OPSAWG Working Group
Internet-Draft
Updates: 8520 (if approved)
Intended status: Best Current Practice
Expires: 30 July 2021

M. Richardson
Sandelman Software Works
W. Pan
Huawei Technologies
E. Lear
Cisco Systems

26 January 2021

Authorized update to MUD URLs
draft-ietf-opsawg-mud-acceptable-urls-00

Abstract

This document provides a way for an RFC8520 Manufacturer Usage Description (MUD) definitions to declare what are acceptable replacement MUD URLs for a device.

RFCEDITOR-please-remove: this document is being worked on at: https://github.com/mcr/iot-mud-acceptable-urls

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 https://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 30 July 2021.

Table of Contents

1. Introduction

[RFC8520] provides a standardized way to describe how a specific purpose device makes use of Internet resources and associated suggested network behavior, which are described in a MUD file hosted in its manufacture's server. By providing a MUD URL, the network manager can locate this MUD file. MUD URLs can come from a number of sources:

* IDevID Extensions

* DHCP option

* LLDP TLV

* [I-D.richardson-opsawg-securehomegateway-mud] proposes to scan them from QRcodes.

The IDevID mechanism provides a URL that is asserted cryptographically by a manufacturer. However, it is difficult for manufacturers to update the IDevID of a device which is already in a box.

The DHCP and LLDP mechanisms are not signed, but are asserted by the device. A firmware update may update what MUD URL is emitted. Sufficiently well targetted malware could also change the MUD URL.

The QRcode mechanism is usually done via paper/stickers, and is typically not under the control of the device itself at all.

While MUD files may include signatures, it is not mandatory to check them, and there is not a clear way to connect the entity which signed the MUD file to the device itself. A malicious device does not need to make up a MUD file if there is already an available, and already trusted MUD file which it can use to impersonate the device.

One defense against this is to not trust MUD URLs which are different from the one that was placed in an IDevID. Or if the initial MUD URL was not taken from an IDevID, it could be trusted on first use. But, if the MUD controller has locked down the URL, then updates to the URL are difficult to do.

2. Updating MUD URLs vs Updating MUD files

There are two ways in which a manufacturer can change what the is processed by the MUD controller: they can change what is in the MUD file (update-in-place), and or they change which file is processed by the MUD controller by changing the URL (updated-url).

2.1. Updating the MUD file in place

One option is for the manufacturer to never change the MUD URL due to firmware updates. The published description is updated whenever the behaviour of the firmware changes.

2.1.1. Adding capabilities

For situations where new capabilities are added to the firmware, the MUD file will detail the new access that the new firmware requires. This may involve new incoming or outgoing connections that should be authorized. Devices which have been upgraded to the new firmware will make use of the new features. Devices which have not been upgraded to the new firmware may have new connections that are authorized, but which the device does not use (outgoing connections), or for which the device is not prepared to respond to (new incoming connections).

It is possible that older versions of the firmware have vulnerabilities which were not easily exploitable due to the MUD file preventing particular kinds of access. As an example, an older firmware could have a no credentials required (or default credentials) access via telnet on port 23 or HTTP on port 80. The MUD file protected the device such that it could either not be accessed at all, or access was restricted to connections from a controller only.

Useful and needed upgrades to the firmware could add credentials to that service, permitting it to be opened up for more general access. The new MUD file would provide for such access, but when combined with the weak security of the old firmware, results in a compromised device.

While there is an argument that old firmware was insecure and should be replaced, it is often the case that the upgrade process involves downtime, or can introduce risks due to needed evaluations not having been completed yet. As an example: moving vehicles (cars, airplanes, etc.) should not perform upgrades while in motion! It is probably undesireable to perform any upgrade to an airplane outside of its service facility. The owner of a vehicle may desire to only perform software upgrades when they are at home, and could make other arrangements for transporation, rather than when parked at a remote cabin. The situation for upgrades of medical devices has even more considerations involving regulatory compliance.

2.1.2. Removing capabilities

For situations where existing capabilities prove to be a problem and are to be turned off or removed in subsequent versions of the firmware, the MUD file will be updated to disallow connections that previously were allowed.

In this case, the new MUD file will forbid some connection which the old firmware still expects to do. As explained in the previous section, upgrades may not always occur immediately upon release of the new firmware.

In this case the old device will be performing unwanted connections, and the MUD controller when be alerting the device owner that the device is mis-behaving. This causes a false positive situation (see [boycrieswolf]), leading to real security issues being ignored. This is a serious issue as documented also in [boywolfinfosec], and [falsemalware].

2.1.3. Significant changes to protocols

[I-D.reddy-opsawg-mud-tls] suggests MUD definitions to allow examination of TLS protocol details. Such a profile may be very specific to the TLS library which is shipped in a device. Changes to the library (including bug fixes) may cause significant changes to the profile, requiring changes to the profile described in the MUD file. Such changes are likely neither forward nor backward compatible with other versions, and in place updates to MUD files are therefore not indicated.

2.2. Motivation for updating MUD URLs

While many small tweaks to a MUD file can be done in place, the situation described above, particularly when it comes to removing capabilities will suggests that changes to the MUD URL. A strategy for doing this securely is needed, and the rest of this document provides a mechanism to do this securely.

3. Threat model for MUD URLs

Only the DHCP and LLDP MUD URL mechanisms are sufficiently close to

the firmware version that they can be easily updated when the firmware is updated. Because of that sensitivity, they may also be easily changed by malware!

There are mitigating mechanisms which may be enough to deal with this problem when MUD files are signed by the manufacturer.

While [RFC8520] has established a mechanism for signing of MUD files, the document does not define a way for a MUD controller to determine who should sign the MUD file for a particular device.

[RFC8520] leaves this for a local policy. There are any number of processes that could be used, but they require coordination of many players. It is expected that each industrial vertical will work out supply chain arrangements or other heuristics.

3.1. Trust on First Use (TOFU): leveraging the manufacturer signature

Many MUD controllers currently use a Trust on First Use (TOFU) mechanism. The first time a signature from a particular device-type is verified, the identity of the signing authority is recorded. It is pinned. Subsequent updates to that MUD file must be signed by the same entity in order to be accepted.

Based upon this process, an update to the MUD URL would be valid if the new MUD file was signed by the same entity that signed the previous entry. This mechanism permits a replacement URL to be any URL that the same manufacturer can provide.

3.2. Concerns about same-signer mechanism

There is still a potential threat: a manufacturer which has many products may have a MUD definition for another product that has the privileges that the malware desires.

The malware could simply change the expressed MUD URL to that of the other product, and it will be accepted by the MUD controller as valid.

This works as long as manufacturers use a single key to sign all products. Some manufacturers could sign each product with a different key. Possibly, all the keys are collected into a single PKI, signed by a common certification authority. In this case, the question as to whether the MUD controller should pin the end-entity (EE) certificate, or the CA certificate. Pinning the EE certificate defends against malware that changes the product type, but keeps the manufacturer from being able to cycle the validity of the End-Entity Certificate for cryptographic hygiene reasons. Pinning the CA certificate allows the EE certificate to change, but may not defend against product type changes.

It is possible to invent policy mechanisms that would link the EE certificate to a value that is in the MUD file. This could be a policy OID, or could involve some content in a subjectAltName. Future work could go in this direction. This document proposes a simpler solution.

4. Outline of proposed mechanism

The document proposes to limit what MUD URLs are considered valid from the device, limiting new MUD URLs to be variations of the initial (presumed to be secure) URL.

5. Changes to RFC8520

The first MUD file which is defined for a device can come from an IDevID (which is considered more secure), or via Trust on First Use with DHCP or LLDP or another mechanism.

This first, initially trusted, MUD file will be called the "root" MUD file.

MUD files contain a self-referential MUD-URL attribute that point to a MUD file located on the vendor's web site. While the IDevID, DHCP and LLDP mechanisms only transmit a URL, there are some newer, not yet standardized proposals that would fetch an entire MUD file from the device, such as [I-D.jimenez-t2trg-mud-coap].

The MUD-URL MUST always be an Absolute URI: see [RFC3986] section 4.3.

The URL found in the MUD-URL attribute is to be called the canonical MUD URL for the device.

The MUD-SIGNATURE attribute in the MUD file SHOULD be a relative URI (see [RFC3986] section 4.2) with the (hierarchical) base URL for this reference being the MUD-URL attribute.

Subsequent MUD files are considered valid if:

* have the same initial Base-URI as the MUD-URL, but may have a different final part

* they are signed by the same End Entity (same trusted CA and same SubjectAltName) as the "root" MUD file.

Section 5.2 of [RFC3986] details many cases for calculating the Base-URI. The test is simplified to: remove everything to the right of the last (rightmost) "/" in the URL of "root" MUD file URL, and the proposed new URL. The resulting two strings MUST be identical.

For as a simple example, if the "root" mud-url is http://example.com/hello/there/file.json then any URL that starts with http://example.com/hello/there/ would be acceptable, such as http://example.com/hello/there/revision2.json.

Once the new MUD file is accepted, then it becomes the new "root" MUD file, and any subsequent updates must be relative to the MUD-URL in the new file.

This process allows a manufacturer to rework their file structure, to change web server hostnames (such as when there is an acquisition or merger), etc. so long as they retain the old structure long enough for all devices to upgrade at least once.

(XXX: how should the trust anchor for the signature be updated when there is Merger&Acquisition)

6. Privacy Considerations

The MUD URL contains sensitive model and even firmware revision numbers. Thus the MUD URL identifies the make, model and revision of a device. [RFC8520] already identifies this privacy concern, and suggests use of TLS so that the HTTP requests that retrieve the MUD file do not divulge that level of detail. However, it is possible that even observing the traffic to that manufacturer may be revealing, and [RFC8520] goes on to suggest use of a proxy as well.

7. Security Considerations

Prior to the standardization of the process in this document, if a device was infiltrated by malware, and said malware wished to make accesses beyond what the current MUD file allowed, the the malware would have to:

1. arrange for an equivalent MUD file to be visible somewhere on the Internet

2. depend upon the MUD-manager either not checking signatures, or

3. somehow get the manufacturer to sign the alternate MUD

4. announce this new URL via DHCP or LLDP, updating the MUD-manager with the new permissions.

One way to accomplish (3) is to leverage the existence of MUD files created by the manufacturer for different classes of devices. Such files would already be signed by the same manufacturer, eliminating the need to spoof a signature.

With the standardization of the process in this document, then the attacker can no longer point to arbitrary MUD files in step 4, but can only make use of MUD files that the manufacturer has already provided for this device.

Manufacturers are advised to maintain an orderly layout of MUD files in their web servers, with each unique producting having its own directory/pathname.

The process described updates only MUD-managers and the processes that manufacturers use to manage the location of their MUD files.

A manufacturer which has not managed their MUD files in the the way described here can deploy new directories of per-product MUD files, and then can update the existing MUD files in place to point to the new URLs using the MUD-URL attribute.

There is therefore no significant flag day: MUD managers may implement the new policy without significant concern about backwards compatibility.

7.1. Updating files vs Updating MUD URLs

Device developers need to consider whether to make a change by updating a MUD file, or updating the MUD URL.

MUD URLs can only be updated by shipping a new firmware. It is reasonable to update the MUD URL whenever a new firmware release causes new connectivity to be required. The updated mechanism defined in this document makes this a secure operation, and there is no practical limitation on the number of files that a web server can hold.

In place updates to a MUD file should be restricted to cases where it turns out that the description was inaccurate: a missing connection, an inadvertent one authorized, or just incorrect information.

Developers should be aware that many enterprise web sites use outsourced content distribution networks, and MUD controllers are likely to cache files for some time. Changes to MUD files will take some time to propogate through the various caches. An updated MUD URL will however, not experience any cache issues, but can not be deployed with a firmware update.

8. References

8.1. Normative References

[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, <https://www.rfc-editor.org/info/rfc3986>.

[RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <https://www.rfc-editor.org/info/rfc8520>.

8.2. Informative References

[boycrieswolf] "The Boy Who Cried Wolf", 18 January 2020, <https://fablesofaesop.com/the-boy-who-cried-wolf.html>.

[boywolfinfosec]
"Security Alerts - A Case of the Boy Who Cried Wolf?", 18 January 2020, <https://www.infosecurity-magazine.com/opinions/security-alerts-boy-cried-wolf/>.

[falsemalware]
"False malware alerts cost organizations $1.27M annually, report says", 18 January 2020, <https://www.scmagazine.com/home/security-news/false-malware-alerts-cost-organizations-1-27m-annually-report-says/ and http://go.cyphort.com/Ponemon-Report-Page.html>.

[I-D.jimenez-t2trg-mud-coap]
Jimenez, J., "Using MUD on CoAP environments", Work in Progress, Internet-Draft, draft-jimenez-t2trg-mud-coap-00, 9 March 2020, <http://www.ietf.org/internet-drafts/draft-jimenez-t2trg-mud-coap-00.txt>.

[I-D.reddy-opsawg-mud-tls]
Reddy.K, T., Wing, D., and B. Anderson, "MUD (D)TLS profiles for IoT devices", Work in Progress, Internet-Draft, draft-reddy-opsawg-mud-tls-05, 31 August 2020, <http://www.ietf.org/internet-drafts/draft-reddy-opsawg-mud-tls-05.txt>.

[I-D.richardson-opsawg-securehomegateway-mud]
Richardson, M., Latour, J., and H. Gharakheili, "On loading MUD URLs from QR codes", Work in Progress, Internet-Draft, draft-richardson-opsawg-securehomegateway-mud-05, 8 September 2020, <http://www.ietf.org/internet-drafts/draft-richardson-opsawg-securehomegateway-mud-05.txt>.

Appendix A. Appendices

Authors' Addresses

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

Wei Pan
Huawei Technologies

Email: william.panwei@huawei.com

Eliot Lear
Cisco Systems

Email: lear@cisco.com