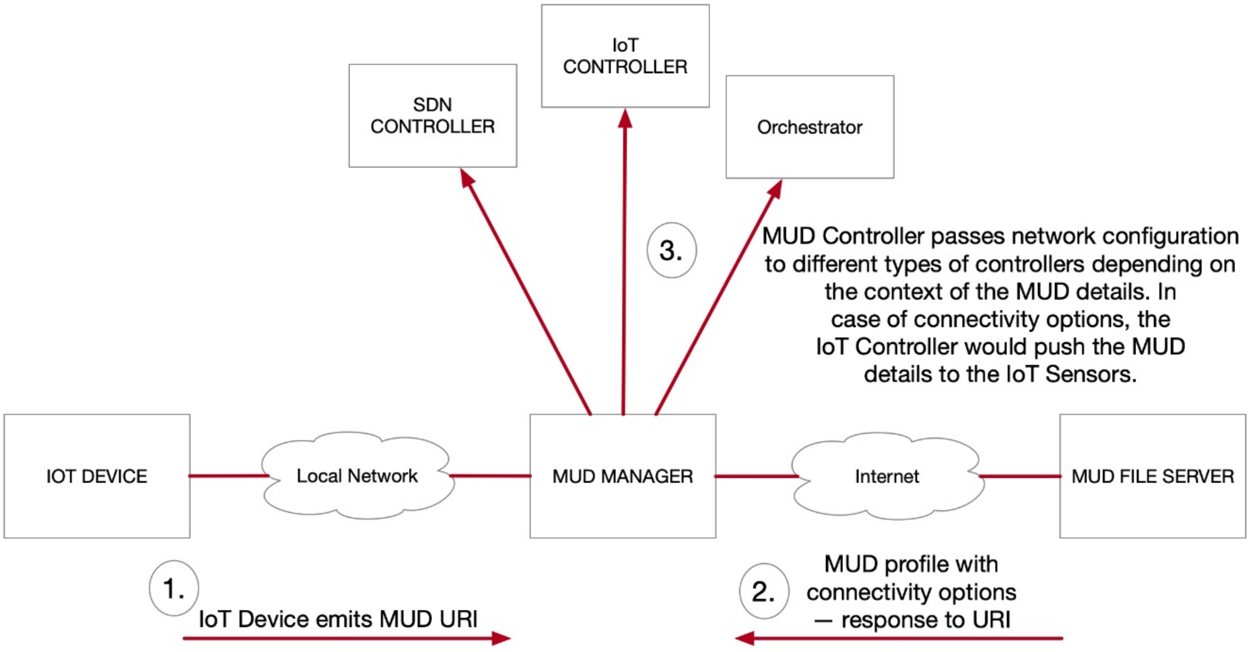
IoT CONTROLLER
SDN CONTROLLER
Orchestrator
3.
MUD Controller passes network configuration to different types of controllers depending on the context of the MUD details. In case of connectivity options, the IoT Controller would push the MUD details to the IoT Sensors.
IOT DEVICE
Local Network
MUD MANAGER
Internet
MUD FILE SERVER
1.
IoT Device emits MUD URI
2.
MUD profile with connectivity options — response to URI

Figure 2: Exemplary Connection Option Processing

As illustrated in Figure 2, above, there are multiple options for how the MUD manager or controller may share information that is received about an IoT device's interface details. For example, the MUD manager may share MUD file details related to

IoT sensor interface details with a software-defined networking (SDN) controller. The SDN controller can then take the appropriate actions based on the received information. The MUD Manager may also integrate through REST API calls with an IoT orchestrator and can share the received details via API calls. It is important to note that the MUD controller does not need to itself perform the configuration.

There are multiple options for how a MUD controller may handle received data and the actions that can be taken upon it. Importantly, for the techniques presented herein is the point that the interface-related data represents a security vulnerability that is received directly from an IoT device manufacturer and, employing aspects of the techniques presented herein, MUD may communicate this information (along with other critical device security data) in an automated way.

The action of the network is to simply pass this information on to a configuration manager, IoT controller, SDN, or other system that can then take action on it. The MUD manager or controller performs its operations with all MUD data from a received file and parses it and takes various actions. For some information that is received through MUD, access lists and firewall specific data needs to be configured or changed and this can be passed to a system or device that handles such tasks. For the interface-related data, this can be passed to a device management, SDN, or configuration management system for processing and actions. If the interfaces need to remain enabled, or be disabled, then this may be communicated to the system that is responsible for those actions.

As noted previously, according to aspects of the techniques presented herein it is not required that a network needs to jam Bluetooth or necessarily account for such interfaces directly. Rather, an IoT controller may pass along the manufacturer-sourced information and alert other systems that may be responsible for IoT device configuration and provisioning. In the future, as intent-based networks evolve, this sort of integration may become more seamless. Additionally, it is important to note that the just described options represent only some of the possible methods and these may change with the developments surrounding IoT, MUD and SDN that will likely take place during the coming years.

As more connectivity options become available, securing multiple network interfaces becomes more difficult. Consider the case of the camera device in the above example also having LiFi as an interface. Another connectivity option adds more vulnerabilities and challenges for the network to deal with. Additionally, this provides more possibilities for a preferred interface and also more options for fallback and redundancy in the device configuration.

It is important to note that the techniques presented herein leverage MUD as a mechanism to ensure that the available interfaces and their potential vulnerabilities are known to the network. Recommended security policies are already being communicated by MUD so extending MUD's security coverage to all of the interfaces of a device, according to the techniques presented herein, is harmonious. And more importantly, MUD allows for the identification of, and the alerting of the network to, such vulnerabilities in an automated manner (that can help with the automatic configuration of devices, that can be used as a double-check for an existing device configuration template, etc.).

Aspects of the techniques presented herein extend MUD in a novel, forward-looking manner beyond its traditional usage of recommending network security configurations and policies. As noted previously, one of the values in the presented techniques includes the automated notification to the network of a possible security vulnerability by an authoritative source (i.e., the device manufacturer) using MUD. Actions from the network may be to simply notify configuration or provisioning systems for the IoT devices, alert administrators or other security systems about these possible vulnerabilities, or in the future utilize intent-based networking (e.g., a device is being deployed into an open, publicly accessible area like a building lobby) to automatically shut down or secure those interfaces.

Under further aspects of the techniques presented herein MUD security policies may be specific to each of a device's interfaces. Presently, a single policy is applied to the device. With different interface connectivity options, there is often a need to require more robust security measures for some interfaces versus others. For example, LiFi does not propagate broadly and is easily blocked by walls and doors whereas Wi-Fi penetrates and has a much larger range. If a network detects that a device is connecting through Wi-Fi then it may elect to enact a different set of security policies and with SDN this may be implemented in real-time. A different security posture could conceivably be implemented for each interface based off of a manufacturer's recommendations as communicated through MUD.

Under still further aspects of the techniques presented herein a manufacturer may also provide notification of API access for an IoT device. Additional details on available configuration tasks and other diagnostic or telemetry data that are available through an API may be communicated. Such capabilities may help with the automated configuration management of, and the automated telemetry gathering for, the IoT device. While this additional API information could be used by the MUD controller, it could also simply be passed on to the Network Policy System as shown in Figure 1, above.

In summary, and as described and illustrated in the narrative that was presented above, for MUD to be comprehensive in its application to the onboarding of IoT devices to a network, it needs to relay details about all of the interfaces of an IoT device to mitigate security threats and allow for the optimal integration of that device into the network. As more connectivity options and multiple physical interfaces become more prevalent in IoT devices, this challenge will only become more complicated. To address these types of challenges, techniques have been presented herein that extend the capabilities of MUD so that it can effectively communicate information about an IoT device's multiple connectivity options to a network in an automated manner.