MUDIS: MUD Inspection System

Anat Bremler-Barr
Reichman University (IDC)
Israel

Bar Meyuhas
Reichman University (IDC)
Israel

Ran Shister
Reichman University (IDC)
Israel

***Abstract*—The Manufacturer Usage Description (MUD) is an IETF white-list protection scheme that formalizes the authorized network behavior in a MUD file; this MUD file can then be used as a type of firewall mechanism.**

This demo introduces MUDIS, a MUD Inspection System that inspects the network behavior of devices, based on their formal description in the MUD file. We present several use-cases in which MUDIS is useful, including examining the impact of device location, the impact of a firmware update, the correlation of network behavior between different devices of the same manufacture, and more.

MUDIS inspects two MUD files, clusters together and graphically visualizes identical, similar, and dissimilar rules. It then calculates a similarity score that measures the similarity between them both. It also generalizes the two MUD files where possible, such that the resulting generalized MUD covers all the permitted (white-list) network behavior for both MUDs.

Our open-source MUDIS tool and proof-of-concept dataset are available for researchers and IoT manufacturers, allowing anyone to gain meaningful insights over the network behavior of IoT devices.

I. Introduction

The Manufacturer Usage Description (MUD) is an IETF white-list protection scheme [1] that formalizes the authorized network behavior in a MUD file. This MUD file can then be used as a type of firewall mechanism that provides security for the highly diverse IoT devices. MUD files consist of Access Control Lists (ACLs), each with several Access Control Entries (ACEs). Each ACE is defined as a 5-tuple:

$$ACE = (legitimate\_endpoints, protocol, source\_port, destination\_port, direction) \quad (1)$$

The MUD file is fetched by the IoT device using DHCP or LLDP, and thus there is a single MUD file for each firmware version, regardless of any other device or network factors.

The MUD can be provided by the manufacture or learned based on information captured from the device network traffic (PCAP) using a MUD generator tool such as MUDGEE [2] or MUD-PD [3]. Environmental variables can influence the network behavior of an IoT device and hence, have a direct impact on IoT security, including the MUD file that is learned [4], [5]. For example, a device's location impacts its behavior [6], which makes learning what is normal and secure behavior for an IoT device more challenging than expected. In many cases, the same IoT device, with the same firmware, can exhibit different behavior or connect to different domains/IPs with different ports and protocols, depending on the device's environment variables. This is even more challenging when learning the behavior of IoT devices that have more than one different environment variable (e.g., internet connection, DNS blocking, human interaction).

978-1-6654-0601-7/22/$31.00

We present a novel and unique tool called MUD Inspection System or MUDIS for short. MUDIS inspects and analyzes two MUD files by comparing their rules, and produces a single generalized MUD file that is comprehensive, tight, and secure for both MUDs. MUDIS is useful for many cases, including analyzing MUDs that were generated from different network traffic due to different environmental factors, analyzing the differences in MUDs between different firmware versions, identifying anomalies based on their network behavior to spot rare actions like firmware updates, find malware infected devices, and more.

MUDIS is a web application with a RESTful web service that is written in Python and uses MongoDB for storage, following the Object-Oriented Programming approach. All the code is open source and available for the use of other researchers and IoT manufacturers at [7], together with our POC dataset [8] and an easy-to-use setup guide based on Docker.

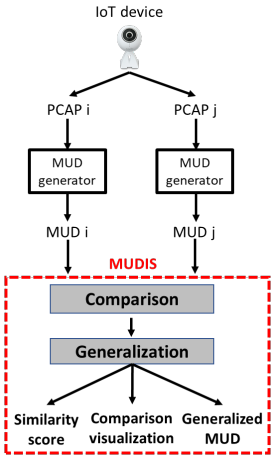

Figure 1: MUDIS architecture

The MUDIS architecture, depicted in Figure 1, receives two MUD files as input and performs four tasks, using a set of algorithms: parsing the input MUDs; comparing their rules; generalizing them into one MUD file; and then graphically visualizing the results. The MUDIS comparison task visualizes the differences between the two files and highlights identical ACEs, similar ACEs, clusters of ACEs, and dissimilar ones. The number of ACEs may differ between the devices and

the network behavior captured, and can range from just a few ACEs to a few dozens of them. This emphasizes the importance of comparing and visualizing the ACEs so we can easily spot similarities and differences between the two MUDs. MUDIS also calculates a similarity score that measures and numerically represent the similarity between the two MUD files. The comparison task helps us drill down and gain insights about the origin of the differences, and analyze and emphasize their impact on the device's network behavior. For example, these may include domain differences, encrypted vs. plain communication, different ports and protocols for the same end-point, the use of cloud services, and much more. The MUDIS generalization task outputs a generalized and comprehensive MUD file that can white-list the network behavior of both MUD files, in a tight and secure manner. The naive method would be to add both sets of rules to form a single unified MUD. However, in MUDIS we use ranges in the domain (e.g. *.iotvendor.com) field to create a generalized MUD with fewer rules. By doing this, we increase the explainability of the resulting MUD and reduce implementation costs in the firewall, which is crucial for network administrators and device manufacturers who need to support the devices' MUDs.

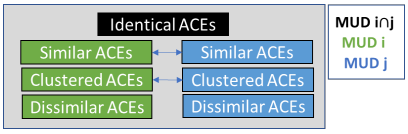

Figure 2: MUDIS comparison visualization

II. Main Features

In the following subsections we present two main features of MUDIS: Comparison and Generalization

A. MUD Comparison

Given two MUD files, MUD_i and MUD_j, our tool compares them to find their differences and similarities. Initially, it outputs a similarity score to measure and numerically represent the similarity between the two MUDs. The similarity scale ranges from 0 to 1, where 0 means that there are no similar ACEs among the MUD files and 1 means the two MUDs are identical. We define MUD similarity as the Jaccard similarity coefficient of the two MUDs and divide the number of equal ACEs in both MUDs by their total number of ACEs. The similarity measure of two MUDs is defined formally as:

$$Similarity(MUD_i, MUD_j) = \frac{|MUD_i \cap MUD_j|}{|MUD_i \cup MUD_j|} \quad (2)$$

MUDIS then divides the ACEs of the two MUDs into four groups: identical ACEs, similar ACEs, clustered ACES, and dissimilar ones. This separation highlights valuable connections and patterns between the ACEs, enabling MUDIS users to gain meaningful insights. The algorithm uses the following four steps, where each step output is visualized in a different frame on the MUDIS visualization screen (see Figure 2):

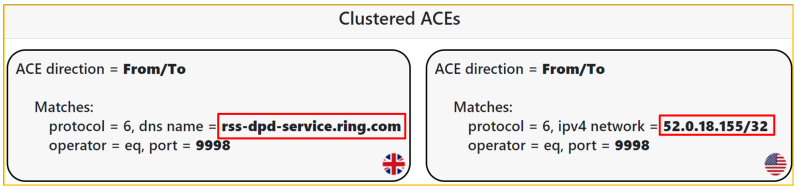

Figure 3: Ring Doorbell clustered ACEs

1) **Find identical ACEs**. ACEs in which all of their fields are identical.
2) **Find similar ACEs**. MUDIS marks two ACEs of two MUDs as similar if they have similar domain names and all other fields in the ACEs are identical (port, protocol, etc.). A similar domain name is defined as follows: let the domain name be in the format subDomain-part.mainDomain.suffixTLD. The suffixTLD is the top level domain (e.g., .com, .net or .us) or a combination of top level domains (e.g., .com.tw). The subDomain part can be empty or include multiple sub-domains (i.e., sub1.sub2.sub3.mainDomain.suffixTLD). Two domain names are similar if and only if their mainDomain part is equal. For example, In Figure 4 we compared two different devices, the Bulb and Plug, of the same manufacture (Tp-link). MUDIS found similar ACEs that differ only in their sub-domain parts: the Bulb uses **n-devs**.tplinkcloud.com whereas the Plug uses **devs**.tplinkcloud.com.
3) **Find Clustered ACEs**. After the first two steps, if there are still unmatched ACEs remaining, we cluster the ACEs with the same traffic directions into two types of clusters: (1) ACEs with similar or equal endpoints but with different ports or protocols (2) ACEs with the same ports and protocol but with different endpoints. Note that each cluster type may contain several clustered ACEs and the same ACE can be clustered with multiple ACEs of the other MUD.
   For example, In Figure 3 we compared the Ring Doorbell device behavior in two different locations (UK and US). MUDIS managed to automatically spot a difference and clustered two ACEs that communicate with the same protocol and unique high port, but use two different endpoints.
4) **Find all dissimilar ACEs** Finally, we gather all the non-clustered ACEs into the dissimilar ACEs section. For example, in Figure 5 we compared two MUDS of the same Xiaomi device, where one of the MUDs was generated out of a PCAP with a rare action such as firmware update. MUDIS found a unique domain of Xiaomi that downloads a new firmware version over port 80.

B. MUD Generalization

The goal of this feature is to create a generalized MUD that is comprehensive, tight, and secure. **Comprehensive** means that the generalized MUD should be applicable to the two

Figure 4: MUDs Comparison (a) and generalization (b) of two different devices of the same manufacturer

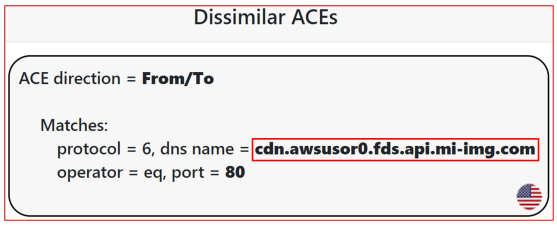

Figure 5: The Xiaomi Bulb's unique domain when downloading a new firmware version

MUDs presented. It must also be **tight and secure** because a MUD's main goal is to whitelist only legitimate flows of the IoT and thereby reduce the device attack surface.

The generalization algorithm has three steps:

1) **Add equal ACEs.** All the identical ACEs that appear in both MUDs are added only once to the generalized MUD.
2) **Generalize similar ACEs by generalizing similar domains.** As mentioned previously, MUDIS marks two ACEs as similar if they have similar domain names, and all other fields are identical. MUDIS use ranges in domain (e.g., *.iotvendor.com) to create a generalized ACE from two similar ACEs, as shown in Figure 4(b). MUDIS only generalizes sub-domains where the whole domain is in the control of the main domain owner i.e., the IoT manufacturer or the exact IoT service that the manufacturer uses. Moreover, to keep the generalized MUD tight and secure, MUDIS automatically identifies problematic scenarios and does not generalize ACEs with different domain suffixes (TLDs) and known cloud services that are shared across clients (e.g., *.s3.amazonaws.com). This is aligned with the IETF Operational Consideration for the use of DNS in IoT [9].
3) **Adding dissimilar and clustered ACEs.** Following previous steps, we are left with any ACEs in both MUDs that are neither identical nor similar. These are added to the generalized MUD as-is. However, to ensure fast convergence, if some ACEs share a domain that can be safely generalized, we generalize it for all the ACEs in which it appears to support future differences that we have not yet encountered.

A naive generalization algorithm that simply unifies all available MUDs, would also be both comprehensive and tight. However, MUDIS generalization algorithm demonstrates superior performance compared to the naıve algorithm in terms of converging velocity and ACEs cardinality [6]. The generalized MUD is also more explainable and easier to implement in a firewall, due to the reduced amount of rules.

III. Demonstration

In this demo, we present some of our system's features. We then use the system to analyze and explain use-cases by inspecting different MUD files for three scenarios: (i) the same device captured in different locations; (ii) different devices from the same manufacture; (iii) the same device captured in unique situations for example, during firmware update.

In each scenario, we show how MUDIS can compare the different MUDs and analyze their different network behavior using the MUDIS visualizer; this demonstrates how easy it is to spot correlations and gain meaningful insights. We also generalize two pairs of MUD files. Each generalization will present a different aspect of the generalization algorithm while examining some of the challenges.

IV. Conclusions

This paper introduces MUDIS, a tool for MUD inspection, comparison, and generalization. We encourage the use of MUDIS to achieve better and deeper understanding over the network behavior of IoT devices.

Acknowledgement: This research was supported in part by a Cisco grant.

References

[1] E. Lear, R. Droms, and D. Romascanu, "Manufacturer Usage Description Specification," RFC 8520, Mar. 2019. [Online]. Available: https://rfc-editor.org/rfc/rfc8520.txt
[2] A. Hamza, D. Ranathunga, H. Gharakheili, M. Roughan, and V. Sivaraman, "Clear as mud: Generating, validating and applying iot behavioral profiles," in *Workshop on IoT Security and Privacy*. USA: Association for Computing, 2018, pp. 8–14.
[3] N. I. o. S. NIST and Technology, "Mud-pd is a tool assist in the characterization of iot device network behavior and the creation and definition of appropriate mud files." [Online]. Available: https://github.com/usnistgov/MUD-PD
[4] NIST, "National institute of standards and technology," Sep 2021. [Online]. Available: https://www.nist.gov/
[5] N. I. o. S. NIST and Technology, "Methodology for characterizing network behavior of internet of things devices," Apr 2020. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04012020-draft.pdf
[6] A. Bremler-Barr, B. Meyuhas, and R. Shister, "One mud to rule them all: Iot location impact," in *NOMS 2022 - short paper*, apr 2022, accepted for publication.
[7] R. Shister, B. Meyuhas, and A. Bremler-Barr, "Mudis - mud inspection system." [Online]. Available: https://github.com/ransh93/MUDIS
[8] B. Meyuhas, Shister, "Mud files dataset in different locations." 2021. [Online]. Available: https://github.com/barmey/IoT_mud_files_locations
[9] M. Richardson and W. Pan, "Operational Considerations for use of DNS in IoT devices," IETF, Internet-Draft draft-ietf-opsawg-mud-iot-dns-considerations-02, Jul. 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-mud-iot-dns-considerations-02