Withdrawn Draft

Warning Notice

The attached draft document has been withdrawn, and is provided solely for historical purposes. It has been superseded by the document identified below.

Withdrawal Date January 11, 2022

Original Release Date April 01, 2020

Superseding Document

Status	Initial Public Draft (IPD)
Series/Number	NIST Interagency or Internal Report 8349
Title	Methodology for Characterizing Network Behavior of Internet of Things Devices
Publication Date	January 11, 2022
DOI	https://doi.org/10.6028/NIST.IR.8349-draft
CSRC URL	https://csrc.nist.gov/publications/detail/nistir/8349/draft
Additional Information	https://www.nccoe.nist.gov/projects/securing-home-iot-devices-using-mud

National Institute of Standards and Technology
U.S. Department of Commerce

Methodology for Characterizing Network Behavior of Internet of Things Devices

Paul Watrobski
Joshua Klosterman
The MITRE Corporation
McLean, VA

William Barker
Dakota Consulting
Gaithersburg, MD

Murugiah Souppaya
Computer Security Division
Information Technology Laboratory

April 1, 2020

This publication is available free of charge from:
https://doi.org/10.6028/NIST.CSWP.04012020-draft

National Institute of Standards and Technology
U.S. Department of Commerce

Abstract

This white paper describes an approach to determining and documenting the device types and communication behaviors of Internet of Things (IoT) devices connected to a network. From this identification and documentation, files based on the Manufacturer Usage Description (MUD) specification can be created and used by manufacturers and network administrators to manage access to and from those devices. The paper also describes the current state of implementation of the approach and proposals for future development.

Keywords

device characterization; Internet of Things; IoT; Manufacturer Usage Description; MUD; MUD-PD.

Disclaimer

Any mention of commercial products or reference to commercial organizations is for information only; it does not imply recommendation or endorsement by the National Institute of Standards and Technology (NIST), nor does it imply that the products mentioned are necessarily the best available for the purpose.

Additional Information

For additional information on NIST's cybersecurity programs, projects, and publications, visit the National Cybersecurity Center of Excellence (NCCoE) and the Computer Security Resource Center. Information on other efforts at NIST and in the Information Technology Laboratory (ITL) is also available.

Supplemental Content

NIST's NCCoE created MUD-PD, a tool to assist in developing MUD files. The tool is described in greater detail in Section 3.2.

See GitHub: https://www.github.com/usnistgov/MUD-PD.

Public comment period: *April 1, 2020 through May 1, 2020*

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930
Email: mitigating-iot-ddos-nccoe@nist.gov .

All comments are subject to release under the Freedom of Information Act (FOIA).

Acknowledgments

The authors wish to thank the following individuals for their generous contributions of expertise and time.

Name	Organization
Luca Deri	Institute for Informatics and Telematics
Donna Dodson	National Institute of Standards and Technology
William Polk	National Institute of Standards and Technology
Karen Scarfone	Scarfone Cybersecurity
Parisa Grayeli	The MITRE Corporation
Blaine Mulugeta	The MITRE Corporation
Susan Symington	The MITRE Corporation
Hassan Habibi Gharakheili	University of New South Wales
Russ Housley	Vigil Security, LLC

Document Conventions

This paper utilizes several terms for which contradictory or generic definitions exist in literature. For purposes of this paper, the following definitions have been coined or adopted:

characterizing: the acts of collecting information intended to be used in describing the behavior and/or characteristics pertaining to a device, analyzing the information, and/or storing the information.

fingerprinting: the act of collecting information intended to help uniquely identify a device.

MUD file: a file containing information that describes an IoT device and associated suggested specific network behavior, as described in the MUD specification [1]. The term "MUD profile" is used throughout existing literature and is synonymous with "MUD file". This paper adheres to the use of "MUD file" as defined in the MUD specification.

MUD file accuracy: how precisely a particular MUD file captures the full communication requirements of an IoT device—in particular, the extent to which it lists all potential communications that the device may need to perform its intended function (comprehensiveness) and the extent to which it avoids listing communications that the device does not need (correctness). Note that it may be impossible to ensure complete accuracy of a MUD file even if the file is created by the manufacturer of the device. For some devices, it may be impractical or even impossible to test every possible situation or network configuration that could alter device behavior, and potential communication requirements that would be revealed by those situations may remain unknown. The simpler the IoT device, the easier it will be to create an accurate MUD file.

Audience

The focus of this paper is on capturing the information needed to develop MUD files for IoT devices, and this paper is written for the benefit of those who would like to build, create, or utilize MUD files, including:

- IoT device manufacturers and developers,
- network administrators,
- IoT device vulnerability researchers and analysts,
- network equipment developers and manufacturers, and
- service providers that develop and utilize components based on the MUD specification.

Trademark Information

Table of Contents

1 Introduction 1
1.1 Purpose and Scope 1
1.2 Challenges 2
1.3 Background 2
2 Network Traffic Capture Methodology 4
2.1 Capture Strategy 4
2.2 Capture Procedure 8
2.3 Documentation Strategy 10
3 Analysis Use Cases and Tools 11
3.1 Manual MUD File Generation 11
3.2 MUD-PD 12
3.3 MUD-PD Support for Privacy Analysis 21
4 Next Steps 23
4.1 Extending MUD-PD Features 23
4.2 Developing a MUD Pipeline 23
4.3 Community Feedback 26
References 28

List of Appendices

Appendix A— Example Capture Environment 30
Appendix B— Acronyms 31

1 Introduction

The National Institute of Standards and Technology's (NIST's) National Cybersecurity Center of Excellence (NCCoE) is working to improve the ability of network administrators and operators of Internet of Things (IoT) networks to identify, understand, and document network communication requirements of IoT devices. Documenting the types of devices and communication behaviors of those devices can allow creation of files based on the Manufacturer Usage Description (MUD) specification, which can be used by network administrators to manage access to and from those devices [1].

Note that the Document Conventions section earlier in this white paper defines several terms as used in this paper. Readers should review those definitions before proceeding.

1.1 Purpose and Scope

The purpose of this publication is to demonstrate how to use device characterization techniques to describe the communication requirements of IoT devices. This publication focuses on the capture of network communications involving IoT devices necessary to generate MUD files. The methodology seeks to allow for analysis of the full range of IoT device network traffic behaviors that can reasonably be expected. This includes examining a variety of factors that could potentially alter an IoT device's behavior at each stage of the device's life cycle. An important item to note is that this work is focused on documenting the behavior of IoT devices, not on establishing the identity of the devices themselves.

One of the primary motivators for developing this methodology is to support developing files that could be used in the application of MUD [1]. MUD provides a standard way to specify the network communications that a device requires to perform its intended function. The MUD specification supports development of MUD file that defines expected and permitted network activity and behavior. Accurately generating a MUD file for a networked device requires a comprehensive picture of the device's potential actions.

A MUD file's accuracy is based on two concepts: comprehensiveness—the extent to which it lists all potential communications that the device may need to perform its intended function, and correctness—the extent to which it avoids listing communications that the device does not need. An accurate MUD file will contain all the potential communications necessary for the device to perform its intended function while not listing any unneeded communications. However, because the final decision of what actions a device may perform is ultimately up to the local network administrator [1], the local administrator tasked with implementing the device may decide that the deployed device's MUD file should be more or less restrictive than the MUD file provided by the manufacturer.

It should be noted that a device may have a minimum set of permissions for the device to operate at all. Additionally, a network administrator may wish to create a MUD file for a legacy device, i.e., a device for which the manufacturer has not provided a MUD file. The goal is to have an accurate MUD file. The methodology described herein provides a framework that allows capture of the often-large range of behaviors required for generating accurate MUD files.

In addition to prescribing a methodology for capturing an IoT device's behavior on a network, use of this behavior information to create MUD files can be leveraged by MUD-PD, described in Section 3.2. The MUD-PD tool can be used in generating MUD files for use on a live network. Developers, network administrators, and researchers can take advantage of the methodology to develop a comprehensive data set that can be used for generating MUD files, investigating security and privacy concerns, developing machine learning algorithms, and more. The methodology described has been developed on internet protocol (IP)-based networks, but it can potentially be utilized with other types of networks as well. It is important to note that this type of analysis assumes that the IoT devices have not been tampered with or compromised by a malicious actor at any point in the process. The analysis method also assumes that the IoT devices are operating as intended by the manufacturers of the devices.

1.2 Challenges

For network administrators to properly secure a network, they need to understand what devices are on the network and what network communication each device requires to perform its intended function. In the case of networks that include IoT devices, it is often difficult to identify each individual device, much less know what access is required by each device to other network components, and what access other network components need to each device. To address this challenge, many organizations are implementing IoT device fingerprinting and characterization methods to identify the types of devices on a network. Once the IoT device type is known for each device, the network administrator can begin to manage security and access control for the devices [2].

Comprehensively describing the characteristics of IoT devices is made difficult by a variety of factors. For example, IoT devices are often subject to internal changes that may affect their behavior. These changes can be caused by software updates, firmware updates, new hardware, and so on. External changes can also occur with hardware replacements, integrations with other IoT devices, connections to new networks, and more. These changes can increase the complexity involved in tracking an IoT device's behavior and, by extension, increase the difficulty of accurately characterizing an IoT device. User activities can also have significant effects on IoT device behavior. For example, two cameras created by the same manufacturer may display drastically different behaviors if they are used for different purposes. Additionally, the behaviors may be distinct for the different firmware or hardware revisions of the same device. Many IoT devices are also created as variants based on the design of an existing IoT device, which can make their behaviors appear similar, even if the IoT devices are technically distinct from one another.

1.3 Background

As stated in Section 1.2, secure and reliable administration of any network requires knowledge of what devices are on the network and what network communication each requires to perform its intended function. In the case of networks that do not support MUD, it can be challenging to detect and identify each individual device and the connection requirements involved, and to make that information available to access management processes. The first challenge is just detecting the devices that are on the network. This paper does not cover device detection and

identification; it focuses on device communication analysis and characterization. However, Section 2.1.5 describes two tools that support manual device identification and analysis.

Once an IoT device's presence and type have been established, a network administrator can begin to develop information regarding the device's characteristics and behavior. Approaches like those of the Princeton IoT Inspector [3] and ProfilIoT's use of machine learning [4] are being used to characterize and identify IoT devices, which can provide insight into security and privacy issues associated with each device. However, not all fingerprinting and characterization schemes are equivalent. These schemes are often created based on a limited set of data derived from network traffic that allows them to accurately identify just the device type. The network traffic information used to develop these schemes include packet headers, network ports, packet timing, handshakes, and other information that might be unique to a particular IoT device [5], [6]. Given the limited set of data used to develop the fingerprints, the fingerprints inherently do not contain the information necessary to determine a device's full range of potential behaviors.

As previously stated, the goal of the MUD specification [1] is to provide a standard method for IoT devices to "signal to the network the access and network functionality they require to properly function". This is accomplished by using a MUD file, which can allow a network administrator to know what access control rules should apply to the IoT device. However, building a comprehensive MUD file for an IoT device requires detailed and accurate knowledge of all potential network behaviors required for that device to perform its intended function. Because a manufacturer may not be able to predict all operational environments in which a device is used, there is no guarantee that all manufacturer-provided MUD files are comprehensive. If a network administrator enforces an inaccurate MUD file, the functionality of the device can be severely impaired or potentially lead to vulnerabilities. Therefore, it is imperative that any MUD file be as accurate as possible. This paper describes a way to build an accurate MUD file based on network traffic data that reveals information about the IoT device's potential behavior. The methodology described is designed to create an accurate set of network traffic data capturing as much of the IoT device's potential behavior as possible. Assuming the captured behavior is deemed permissible, it would be included in the device's MUD file.

The NCCoE has developed the MUD-PD tool, which allows creation of MUD files based on capture of relevant network traffic information. MUD-PD requires a diverse set of network traffic captures to generate accurate MUD files. The tool extracts and aggregates pertinent information that allows creation of accurate MUD files without manually parsing a large set of network traffic data. This tool can drastically reduce the time and effort required to generate MUD files compared with manually creating MUD files. Developing MUD files consists of two major steps: traffic capture and traffic analysis. Section 2 discusses traffic capture strategy, tools, example procedures, and documentation. Section 3 discusses analysis of traffic communications, privacy implications, and MUD file generation using the MUD-PD tool.

2 Network Traffic Capture Methodology

Properly generating an accurate MUD file requires a comprehensive data set that reflects the greatest possible range of intended device behaviors for each networked device. In the case of MUD files that can and will be used for network security and access control, it is imperative that each generated file be sufficiently accurate to prevent false reporting of network activity and placement of restrictions on devices that may prevent them from functioning properly. The methodology described in this section is designed to support capture of the information needed for IoT device analysis and MUD file generation. This methodology is based on network traffic and does not account for device behavior that cannot be observed from network traffic. Observed device behaviors outside the scope of this methodology should be documented through other means.

2.1 Capture Strategy

Capturing a wide range of intended device behaviors requires that communications to and from the IoT device be captured under a wide range of states and environmental conditions. This section describes capture during different life-cycle stages; environmental variables that may affect device behavior; capture approaches; placement of capture tools within network architectures; and available capture tools. The information listed in this section should be documented for each capture activity for each IoT device to support analysis of the device's behavior.

2.1.1 IoT Device Life-Cycle Phases

There are varying taxonomies of IoT device life cycles, but this white paper organizes device life-cycle components into three broad phases for IoT device traffic analysis: setup, normal operation, and decommissioning/removal.

2.1.1.1 Setup

The setup phase includes everything needed to initially connect an IoT device to a network and to take configuration actions necessary for the device to be fully functional and ready to begin normal operations. Setup typically begins with a wired or wireless connection of the device to the network. Once the device is connected, setup processes can include firmware updates; connections to smart hubs, smartphones, and other devices; and other processes that must be completed for all of the device's intended functions and features to be enabled. While following the manufacturer's instructions may be adequate for most situations involving setup behaviors, deviation from those instructions may be necessary to capture the device's behavior under some circumstances (e.g., not connecting an IoT device to an associated cloud service may result in unique behavior for devices that a manufacturer assumes will be connected to a cloud service). Initial connection to cloud/internet-based services may be required for some devices. This phase may also include connection of an IoT device to a smartphone or another device that is expected to manage the device (such as a controller/smart hub). Setup failure situations (such as being unable to properly register with a cloud service) can also produce setup connectivity behaviors different from those anticipated by the manufacturer.

2.1.1.2 Normal Operation

The "normal operation" phase captures an IoT device's behavior for the majority of its service life after it has been set up and is performing its intended function. This phase covers a wide range of behaviors, such as human-to-device interactions, controller or smart hub-to-device interactions, and cloud service-to-device interactions. It also covers device-initiated behaviors that can occur without human interaction. Software and firmware updates may occur with or without human initiation or interaction and can cause an intended change in device behavior. Capture of both human-initiated updates and automatic updates is important, though capture of automatic updates may be the more challenging. Other types of interactions during normal operation may include remote control through smartphones and cloud-based services. Normal operation failure situations, such as being unable to access required resources, can also produce anomalous behaviors. "Unexpected" scenarios, including removing essential devices, removing the controller/smart hub, or performing a hard reset on the IoT device, are still considered normal operation and should also be examined.

2.1.1.3 Decommissioning/Removal

The final phase in an IoT device's life cycle (before the device is reused elsewhere or reaches end of life) includes the process of de-registering the IoT device from other devices, such as controllers/smart hubs, and/or cloud services (decommissioning) and removing it from the network (removal). If manufacturer instructions for this process exist, they should be included as part of the capture-planning process if possible. If no instructions exist, a factory reset is recommended. Factory reset brings the device back to its initial configuration. (Note: Firmware updates may not be rolled back during the factory reset process.) This paper treats the factory reset process as falling under the decommissioning/removal phase because a factory reset can sometimes de-register the device from a cloud service and/or disconnect the IoT device from the network. Inclusion of other types of removal situations is also recommended because IoT devices can sometimes be removed from a network without taking prior decommissioning actions. If the device is used in a different role or by a new owner, subsequent actions are treated here as falling within a new setup phase. Capture plans should cover both device-initiated behaviors and behaviors triggered by human interaction during decommissioning and removal.

2.1.2 Environmental Variables

The IoT device should be examined under a wide variety of environmental conditions to capture the largest possible range of intended device behaviors. For example, if an IoT device is not permitted access to the internet, it may not be able to complete some of the communications on which it relies to function as intended (e.g., cloud-based manufacturer support services or network time services). This can cause the IoT device to exhibit different behaviors on the network than those originally anticipated or documented by the manufacturer. As discussed in Section 1.3, there is currently no guarantee that the manufacturer-provided MUD file will cover every communication pattern that the device may exhibit. For example, it is possible that the device's apparent behavior may have changed due to updates of third-party libraries, the characteristics of which may have been overlooked. Behaviors like this need to be captured to provide a more accurate characterization of the IoT device.

This subsection provides an example set of environmental variables that can be applied during each of the three life-cycle phases described in Section 2.1.1. This is not a complete list, but depending on the device type and design, each of the variables has the potential to change the behavior of an IoT device. For consistency and to limit confusion, these variables should persist throughout the duration of a network traffic capture process and should not be added or removed after the capture has begun. There are exceptions to this rule, such as capturing behaviors when emulating an internet outage. Any deviations from persistent variables should be clearly documented.

- **No internet** removes internet access from the local network to which the IoT device is connected. This can limit an IoT device's access to resources and modify the IoT device's behavior.
- **Preferred DNS servers blocked** tests a device's behavior when its preferred domain name system (DNS) servers have been blocked. For example, an IoT device may be configured to rely on DNS servers managed by the manufacturer. If access to these DNS servers is restricted, the IoT device's functionality will be reduced unless compensating measures are taken. This can result in modification of the IoT device's behavior.
- **Device isolation** indicates that the device is alone on the local network; that is, no other devices are connected except essential network or other communication components needed for the IoT device to function properly. For example, if the IoT device needs to be controlled by a controller/smart hub or smartphone, this device may also be connected during the capture.
- **No human interaction** means that no human interaction or configuration of the device has taken place for the duration of the capture activity. The device will not be preprogrammed by the analysts to take any actions prior to the start of the capture process.
- **Controller/smart hub control** indicates that the device has been or will be connected to a controller/smart hub during the capture. An IoT device connected to a controller/smart hub will typically display behavior that is different from that of a device that is not.
- **Same manufacturer** means that at least one device from the same manufacturer has been connected to the network before the capture has begun. It is likely that a network may have two IoT devices from the same manufacturer. Additionally, many manufacturers have been working to create their own IoT "ecosystems." Because some IoT devices are designed to communicate with other IoT devices from the same manufacturer, connecting multiple devices from the same manufacturer may reveal additional behavior not seen when one of the IoT devices is the only one from that manufacturer connected to the network.
- **Full network** indicates that enough active devices to simulate an IoT application are connected to the local network before the beginning of the capture. As the purpose and scope of networks that support IoT devices can vary widely and are often application-dependent, it is up to the analyst to determine how many and/or what variety of devices is considered a full network. The presence of other devices on the same network may affect the behavior of IoT devices being characterized.

2.1.3 Activity-Based and Time-Based Capture Approaches

Activity-based captures are focused on IoT device behavior solely during a specified set of actions. For example, capturing IoT device setup behaviors does not require a specific amount of time; its beginning and completion are determined only by the duration of the setup process.

Time-based captures are focused on capturing IoT device behavior during a specific time period. For example, capturing IoT device behaviors throughout an entire day of normal operation can allow observation and documentation of a wide range of behaviors (e.g., device-initiated behaviors). Some behaviors may be observed only over a longer term. One example of this property involves devices that "learn" the user's behavior and modify functionality accordingly. These devices may behave in a different way over the weekend from during the week or when the learned pattern is broken, such as on a holiday or when the user is traveling for an extended period.

2.1.4 Network Architecture and Capture Approach

The ideal capture activity will capture the network traffic among all hosts on the local network and all communications entering and leaving the local network. In cases of smaller and/or simpler networks, capture of network traffic directly from a single gateway may be sufficient because the gateway will receive all communication both to and from the local network and among all network devices. An example of a capture setup using a single gateway can be found in Appendix A. In larger networks where network traffic does not flow through a single gateway, capture of network traffic from multiple locations throughout the network is recommended where possible. These capture locations should be carefully chosen to ensure that all relevant traffic can be properly captured.

The capture approach adopted may depend on the hardware available. The capture device will need sufficient resources to store all captured traffic. The absence of sufficient processing power, memory, or storage is likely to cause network packets to be dropped and may compromise the accuracy and integrity of the capture.

Once network capture locations have been determined, the method of capture should be chosen. Capture of traffic directly on the chosen gateway/router/switch is ideal if the network device's resources are sufficient for the task. This allows capturing network traffic from any or all of the Ethernet ports and wireless radios managed by the network device and saving the captured information directly. It is not always possible to capture traffic directly on the network device, but alternatives are available for situations that do not permit capture in this manner. For example, placing a network tap in-line on a wired IoT device can provide access to the desired communication. Another alternative is using a mirrored or switched port analyzer (SPAN) port to send all traffic from a port or virtual local area network to a capture device that is listening on a selected port. For IoT devices that communicate over a wireless network, using a wireless network adapter in promiscuous mode will allow capture of wireless traffic. This is not always an ideal option, as there may be instances where interference with capturing wireless traffic is unavoidable (e.g., wireless isolation is being used).

2.1.5 Capture Tools

Various tools are available for capturing network traffic. Two of the most widely used are tcpdump and Wireshark.

2.1.5.1 tcpdump

tcpdump is a lightweight command-line-based tool that can be used on Cisco IOS, Junos OS, and many Linux-based router and switch operating systems. Packet captures (pcaps) can be saved to a standard pcap file format, which is commonly used to store network traffic data. The following command demonstrates tcpdump usage:

```
bash$ tcpdump -i eth0 -s0 -n -B 2000000 -w capture.pcap
```

- "tcpdump" starts the capture program.
- "-i eth0" instructs tcpdump to start capturing packets from the interface eth0.
- "-s0" sets the snapshot length to an unlimited size, allowing capture of larger packets. tcpdump normally truncates IPv4 packets larger than 68 bytes.
- "-n" turns off host name resolution, which reduces the processing and buffer resources needed to capture properly.
- "-B 2000000" sets the operating system capture buffer size to 2,000,000 kibibytes, allowing capture of a greater amount of network traffic. It is important to note that packet drops can still occur in the driver and in the kernel, so it is important to ensure the capture hardware is adequate to the task.
- "-w capture.pcap" saves network traffic to a file named capture.pcap.

2.1.5.2 Wireshark

Wireshark is one of the most readily available packet capture and analysis tools, and it is open source. Wireshark provides a graphical user interface during both capture and analysis. It also has a command-line-based capture utility called tshark, which can perform both capture and analysis functions.

Wireshark is supported by Windows, macOS, and a wide range of Unix and Unix-like platforms, including Linux and BSD. Use of Wireshark as a capture tool often involves setting up a mirrored/SPAN port or a network tap to ensure that Wireshark can capture as much relevant network traffic as possible. Wireshark also supports putting network interfaces into promiscuous mode, which is often necessary to properly capture wireless network traffic. Wireshark supports the PCAP Next Generation Dump (PcapNg) file format, which allows addition of metadata to network traffic captures. See Section 2.3 for further details.

2.2 Capture Procedure

This section lists example procedures for capturing network traffic. These examples focus on capturing directly from a router. They are purposely generalized to be applicable to many

situations and may be modified/customized as required. See Appendix A for an example of a network in which these procedures could be used.

2.2.1 Device Setup Capture

Device setup captures are mainly activity-based captures. An example process for this capture type is as follows:

1. Select, implement, and document environmental variables to be used for this capture.
2. Start packet capture on router.
3. Begin device setup according to manufacturer instructions.
4. Complete device setup.
5. End packet capture.
6. Transfer packet capture file from router to external storage for analysis.

2.2.2 Normal Operation Capture

Capture of normal operation can be either activity-based or time-based. The process for this capture type is as follows:

1. Select, implement, and document environmental variables to be used for this capture.
2. Start packet capture on router.
3. Begin normal operation for device (following manufacturer directions, if available).
4. Document actions/activity taken.
5. End device operations.
6. End packet capture.
7. Transfer packet capture file from router to external storage for analysis.

2.2.3 Decommissioning/Removal Capture

Decommissioning/removal captures are mainly activity-based. The process for this capture type is as follows:

1. Start packet capture on router.
2. Begin decommissioning process for device (remove from smartphone application/smart hub/cloud service)
3. End decommissioning process.
4. Remove the device from the network.
5. End packet capture.
6. Transfer packet capture file from router to external storage for analysis.

2.3 Documentation Strategy

After the network traffic captures have been completed, it is important to document the conditions and other details that were applicable to each packet capture. Documenting the life-cycle phase, environmental variables involved, and other important factors can greatly help with subsequent analysis of the network traffic. Options for recording this information include editing the file name, using a text document, storing information in a database, or recording metadata to the capture file itself.

Note that the MUD specification does not contain mechanisms for allowing or blocking traffic under specific conditions. However, it may be useful to a network administrator to trace network activity to a particular event. For a situation like this, and to gain a better understanding of a device's behavior, it is important to keep a log of the activities, actions performed, and environmental variables during each capture.

There are a number of ways to document this information. The simplest is to manually write descriptions for each capture and store the text documents along with the captures. This approach is not scalable and may lead to mistakes where capture-document pairs are separated. An alternative is to use the comment field in the PcapNg. PcapNg extends the capabilities of the libpcap format. Wireshark can convert pcapfiles to PcapNg, and comments can be added by using the graphical user interface (GUI). The terminal-based interface to Wireshark, tshark, allows inclusion of comments while taking a network capture. The following command allows insertion of a text description of the capture environment and variables. This way, the information is contained within the capture itself.

```
bash$ tshark -w capture.pcapng --capture-comment "Example comment."
```

- The same `-i`, `-s`, `-n`, and `-B` options used in Section 2.1.5.1 (tcpdump) can be used here.
- The default file type for tshark captures is PcapNg.
- The `--capture-comment` option allows text comments to be added during a capture.

Use of the comment field in PcapNg is still not an optimal solution. PcapNg is limited in that it requires further manual interaction for the information to be consumed and used by interested parties. As the comment field allows arbitrary text input, it is possible to embed information in JavaScript Object Notation (JSON) format. JSON is computer parsable/readable. Consequently, the NCCoE has begun developing a tool to format the desired information as JSON and insert it into the comment field of a pcapng file. This can be initiated at the start of a capture or inserted afterward. As JSON is somewhat human readable and the data being added is fairly simple, a user can still understand the necessary information from the output.

3 Analysis Use Cases and Tools

This section describes several use cases for the characterization methodology along with useful analysis tools.

3.1 Manual MUD File Generation

Currently, MUD files are often generated manually. Although there are tools such as MUD Maker [7] (mudmaker.org) that allow a user to input the necessary values without concern for the computer syntax, most MUD files are still written by hand and require significant effort to complete. After capturing the necessary data through network traffic captures (as described in Section 2), manual analysis is needed to extract the information needed. Relevant information often includes network destinations with which the IoT device has communicated, ports and protocols utilized, and other data regarding the device's behavior. This may be achieved using network traffic-analysis tools like Wireshark and NetworkMiner, which enable extraction of the information necessary for a MUD file.

3.1.1 Wireshark

Wireshark is a well-known open-source tool for network traffic analysis (as well as for packet capture, as discussed in Section 2.1.5.2). It can be run on Windows, OSX/macOS, and Linux. It supports deep packet inspection for hundreds of protocols, which allows the user to sift through packet bytes and extract the relevant information. Analysis can be performed using a wide array of display filters, and results can be exported in a variety of formats. In addition, Wireshark includes decryption support for Secure Sockets Layer/Transport Layer Security, and Wi-Fi Protected Access (WPA)/WPA2. The combination of capabilities allows analysis needed to generate a MUD file from the packet capture file generated as described in Section 2.

3.1.2 NetworkMiner

NetworkMiner is another popular open-source network traffic-analysis tool, and it is built and maintained by Netresec. It is officially supported only on Windows but can be run in macOS through Mono. While it can also be used for packet capture, NetworkMiner's strengths lie in processing network traffic captures and displaying relevant information quickly and easily. It automatically displays network hosts involved and extracts files, images, messages, and credentials. NetworkMiner also compiles a list of individual sessions between hosts and DNS requests throughout the network traffic capture. NetworkMiner does not have the deep packet inspection capabilities that Wireshark has, but it is a quick and helpful tool that complements Wireshark's depth.

3.1.3 Overview of Manual MUD File Generation Process

The process for generating/developing a MUD file begins with a set of network communication capture files. The assumption is that this set includes diverse behaviors such as those described in Section 2. For each network communication capture file, the following steps may be performed:

1. Inspect packets to locate and record:
    a. IoT device (source) addresses (media access control [MAC], IPv4/6)
    b. destinations
        i. addresses (MAC, IPv4/6)
        ii. domain names
    c. protocols and ports (transmission control protocol [TCP]/User Datagram Protocol [UDP], IPv4/6)
        i. source-initiated (the IoT device being characterized)
        ii. destination-initiated (a device outside the IoT device being characterized)
2. Identify the destination devices and servers:
    a. type of device
    b. manufacturer

Once all of this information has been collected for every packet capture, the final steps are to consolidate it and write the MUD file. The information should be consolidated into a unique list, as some devices and protocols may appear in multiple network communication capture files. As mentioned above, writing the MUD file may be done manually in a simple text editor or through text entry into MUD Maker [7]. Before any MUD file is deployed, it should be manually verified, and the contents of the MUD file should be confirmed to accurately depict the intended and accepted communication requirements of the IoT device.

3.2 MUD-PD

The NCCoE is developing the open-source MUD-PD tool as an example of how to reduce the barrier to entry for vendors to create accurate MUD files for their devices. MUD-PD supplements currently available methodologies for writing MUD files that use packet inspection tools like Wireshark and NetworkMiner. Several approaches to automated MUD file generation currently exist. These include one devised by a researcher at the University of Twente [8], and an open-source tool created by the University of New South Wales (UNSW) called MUDgee [9]. The MUDgee tool takes a single network traffic capture file and generates a MUD file based on the observed network behavior. MUDgee assumes that all the activity seen is intended and is nonmalicious.

MUD-PD builds on MUDgee and supports several enhancements. MUDgee currently supports a single network traffic capture file for use in MUD file generation. This can be augmented by using packet merging tools (where multiple network traffic captures are concatenated into one file), but packet merging adds to the complexity of using the tool. The interface is terminal-based, a limitation with respect to user friendliness. Currently, the MUD files generated by MUDgee are missing a number of features that are included in the MUD specification. Certain features, like support for the “same manufacturer” and “controller” classes, must be added manually based on additional documentation or user input. The core of the MUD file generation function in MUD-PD is built upon MUDgee and will continue to address these constraints. The

NCCoE's goal is to extend the ability to generate a complete MUD file from network traffic captures.

The initial version of MUD-PD required that the user manually enter all of the metadata as the files are imported. While this functionality is still present, its user interface has been simplified. Also, compatibility with the PcapNg file format, specifically extraction of JSON-formatted data, is in development. The goal of these enhancements is to simplify the import process and embed information on the nature of the capture within the packet capture itself to enable metadata to automatically be extracted and imported. The combination of network capture data and documentation allows the MUD file-generation process to be more comprehensive and to be automated, requiring very little user input.

MUD-PD parses and extracts data from packet captures and organizes it in a relational database. The GUI allows the user to examine individual packets or any combination of packets when inspecting the communications of specific devices. As the metadata about the physical actions and activities that occurred during the network captures are also stored, the user can gain greater insight as to how the network activity and physical world may be associated. In addition to being an exploratory tool intended to aid MUD file development, the database at its core can be queried through any MySQL interface. This allows more potential uses.

Additional functions built into MUD-PD include generation of a human-readable device report that summarizes what is discovered on the network and general metadata for each individual network traffic capture. Another significant added function is the automated generation of a MUD file. The MUD file can then be used as is or adjusted and tweaked by the developer or network administrator as they see fit to protect the device and MUD-enabled network. MUD files are currently generated through a custom interface to MUDgee. To avoid concerns about future compatibility and to extend the characteristics generated, work is underway to generate the MUD file directly from the database itself without relying on MUDgee.

3.2.1 Current Feature Set

This subsection provides a high-level overview of MUD-PD as it currently stands. In Section 3.2.2, a tour of the tool illustrates its finer details. MUD-PD has three main functions:

- **information import:** The first function is to import network traffic captures. During this step, the user is provided the opportunity to input metadata about the capture. The goal of importing the network traffic capture is to parse the packets—extracting features of interest such as the source, destination, ports, and protocols. This information is at the heart of MUD files. Parsing and importing the network traffic captures permits MUD-PD to extract local network devices and allows them to be tagged as devices of interest.
- **database viewing:** The second function is to present a user with a view of information of interest that has been imported into the database. The user can view a list of all the imported packet captures and the devices seen in any and all of the selected network traffic capture files. The user can then select a device or combination of devices to view some information about the packets coming from or to them. For deeper inspection, the user can open the file in Wireshark.

- **file generation:** The third and most useful function is to generate device reports and MUD files. The device reports summarize the captures in which the device is found, including metadata of the capture environment and a summary of what other devices were communicating on the local network. Currently, MUD files are generated through a behind-the-scenes interface to MUDgee but will eventually exclusively use the MySQL database. It is up to the user to determine whether the MUD files created are accurate enough to be put in service.

3.2.2 GUI Overview

Upon starting MUD-PD for the first time (installation instructions can be found at https://github.com/usnistgov/MUD-PD), the user is greeted with the MUD-PD main window (Figure 1). The labels contained in Figure 1 highlight the components of this window:

- (A) button to connect to an existing database
- (B) button to create and (re)initialize a database
- (C) button to import a capture file
- (D) button to generate a MUD file
- (E) button to generate a device report
- (F) box to contain a list of imported capture files
- (G) box to contain a list of active local network devices
- (H) box to contain a list of communications
- (I) button to inspect a previously imported capture file
- (J) toggles to limit view of communications to north/south (i.e., external) traffic or east/west (i.e., internal) traffic
- (K) toggles for a future feature described below
- (L) buttons to select how many packets to view in the communication box

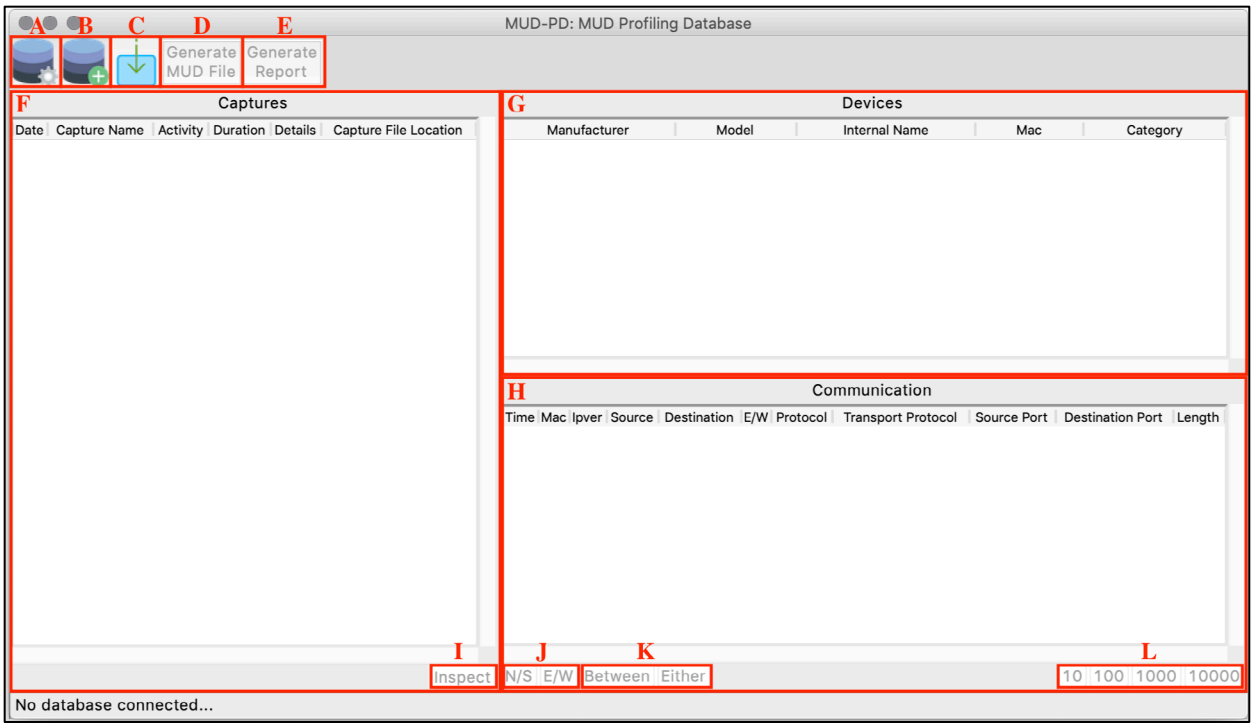


Figure 1: MUD-PD main window with buttons and list boxes labeled

The next step, after running MUD-PD for the first time, is to select the button labeled B to initiate the prompt to create a new database (Figure 2).

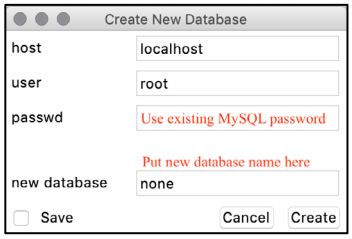


Figure 2: Prompt for creating a new database

Every time MUD-PD is run from this point forward, the user can select the button labeled A to connect to an existing database (see Figure 1 and Figure 3). When connected to an existing database, the button for creating a new database may also be used to reinitialize the database, wiping all existing data. The process is irreversible, so this should be done with caution.

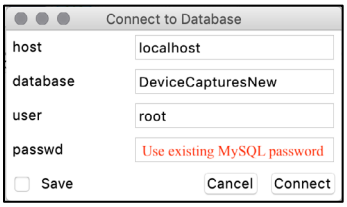


Figure 3: Prompt for connecting to an existing database

After connecting to a database, the user can examine any data contained within it. Referring to Figure 1, the user can view a list of pcap files that have been imported thus far in the Captures box (F) on the left side. On the upper right is the section called Devices (G), which contains a list of local devices communicating in the selected pcap files. The lower right section called Communication (H) contains a list of the packets sent by the selected devices in the pcap files. Above these boxes is a short toolbar with some options. From left to right, these are connect to a database (A), create a new database (B), import a pcap file (C), generate a MUD file (D), and generate a device report (E).

The Captures list (F) contains metadata for the imported pcap files, including the time of capture, the event captured, the duration of the capture (in seconds), the file location, and any additional details input during the import process. Below the list is an option to inspect (I) the currently selected packet capture. If more than one pcap is selected, only the pcap closest to the top will be opened. Inspecting a packet capture presents the same window that is opened when importing a capture file but allows the user to update/modify the details in the database. The details are identical to the import process, which will be covered in detail in Section 3.2.2.1. The user can select any number of pcap files, which will modify the list of devices to show any/all local devices that have sent or received packets during the captures.

The Devices list (G) includes information that either can be inferred from capture information or that has been input by the user during the import process. This includes the manufacturer, the model, a unique name for internal/lab use, the MAC address, and the general category of the device. The selection of an entry in the Devices list will determine what is listed in the Communication box. The user can either select All… to view all of the packets communicated across the network, or a single device to view only the communication to/from that device. Work is underway to allow the user to select multiple devices to view the communication that occurs between any combination of the devices selected or to/from any of the selected devices from/to other internal or external hosts or servers.

The Communication list (H) displays parsed packet information such as the time, MAC address of the sender, IP version, source and destination addresses, scope of traffic, innermost protocol layer, transport protocol, source and destination ports, and packet length. The IP version is given as either 4 or 6. If it is blank, the packet is below the IP layer (i.e., layer 3). By scope of traffic, we mean whether it would be considered east/west (i.e., internal/local network) traffic indicated by a value of 1, or north/south (i.e., to/from an external address/network) indicated by a value of 0. The source and destination ports are those of TCP or UDP. The user can choose to filter by north/south (N/S) or east/west (E/W) traffic and can select the number of packets displayed (J). There are two additional buttons (K) that hint at future capabilities planned for allowing the user to view traffic between two or more selected devices or to view the traffic to/from any of the selected devices. Last, the user may select to view the first 10, 100, 1,000, or 10,000 packets that satisfy the above filters (L).

3.2.2.1 Importing a New Packet Capture

The real potential of this tool begins to be realized when importing a packet capture. Here, the user is prompted to select the pcap file to import (Figure 4). Then metadata regarding the capture

can be input. This includes the phase of the device life cycle being captured. In most cases, this will be normal operation. The other two options are setup and decommissioning/removal, as described in Section 2.1.1. The user can also select all the environmental variables that apply, including whether internet connectivity was enabled, there was human interaction with the device, the device was isolated on the network, and/or the device's preferred DNS was blocked. Whether the capture was duration-based or action-based should also be selected. The specific duration (in seconds) or action can be input, and doing so is highly recommended for auditability and ease of use.

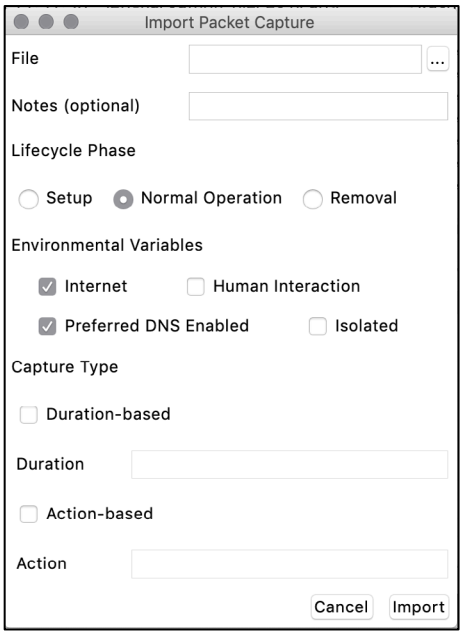
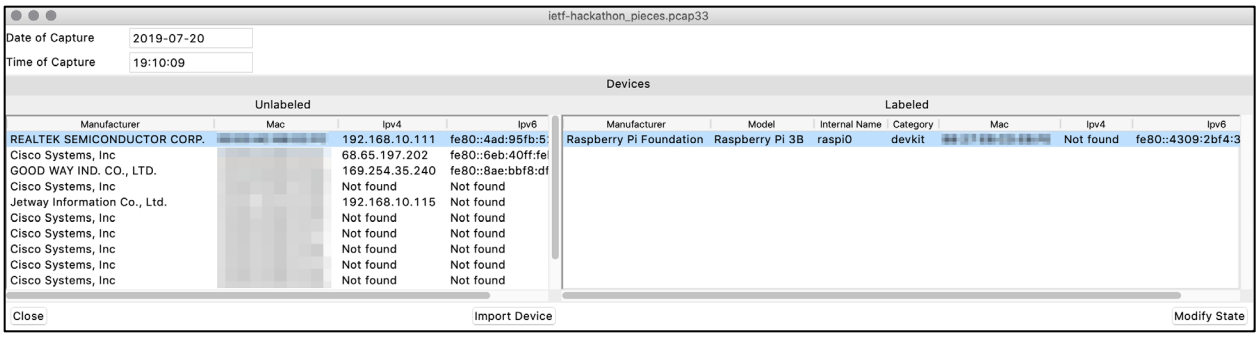


Figure 4: Prompt for importing packet captures into database

3.2.2.2 Viewing and Importing Devices

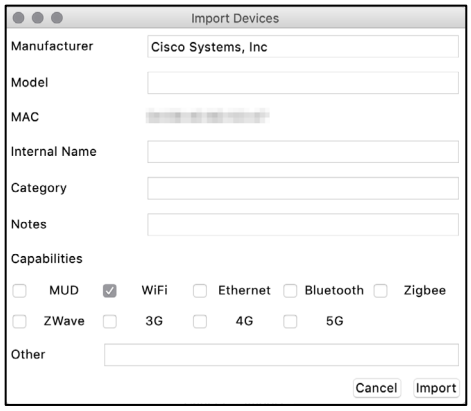
During the pcap import process, the user is presented with lists of the labeled and unlabeled devices that were seen in the capture file (Figure 5). A *labeled device* is one that has been seen in a previously imported capture and has been imported to the database. An *unlabeled device* may have been seen in a previous capture but has not yet been imported. This packet capture import window also includes the time and date of the capture, which is extracted from the capture file, but can be edited if the user believes either or both are incorrect for some reason. The left list is the unlabeled devices. MUD-PD attempts to look up the manufacturer based on the MAC address and also lists the IP addresses (both v4 and v6 when available). The user can select any device in this list and import it into the database, moving it to the list of labeled devices on the right. In addition to the information found in the unlabeled list, this one includes all the information available in the device list of the main window (Figure 1). The state of the device (i.e., the firmware version) can also be updated here. This field is not currently used in MUD-PD but can be queried through MySQL.



700

701 **Figure 5: Window listing devices imported and to import during the packet capture import process**

702 Selecting the Import Device button presents the user with a window with fields for adding or
703 modifying the manufacturer, model, internal name, category, notes, and list of capabilities
704 (Figure 6). The capabilities are MUD, Wi-Fi, Ethernet, Bluetooth, Zigbee, ZWave, 3G, 4G, 5G,
705 and other. Wi-Fi is automatically selected as default because the vast majority of consumer IoT
706 devices are Wi-Fi enabled. Currently, other capabilities must be selected manually; however, the
707 NCCoE is considering implementing a capability to extract or infer their presence from the
708 capture in future releases. The MAC address of the device is also listed but may not be modified,
709 as this is determined from the capture itself and is used as an identifier.



710

711 **Figure 6: Window prompt for importing a device**

712 After the metadata has been input and the Import button has been selected, the user is prompted
713 to input the firmware version of the device (Figure 7).

714

715 **Figure 7: Window prompting to update the firmware version logged in the database**

3.2.2.3 Generating Device Reports

The process for generating a device report is straightforward (Figure 8). The user may generate reports for any combination of devices or a single device. After selecting the devices for which to generate the report, the list of packet captures is updated to only those in which the device has sent or received packets. The user may select all or any combination of packets to report on.

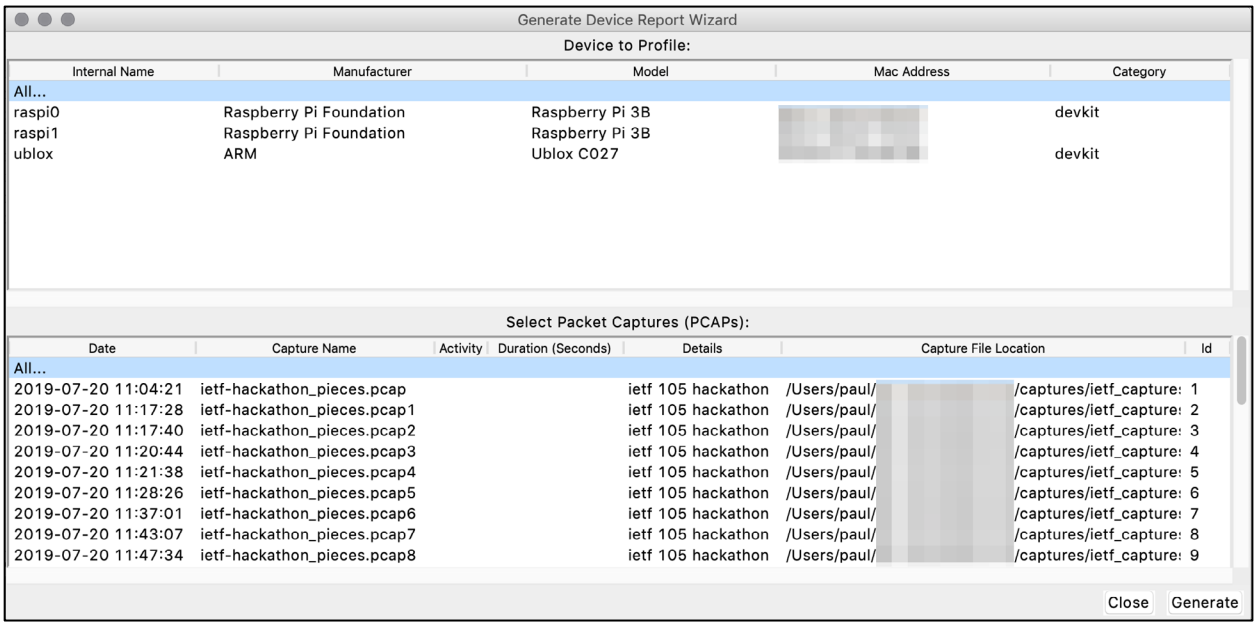


Figure 8: Prompt for generating a human-readable device report

The generated report lists the packet captures in which the device is seen, including the hash of the file. The example report, shown in Figure 9, contains only one file, whereas a typical report may contain many. The capture metadata is also listed for each file. In addition, listed under each capture file are the other local devices seen on the network during the capture. The internal name (if the device is labeled) is also given. Eventually, this report may include more specific information about the communication to/from the device, similar to what would be listed in the device's MUD file (if it had one). Current plans are to list the ports used, as well as the specific hosts and servers with which the device has communicated. In the future, the NCCoE may also provide a checklist of the types of captures performed to indicate to the user where gaps may exist in the MUD file that would be generated for the device.

```
This document serves to indicate the devices and operations captured in addition
to any specific procedures or environmental details of the captures.

Device: raspi0
MAC:    b8:27:eb:01:23:45

Capture File:  example_file.pcap
SHA256 Hash:   e83a34cbf4eab7bd8726bb9f4fce1db89b3928625c27a300d3c557ea7056466f
Device Phase:  Normal Operation
Environmental Variables:
    Internet enabled       True
    Human Interaction      False
    Preferred DNS Enabled  True
    Device Isolated        False
Action-based Capture:   False
Duration-based Capture: True
    Intended Duration:  600
    Actual Duration:    754
Start Time:     2020-02-02 12:34:56
End Time:       2020-02-02 12:47:30
Other Devices:
    Name:  router0
     MAC:  01:23:45:67:89:ab
    Name:  controller0
     MAC:  fe:dc:ba:98:76:54
    Name:  raspi1
     MAC:  d8:27:eb:67:89:ab
Notes:
        Example capture with made-up devices
```

Figure 9: Example device report showing the details of a single packet capture

3.2.2.4 Generating a MUD File

Generating the MUD file is currently a little more complex than generating the device report and will be streamlined in a future version. In the current version, the user is prompted to select a device for which to generate a MUD file (Figure 10). Then the gateway (typically the router or switch) must be selected. In most scenarios and a home environment, this could potentially be inferred from the IP addresses in the capture. For the present, for compatibility with the MUDgee MUD file generation tool, this must be selected manually. Finally, the user may select what packet capture files to use to generate the MUD file. The reason for this option is that there may be instances where a packet capture contains erroneous behavior that should not be included in the MUD file. Examples are if the capture contains an attack on the device (intentional/investigative or otherwise) and if the device sent or received packets that lead to compromise. Inclusion of such communication may enable the device to be successfully compromised in the future. It is, however, desirable to include as great a variety of captures as possible so that the MUD file is as complete as possible.

The NCCoE may eventually incorporate a feature where warnings are issued or a level of confidence displayed that indicates the level of accuracy that can be expected from the MUD file generated. This will require more work. However, it could be inferred to some degree from the variety of captures taken and selected, assuming that the device has not been compromised and is behaving as intended by the manufacturer. The format of a MUD file is outside the scope of this paper; the full specification and examples can be found in the MUD Request for Comments [1].

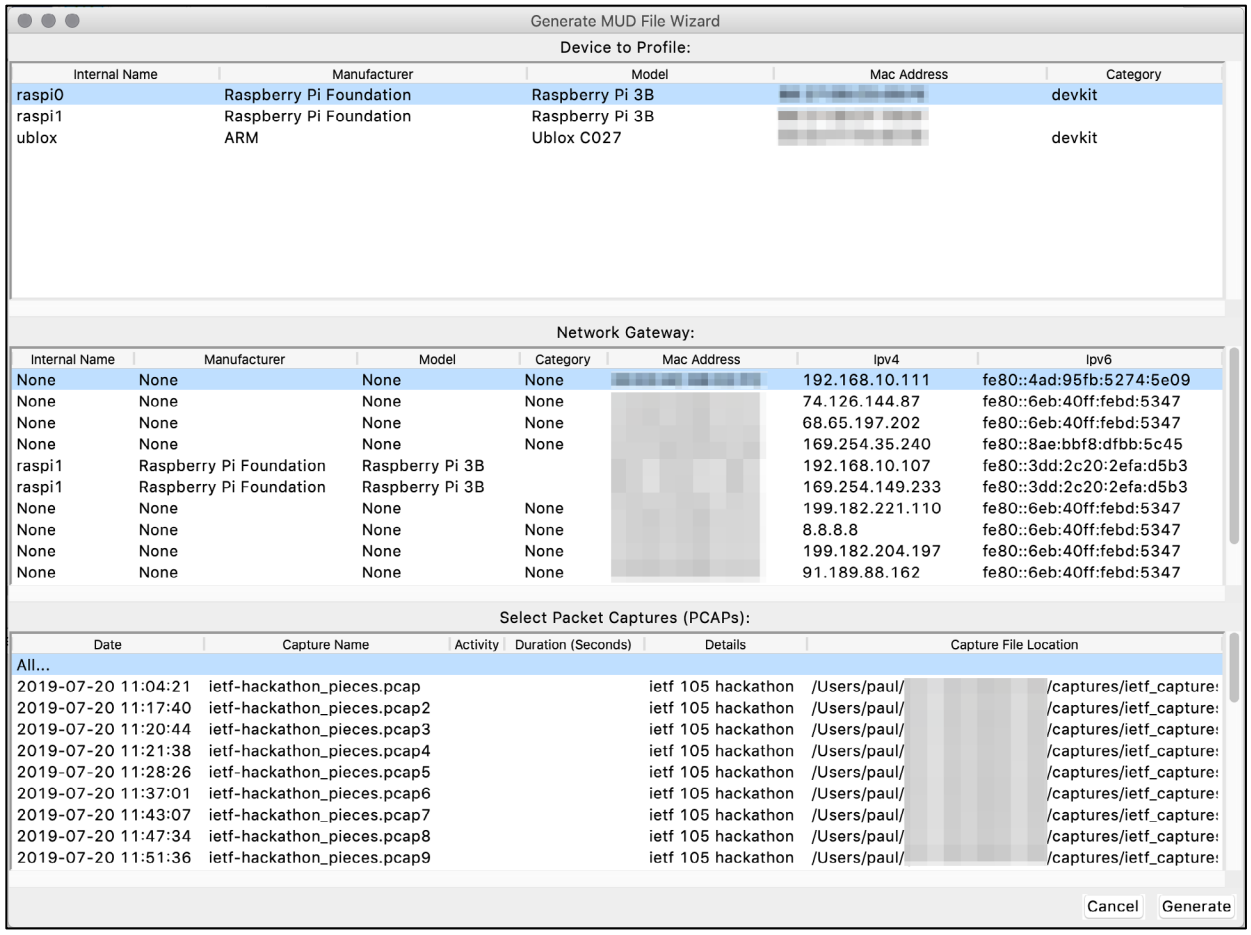


Figure 10: Prompt for generating a MUD file for a device

3.2.3 MUD-PD Uses

To an extent, MUD-PD is relatively purpose-agnostic. While its original intention was to assist in generating MUD files for IoT devices, the data it contains can be analyzed in other ways for other purposes. Because the data set will inevitably get large and it is labeled, machine learning techniques could be applied in an effective manner. The applications of machine learning and this data set are plentiful, including those not foreseen.

As the next section discusses, the same data collected for generating MUD files can be used to examine the privacy implications of these devices. Investigation into what the devices are communicating (the content of the communication) rather than simply how they are communicating can lead to a deeper understanding and greater awareness of the implications of putting smart devices in our homes.

3.3 MUD-PD Support for Privacy Analysis

As mentioned above, MUD-PD can be applied for more purposes than generating MUD files for IoT devices. While MUD files define the suggested behavior of a device, and one could argue that the content communicated is a component of a device's behavior, they do not necessarily

capture the privacy implications associated with the device or its associated networks. The NCCoE recommends this tool be used for privacy analysis only in a research and development setting. To understand the privacy implications in such a setting requires understanding the data content being transmitted from the device to outside services. Depending on device and protocols implemented, the content in the network packets may or may not be encrypted. Even where they are encrypted, the protocol under analysis may be susceptible to a man-in-the-middle attack that reveals some or all of the contents of the packets. Utilizing such an attack may be useful for an investigation into privacy, but again, should be implemented only in a research and development setting. There may be some moral, ethical, and privacy implications in implementing such an evaluation technique; these should be mitigated by limiting use of the tool to a controlled environment (i.e., a laboratory) and by adhering to the NIST Privacy Framework [10] and the Common Rule for the Protection of Human Subjects [11]. The same techniques for collection and logging can be beneficial to privacy investigations—tracking what potentially private information is transmitted and tracing the risks to all the devices and parties involved.

4 Next Steps

The NCCoE is considering a number of follow-on activities. The NCCoE needs to work to ensure that any methodology prescribed for characterizing devices is robust from security and reliability points of view. Going forward, the NCCoE will work to find and document additional situations and environmental variables that could modify the behavior of an IoT device, as well as work to capture additional interactions between and among devices. A final proposal is to explore usage of the PcapNg file format to document captures more effectively.

4.1 Extending MUD-PD Features

MUD-PD is still in development. Existing features will be streamlined, simplifying and speeding the collection, logging, and file generation processes. A number of additional features are planned, including extracting more information from packets (to include DNS resolutions). Deeper investigation into the packets captured, such as limiting the view of communication to only that between the two selected devices, will be enabled in the GUI and accessible in the communications tab of the main window (Figure 1). The generated device reports may also be extended to include combinations of devices if there is interest from developers or network administrators.

MUD files will be generated from the database itself rather than by a third party. In addition to the computer-readable MUD file, development is underway to provide the option of simultaneously producing a more human-readable report of its contents. This may aid in more rapid comprehension and development.

A number of enhancements to the usability and user experience of the MUD-file generation process itself are also being considered. This includes presenting the user with coarse estimates or warnings of the potential quality of the produced MUD file that can be expected based upon the network traffic captures selected, the goal being to highlight where gaps and deficiencies may exist in the resulting MUD file. One usability enhancement may be a wizard, or extended MUD file-generation process, that walks the user through each of the automatically generated rules to allow modifications as needed. This may be a useful and desirable feature for network administrators.

The NCCoE is also examining application of this tool and its data sets to investigate the privacy implications of IoT devices. To do so will require that packet payload information be extracted and stored. This includes strings, images, credentials seen, and certificates. It may also be worth logging whether packets are encrypted as well as the type and strength of the algorithm.

4.2 Developing a MUD Pipeline

The NCCoE is working on creating a set of pipelines focused on MUD file development, which address different use cases for MUD. Three use cases are being considered thus far: (1) a device manufacturer or developer that needs to provide a MUD file for its users, (2) a network administrator who may wish to inspect an official MUD file or a device's adherence to said file and who may wish to augment or modify its allowed behavior, and (3) a researcher who may be

interested in all of the above in addition to investigating the intricacies of existing MUD rules and proposed extensions.

In the first use case, a device manufacturer or developer might find it useful to have access to a suite of interoperable tools that make the generation, inspection, and validation of MUD files easy and straightforward (Figure 11). To begin the process, the two options are to build a MUD file by hand by using a tool like MUD Maker [7] or to generate one from a capture of network traffic by using MUD-PD/MUDgee. The next steps are to inspect the MUD file, which can be done visually using the MUD Visualizer [12], and validate that no rules are missing that should be present and no rules are present that should not be; and to edit where necessary. After a number of iterations through these steps, manufacturers may reach a point where they are confident in the MUD files and publish them for user consumption. The process depicted in Figure 11 can also be used to generate MUD files for legacy devices.

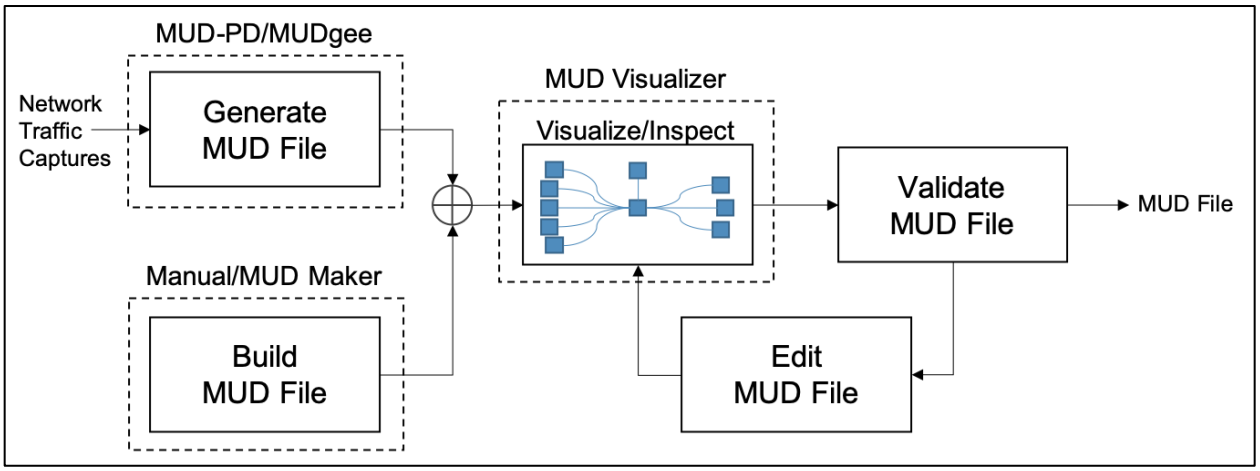


Figure 11: MUD pipeline for the device manufacturer or developer use case

In the second use case, it may be useful for network administrators to have a view of the network with an overlay of the MUD rules that have been defined by a manufacturer (Figure 12). To drive this capability, they must be able to ingest a MUD file and compare it against the behavior observed on the network. The MUD file may be manufacturer-defined or user-defined. When the MUD file and observed behavior are inspected and compared, the user could be presented with a diagram highlighting where the observed behavior does not comply with the MUD file. The UNSW researchers have developed a tool for comparing a provided MUD file with observed activity [13]. One also could imagine the MUD Visualizer tool being extended to include this capability. Because the network administrator may also be interested in reducing or expanding a device's capabilities, tailoring it to their specific network, the ability to build and/or edit MUD files would be desirable. MUD files can currently be built/written using MUD Maker, but there is not a dedicated tool for editing MUD files. To assist in live network administration and monitoring, it may be useful for the comparisons to be done on the fly on a live network, issuing live reports or warnings when noncompliance is detected.

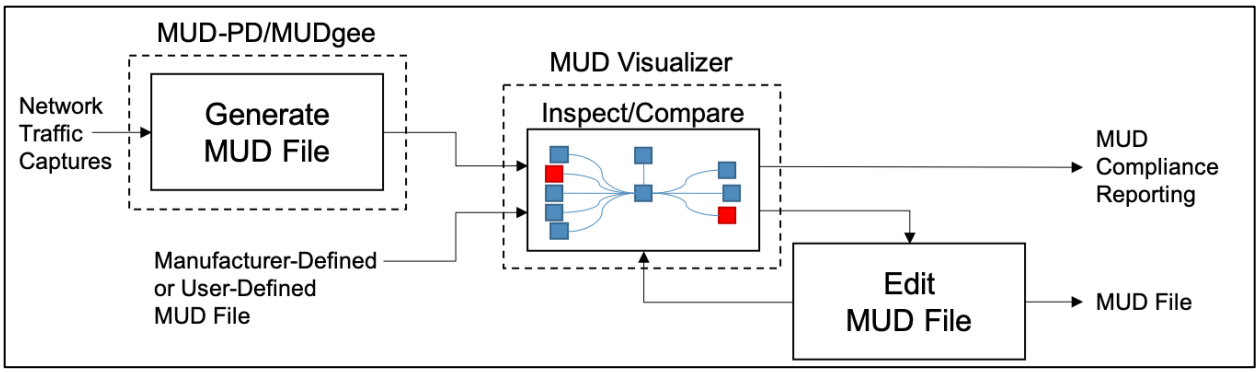


Figure 12: MUD pipeline for the network administrator use case

The third use case is more open-ended. Researchers may also want access to all the same tools useful to manufacturers and network administrators, and even more. There could be interest in studying existing MUD files or investigating the implications of various MUD rules or offering extensions (see Figure 13). For researchers, it may be useful to emulate a network of devices based on the MUD files to understand how networks scale and devices interact.

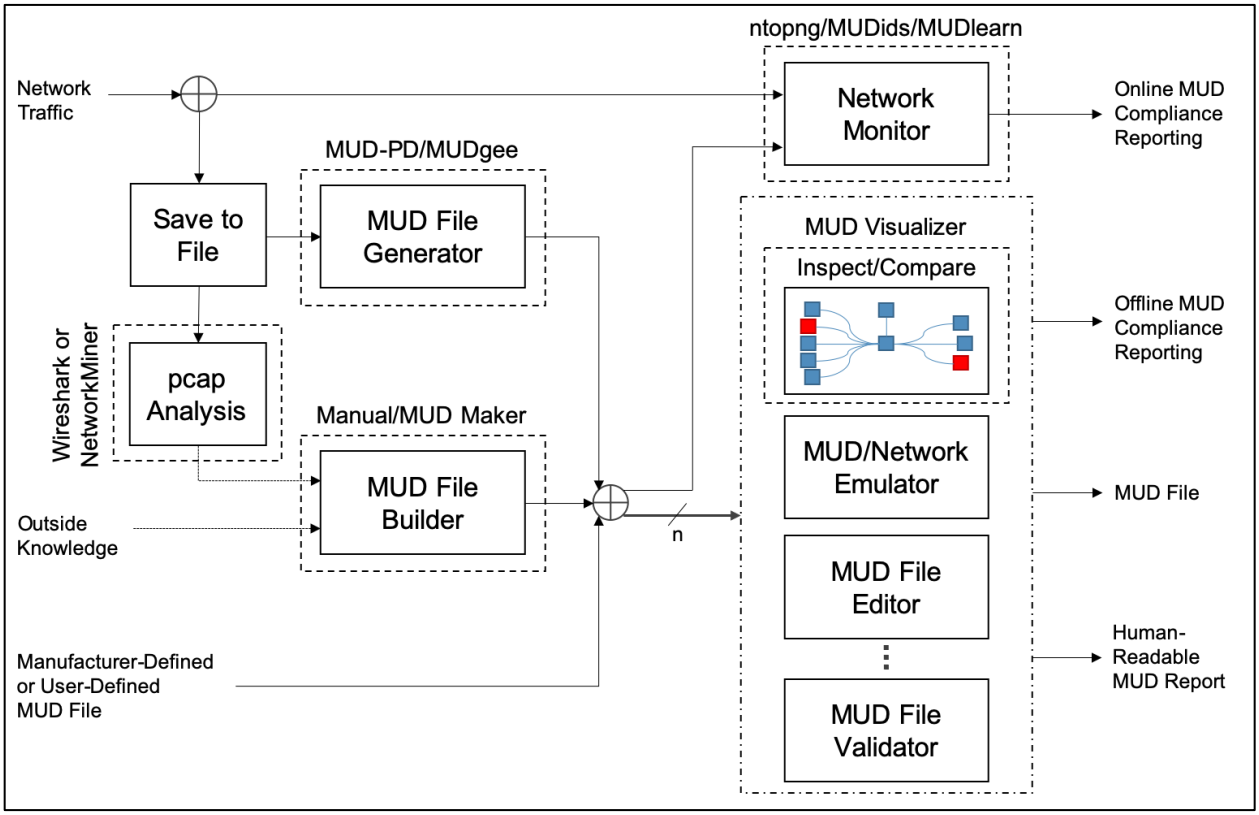


Figure 13: The overarching MUD pipeline, particularly as it may be used for research and development

Figure 13 demonstrates how a number of existing and proposed future tools relevant to MUD can be leveraged to achieve the research and development goals of the use cases described above. Several boxes in Figure 13 are labeled with existing tools that could potentially fill the

associated roles in their current state or with future development. The boxes that lack a dashed outline have not been associated with any existing tools that could potentially fill the role.

There are a number of ways in which a MUD file may be generated or selected. MUD files may come from the manufacturer or be generated by the user using network captures through MUD-PD/MUDgee or be written by hand with assistance from MUD Maker and Wireshark and/or NetworkMiner. These MUD files can then be used for several purposes or processed in a number of ways. Some may require using one version while others may require two or more, as indicated by the *n* in Figure 13.

A MUD plug-in is in development for the ntopng network monitoring tool [14]. When using a MUD file with live analysis of network activity, there is the potential for real-time MUD compliance reporting. Additionally, extensions to MUD's functionality are being proposed for use within the tool. Interest has been expressed in developing other MUD reporting tools. For example, the UNSW researchers have been using MUD in combination with software-defined networking to develop an intrusion detection system as well as a tool for detecting volumetric attacks, both of which have the potential for live reporting. These are called MUDids and MUDlearn, respectively [15], [16]. MUD files can also be visualized using the MUD Visualizer tool that is paired with MUD Maker. This tool could potentially be extended to compare two MUD files for offline compliance and manual validation. Additionally, tools are being proposed for automated validation of MUD files and network emulation based on these files. Development of application programming interfaces for these tools would greatly enhance interoperability and future development. The NCCoE hopes that the community of IoT manufacturers, developers, network administrators, and researchers will continue to contribute to improvements in this area.

4.3 Community Feedback

The NCCoE is seeking feedback on this document from all interested parties. In particular, input is needed on these challenges:

- Because it may be impossible to capture all potential aspects of an IoT device's behavior, how can the accuracy of a MUD file be measured?
    - How can the correctness of a MUD file be verified (and ensure that unnecessary behavior is not included)?
    - What combination of captures is needed to create a comprehensive MUD file (and ensure behavior that should be permissible is not omitted)?
- What are other applications of a MUD-PD tool or its data sets?
- What other tools should be considered for connecting in the MUD pipeline (or other pipelines)?
- What features are desirable for a tool like this?
- What other extractable features of packet captures might be of use to developers, network administrators, and researchers?
- How can the NCCoE improve the quality and efficiency of the tool?

- Is the NCCoE reinventing the wheel in some respects where existing open-source code might be better leveraged instead?

905 ## References

[1] Lear E, Droms R, Romascanu D (2019) Manufacturer Usage Description Specification. (Internet Engineering Task Force (IETF)), IETF Request for Comments (RFC) 8520. https://doi.org/10.17487/RFC8520

[2] Thangavelu V, Divakaran DM, Sairam R, Bhunia SS, Gurusamy M (2019) DEFT: A Distributed IoT Fingerprinting Technique. *IEEE Internet of Things Journal* 6(1):940-952. https://doi.org/10.1109/JIOT.2018.2865604

[3] Huang DY, Apthorpe N, Acar G, Li F, Feamster N (2019) IoT Inspector: Crowdsourcing Labeled Network Traffic from Smart Home Devices at Scale. *arXiv preprint*. https://arxiv.org/abs/1909.09848

[4] Meidan Y, Bohadana M, Shabtai A, Guarnizo JD, Ochoa M, Tippenhauer NO, Elovici Y (2017) ProfilIoT: A Machine Learning Approach for IoT Device Identification Based on Network Traffic Analysis. *Proceedings of the Symposium on Applied Computing (SAC '17)* (ACM, Marrakech, Morocco), pp 506-509. https://doi.org/10.1145/3019612.3019878

[5] Bezawada B, Bachani M, Peterson J, Shirazi H, Ray I, Ray I (2018) Behavioral Fingerprinting of IoT Devices. *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security (ASHES '18)* (ACM, Toronto, Canada), pp 41-50. https://doi.org/10.1145/3266444.3266452

[6] Aneja S, Aneja N, Islam MS (2018) IoT Device Fingerprint using Deep Learning. *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)* (IEEE, Bali, Indonesia), pp 174-179. https://doi.org/10.1109/IOTAIS.2018.8600824

[7] Lear E (2020) *MUD Maker Tool*. Available at https://mudmaker.org/mudmaker.html

[8] Schutijser CJTM (2018) Towards Automated DDoS Abuse Protection Using MUD Device Profiles. (University of Twente, Enschede, The Netherlands). Available at http://purl.utwente.nl/essays/76207

[9] Hamza A, Ranathunga D, Gharakheili HH, Roughan M, Sivaraman V (2018) Clear as MUD: Generating, Validating and Applying IoT Behavioral Profiles. *Proceedings of the 2018 Workshop on IoT Security and Privacy (IoT S&P '18)* (ACM, Budapest, Hungary), pp 8-14. https://doi.org/10.1145/3229565.3229566

[10] National Institute of Standards and Technology (2020) NIST Privacy Framework: A Tool for Improving Privacy through Enterprise Risk Management, Version 1.0. (National Institute of Standards and Technology, Gaithersburg, MD). Available at https://www.nist.gov/privacy-framework

[11] U.S. Department of Health and Human Services, Office for Human Research Protections (2020) *Federal Policy for the Protection of Human Subjects ('Common Rule')*. Available at https://www.hhs.gov/ohrp/regulations-and-policy/regulations/common-rule/index.html

[12] Andalibi V, Lear E (2020) *MUD Visualizer Tool*. Available at https://www.mudmaker.org/mudvisualizer.php

[13] Hamza A, Ranathunga D, Gharakheili HH, Benson TA, Roughan M, Sivaraman V (2019) Verifying and Monitoring IoTs Network Behavior using MUD Profiles. *arXiv preprint*. https://arxiv.org/abs/1902.02484

[14] ntop (2020) *ntopng: High-Speed Web-based Traffic Analysis and Flow Collection*. Available at https://www.ntop.org/products/traffic-analysis/ntop/

[15] Hamza A, Gharakheili HH, Sivaraman V (2018) Combining MUD Policies with SDN for IoT Intrusion Detection. *Proceedings of the 2018 Workshop on IoT Security and Privacy (IoT S&P '18)* (ACM, Budapest, Hungary), pp 1-7. https://doi.org/10.1145/3229565.3229571

[16] Hamza A, Gharakheili HH, Benson TA, Sivaraman V (2019) Detecting Volumetric Attacks on loT Devices via SDN-Based Monitoring of MUD Activity. *Proceedings of the 2019 ACM Symposium on SDN Research (SOSR '19)* (ACM, San Jose, California), pp 36-48. https://doi.org/10.1145/3314148.3314352

Appendix A—Example Capture Environment

This appendix presents an example capture environment that supports analysis of both wired and wireless Internet of Things (IoT) devices. Example procedures for capture are identified in Section 2.2. The following components compose the example environment:

- home router with tcpdump capability for capturing all network traffic, both wired and wireless (Linksys WRT1900ACS running OpenWRT)
- external storage to increase capture storage capacity of the home router (such as a flash drive)
- computer running Linux or macOS X (can be used for both capture and analysis as needed)
- IoT devices to characterize (camera, smart light, smart TV, smart switch)
- other devices that interact/communicate with the IoT devices (such as smart hubs/controllers/smartphones)

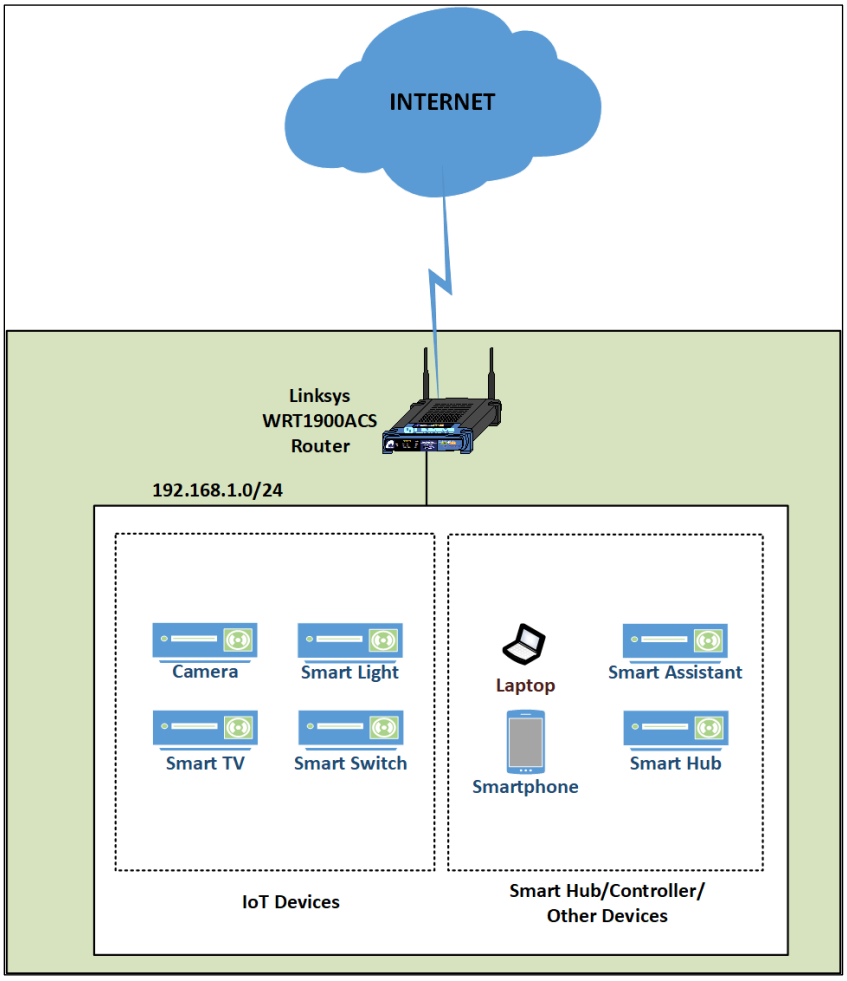


Figure 14: Example capture architecture

Appendix B—Acronyms

Selected acronyms and abbreviations used in this paper are defined below.

DNS	Domain Name System
GUI	Graphical User Interface
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
MAC	Media Access Control
MUD	Manufacturer Usage Description
NCCoE	National Cybersecurity Center of Excellence
NIST	National Institute of Standards and Technology
pcap	Packet Capture
PcapNg	Packet Capture Next Generation
SDN	Software-Defined Networking
SPAN	Switched Port Analyzer
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UNSW	University of New South Wales
WPA	Wi-Fi Protected Access