

2 Methodology for Characterizing Network 3 Behavior of Internet of Things Devices

4 Paul Watrobski

5 Joshua Klosterman

6 *The MITRE Corporation*

7 *McLean, VA*

8
9 William Barker

10 *Dakota Consulting*

11 *Gaithersburg, MD*

12
13 Murugiah Souppaya

14 *Computer Security Division*

15 *Information Technology Laboratory*

16 April 1, 2020

17
18 This publication is available free of charge from:

19 <https://doi.org/10.6028/NIST.CSWP.04012020-draft>

21

Abstract

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

28

Keywords

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

31

Disclaimer

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

36

Additional Information

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

41

Supplemental Content

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

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

45

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

46

47

48

49

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 .

50

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

51 **Acknowledgments**

52 The authors wish to thank the following individuals for their generous contributions of expertise
53 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

54 **Document Conventions**

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

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

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

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

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

75 **Audience**

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

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

84 **Trademark Information**

85 All registered trademarks or trademarks belong to their respective organizations.

86	Table of Contents	
87	1 Introduction	1
88	1.1 Purpose and Scope	1
89	1.2 Challenges	2
90	1.3 Background.....	2
91	2 Network Traffic Capture Methodology	4
92	2.1 Capture Strategy	4
93	2.2 Capture Procedure.....	8
94	2.3 Documentation Strategy	10
95	3 Analysis Use Cases and Tools	11
96	3.1 Manual MUD File Generation	11
97	3.2 MUD-PD	12
98	3.3 MUD-PD Support for Privacy Analysis	21
99	4 Next Steps	23
100	4.1 Extending MUD-PD Features	23
101	4.2 Developing a MUD Pipeline	23
102	4.3 Community Feedback	26
103	References	28

104	List of Appendices	
105	Appendix A— Example Capture Environment	30
106	Appendix B— Acronyms	31
107		

108 **1 Introduction**

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

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

118 **1.1 Purpose and Scope**

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

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

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

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

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

158 **1.2 Challenges**

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

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

181 **1.3 Background**

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

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

190 Once an IoT device's presence and type have been established, a network administrator can
191 begin to develop information regarding the device's characteristics and behavior. Approaches
192 like those of the Princeton IoT Inspector [3] and ProfillIoT's use of machine learning [4] are
193 being used to characterize and identify IoT devices, which can provide insight into security and
194 privacy issues associated with each device. However, not all fingerprinting and characterization
195 schemes are equivalent. These schemes are often created based on a limited set of data derived
196 from network traffic that allows them to accurately identify just the device type. The network
197 traffic information used to develop these schemes include packet headers, network ports, packet
198 timing, handshakes, and other information that might be unique to a particular IoT device [5],
199 [6]. Given the limited set of data used to develop the fingerprints, the fingerprints inherently do
200 not contain the information necessary to determine a device's full range of potential behaviors.

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

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

225 **2 Network Traffic Capture Methodology**

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

236 **2.1 Capture Strategy**

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

244 **2.1.1 IoT Device Life-Cycle Phases**

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

248 **2.1.1.1 Setup**

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

264 **2.1.1.2 Normal Operation**

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

278 **2.1.1.3 Decommissioning/Removal**

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

293 **2.1.2 Environmental Variables**

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

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

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

347 **2.1.3 Activity-Based and Time-Based Capture Approaches**

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

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

359 **2.1.4 Network Architecture and Capture Approach**

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

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

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

386 **2.1.5 Capture Tools**

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

389 **2.1.5.1 tcpdump**

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

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

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

406 **2.1.5.2 Wireshark**

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

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

418 **2.2 Capture Procedure**

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

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

423 **2.2.1 Device Setup Capture**

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

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

432 **2.2.2 Normal Operation Capture**

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

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

442 **2.2.3 Decommissioning/Removal Capture**

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

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

452 **2.3 Documentation Strategy**

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

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

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

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

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

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

485 **3 Analysis Use Cases and Tools**

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

488 **3.1 Manual MUD File Generation**

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

498 **3.1.1 Wireshark**

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

507 **3.1.2 NetworkMiner**

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

517 **3.1.3 Overview of Manual MUD File Generation Process**

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

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

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

540 **3.2 MUD-PD**

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

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

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

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

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

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

584 **3.2.1 Current Feature Set**

585 This subsection provides a high-level overview of MUD-PD as it currently stands. In Section
586 3.2.2, a tour of the tool illustrates its finer details. MUD-PD has three main functions:

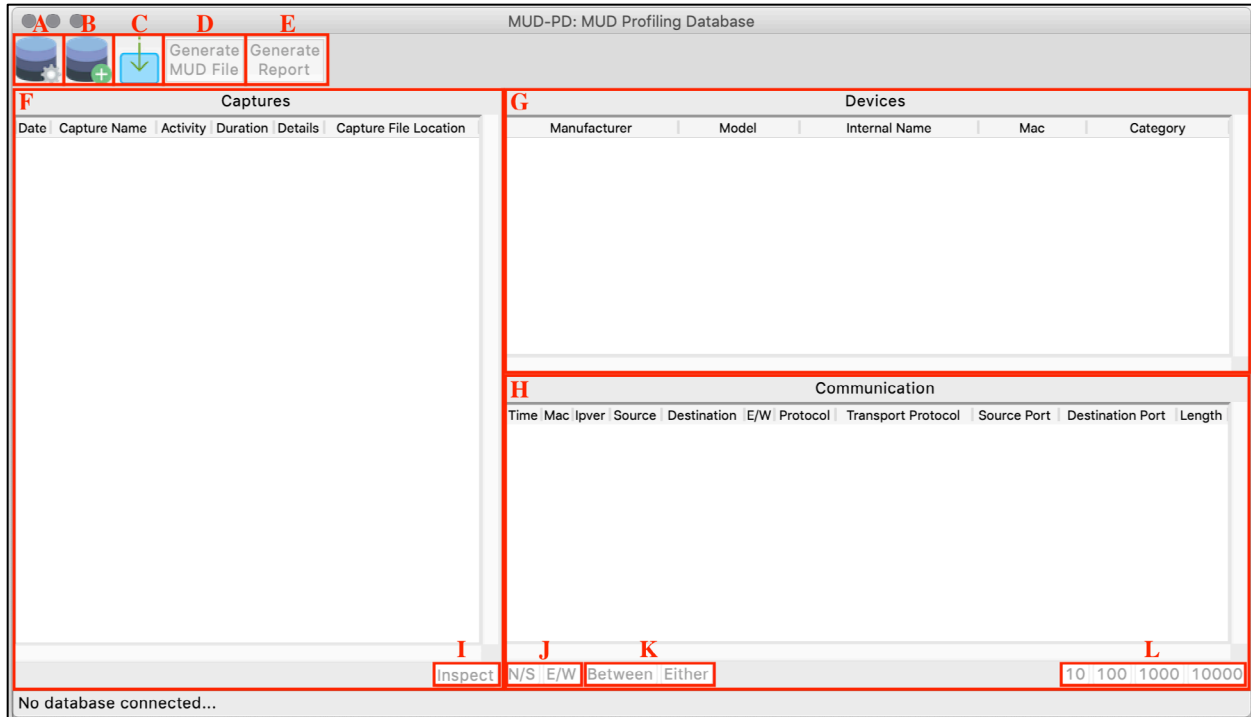
- 587 • **information import:** The first function is to import network traffic captures. During this
588 step, the user is provided the opportunity to input metadata about the capture. The goal of
589 importing the network traffic capture is to parse the packets—extracting features of
590 interest such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

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

606 3.2.2 GUI Overview

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

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

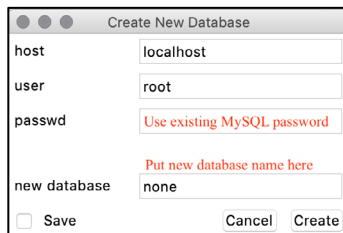


623

624

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

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

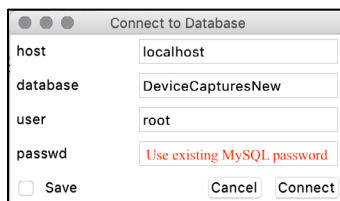


627

628

Figure 2: Prompt for creating a new database

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



633

634

Figure 3: Prompt for connecting to an existing database

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

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

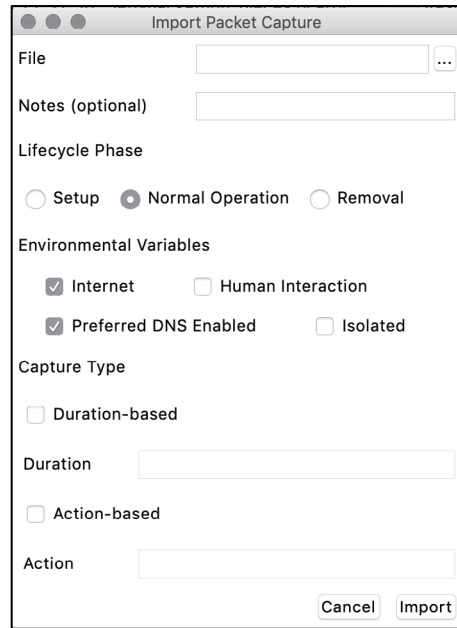
652 The Devices list (G) includes information that either can be inferred from capture information or
653 that has been input by the user during the import process. This includes the manufacturer, the
654 model, a unique name for internal/lab use, the MAC address, and the general category of the
655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

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

673 **3.2.2.1 Importing a New Packet Capture**

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

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



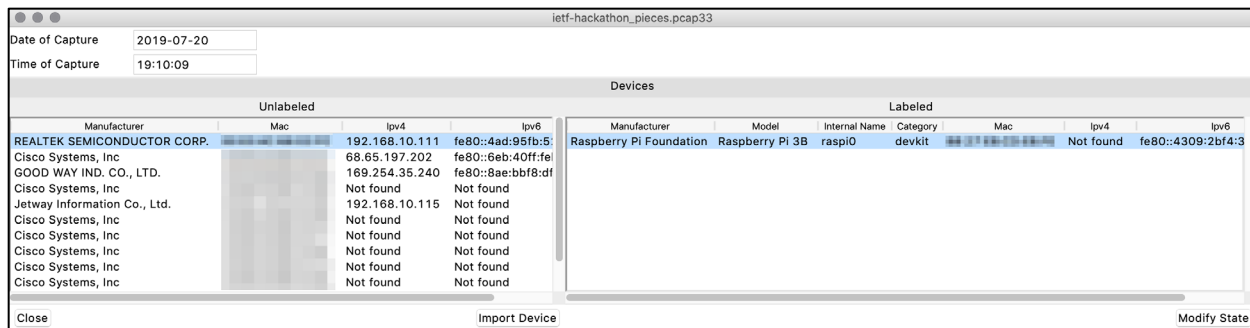
684

685

Figure 4: Prompt for importing packet captures into database

686 3.2.2.2 Viewing and Importing Devices

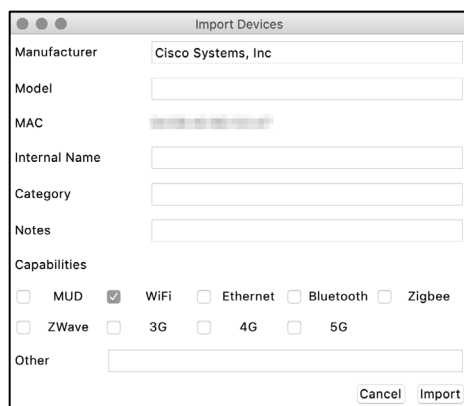
687 During the pcap import process, the user is presented with lists of the labeled and unlabeled
688 devices that were seen in the capture file (Figure 5). A *labeled device* is one that has been seen in
689 a previously imported capture and has been imported to the database. An *unlabeled device* may
690 have been seen in a previous capture but has not yet been imported. This packet capture import
691 window also includes the time and date of the capture, which is extracted from the capture file,
692 but can be edited if the user believes either or both are incorrect for some reason. The left list is
693 the unlabeled devices. MUD-PD attempts to look up the manufacturer based on the MAC
694 address and also lists the IP addresses (both v4 and v6 when available). The user can select any
695 device in this list and import it into the database, moving it to the list of labeled devices on the
696 right. In addition to the information found in the unlabeled list, this one includes all the
697 information available in the device list of the main window (Figure 1). The state of the device
698 (i.e., the firmware version) can also be updated here. This field is not currently used in MUD-PD
699 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).

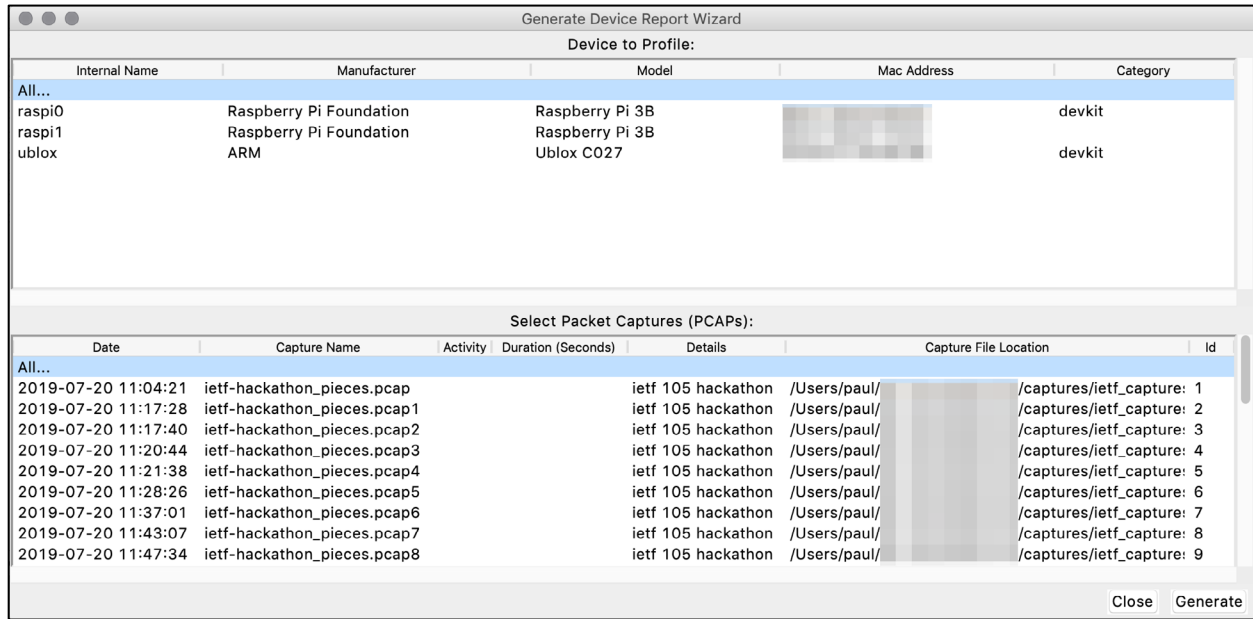


714

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

716 **3.2.2.3 Generating Device Reports**

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



721

722

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

723 The generated report lists the packet captures in which the device is seen, including the hash of
724 the file. The example report, shown in Figure 9, contains only one file, whereas a typical report
725 may contain many. The capture metadata is also listed for each file. In addition, listed under each
726 capture file are the other local devices seen on the network during the capture. The internal name
727 (if the device is labeled) is also given. Eventually, this report may include more specific
728 information about the communication to/from the device, similar to what would be listed in the
729 device's MUD file (if it had one). Current plans are to list the ports used, as well as the specific
730 hosts and servers with which the device has communicated. In the future, the NCCoE may also
731 provide a checklist of the types of captures performed to indicate to the user where gaps may
732 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
```

733

734

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

735

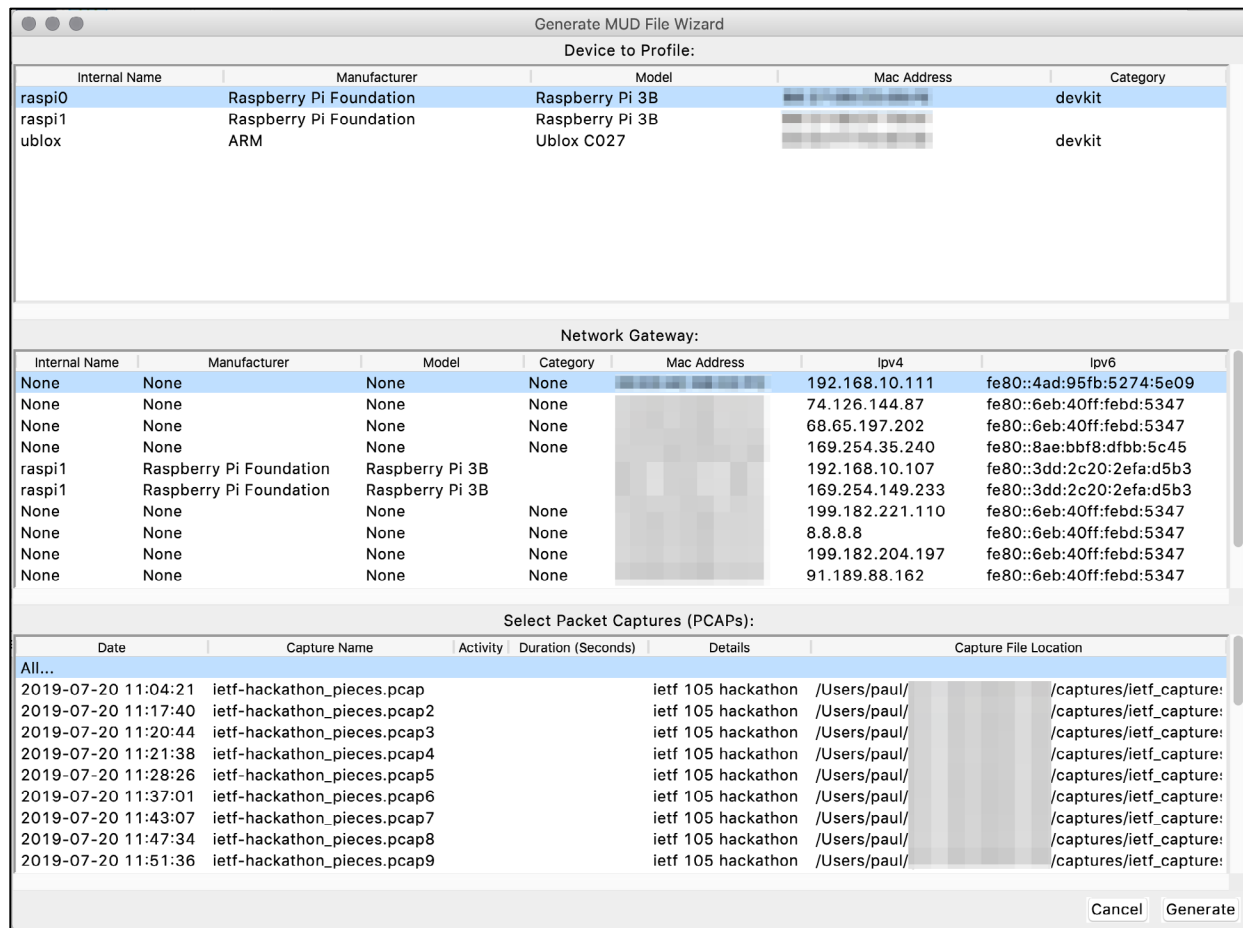
3.2.2.4 Generating a MUD File

736

737 Generating the MUD file is currently a little more complex than generating the device report and
738 will be streamlined in a future version. In the current version, the user is prompted to select a
739 device for which to generate a MUD file (Figure 10). Then the gateway (typically the router or
740 switch) must be selected. In most scenarios and a home environment, this could potentially be
741 inferred from the IP addresses in the capture. For the present, for compatibility with the MUDgee
742 MUD file generation tool, this must be selected manually. Finally, the user may select what
743 packet capture files to use to generate the MUD file. The reason for this option is that there may
744 be instances where a packet capture contains erroneous behavior that should not be included in
745 the MUD file. Examples are if the capture contains an attack on the device
746 (intentional/investigative or otherwise) and if the device sent or received packets that lead to
747 compromise. Inclusion of such communication may enable the device to be successfully
748 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.

749

750 The NCCoE may eventually incorporate a feature where warnings are issued or a level of
751 confidence displayed that indicates the level of accuracy that can be expected from the MUD file
752 generated. This will require more work. However, it could be inferred to some degree from the
753 variety of captures taken and selected, assuming that the device has not been compromised and is
754 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].



755

756

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

757 **3.2.3 MUD-PD Uses**

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

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

768 **3.3 MUD-PD Support for Privacy Analysis**

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

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

786 **4 Next Steps**

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

793 **4.1 Extending MUD-PD Features**

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

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

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

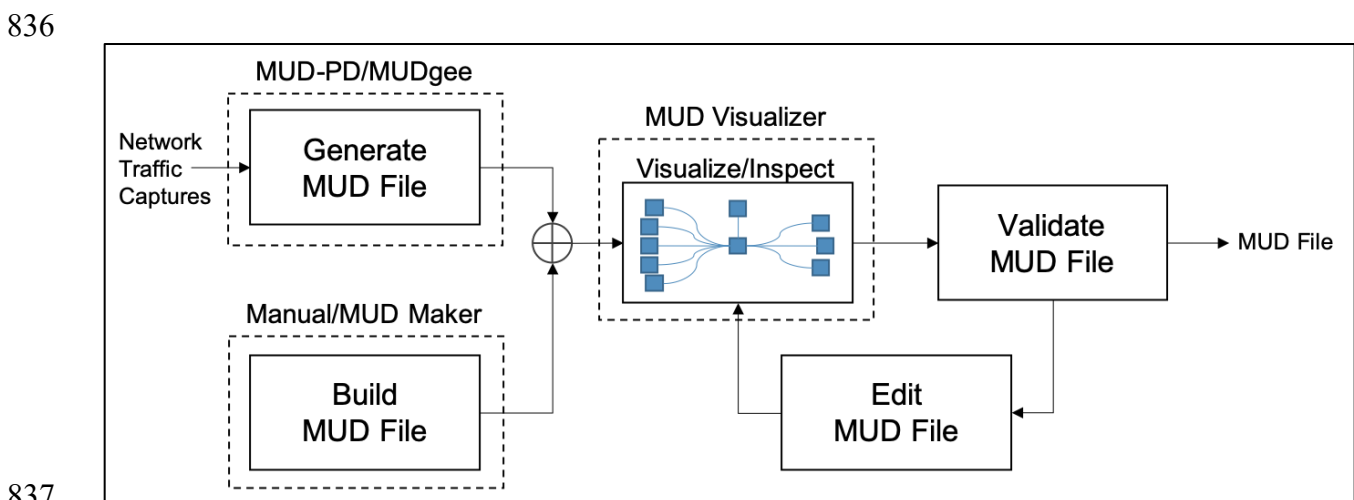
814 The NCCoE is also examining application of this tool and its data sets to investigate the privacy
815 implications of IoT devices. To do so will require that packet payload information be extracted
816 and stored. This includes strings, images, credentials seen, and certificates. It may also be worth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

818 **4.2 Developing a MUD Pipeline**

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

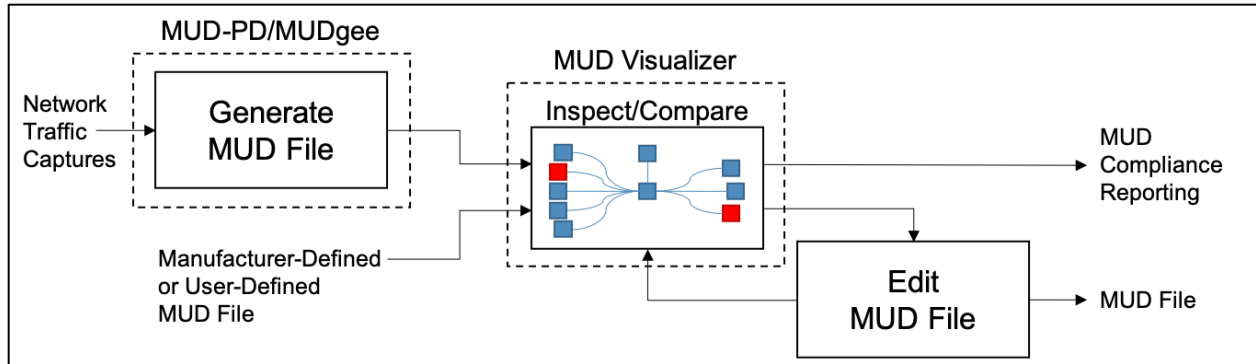
824 interested in all of the above in addition to investigating the intricacies of existing MUD rules
 825 and proposed extensions.

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



837
 838 **Figure 11: MUD pipeline for the device manufacturer or developer use case**

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



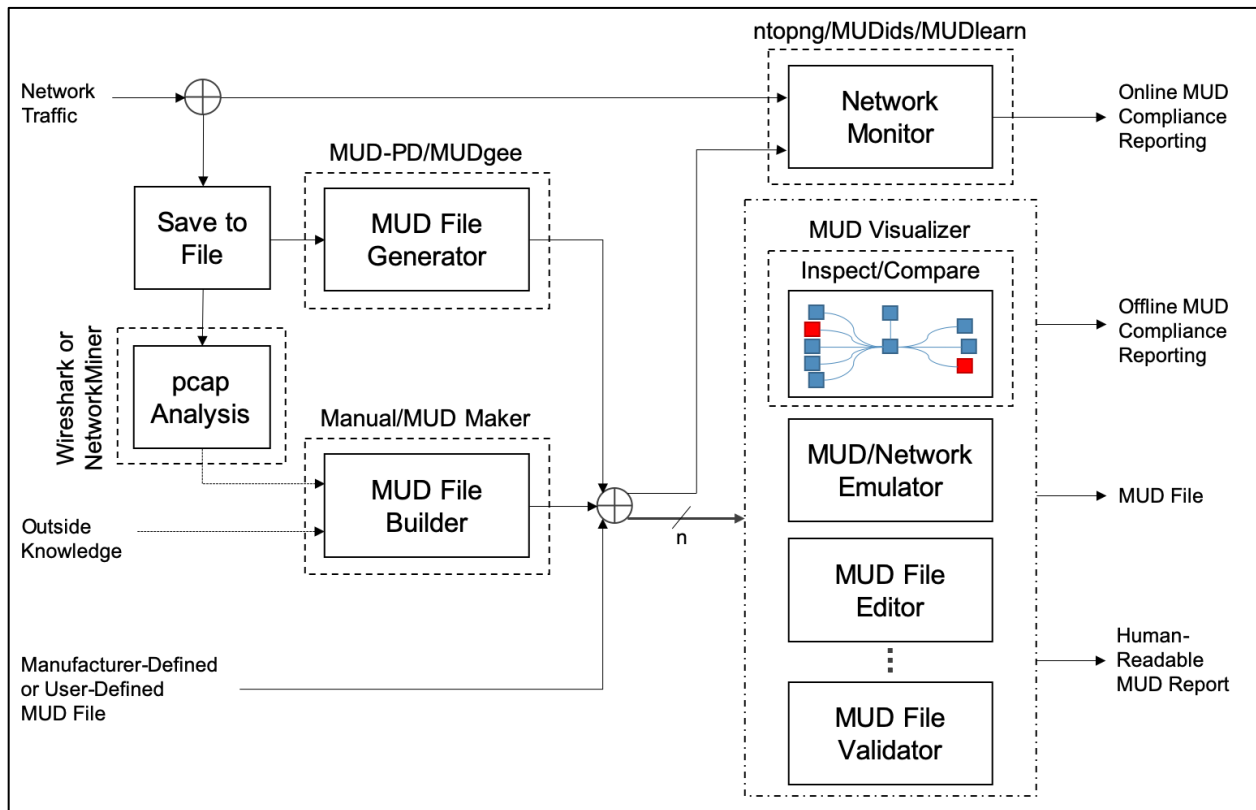
853

854

Figure 12: MUD pipeline for the network administrator use case

855

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



860

861

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

862

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

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

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

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

887 **4.3 Community Feedback**

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

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

- 903
- 904
- Is the NCCoE reinventing the wheel in some respects where existing open-source code might be better leveraged instead?

905 **References**

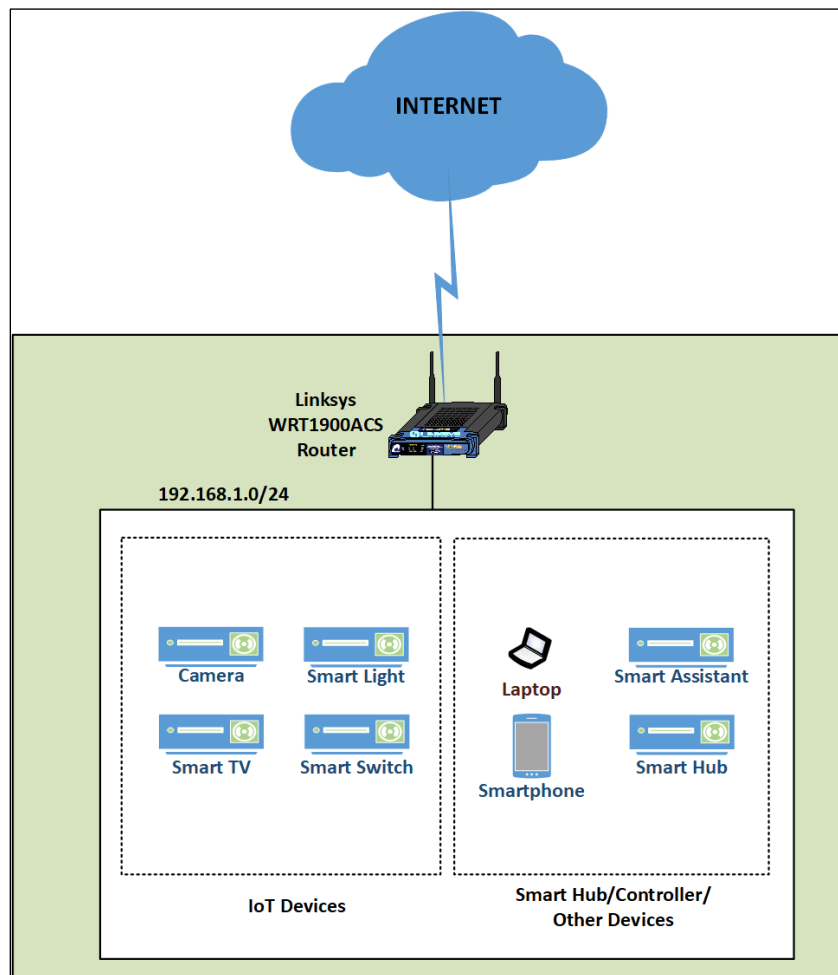
- [1] Lear E, Droms R, Romascanu D (2019) Manufacturer Usage Description Specification. (Internet Engineering Task Force (IETF)), IETF Request for Comments (RFC) 8520. <https://doi.org/10.17487/RFC8520>
- [2] Thangavelu V, Divakaran DM, Sairam R, Bhunia SS, Gurusamy M (2019) DEFT: A Distributed IoT Fingerprinting 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 IoT 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>

906 **Appendix A—Example Capture Environment**

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

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



919

920

Figure 14: Example capture architecture

921 **Appendix B—Acronyms**

922 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