NIST SPECIAL PUBLICATION 1800-15B

Securing Small-Business and Home Internet of Things (IoT) Devices

Mitigating Network-Based Attacks Using Manufacturer Usage Description (MUD)

Volume B:
Approach, Architecture, and Security Characteristics

Douglas Montgomery
Tim Polk
Mudumbai Ranganathan
Murugiah Souppaya
NIST

William C. Barker
Dakota Consulting

Drew Cohen
Kevin Yeich
MasterPeace Solutions, Ltd.

Steve Johnson
Ashwini Kadam
Craig Pratt
Darshak Thakore
Mark Walker
CableLabs

Dean Coclin
Avesta Hojjati
Clint Wilson
DigiCert

Tim Jones
Forescout

Yemi Fashina
Parisa Grayeli
Joshua Harrington
Joshua Klosterman
Blaine Mulugeta
Susan Symington
The MITRE Corporation

Eliot Lear
Brian Weis
Cisco

Adnan Baykal
Global Cyber Alliance

Jaideep Singh
Molex

September 2020

DRAFT

This publication is available free of charge from:
https://www.nccoe.nist.gov/projects/building-blocks/mitigating-iot-based-ddos

DISCLAIMER

Certain commercial entities, equipment, products, or materials may be identified by name or company logo or other insignia in order to acknowledge their participation in this collaboration or to describe an experimental procedure or concept adequately. Such identification is not intended to imply special status or relationship with NIST or recommendation or endorsement by NIST or NCCoE; neither is it intended to imply that the entities, equipment, products, or materials are necessarily the best available for the purpose.

National Institute of Standards and Technology Special Publication 1800-15B, Natl. Inst. Stand. Technol. Spec. Publ. 1800-15B, 219 pages, (September 2020), CODEN: NSPUE2

FEEDBACK

You can improve this guide by contributing feedback. As you review and adopt this solution for your own organization, we ask you and your colleagues to share your experience and advice with us.

Comments on this publication may be submitted to: mitigating-iot-ddos-nccoe@nist.gov.

Public comment period: September 16, 2020 through October 16, 2020

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

National Cybersecurity Center of Excellence
National Institute of Standards and Technology
100 Bureau Drive
Mailstop 2002
Gaithersburg, MD 20899
Email: nccoe@nist.gov

NATIONAL CYBERSECURITY CENTER OF EXCELLENCE

The National Cybersecurity Center of Excellence (NCCoE), a part of the National Institute of Standards and Technology (NIST), is a collaborative hub where industry organizations, government agencies, and academic institutions work together to address businesses' most pressing cybersecurity issues. This public-private partnership enables the creation of practical cybersecurity solutions for specific industries, as well as for broad, cross-sector technology challenges. Through consortia under Cooperative Research and Development Agreements (CRADAs), including technology partners—from Fortune 50 market leaders to smaller companies specializing in information technology security—the NCCoE applies standards and best practices to develop modular, easily adaptable example cybersecurity solutions using commercially available technology. The NCCoE documents these example solutions in the NIST Special Publication 1800 series, which maps capabilities to the NIST Cybersecurity Framework and details the steps needed for another entity to re-create the example solution. The NCCoE was established in 2012 by NIST in partnership with the State of Maryland and Montgomery County, Maryland.

To learn more about the NCCoE, visit https://www.nccoe.nist.gov/. To learn more about NIST, visit https://www.nist.gov.

NIST CYBERSECURITY PRACTICE GUIDES

NIST Cybersecurity Practice Guides (Special Publication 1800 series) target specific cybersecurity challenges in the public and private sectors. They are practical, user-friendly guides that facilitate the adoption of standards-based approaches to cybersecurity. They show members of the information security community how to implement example solutions that help them align more easily with relevant standards and best practices, and provide users with the materials lists, configuration files, and other information they need to implement a similar approach.

The documents in this series describe example implementations of cybersecurity practices that businesses and other organizations may voluntarily adopt. These documents do not describe regulations or mandatory practices, nor do they carry statutory authority.

ABSTRACT

The goal of the Internet Engineering Task Force's Manufacturer Usage Description (MUD) specification is for Internet of Things (IoT) devices to behave as intended by the manufacturers of the devices. MUD provides a standard way for manufacturers to indicate the network communications that a device requires to perform its intended function. When MUD is used, the network will automatically permit the IoT device to send and receive only the traffic it requires to perform as intended, and the network will prohibit all other communication with the device, thereby increasing the device's resilience to network-based attacks. In this project, the NCCoE demonstrated the ability to ensure that when an IoT device connects to a home or small-business network, MUD can automatically permit the device to send and

receive only the traffic it requires to perform its intended function. This NIST Cybersecurity Practice Guide explains how MUD protocols and tools can reduce the vulnerability of IoT devices to botnets and other network-based threats as well as reduce the potential for harm from exploited IoT devices. It also shows IoT device developers and manufacturers, network equipment developers and manufacturers, and service providers who employ MUD-capable components how to integrate and use MUD to satisfy IoT users' security requirements.

KEYWORDS

access control; bootstrapping; botnets; firewall rules; flow rules; Internet of Things; IoT; Manufacturer Usage Description; MUD; network segmentation; onboarding; router; server; software update server; threat signaling; Wi-Fi Easy Connect.

DOCUMENT CONVENTIONS

The terms "shall" and "shall not" indicate requirements to be followed strictly to conform to the publication and from which no deviation is permitted.

The terms "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable without mentioning or excluding others or that a certain course of action is preferred but not necessarily required or that (in the negative form) a certain possibility or course of action is discouraged but not prohibited.

The terms "may" and "need not" indicate a course of action permissible within the limits of the publication.

The terms "can" and "cannot" indicate a possibility and capability, whether material, physical, or causal.

Acronyms used in figures can be found in the Acronyms appendix.

CALL FOR PATENT CLAIMS

This public review includes a call for information on essential patent claims (claims whose use would be required for compliance with the guidance or requirements in this Information Technology Laboratory [ITL] draft publication). Such guidance and/or requirements may be directly stated in this ITL publication or by reference to another publication. This call also includes disclosure, where known, of the existence of pending U.S. or foreign patent applications relating to this ITL draft publication and of any relevant unexpired U.S. or foreign patents.

ITL may require from the patent holder, or a party authorized to make assurances on its behalf, in written or electronic form, either:

1. assurance in the form of a general disclaimer to the effect that such party does not hold and does not currently intend holding any essential patent claim(s); or

2. assurance that a license to such essential patent claim(s) will be made available to applicants desiring to utilize the license for the purpose of complying with the guidance or requirements in this ITL draft publication either:

    a. under reasonable terms and conditions that are demonstrably free of any unfair discrimination or

    b. without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination

Such assurance shall indicate that the patent holder (or third party authorized to make assurances on its behalf) will include in any documents transferring ownership of patents subject to the assurance, provisions sufficient to ensure that the commitments in the assurance are binding on the transferee, and that the transferee will similarly include appropriate provisions in the event of future transfers with the goal of binding each successor-in-interest.

The assurance shall also indicate that it is intended to be binding on successors-in-interest regardless of whether such provisions are included in the relevant transfer documents.

Such statements should be addressed to mitigating-iot-ddos-nccoe@nist.gov.

ACKNOWLEDGMENTS

We are grateful to the following individuals for their generous contributions of expertise and time.

Name	Organization
Allaukik Abhishek	Arm
Michael Bartling	Arm
Tao Wan	CableLabs
Russ Gyurek	Cisco
Peter Romness	Cisco
Rob Cantu	CTIA
Katherine Gronberg	Forescout

Name	Organization
Rae'-Mar Horne	MasterPeace Solutions, Ltd.
Nate Lesser	MasterPeace Solutions, Ltd.
Tom Martz	MasterPeace Solutions, Ltd.
Daniel Weller	MasterPeace Solutions, Ltd.
Nancy Correll	The MITRE Corporation
Sallie Edwards	The MITRE Corporation
Drew Keller	The MITRE Corporation
Sarah Kinling	The MITRE Corporation
Karri Meldorf	The MITRE Corporation
Mary Raguso	The MITRE Corporation
Allen Tan	The MITRE Corporation
Mo Alhroub	Molex
Bill Haag	National Institute of Standards and Technology
Paul Watrobski	National Institute of Standards and Technology
Bryan Dubois	Patton Electronics
Stephen Ochs	Patton Electronics

Name	Organization
Karen Scarfone	Scarfone Cybersecurity
Matt Boucher	Symantec
Petros Efstathopoulos	Symantec
Bruce McCorkendale	Symantec
Susanta Nanda	Symantec
Yun Shen	Symantec
Pierre-Antoine Vervier	Symantec
John Bambenek	ThreatSTOP
Russ Housley	Vigil Security

The Technology Partners/Collaborators who participated in this project submitted their capabilities in response to a notice in the Federal Register. Respondents with relevant capabilities or product components were invited to sign a Cooperative Research and Development Agreement (CRADA) with NIST, allowing them to participate in a consortium to build these example solutions. We worked with:

Technology Partner/Collaborator	Build Involvement
Arm	Subject matter expertise
CableLabs	Micronets Gateway Micronets cloud infrastructure Prototype IoT devices–Raspberry Pi with Wi-Fi Easy Connect support Micronets mobile application
Cisco	Cisco Catalyst 3850-S MUD manager

Technology Partner/Collaborator	Build Involvement
CTIA	Subject matter expertise
DigiCert	Private Transport Layer Security certificate Premium Certificate
Forescout	Forescout appliance–VCT-R Enterprise manager–VCEM-05
Global Cyber Alliance	Quad9 threat agent and Quad 9 MUD manager (integrated in Yikes! router) Quad9 domain name system Quad9 threat application programming interface ThreatSTOP threat MUD file server
MasterPeace Solutions, Ltd.	Yikes! router Yikes! cloud Yikes! mobile application
Molex	Molex light-emitting diode light bar Molex Power over Ethernet Gateway
Patton Electronics	Subject matter expertise
Symantec	Subject matter expertise

Contents

List of Figures

List of Tables

1 Summary

The Manufacturer Usage Description Specification (Internet Engineering Task Force [IETF] Request for Comments [RFC] 8520) provides a means for increasing the likelihood that Internet of Things (IoT) devices will behave as intended by the manufacturers of the devices. This is done by providing a standard way for manufacturers to indicate the network communications that the device requires to perform its intended function. When the Manufacturer Usage Description (MUD) is used, the network will automatically permit the IoT device to send and receive only the traffic it requires to perform as intended, and the network will prohibit all other communication with the device, thereby increasing the device's resilience to network-based attacks. This project focuses on the use of IoT devices in home and small-business environments. Its objective is to show how MUD can practically and effectively reduce the vulnerability of IoT devices to network-based threats, and how MUD can limit the usefulness of any compromised IoT devices to malicious actors.

This volume describes a reference architecture that is designed to achieve the project's objective, the laboratory architecture employed for the demonstrations, and the security characteristics supported by the reference design. Four implementations of the reference design are demonstrated. These implementations are referred to as *builds*, and this volume describes all of them in detail:

- Build 1 uses products from Cisco Systems, DigiCert, Forescout, and Molex.
- Build 2 uses products from MasterPeace Solutions, Ltd.; Global Cyber Alliance (GCA); ThreatSTOP; and DigiCert.
- Build 3 uses products from CableLabs and DigiCert.
- Build 4 uses software developed at the National Institute of Standards and Technology (NIST) Advanced Networking Technologies laboratory and products from DigiCert.

The primary technical elements of this project include components that are designed and configured to support the MUD protocol. We describe these components as being *MUD-capable*. The components used include MUD-capable network gateways, routers, and switches that support wired and wireless network access; MUD managers; MUD file servers; MUD-capable Dynamic Host Configuration Protocol (DHCP) servers; update servers; threat-signaling servers; MUD-capable IoT devices; and MUD files and their corresponding signature files. We also used devices that are not capable of supporting the MUD protocol, which we call *non-MUD-capable* or *legacy* devices, to demonstrate the security benefits of the demonstrated approach that are independent of the MUD protocol, such as threat signaling and device onboarding. Non-MUD-capable devices used include laptops, phones, and IoT devices that cannot emit or otherwise convey a uniform resource locator (URL) for a MUD file as described in the MUD specification.

The demonstrated approach, which deploys MUD as an additional security tool rather than as a replacement for other security mechanisms, shows that MUD can make it more difficult to compromise

IoT devices on a home or small-business network by using a network-based attack. While MUD can be used to protect networks of any size, the scenarios examined by this National Cybersecurity Center of Excellence (NCCoE) project involve IoT devices being used in home and small-business networks. Owners of such networks cannot be assumed to have extensive network administration experience. This makes plug-and-play deployment a requirement. Although the focus of this project is on home and small-business network applications, the home and small-business network users are not the guide's intended audience. This guide is intended primarily for IoT device developers and manufacturers, network equipment developers and manufacturers, and service providers whose services may employ MUD-capable components. MUD-capable IoT devices and network equipment are not yet widely available, so home and small-business network owners are dependent on these groups to make it possible for them to obtain and benefit from MUD-capable equipment and associated services.

1.1 Challenge

The term *IoT* is often applied to the aggregate of single-purpose, internet-connected devices, such as thermostats, security monitors, lighting control systems, and connected television sets. The IoT is experiencing what some might describe as hypergrowth. Gartner forecasts that the number of IoT devices will reach 25 billion by 2021, while Forbes forecast that the market will exceed $457 billion before 2021. While such rapid growth has the potential to provide many benefits, it is also a cause for concern because IoT devices are tempting targets for attackers. State-of-the-art security software protects full-featured devices, such as laptops and phones, from most known threats, but many IoT devices, such as connected thermostats, security cameras, and lighting control systems, have minimal security or are unprotected. Because they are designed to be inexpensive and limited purpose, IoT devices may have unpatched software flaws. They also often have processing, timing, memory, and power constraints that make them challenging to secure. Users often do not know what IoT devices are on their networks and lack means for controlling access to them over their life cycles. However, the consequences of not addressing the security of IoT devices can be catastrophic. For instance, in typical networking environments, malicious actors can detect and attack an IoT device within minutes of it connecting to the internet. If it has a known vulnerability, this weakness can be exploited at scale, enabling an attacker to commandeer sets of compromised devices, called *botnets*, to launch large-scale distributed denial of service (DDoS) attacks, as well as other network-based attacks. A DDoS attack involves multiple computing devices in disparate locations sending repeated requests to a server with the intent to overload it and ultimately render it inaccessible. On October 12, 2016, a botnet consisting of more than 100,000 devices, called Mirai, launched a large DDoS attack on the internet infrastructure firm Dyn. Mirai interfered with Dyn's ability to provide domain name system (DNS) services to many large websites, effectively taking those websites offline for much of a day.

A DDoS or other network-based attack may result in substantial revenue losses and potential liability exposure, which can degrade a company's reputation and erode customer trust. Victims of a DDoS attack can include

- businesses that rely on the internet, who may suffer if their customers cannot reach them
- IoT device manufacturers, who may suffer reputational damage if their devices are exploited
- service providers, who may suffer service degradation that affects their customers
- users of IoT devices, who may suffer service degradation and potentially incur extra costs due to increased activity by their compromised machines

Because IoT devices are designed to be low cost and for limited purposes, it is not realistic to try to solve the problem of IoT device vulnerability by requiring that all IoT devices be equipped with robust state-of-the-art security mechanisms. Instead, we are challenged to develop ways to improve IoT device security without requiring costly or complicated improvements to the devices themselves. A second challenge lies in the need to develop security mechanisms that will be effective even though IoT devices will, by their very nature, remain vulnerable to attack, and some will inevitably be compromised. These security mechanisms should protect the rest of the network from any devices that become compromised. Given the widespread use of IoT devices by consumers who may not even be aware that the devices are accessing their network, a third challenge is the practical need that IoT security mechanisms be easy to use. Ideally, security features should be so transparent that a user need not even be aware of their operation. To address these challenges, the National Cybersecurity Center of Excellence (NCCoE) and its collaborators have demonstrated the practicality and effectiveness of using the Internet Engineering Task Force's Manufacturer Usage Description (MUD) standard to reduce both the vulnerability of IoT devices to network-based attacks and the potential for harm from any IoT devices that become compromised.

1.2 Solution

This project demonstrates how to use MUD to strengthen security when deploying IoT devices on home and small-business networks. The demonstrated approach uses MUD to constrain the communication abilities of MUD-capable IoT devices, thereby reducing the potential for these devices to be attacked as well as reducing the potential for them to be used to launch network-based attacks—both attacks that could be launched across the internet and attacks on the MUD-capable IoT device's local network. Using MUD combats IoT-based, network-based attacks by providing a standardized and automated method for making access control information available to network control devices capable of prohibiting unauthorized traffic to and from IoT devices. When MUD is used, the network will automatically permit the IoT device to send and receive the traffic it requires to perform as intended, and the network will prohibit all other communication with the device. Even if an IoT device becomes compromised, MUD prevents it from being used in any attack that would require the device to send traffic to an unauthorized destination.

In developing the demonstrated approach, the NCCoE sought existing technologies that use the MUD specification (RFC 8520). The NCCoE envisions using MUD as one of many possible tools that can be deployed, in accordance with best practices, to improve IoT security. This practice guide describes four implementations of the MUD specification that support MUD-capable IoT devices. It describes how Build 2 uses threat signaling to prevent both MUD-capable and non-MUD-capable IoT devices from connecting to internet locations that are known to be potentially malicious. It describes how Build 3 supports secure and automated onboarding of both MUD-capable and non-MUD-capable devices using the Wi-Fi Alliance's Wi-Fi Easy Connect protocol. It also describes the importance of using update servers to perform periodic updates to all IoT devices so that the devices will be protected with up-to-date software patches. It shows IoT device developers and manufacturers, network equipment developers and manufacturers, and service providers who employ MUD-capable components how to integrate and use MUD to help make home and small-business networks more secure.

1.3 Benefits

The demonstrated approach offers specific benefits to several classes of stakeholders:

- Organizations and others who rely on the internet, including businesses that rely on their customers being able to reach them over the internet, can understand how MUD can be used to protect internet availability and performance against network-based attacks.
- IoT device manufacturers can see how MUD can protect against reputational damage resulting from their devices being easily exploited to support DDoS or other network-based attacks.
- Service providers can benefit from a reduction in the number of IoT devices that malicious actors can use to participate in DDoS attacks against their networks and degrade service for their customers.
- Users of IoT devices, including small businesses and homeowners, can better understand what to ask for with respect to the set of tools available to protect their internal networks from being subverted by malicious actors. They will also better understand what they can expect regarding reducing their vulnerability to threats that can result from such subversion. By protecting their networks, they also avoid suffering increased costs and bandwidth saturation that could result from having their machines captured and used to launch network-based attacks.

2 How to Use This Guide

This NIST Cybersecurity Practice Guide demonstrates a standards-based reference design and provides users with the information they need to replicate deployment of the MUD protocol to mitigate the threat of IoT devices being used to perform DDoS and other network-based attacks. This reference design is modular and can be deployed in whole or in part.

This guide contains three volumes and a supplement:

- NIST SP 1800-15A: *Executive Summary – why we wrote this guide, the challenge we address, why it could be important to your organization, and our approach to solving this challenge*
- NIST SP 1800-15B: *Approach, Architecture, and Security Characteristics – what we built and why, including the risk analysis performed, and the security control map* ***(you are here)***
- NIST SP 1800-15C: *How-To Guides – instructions for building the example implementations including all the security-relevant details that would allow you to replicate all or parts of this project*
- Functional Demonstration Results - supplement to NIST SP 1800-15B: *describes the functional demonstration results for the four implementations of the MUD-based reference solution*

It is intended for IoT device developers and manufacturers, network equipment developers and manufacturers, and service providers who employ MUD-capable components.

Depending on your role in your organization, you might use this guide in different ways:

Business decision makers, including chief security and technology officers, will be interested in the *Executive Summary,* NIST SP 1800-15A, which describes the following topics:

- challenges that enterprises face in mitigating IoT-based DDoS threats
- example solutions built at the NCCoE
- benefits of adopting the example solutions

Technology or security program managers who are concerned with how to identify, understand, assess, and mitigate risk will be interested in this part of the guide, NIST SP 1800-15B, which describes what we did and why. The following sections will be of particular interest:

- Section 3.4.3, Risk, provides a description of the risk analysis we performed.
- Section 5.2, Security Control Map, maps the security characteristics of this solution to cybersecurity standards and best practices.

You might share the *Executive Summary,* NIST SP 1800-15A, with your leadership team members to help them understand the importance of adopting standards-based mitigation of network-based distributed denial of service by using MUD protocols.

Information Technology (IT) professionals who want to implement an approach like this will find the whole practice guide useful. You can use the how-to portion of the guide, NIST SP 1800-15C, to replicate all or parts of the builds created in our lab. The how-to portion of the guide provides specific product installation, configuration, and integration instructions for implementing the example solutions. We do not re-create the product manufacturers' documentation, which is generally widely available. Rather, we show how we incorporated the products together in our environment to create each example solution.

This guide assumes that IT professionals have experience implementing security products within the enterprise. While we have used a suite of commercial and open-source products to address this challenge, this guide does not endorse these particular products. Your organization can adopt one of these example solutions or one that adheres to these guidelines in whole, or you can use this guide as a starting point for tailoring and implementing parts of the MUD protocol. Your organization's security experts should identify the products that will best integrate with your existing tools and IT system infrastructure. We hope you will seek products that are congruent with applicable standards and best practices. Section 5, Security Characteristic Analysis, maps the characteristics of the demonstrated approach to the cybersecurity controls provided by this reference solution.

A NIST Cybersecurity Practice Guide does not describe "the" solution, but a possible solution. This is a draft guide. We seek feedback on its contents and welcome your input. Comments, suggestions, and success stories will improve subsequent versions of this guide. Please contribute your thoughts to mitigating-iot-ddos-nccoe@nist.gov.

2.1 Typographic Conventions

The following table presents typographic conventions used in this volume.

Typeface/ Symbol	Meaning	Example
Italics	file names and path names; references to documents that are not hyperlinks; new terms; and placeholders	For language use and style guidance, see the *NCCoE Style Guide.*
Bold	names of menus, options, command buttons, and fields	Choose **File > Edit.**

Typeface/ Symbol	Meaning	Example
`Monospace`	command-line input, onscreen computer output, sample code examples, and status codes	`Mkdir`
`Monospace Bold`	command-line user input contrasted with computer output	**`service sshd start`**
blue text	link to other parts of the document, a web URL, or an email address	All publications from NIST's NCCoE are available at https://www.nccoe.nist.gov.

3 Approach

The NCCoE issued an open invitation to technology providers to participate in demonstrating an approach to deploying IoT devices in home and small-business networks in a manner that provides higher security than is typically achieved in today's environments. In this project, the MUD specification (RFC 8520) is applied to home and small-business networks that are composed of both IoT and fully featured devices (e.g., personal computers and mobile devices). MUD constrains the communication abilities of MUD-capable IoT devices, thereby reducing the potential for these devices to be attacked as well as the potential for them to be used to launch attacks. Network gateway components and IoT devices leverage MUD to ensure that IoT devices send and receive only the traffic they require to perform their intended function. The resulting constraints on the MUD-capable IoT device's communication abilities reduce the potential for MUD-capable devices to be the victims of network-based attacks, as well as reduce the ability for these devices to be used in a DDoS or other network-based attack. In addition, in Build 2, we provide network-wide access controls based on threat signaling to protect legacy IoT devices, MUD-capable IoT devices, and fully featured devices (e.g., personal computers). In Build 3, the Wi-Fi Alliance's Wi-Fi Easy Connect protocol is used to securely onboard both MUD-capable and non-MUD-capable IoT devices that are Wi-Fi Easy Connect-capable. Automatic secure updates are also recommended for all devices.

The NCCoE prepared a Federal Register Notice inviting technology providers to provide products and/or expertise to compose prototypes. Components sought included MUD-capable routers or switches; MUD managers; MUD file servers; MUD-capable DHCP servers; IoT devices capable of emitting or otherwise conveying a MUD URL; and network access control based on threat signaling. Cooperative Research and

Development Agreements (CRADAs) were established with qualified respondents, and build teams were assembled. The build teams fleshed out the initial architectures, and the collaborators' components were composed into example implementations, i.e., builds. Each build team documented the architecture and design of its build. As each build progressed, its team documented the steps taken to install and configure each component of the build. The teams then conducted functional testing of the builds, including demonstrating the ability to retrieve a device's MUD file and use it to determine what traffic the device would be permitted to send and receive. We verified that attempts to perform prohibited communications would be blocked. Each team conducted a risk assessment and a security characteristic analysis and documented the results, including mapping the security contributions of the demonstrated approach to the *Framework for Improving Critical Infrastructure Cybersecurity* (NIST Cybersecurity Framework) and other relevant standards. Finally, the NCCoE worked with industry collaborators to suggest considerations for enhancing future support for MUD.

3.1 Audience

The focus of this project is on home and small-business deployments. Its solution is targeted to address the needs of home and small-business networks, which have users who cannot be assumed to have extensive network administration experience and who therefore require plug-and-play functionality. Although the focus of this project is on home and small-business network applications, we do not intend home and small-business network users to be this guide's primary audience. This guide is intended for the following types of organizations that provide products and services to homes and small businesses:

- IoT device developers and manufacturers
- network equipment developers and manufacturers
- service providers that employ MUD-capable components

3.2 Scope

The scope of this NCCoE project is IoT deployments in those home and small-business applications where plug-and-play deployment is required. The demonstrated approach includes MUD-capable IoT devices that interact with traditional computing devices, as permitted by their MUD files, and that also interact with external systems to access update servers and various cloud services. It employs both MUD-capable and non-MUD-capable IoT devices, such as connected lighting controllers, cameras, mobile phones, printers, baby monitors, digital video recorders, and connected assistants.

The primary focus of this project is on the technical feasibility of implementing MUD to mitigate network-based attacks. We show use of threat signaling to protect both MUD-capable devices and devices that are not MUD capable from known threats. We also show how Wi-Fi Easy Connect protocol can onboard both MUD-capable and non-MUD-capable devices, thereby securely providing each device with unique credentials for connecting to the network.

The reference architecture for the demonstrated approach includes support for automatic secure software updates. All builds include a server that is meant to represent an update server to which MUD will permit devices to connect. However, demonstrations of actual IoT device software updates and patching were not included in the scope of the project.

Providing security protections for each of the components deployed in the demonstrated approach is important. However, demonstrating these protections is outside the scope of this project. It is assumed that network owners deploying the architecture will implement best practices for securing it. Also, governance, operational, life cycle, cost, legal, and privacy issues are outside the project's current scope.

3.3 Assumptions

This project is guided by the following assumptions:

- IoT devices, by definition, are not general-purpose devices.
- Each IoT device has an intended function, and this function is specific enough that the device's communication requirements can be defined accurately and completely.
- An IoT device's communication should be limited to only what is required for the device to perform its function.
- Cost is a major factor affecting consumer purchasing decisions and consequent product development decisions. Therefore, it is assumed that IoT devices will not typically include organic support for all their own security needs and would therefore benefit from protections provided by an outside mechanism, such as MUD.
- IoT device manufacturers will use the MUD file mechanism to indicate the communications that each device needs.
- Network routers can be automatically configured to enforce these communications so that
    - intended communications are permitted
    - unintended communications are prohibited
- If all MUD-capable network components are deployed and functioning as intended, a malicious actor would need to compromise one of the systems with which an IoT device is permitted to communicate to launch a network-based attack on the device. If a device were to be compromised, it could be used in a network-based attack only against systems with which it is permitted to communicate.
- Network owners who want to provide the security protections demonstrated in this project will:
    - be able to acquire and deploy all necessary components of the architecture on their own network, including MUD-capable IoT devices, Wi-Fi Easy Connect-capable IoT devices, a MUD manager, a MUD-capable gateway/router/switch, a threat-signaling-

capable gateway/router/switch, a Wi-Fi Easy Connect-capable gateway, and a mobile application or other mechanism for scanning the quick response (QR) code of a Wi-Fi Easy Connect-capable device

- o have access to MUD file servers that host the MUD files for their IoT devices, update servers, threat-signaling servers, and current threat intelligence

- All deployed architecture components are secure and can be depended upon to perform as designed.
- Best practices for administrative access and security updates will be implemented, and these will reduce the success rate of compromise attempts.

3.4 Risk Assessment

NIST SP 800-30 Revision 1, *Guide for Conducting Risk Assessments*, states that risk is “a measure of the extent to which an entity is threatened by a potential circumstance or event, and typically a function of: (i) the adverse impacts that would arise if the circumstance or event occurs; and (ii) the likelihood of occurrence.” The guide further defines risk assessment as “the process of identifying, estimating, and prioritizing risks to organizational operations (including mission, functions, image, reputation), organizational assets, individuals, other organizations, and the Nation, resulting from the operation of an information system. Part of risk management incorporates threat and vulnerability analyses, and considers mitigations provided by security controls planned or in place.”

The NCCoE recommends that any discussion of risk management, particularly at the enterprise level, begins with a comprehensive review of NIST SP 800-37 Revision 2, *Risk Management Framework for Information Systems and Organizations*—material that is available to the public. The Risk Management Framework (RMF) guidance, as a whole, proved to be invaluable in giving us a baseline to assess risks, from which we developed the project, the security characteristics of the builds, and this guide.

Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks, NIST Interagency or Internal Report (NISTIR) 8228, identified security and privacy considerations and expectations that, together with the NIST Cybersecurity Framework and *Security and Privacy Controls for Federal Information Systems and Organizations* (NIST Special Publication [SP] 800-53 Revision 5), informed our risk assessment and subsequent recommendations from which we developed the security characteristics of the builds and this guide.

3.4.1 Threats

Historically, internet devices have enjoyed full connectivity at the network and transport layers. Any pair of devices with valid internet protocol (IP) addresses was, in general, able to communicate by using transmission control protocol (TCP) for connection-oriented communications or User Datagram Protocol (UDP) for connectionless protocols. Full connectivity was a practical architectural option for fully featured devices (e.g., servers and personal computers) because the identity of communicating hosts depended largely on the needs of inherently unpredictable human users. Requiring a reconfiguration of hosts to permit communications to meet the needs of system users as they evolved was not a scalable solution. However, a combination of allowing only certain device capabilities and blocking devices or domains that are considered suspicious allowed network administrators to mitigate some threats.

With the evolution of internet hosts from multiuser systems to personal devices, this security posture became impractical, and the emergence of IoT has made it unsustainable. In typical networking environments, a malicious actor can detect an IoT device and launch an attack on that device from any system on the internet. Once compromised, that device can be used to attack any other system on the internet. Anecdotal evidence indicates that a new device will be detected and will experience its first attack within minutes of deployment. Because the devices being deployed often have known security flaws, the success rate for compromising detected systems is very high. Typically, malware is designed to compromise a list of specific devices, making such attacks very scalable. Once compromised, an IoT device can be used to compromise other internet-connected devices, launch attacks on any victim device on the internet, or launch attacks on devices within the local network hosting the device.

3.4.2 Vulnerabilities

The vulnerability of IoT devices in this environment is a consequence of full connectivity, exacerbated by the large number of security vulnerabilities in complex software systems. Modern systems ship with millions of lines of code, creating a target-rich environment for malicious actors. Some vendors provide patches for security vulnerabilities and an efficient means for securely updating their products. However, patches are often unavailable or nearly impossible to install on many other products, including many IoT devices. In addition, poorly designed and implemented default configuration baselines and administrative access controls, such as hard-coded or widely known default passwords, provide a large attack surface for malicious actors. Many IoT devices include those types of vulnerabilities. The Mirai malware, which launched a large DDoS attack on the internet infrastructure firm Dyn that took down many of the internet’s top destinations offline for much of a day, relied heavily on hard-coded administrative access to assemble botnets consisting of more than 100,000 devices.

3.4.3 Risk

The demonstrated approach implements a set of protocols designed to permit users and product support staff to constrain access to MUD-capable IoT devices. A network that includes IoT devices will be vulnerable to exploitation if some but not all IoT devices are MUD-capable. MUD may help prevent a compromised IoT device from doing harm to other systems on the network, and a device acting out of profile may indicate that it is compromised. However, MUD does not necessarily help owners find and identify already-compromised systems, and it does not help owners correct compromised systems without replacing or reprogramming existing system components. For example, if a system is compromised so that it emits a new URL referencing a MUD file that permits malicious actors to send traffic to and from the IoT device, MUD may not be able to help owners detect such compromised systems and stop the communications that should be prohibited. However, if a system is compromised but it is still emitting the correct MUD URL, MUD can detect and stop any unauthorized communications that the device attempts. Such attempts would also indicate potential compromises.

If a network is set up so that it uses legacy IoT devices that do not emit MUD URLs, these devices could be associated with MUD URLs or with MUD files themselves by using alternative means, such as a device serial number or a public key. If the device is compromised and attempts unauthorized communication, the attempt should be detected, and the device would be subjected to the constraints specified in its MUD file. Under these circumstances, MUD can permit the owner to find and identify already-compromised systems. Moreover, where threat signaling is employed, a compromised system that reaches back to a known malicious IP address can be detected, and the connection can be refused.

4 Architecture

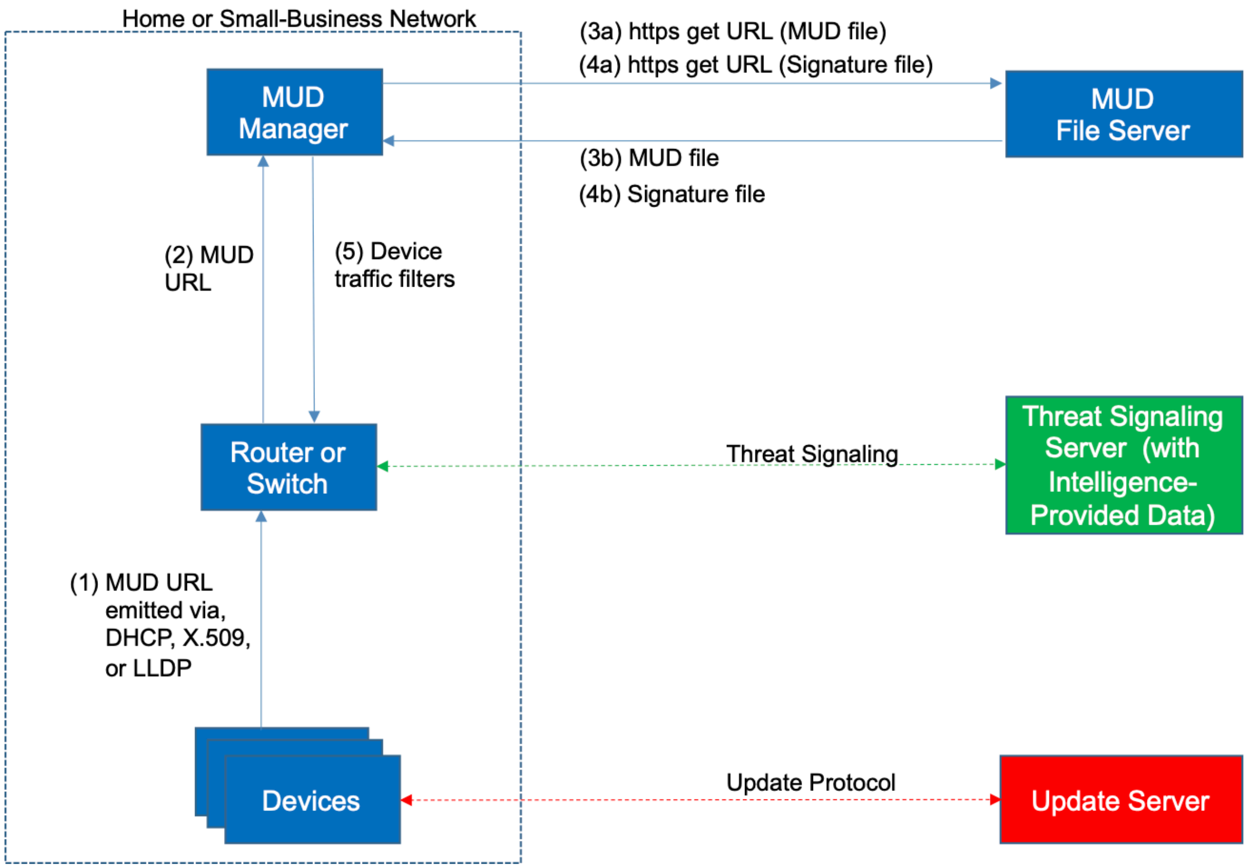
The project architecture is intended for home and small-business networks that are composed of both IoT components and fully featured devices (e.g., personal computers). The architecture is designed to provide three forms of protection:

- use of the MUD specification to automatically permit an IoT device to send and receive only the traffic it requires to perform as intended, thereby reducing the potential for the device to be the victim of a communications-based malware exploit or other network-based attack, and reducing the potential for the device, if compromised, to be used in a DDoS or other network-based attack
- use of network-wide access controls based on threat signaling to protect legacy (non-MUD-capable) IoT devices and fully featured devices, in addition to MUD-capable IoT devices, from connecting to domains that are known current threats
- automated secure software updates to all devices to ensure that operating system patches are installed promptly

4.1 Reference Architecture

Figure 4-1 depicts the logical architecture of the reference design. It consists of three main components: support for MUD, support for threat signaling, and support for periodic updates.

Figure 4-1 Reference Architecture



4.1.1 Support for MUD

A new functional component, the MUD manager, is introduced to augment the existing networking functionality offered by the home/small-business network router or switch. Note that the MUD manager is a logical component. Physically, the functionality that the MUD manager provides can and often is combined with that of the network router in a single device.

IoT devices must somehow be associated with a MUD file. The MUD specification describes three possible mechanisms through which the IoT device can provide the MUD file URL to the network: inserting the MUD URL into DHCP address requests that they generate when they attach to the network (e.g., when powered on) (supported by Builds 1, 2, and 4), providing the MUD URL in a Link Layer Discovery Protocol (LLDP) frame (also supported by Build 1), or providing the MUD URL as a field in an X.509 certificate that the device provides to the network via a protocol such as Tunnel Extensible Authentication Protocol. Each of these MUD URL emission mechanisms is listed as a possibility in Figure 4-1. In addition, the MUD specification provides flexibility to enable other mechanisms by which MUD file URLs can be associated with IoT devices. One alternative mechanism is to associate the device with its MUD file by using the bootstrapping information that the device conveys as part of the Wi-Fi Easy Connect onboarding process (supported by Build 3).

Figure 4-1 uses labeled arrows to depict the steps involved in supporting MUD when an IoT device emits its MUD file URL using one of the mechanisms specified in the MUD specification:

- The IoT device emits a MUD URL by using a mechanism such as DHCP, LLDP, or X.509 certificate (step 1).
- The router extracts the MUD URL from the protocol frame of whatever mechanism was used to convey it and forwards this MUD URL to the MUD manager (step 2).
- Once the MUD URL is received, the MUD manager uses hypertext transfer protocol secure (https) to request the MUD file from the MUD file server by using the MUD URL provided in the previous step (step 3a); if successful, the MUD file server at the specified location will serve the MUD file (step 3b).
- Next, the MUD manager uses https to request the signature file associated with the MUD file (step 4a) and upon receipt (step 4b) verifies the MUD file by using its signature file.
- The MUD file describes the communications requirements for the IoT device. Once the MUD manager has determined the MUD file to be valid, the MUD manager converts the access control rules in the MUD file into access control entries (e.g., access control lists—ACLs, firewall rules, or flow rules) and installs them on the router or switch (step 5).

If an alternative method of conveying the device's MUD file URL to the MUD manager is used (i.e., a mechanism other than emission of the MUD file URL via DHCP, X.509, or LLDP), steps 1 and 2 in Figure 4-1 would be replaced by that alternative mechanism.

Once the device's access control rules are applied to the router or switch, the MUD-capable IoT device will be able to communicate with approved local hosts and internet hosts as defined in the MUD file, and any unapproved communication attempts will be blocked.

As described in the MUD specification, the MUD file rules can limit both traffic between the device and external internet domains (north/south traffic), as well as traffic between the device and other devices on the local network (east/west traffic). East/west traffic can be limited by using the following constructs:

- controller—class of devices known to be controllers (could describe well-known services such as DNS or Network Time Protocol [NTP])
- my-controller—class of devices that the local network administrator admits to the class
- local-networks—class of IP addresses that are scoped within some local administrative boundary
- same-manufacturer—class of devices from the same manufacturer as the IoT device in question

- manufacturer—class of devices made by a particular manufacturer as identified by the authority component of its MUD URL

It is worth noting that while MUD requires use of a MUD-capable router on the local network, whether this router is stand-alone equipment provided by a third-party network equipment vendor (as is the case in Builds 1, 2, and 4) or integrated with the service provider's residential gateway equipment (Build 3) is not relevant to the ability of MUD to protect the network. While a service provider will be free to support MUD in its internet gateway equipment and infrastructure, such Internet Service Provider (ISP) support is not necessary. A home or small-business network can benefit from the protections that MUD has to offer without ISPs needing to make any changes or provide any support other than basic internet connectivity.

4.1.2 Support for Updates

To provide additional security, the reference architecture also supports periodic updates. All builds include a server that is meant to represent an update server to which MUD will permit devices to connect. Each device on an operational network should be configured to periodically contact its update server to download and apply security patches, ensuring that it is running the most up-to-date and secure code available. To ensure that such updates are possible, an IoT device's MUD file must explicitly permit the IoT device to receive traffic from the update server. Although regular manufacturer updates are crucial to security, the builds described in this practice guide demonstrate only the ability for IoT devices to receive faux updates from a notional update server. Communications between IoT devices and their corresponding update servers are not standardized.

4.1.3 Support for Threat Signaling

To provide additional protection for both MUD-capable and non-MUD-capable devices, the reference architecture also envisions support for threat signaling. The router or switch can receive threat feeds from a notional threat-signaling server to use as a basis for restricting certain types of network traffic. For example, both MUD-capable and non-MUD-capable devices can be prevented from connecting to internet domains that have been identified as being potentially malicious. Communications between the threat-signaling server and the router/switch are not standardized.

4.1.4 Build-Specific Features

The reference architecture depicted in Figure 4-1 is intentionally general. Each build instantiates this reference architecture in a unique way, depending on the equipment used and the capabilities supported. While all four builds support MUD and the ability to receive faux updates from a notional update server, only Build 2 currently supports threat signaling. Build 1 and Build 2 include nonstandard device discovery technology to discover, inventory, profile, and classify attached devices. Such classification can be used to validate that the access that is being granted to each device is consistent with that device's manufacturer and model. In Build 2, a device's manufacturer and model can be used as a basis for identifying and enforcing that device's traffic profile. Build 3 implements the Wi-Fi Easy Connect protocol to onboard both MUD-capable and non-MUD-capable devices, thereby securely providing each device with unique credentials for connecting to the network. For those devices that are both Easy Connect- and MUD-capable, the device's MUD rules are retrieved and installed on the local gateway during the onboarding process, ensuring that the device's MUD-based communication constraints are already in effect when the device connects to the network. Build 3 also creates and enforces separate trust zones (e.g., network segments) called *micronets* to which devices are assigned according to their intended network function.

The four builds of the reference architecture that have been completed and demonstrated are as follows:

- Build 1 uses products from Cisco Systems, DigiCert, Forescout, and Molex. The Cisco MUD manager supports MUD, and the Forescout virtual appliances and enterprise manager perform non-MUD-related device discovery on the network. Molex Power over Ethernet (PoE) Gateway and Light Engine are used as MUD-capable IoT devices. Certificates from DigiCert are also used.
- Build 2 uses products from MasterPeace Solutions, Ltd.; GCA,; ThreatSTOP; and DigiCert. The MasterPeace Solutions Yikes! router, cloud service, and mobile application support MUD as well as perform device discovery on the network and apply additional traffic rules to both MUD-capable and non-MUD-capable devices based on device manufacturer and model. The Yikes! router also integrates with the GCA Quad9 DNS service and the ThreatSTOP threat MUD file server to prevent devices (MUD-capable or not) from connecting to domains that have been identified as potentially malicious based on current threat intelligence. Certificates from DigiCert are also used.
- Build 3 uses products from CableLabs and DigiCert. CableLabs Micronets (e.g., Micronets Gateway, Micronets Manager, Micronets mobile phone application, and related service provider cloud-based infrastructure) supports MUD and implements the Wi-Fi Alliance's Wi-Fi Easy Connect protocol to securely onboard devices to the network. It also uses software-defined networking to create separate trust zones (e.g., network segments) called micronets to which devices are assigned according to their intended network function. Certificates from DigiCert are also used.
- Build 4 uses software developed at the NIST Advanced Networking Technologies laboratory. This software supports MUD and is intended to serve as a working prototype of the MUD specification to demonstrate feasibility and scalability. Certificates from DigiCert are also used.

The logical architectures and detailed descriptions of the builds mentioned above are in Section 6 (Build 1), Section 7 (Build 2), Section 8 (Build 3), and Section 9 (Build 4).

4.2 Physical Architecture

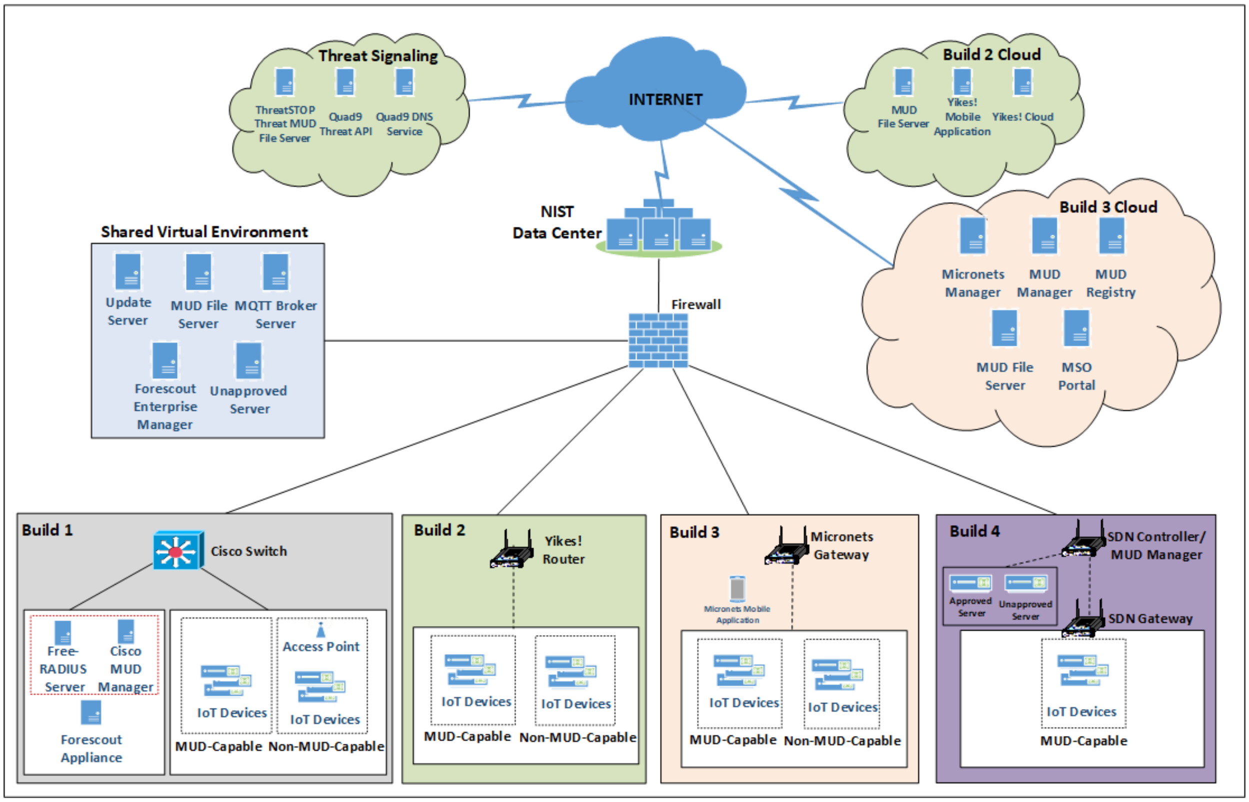
Figure 4-2 depicts the high-level physical architecture of the NCCoE laboratory environment. As depicted, the NCCoE laboratory network is connected to the internet via the NIST data center. Access to and from the NCCoE network is protected by a firewall. The NCCoE network includes a shared virtual environment that houses an update server, a MUD file server, an unapproved server (i.e., a server that is not listed as a permissible communications source or destination in any MUD file), a Message Queuing Telemetry Transport (MQTT) broker server, and a Forescout enterprise manager. These components are hosted at the NCCoE and are used across builds where applicable. DigiCert provided the Transport Layer Security (TLS) certificate and Premium Certificate used by the MUD file server.

All four builds, as depicted in the diagram, have been implemented:

- Build 1 network components consist of a Cisco Catalyst 3850-S switch, a Cisco MUD manager, a FreeRADIUS server, and a virtualized Forescout appliance on the local network. Build 1 also requires support from all components that are in the shared virtual environment, including the Forescout enterprise manager.
- Build 2 network components consist of a MasterPeace Solutions, Ltd. Yikes! router on the local network. Build 2 requires support from the MUD file server, Yikes! cloud, and a Yikes! mobile application that are resident on the Build 2 cloud. The Yikes! router includes threat-signaling capabilities (not depicted) that have been integrated with it. Build 2 also requires support from threat-signaling cloud services that consist of the ThreatSTOP threat MUD file server, Quad9 threat application programming interface (API), and Quad9 DNS service. Build 2 uses only the update server and unapproved server components that are in the shared virtual environment.
- Build 3 network components consist of a CableLabs Micronets Gateway/wireless access point (AP) that resides on the local network and that operates in conjunction with various service provider components and partner/service provider offerings that reside in the Micronets virtual environment in the Build 3 cloud. The Micronets Gateway is controlled by a Micronets Manager that resides in the Build 3 cloud and that coordinates a number of cloud-based Micronets micro-services, some of which are depicted. Build 3 also includes a Micronets mobile Application that provides the user and device interfaces for performing device onboarding.
- Build 4 network components consist of a software-defined networking (SDN)-capable gateway/switch on the local network, an SDN controller/MUD manager, and approved and unapproved servers that are located remotely from the local network. Build 4 also uses the MUD file server that is resident in the shared virtual environment.

IoT devices used in all four builds include those that are both MUD-capable and non-MUD-capable. The MUD-capable IoT devices used, which vary across builds, include Raspberry Pi, ARTIK, u-blox, Intel UP Squared, BeagleBone Black, NXP i.MX 8M (devkit), and the Molex Light Engine controlled by PoE Gateway. Non-MUD-capable devices used, which also vary across builds, include a wireless access point, cameras, a printer, mobile phones, lighting devices, a connected assistant device, a baby monitor, and a digital video recorder. Each of the completed builds and the roles that their components play in their architectures are explained in more detail in Section 6 (Build 1), Section 7 (Build 2), Section 8 (Build 3), and Section 9 (Build 4).

Figure 4-2 Physical Architecture



5 Security Characteristic Analysis

The purpose of the security characteristic analysis is to understand the extent to which the project meets its objective of demonstrating the ability to identify IoT components to MUD managers and manage access to those components while limiting unauthorized access to and from the components. In addition, it seeks to understand the security benefits of the demonstrated approach.

5.1 Assumptions and Limitations

The security characteristic analysis has the following limitations:

- It is neither a comprehensive test of all security components nor a red-team exercise.
- It cannot identify all weaknesses.
- It does not include the lab infrastructure. It is assumed that devices are hardened. Testing these devices would reveal only weaknesses in implementation that would not be relevant to those adopting this reference architecture.

5.2 Security Control Map

One aspect of our security characteristic analysis involved assessing how well the reference design addresses the security characteristics that it was intended to support. The Cybersecurity Framework Subcategories were used to provide structure to the security assessment by consulting the specific sections of each standard that are cited in reference to a Subcategory. The cited sections provide validation points that an example solution would be expected to exhibit. Using the Cybersecurity Framework Subcategories as a basis for organizing our analysis allowed us to systematically consider how well the reference design supports the intended security characteristics.

The characteristic analysis was conducted in the context of home network and small-business usage scenarios.

The capabilities demonstrated by the architectural elements described in Section 4 and used in the home networks and small-business environments are primarily intended to address requirements, best practices, and capabilities described in the following NIST documents: *Framework for Improving Critical Infrastructure Cybersecurity* (NIST Cybersecurity Framework), *Security and Privacy Controls for Federal Information Systems and Organizations* (NIST SP 800-53), and *Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks* (NIST Interagency or Internal Report 8228). NISTIR 8228 identifies a set of 25 security and privacy expectations for IoT devices and subsystems. These include expectations regarding meeting device protection, data protection, and privacy protection goals. The reference architecture directly addresses the PR.AC-1, PR.AC-2, PR.AC-3, PR.AC-7, and PR.PT-3 Cybersecurity Framework Subcategories and supports activities addressing the ID.AM-1, ID.AM-2, ID.AM-3, ID.RA-2, ID.RA-3, PR.AC-5, PR.AC-4, PR.DS-5, PR.DS-6, PR.IP-1, PR.IP-3, and DE.CM-8 Subcategories. Also, the reference architecture directly addresses NIST SP 800-53 controls AC-3, AC-18, CM-7, IA-6, SC-5, SC-7, SC-23, and SI-2, and it supports activities addressing NIST SP 800-53 controls AC-4, AC-6, AC-24, CM-7, CM-8, IA-2, IA-5, IA-8, PA-4, PM-5, RA-5, SC-8, and SI-5. In addition, the reference architecture addresses eight of the NISTIR 8228 expectations. Table 5-1 describes how MUD-specific example implementation characteristics address NISTIR 8228 expectations, NIST SP 800-53 controls, and NIST Cybersecurity Framework Subcategories.

883 Table 5-1 Mapping Characteristics of the Demonstrated Approach, as Instantiated in at Least One of
884 Builds 1-4, to NISTIR 8228 Expectations, NIST SP 800-53 Controls, and NIST Cybersecurity Framework
885 Subcategories

Applicable Project Description Element that Addresses the Expectation	Applicable NISTIR 8228 Expectations	NIST SP 800-53 Controls Supported	Cybersecurity Framework Subcategories Supported
There exists some mechanism for associating each device with a URL that can be used to identify and locate its MUD file. The device itself may emit the MUD file URL in one of three ways: • IoT devices insert the MUD URL into DHCP address requests when the device attaches to the network (e.g., powers on) (Build 1, Build 2, and Build 4). • MUD URL is provided in LLDP (Build 1). • MUD URL is included in X.509 certificate. However, a MUD URL may be learned by a network by other means, and the MUD specification is designed to allow flexibility in this regard. (In Build 3, the information required to retrieve the MUD URL from the MUD registry is conveyed using two fields in the device bootstrapping information, which is encoded in the device's Wi-Fi Easy Connect protocol QR code.)	Device has a built-in identifier.	Supports CM-8 System Component Inventory PM-5 System Inventory	Supports ID.AM-1 Physical devices and systems within the organization are inventoried.
Devices that support the Wi-Fi Easy Connect protocol have been preconfigured with their own unique bootstrapping public/private key pair before they initiate onboarding. Although the private key is not actually a device identifier, the device's possession of this unique private key is what enables the device to be authenticated as part of the onboarding protocol. (Build 3)	Device has a built-in unique identifier.	Supports CM-8 System Component Inventory PM-5 System Inventory	Supports ID.AM-1 Physical devices and systems within the organization are inventoried.
The MUD file URL, which identifies the device type, among other things, is passed to the MUD manager, which retrieves a MUD file by	Device can interface with	Provides AC-3 Access Enforcement	Provides PR.PT-3 The principle of least functionality

Applicable Project Description Element that Addresses the Expectation	Applicable NISTIR 8228 Expectations	NIST SP 800-53 Controls Supported	Cybersecurity Framework Subcategories Supported
using https. The MUD file describes the communications requirements for this device. The MUD manager converts the requirements into access control information for enforcement by the router or switch. (all builds)	enterprise asset management systems.	AC-18 Wireless Access CM-7 Least Functionality SC-5 Denial of Service Protection SC-7 Boundary Protection Supports AC-4 Information Flow Enforcement AC-6 Least Privilege AC-24 Access Control Decisions CM-8 System Component Inventory PM-5 System Inventory	is incorporated by configuring systems to provide only essential capabilities. Supports ID.AM-1 Physical devices and systems within the organization are inventoried. ID.AM-2 Software platforms and applications within the organization are inventoried. ID.AM-3 Organizational communication and data flows are mapped. PR.AC-4 Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. PR.AC-5 Network integrity is protected (e.g.,

Applicable Project Description Element that Addresses the Expectation	Applicable NISTIR 8228 Expectations	NIST SP 800-53 Controls Supported	Cybersecurity Framework Subcategories Supported
			network segregation, network segmentation). PR.DS-5 Protections against data leaks are implemented. DE.AE-1 A baseline of network operations and expected data flows for users and systems is established and managed.
IoT devices periodically contact the appropriate update server to download and apply security patches. (all builds)	The manufacturer will provide patches or upgrades for all software and firmware throughout each device’s life span.	Provides SI-2 Flaw Remediation	Supports PR.IP-1 A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). PR.IP-3 Configuration change control processes are in place.
The router or switch receives threat feeds from the threat-signaling server to use as a basis for restricting certain types of network traffic. (Build 2)	The device either supports the use of vulnerability	Supports AC-24 Access Control Decisions	Supports ID.RA-2 Cyber threat intelligence is received

Applicable Project Description Element that Addresses the Expectation	Applicable NISTIR 8228 Expectations	NIST SP 800-53 Controls Supported	Cybersecurity Framework Subcategories Supported
	scanners or provides built-in vulnerability identification and reporting capabilities.	RA-5 Vulnerability Scanning SI-5 Security Alerts, Advisories, and Directives	from information-sharing forums and sources. ID.RA-3 Threats, both internal and external, are identified and documented. DE.CM-8 Vulnerability scans are performed.
Using the Wi-Fi Easy Connect protocol to onboard devices ensures that there is no need for anyone to be privy to the device's network credentials. The onboarding protocol provisions the network credentials onto the device automatically, using a secure channel, and the device is then able to present its credentials to the network as part of the standard Wi-Fi network connection handshake. There is no need for the device's network password to be input by a human, and the credentials are never displayed, so presentation of the device's network credentials to the network does not pose any risk that the credentials will be viewed and thereby disclosed. (Build 3)	The device can conceal password characters from display when a person enters a password for a device, such as on a keyboard or touchscreen.	Supports IA-6 Authenticator Feedback	Provides PR.AC-7 Users, devices, and other assets are authenticated commensurate with the risk of the transaction.
The MUD file URL is passed to the MUD manager, which retrieves a MUD file from the designated website (denoted as the MUD file server) by using https. The MUD file server must have a valid TLS certificate, and the MUD file itself must have a valid signature. The MUD file describes the communications requirements for this device. The MUD manager converts the requirements into access	The device can use existing enterprise authenticators and authentication mechanisms.	Supports IA-2 Identification and Authentication (Organizational Users) IA-5 Authenticator Management	Provides PR.AC-1 Identities and credentials are issued, managed, verified, revoked, and audited for

Applicable Project Description Element that Addresses the Expectation	Applicable NISTIR 8228 Expectations	NIST SP 800-53 Controls Supported	Cybersecurity Framework Subcategories Supported
control information for enforcement by the router or switch. (all builds)		IA-8 Identification and Authentication (Non-Organizational Users)	authorized devices, users, and processes. PR.AC-3 Remote access is managed. PR.AC-7 Users, devices, and other assets are authenticated commensurate with the risk of the transaction.
Each device that is onboarded using the Wi-Fi Easy Connect protocol is provisioned with unique network credentials that enable the device to authenticate to the network as part of the standard Wi-Fi network connection handshake. (Build 3)	The device can use existing enterprise authenticators and authentication mechanisms.	Supports IA-2 Identification and Authentication (Organizational Users) IA-5 Authenticator Management IA-8 Identification and Authentication (Non-Organizational Users)	Provides PR.AC-1 Identities and credentials are issued, managed, verified, revoked, and audited for authorized devices, users, and processes. PR.AC-3 Remote access is managed. PR.AC-7 Users, devices, and other assets are authenticated commensurate with the risk of the transaction.
There exists some mechanism for associating each device with a URL that can identify and locate its MUD file. The MUD file URL is passed to the MUD manager, which retrieves	Device can prevent unauthorized ac-	Provides SC-23 Session Authenticity	Provides PR.PT-3 The principle of least functionality

Applicable Project Description Element that Addresses the Expectation	Applicable NISTIR 8228 Expectations	NIST SP 800-53 Controls Supported	Cybersecurity Framework Subcategories Supported
a MUD file from the designated website (denoted as the MUD file server) by using https. The MUD file describes the communications requirements for this device. The MUD manager converts the requirements into access control information for enforcement by the router or switch. (all builds)	cess to all sensitive data transmitted from it over networks.	Supports AC-18 Wireless Access SC-8 Transmission Confidentiality and Integrity	is incorporated by configuring systems to provide only essential capabilities. Supports PR.DS-5 Protections against data leaks are implemented. PR.DS-6 Integrity-checking mechanisms are used to verify software, firmware, and information integrity.
There exists some mechanism for associating each device with a URL that can identify and locate its MUD file. The MUD file URL is passed to the MUD manager, which retrieves a MUD file from the designated website (denoted as the MUD file server) by using https. The MUD file describes the communications requirements for this device. The MUD manager converts the requirements into access control information for enforcement by the router or switch. (all builds) The router or switch periodically receives threat feeds from the threat-signaling server to use as a basis for restricting certain types of network traffic. (Build 2)	There is sufficient centralized control to apply policy or regulatory requirements to personally identifiable information.	Supports PA-4 Information Sharing with External Parties	None

886 Table 5-2 details Cybersecurity Framework Identify, Protect, and Detect Categories and Subcategories
887 that the example implementations directly address or for which the example implementations may

serve a supporting role. Entries in the Cybersecurity Framework Subcategory column that are directly addressed are highlighted in green. Informative references are made for each Subcategory. The following sources are used for informative references: Center for Internet Security (CIS), Control Objectives for Information and Related Technology (COBIT), International Society of Automation (ISA), International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), and NIST SP 800-53. While some of the references provide general guidance that informs implementation of referenced Cybersecurity Framework Core Functions, the NIST SP and Federal Information Processing Standard (FIPS) references provide specific recommendations that should be considered when composing and configuring security platforms. (Note that not all of the informative references apply to this example implementation.)

Table 5-2 Mapping Project Objectives to the Cybersecurity Framework and Informative Security Control References

Cybersecurity Framework Category	Cybersecurity Framework Subcategory	Informative References
Asset Management (ID.AM): The data, personnel, devices, systems, and facilities that enable the organization to achieve business purposes are identified and managed consistent with their relative importance to business objectives and the organization's risk strategy.	**ID.AM-1:** Physical devices and systems within the organization are inventoried.	**CIS** CSC 1 **COBIT 5** BAI09.01, BAI09.02 **ISA 62443-2-1:2009** 4.2.3.4 **ISA 62443-3-3:2013** SR 7.8 **ISO/IEC 27001:2013** A.8.1.1, A.8.1.2 **NIST SP 800-53 Rev. 4** CM-8, PM-5
	ID.AM-2: Software platforms and applications within the organization are inventoried.	**CIS** CSC 2 **COBIT 5** BAI09.01, BAI09.02, BAI09.05 **ISA 62443-2-1:2009** 4.2.3.4 **ISA 62443-3-3:2013** SR 7.8 **ISO/IEC 27001:2013** A.8.1.1, A.8.1.2, A.12.5.1 **NIST SP 800-53 Rev. 4** CM-8, PM-5
	ID.AM-3: Organizational communication and data flows are mapped.	**CIS** CSC 12 **COBIT 5** DSS05.02 **ISA 62443-2-1:2009** 4.2.3.4 **ISA 62443-3-3:2013** SR 7.8 **ISO/IEC 27001:2013** A.8.1.1, A.8.1.2, A.12.5.1 **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8
Risk Assessment (ID.RA): The organization understands the	**ID.RA-2:** Cyber threat intelligence is received from information-sharing forums and sources.	**CIS** CSC 4 **COBIT 5** BAI08.01 **ISA 62443-2-1:2009** 4.2.3, 4.2.3.9,

Cybersecurity Framework Category	Cybersecurity Framework Subcategory	Informative References
cybersecurity risk to organizational operations (including mission, functions, image, or reputation), organizational assets, and individuals.		4.2.3.12 **ISO/IEC 27001:2013** A.6.1.4 **NIST SP 800-53 Rev. 4** SI-5, PM-15, PM-16
	ID.RA-3: Threats, both internal and external, are identified and documented.	**CIS** CSC 4 **COBIT 5** APO12.01, APO12.02, APO12.03, APO12.04 **ISA 62443-2-1:2009** 4.2.3, 4.2.3.9, 4.2.3.12 **ISO/IEC 27001:2013** Clause 6.1.2 **NIST SP 800-53 Rev. 4** RA-3, SI-5, PM-12, PM-16
Identity Management, Authentication, and Access Control (PR.AC): Access to physical and logical assets and associated facilities is limited to authorized users, processes, and devices and is managed consistent with the assessed risk of unauthorized access to authorized activities and transactions.	**PR.AC-1:** Identities and credentials are issued, managed, verified, revoked, and audited for authorized devices, users, and processes.	**CIS** CSC 1, 5, 15, 16 **COBIT 5** DSS05.04, DSS06.03 **ISA 62443-2-1:2009** 4.3.3.5.1 **ISA 62443-3-3:2013** SR 1.1, SR 1.2, SR 1.3, SR 1.4, SR 1.5, SR 1.7, SR 1.8, SR 1.9 **ISO/IEC 27001:2013** A.9.2.1, A.9.2.2, A.9.2.3, A.9.2.4, A.9.2.6, A.9.3.1, A.9.4.2, A.9.4.3 **NIST SP 800-53 Rev. 4** AC-1, AC-2, IA-1, IA-2, IA-3, IA-4, IA-5, IA-6, IA-7, IA-8, IA-9, IA-10, IA-11
	PR.AC-3: Remote access is managed.	**CIS** CSC 12 **COBIT 5** APO13.01, DSS01.04, DSS05.03 **ISA 62443-2-1:2009** 4.3.3.6.6 **ISA 62443-3-3:2013** SR 1.13, SR 2.6 **ISO/IEC 27001:2013** A.6.2.1, A.6.2.2, A.11.2.6, A.13.1.1, A.13.2.1 **NIST SP 800-53 Rev. 4** AC-1, AC-17, AC-19, AC-20, SC-15
	PR.AC-4: Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties.	**CIS** CSC 3, 5, 12, 14, 15, 16, 18 **COBIT 5** DSS05.04 **ISA 62443-2-1:2009** 4.3.3.7.3 **ISA 62443-3-3:2013** SR 2.1 **ISO/IEC 27001:2013** A.6.1.2, A.9.1.2,

Cybersecurity Framework Category	Cybersecurity Framework Subcategory	Informative References
		A.9.2.3, A.9.4.1, A.9.4.4, A.9.4.5 **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC-5, AC-6, AC-14, AC-16, AC-24
	PR.AC-5: Network integrity is protected, incorporating network segregation where appropriate.	**CIS** CSC 9, 14, 15, 18 **COBIT 5** DSS01.05, DSS05.02 **ISA 62443-2-1:2009** 4.3.3.4 **ISA 62443-3-3:2013** SR 3.1, SR 3.8 **ISO/IEC 27001:2013** A.13.1.1, A.13.1.3, A.13.2.1, A.14.1.2, A.14.1.3 **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7
	PR.AC-7: Users, devices, and other assets are authenticated (e.g., single-factor, multifactor) commensurate with the risk of the transaction (e.g., individuals' security and privacy risks and other organizational risks).	**CIS** CSC 1, 12, 15, 16 **COBIT 5** DSS05.04, DSS05.10, DSS06.10 **ISA 62443-2-1:2009** 4.3.3.6.1, 4.3.3.6.2, 4.3.3.6.3, 4.3.3.6.4, 4.3.3.6.5, 4.3.3.6.6, 4.3.3.6.7, 4.3.3.6.8, 4.3.3.6.9 **ISA 62443-3-3:2013** SR 1.1, SR 1.2, SR 1.5, SR 1.7, SR 1.8, SR 1.9, SR 1.10 **ISO/IEC 27001:2013** A.9.2.1, A.9.2.4, A.9.3.1, A.9.4.2, A.9.4.3, A.18.1.4 **NIST SP 800-53 Rev. 4** AC-7, AC-8, AC-9, AC-11, AC-12, AC-14, IA-1, IA-2, IA-3, IA-4, IA-5, IA-8, IA-9, IA-10, IA-11
Data Security (PR.DS): Information and records (data) are managed consistent with the organization's risk strategy to protect the confidentiality, integrity, and availability of information.	**PR.DS-5:** Protections against data leaks are implemented.	**CIS** CSC 13 **COBIT 5** APO01.06, DSS05.04, DSS05.07, DSS06.02 **ISA 62443-3-3:2013** SR 5.2 **ISO/IEC 27001:2013** A.6.1.2, A.7.1.1, A.7.1.2, A.7.3.1, A.8.2.2, A.8.2.3, A.9.1.1, A.9.1.2, A.9.2.3, A.9.4.1, A.9.4.4, A.9.4.5, A.10.1.1, A.11.1.4, A.11.1.5, A.11.2.1, A.13.1.1, A.13.1.3, A.13.2.1, A.13.2.3, A.13.2.4, A.14.1.2, A.14.1.3 **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-

Cybersecurity Framework Category	Cybersecurity Framework Subcategory	Informative References
		6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4
	PR.DS-6: Integrity-checking mechanisms are used to verify software, firmware, and information integrity.	**ISA 62443-3-3:2013** SR 3.1, SR 3.3, SR 3.4, SR 3.8 **ISO/IEC 27001:2013** A.12.2.1, A.12.5.1, A.14.1.2, A.14.1.3 **FIPS 140-2** Sec. 4 **NIST SP 800-45 Ver. 2** 2.4.2, 3, 4.2.3, 4.3, 5.1, 6.1, 7.2.2, 8.2, 9.2 **NIST SP 800-49** 2.2.1, 2.3.2, 3.4 **NIST SP 800-52 Rev. 1** 3, 4, D1.4 **NIST SP 800-53 Rev. 4** SI-7 **NIST SP 800-57 Part 1 Rev. 4** 5.5, 6.1, 8.1.5.1, B.3.2, B.5 **NIST SP 800-57 Part 2** 1, 3.1.2.1.2, 4.1, 4.2, 4.3, A.2.2, A.3.2, C.2.2 **NIST SP 800-81-2** All **NIST SP 800-130** 2.2, 4.3, 6.2.1, 6.3, 6.4, 6.5, 6.6.1 **NIST SP 800-152** 6.1.3, 6.2.1, 8.2.1, 8.2.4, 9.4 **NIST SP 800-177** 2.2, 4.1, 4.4, 4.5, 4.7, 5.2, 5.3
Information Protection Processes and Procedures (PR.IP): Security policies (that address purpose, scope, roles, responsibilities, management commitment, and coordination among organizational entities), processes, and procedures are maintained and used to manage protection of information systems and assets.	**PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality).	**CIS** CSC 1 **COBIT 5** BAI10.01, BAI10.02, BAI10.03, BAI10.05 **ISA 62443-2-1:2009** 4.3.4.3.2, 4.3.4.3.3 **ISA 62443-3-3:2013** SR 7.6 **ISO/IEC 27001:2013** A.12.1.2, A.12.5.1, A.12.6.2, A.14.2.2, A.14.2.3, A.14.2.4 **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10
	PR.IP-3: Configuration change control processes are in place.	**CIS** CSC 3, 11 **COBIT 5** BAI01.06, BAI06.01 **ISA 62443-2-1:2009** 4.3.4.3.2,

Cybersecurity Framework Category	Cybersecurity Framework Subcategory	Informative References
		4.3.4.3.3 **ISA 62443-3-3:2013** SR 7.6 **ISO/IEC 27001:2013** A.12.1.2, A.12.5.1, A.12.6.2, A.14.2.2, A.14.2.3, A.14.2.4 **NIST SP 800-53 Rev. 4** CM-3, CM-4, SA-10
Protective Technology (PR.PT): Technical security solutions are managed to ensure the security and resilience of systems and assets, consistent with related policies, procedures, and agreements.	**PR.PT-3:** The principle of least functionality is incorporated by configuring systems to provide only essential capabilities.	**CIS** CSC 3, 11, 14 **COBIT 5** DSS05.02, DSS05.05, DSS06.06 **ISA 62443-2-1:2009** 4.3.3.5.1, 4.3.3.5.2, 4.3.3.5.3, 4.3.3.5.4, 4.3.3.5.5, 4.3.3.5.6, 4.3.3.5.7, 4.3.3.5.8, 4.3.3.6.1, 4.3.3.6.2, 4.3.3.6.3, 4.3.3.6.4, 4.3.3.6.5, 4.3.3.6.6, 4.3.3.6.7, 4.3.3.6.8, 4.3.3.6.9, 4.3.3.7.1, 4.3.3.7.2, 4.3.3.7.3, 4.3.3.7.4 **ISA 62443-3-3:2013** SR 1.1, SR 1.2, SR 1.3, SR 1.4, SR 1.5, SR 1.6, SR 1.7, SR 1.8, SR 1.9, SR 1.10, SR 1.11, SR 1.12, SR 1.13, SR 2.1, SR 2.2, SR 2.3, SR 2.4, SR 2.5, SR 2.6, SR 2.7 **ISO/IEC 27001:2013** A.9.1.2 **NIST SP 800-53 Rev. 4** AC-3, CM-7
Security Continuous Monitoring (DE.CM): The information system and assets are monitored to identify cybersecurity events and verify the effectiveness of protective measures.	**DE.CM-8:** Vulnerability scans are performed.	**CIS** CSC 4, 20 **COBIT 5** BAI03.10, DSS05.01 **ISA 62443-2-1:2009** 4.2.3.1, 4.2.3.7 **ISO/IEC 27001:2013** A.12.6.1 **NIST SP 800-53 Rev. 4** RA-5

Additional resources required to develop this solution are identified in Appendix C. The core standards, secure update standards, industry best practices for software quality, and best practices for identification and authentication are generally stable, well understood, and available in the commercial off-the-shelf market. Standards associated with the MUD protocol are in an advanced level of development by the IETF.

5.3 Scenarios

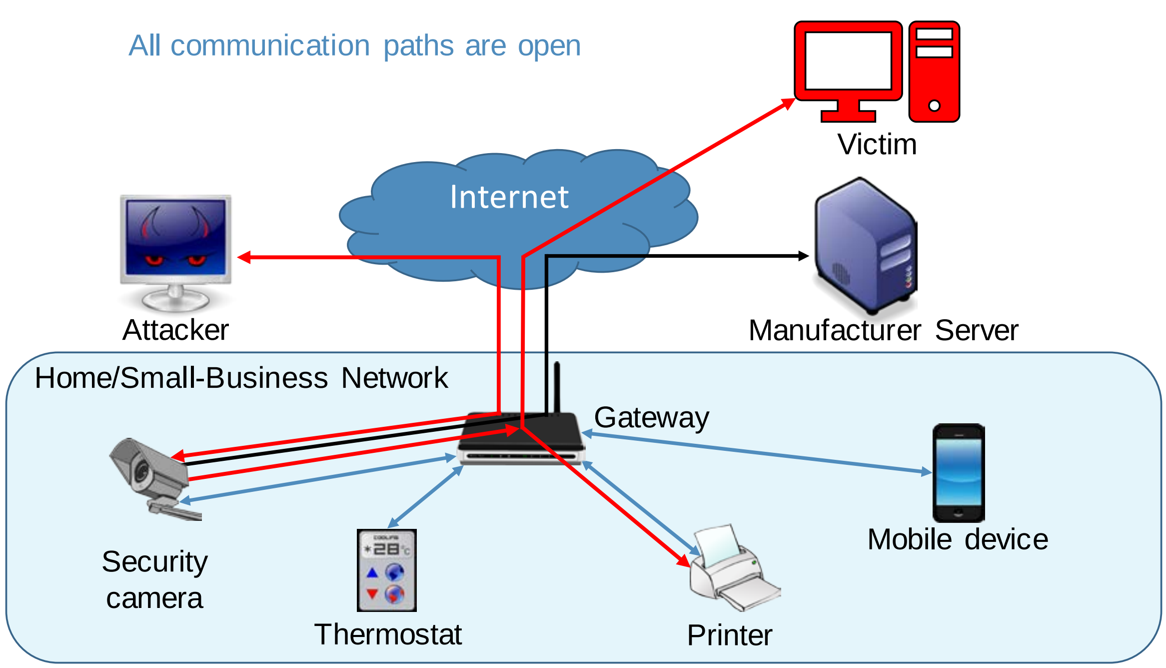
This section presents two scenarios involving home and small-business networks that have IoT devices. In the first scenario, MUD is not deployed on the network, so IoT devices are vulnerable to being port scanned and are not restricted from exchanging traffic with either external sites or other devices on the local network. IoT devices in this first scenario are highly vulnerable to attack. Threat signaling is not deployed either, so none of the devices on the local network are being protected from traffic sent from known malicious actors.

In the second scenario, both MUD and threat signaling are deployed on the network. The MUD files are being used to restrict traffic from being sent between the local IoT devices and some external internet domains (i.e., north/south traffic) as well as traffic among the local IoT devices themselves (i.e., east/west traffic). MUD ensures that each IoT device is permitted to exchange traffic with only external domains and internal devices that are explicitly specified in its MUD file. Threat signaling protects all devices, not just IoT devices, from communicating with sites that are known to be malicious.

5.3.1 Scenario 1: No MUD or Threat-Signaling Protection

In the No MUD or Threat-Signaling Protection scenario, as shown in Figure 5-1, the home/small-business network (depicted by the light blue rectangular box) does not have MUD deployed to provide security for its IoT devices, nor does it use threat signaling.

Figure 5-1 No MUD or Threat-Signaling Protection

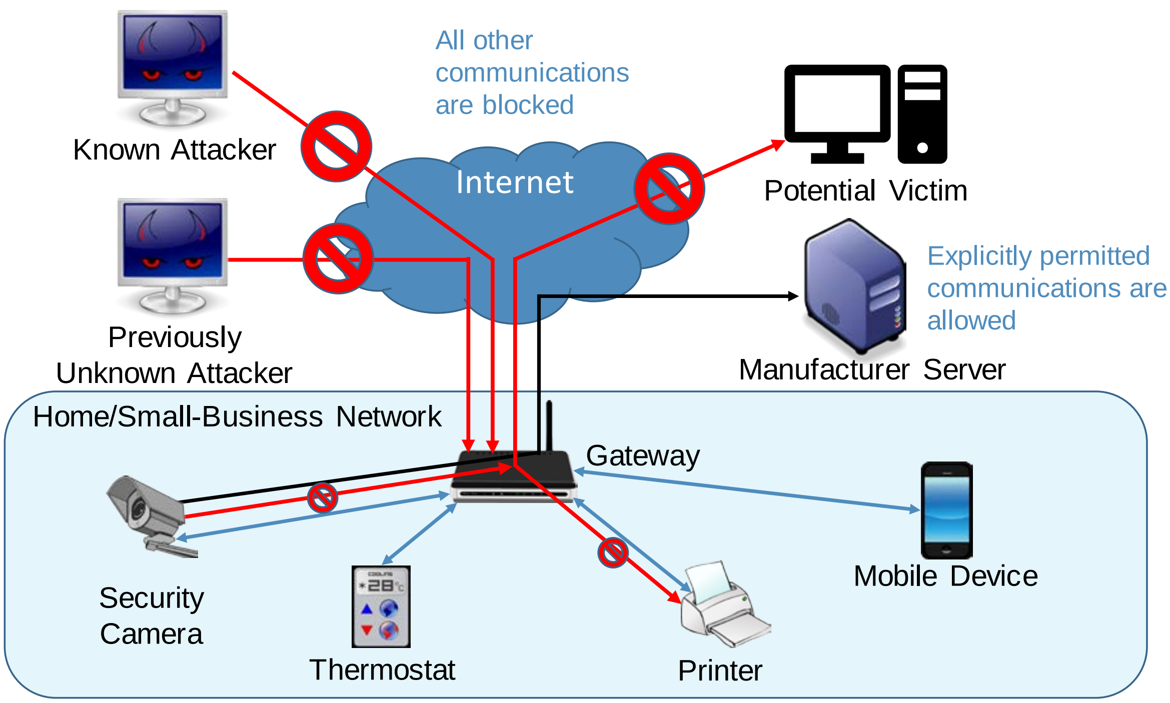


All communication paths are open. The IoT devices on the network can be port scanned (and perhaps hijacked) by an attacker on the internet. IoT devices are permitted to communicate to and from intended services, such as a manufacturer update server, as desired. However, the IoT devices are also reachable by malicious external devices and by compromised devices that are on their local network, making them vulnerable to attacks from these malicious and compromised devices. In addition, if an IoT device on the local network becomes compromised, there are no protections in place to stop it from launching an attack on outside or local devices, creating additional potential victims. As shown in Figure 5-1, an external malicious actor can attack a security camera on the local network, compromise that camera, and use it to launch additional attacks on both local and remote targets.

5.3.2 Scenario 2: MUD and Threat-Signaling Protection

In the MUD and Threat-Signaling Protection scenario, as shown in Figure 5-2, the home/small-business network (depicted by the light blue rectangle) has both MUD and threat signaling deployed. (For simplicity, the components of the MUD deployment such as the MUD manager and MUD file server are not depicted, nor are the components of the threat-signaling deployment.) The MUD file for each MUD-capable IoT device lists the domains of all external services with which the MUD-capable device is permitted to exchange traffic. All external domains that are not explicitly permitted in the MUD file are denied. Therefore, each MUD-capable IoT device on the network can freely communicate with its intended external services, but all other attempted communications between that MUD-capable IoT device and external sites are blocked. The MUD-capable IoT device cannot be port scanned or receive traffic from external malicious domains if communication with those domains is not explicitly permitted in the IoT device's MUD file, even if those domains are not known to be malicious. Furthermore, even if the MUD-capable IoT device is compromised in some way after it has connected to the local network, it will not be permitted to attack any external domains if communication with those domains is not explicitly permitted in the MUD-capable IoT device's MUD file.

Figure 5-2 MUD and Threat-Signaling Protection



In Figure 5-2, the symbol prohibiting traffic sent from the previously unknown attacker depicts the fact that MUD prevents MUD-capable devices from receiving traffic from external sites that are not listed in those devices' MUD files. The symbol prohibiting traffic sent from the security camera to the potential external victim depicts the fact that MUD prevents MUD-capable devices from sending traffic to external targets that are not explicitly permitted in their MUD files.

One of the external sites with which a MUD-capable IoT device is permitted to communicate is a manufacturer update server, from which the IoT device receives regular software updates to ensure that it installs the most recent security patches as needed.

In addition to listing external domains with which each MUD-capable device is permitted to communicate, the MUD file for each MUD-capable device restricts the local devices that each MUD-capable IoT device is permitted to exchange traffic with based on characteristics such as those devices' manufacturer or model or whether those other devices are controllers for the IoT device in question. If a local device is not from the specified manufacturer, for example, it will not be permitted to exchange traffic with the MUD-capable IoT device. So, if a device on the local network attempts to attack another device on the local network that is MUD-capable, the traffic will not be received by that MUD-capable device if the attacking device is not from a manufacturer specified in the MUD-capable device's MUD file. Conversely, if a MUD-capable IoT device becomes compromised, it will not be permitted to attack any local devices that are not from a manufacturer specified in the MUD-capable IoT device's MUD file.

In Figure 5-2, the symbol prohibiting traffic received at the printer depicts the fact that MUD prevents MUD-capable devices from receiving traffic from all local devices that are not permitted in their MUD files. The symbol prohibiting traffic sent from the security camera to the printer depicts the fact that MUD prevents MUD-capable devices from sending traffic to other local devices that are not explicitly permitted in their MUD files.

In addition to MUD, threat signaling is deployed. Threat signaling prevents all devices on the local network from communicating with external domains that are known to be malicious. It protects not just MUD-capable IoT devices but also non-MUD-capable IoT devices and fully functional devices such as cell phones and laptops. This protection is depicted in Figure 5-2 by the symbol prohibiting receipt of traffic sent from the known attacker.

6 Build 1

The Build 1 implementation uses products from Cisco Systems, DigiCert, Forescout, and Molex. Cisco equipment supports MUD. Build 1 uses the Cisco MUD manager, which is available as open-source software; and the Cisco Catalyst 3850-S switch, which has been customized to work with the MUD manager, to provide switching, DHCP, and LLDP services. Build 1 also uses the Forescout virtual appliances and enterprise manager to perform discovery of all types of devices on the network—both MUD-capable and non-MUD-capable. Build 1 uses Molex PoE Gateway and Light Engine as MUD-capable IoT devices. Build 1 also uses certificates from DigiCert.

6.1 Collaborators

Collaborators that participated in this build are described briefly in the subsections below.

6.1.1 Cisco Systems

Cisco Systems is a provider of enterprise, telecommunications, and industrial networking solutions. The work in this project was undertaken within Cisco's Enterprise Central Software Group with an eye toward improving the product offering over time. Cisco provided a proof-of-concept MUD manager as well as a Catalyst 3850-S switch with Power over Ethernet. Learn more about Cisco Systems at https://www.cisco.com.

6.1.2 DigiCert

DigiCert is a major provider of scalable TLS/secure sockets layer (SSL), and public key infrastructure (PKI) solutions for identity and encryption. The company is known for its expertise in identity and encryption for web servers and Internet of Things devices. DigiCert supports TLS/SSL and other digital certificates for PKI deployments at any scale through its certificate life-cycle management platform, CertCentral®. The company provides enterprise-grade certificate management platforms, responsive customer support, and advanced security solutions. Learn more about DigiCert at https://www.digicert.com.

6.1.3 Forescout

Forescout Technologies is an industry leader in device visibility and control. Forescout's unified security platform enables enterprises and government agencies to gain complete situational awareness of their extended enterprise environment and to orchestrate actions to reduce cyber and operational risk. Forescout products deploy quickly with agentless, real-time discovery and classification of every connected device, as well as with continuous posture assessment. As of December 31, 2019, more than 3,700 customers in over 90 countries rely on Forescout's infrastructure-agnostic solution to reduce the risk of business disruption from security incidents or breaches, to demonstrate security compliance, and to increase security operations productivity. Learn more about Forescout at https://www.forescout.com.

6.1.4 Molex

Molex brings together innovation and technology to deliver electronic solutions to customers worldwide. With a presence in more than 40 countries, Molex offers a full suite of solutions and services for many markets, including data communications, consumer electronics, industrial, automotive, commercial vehicle, and medical. Learn more about Molex at https://www.molex.com.

6.2 Technologies

Table 6-1 lists all of the products and technologies used in Build 1 and provides a mapping among the generic component term, the specific product used to implement that component, and the security Function Subcategories that the product provides. When applicable, both the Function Subcategories that a component provides directly and those that it supports but does not provide directly are listed and labeled as such. For rows in which the provides/supports distinction is not noted, the component directly provides all listed Categories. Refer to Table 5-1 for an explanation of the NIST Cybersecurity Framework Subcategory codes.

1022 **Table 6-1 Products and Technologies Used in Build 1**

Component	Product	Function	Cybersecurity Framework Subcategories
MUD manager	Cisco MUD manager (open source) and a FreeRADIUS server	Fetches, verifies, and processes MUD files from the MUD file server; configures router or switch with traffic filters to enforce access control based on the MUD file	Provides PR.PT-3 Supports ID.AM-1 ID.AM-2 ID.AM-3 PR.AC-4 PR.AC-5 PR.DS-5 DE.AE-1
MUD file server	NCCoE-hosted Apache server	Hosts MUD files; serves MUD files to the MUD manager by using https	ID.AM-1 ID.AM-2 ID.AM-3 PR.AC-4 PR.AC-5 PR.DS-5 PR.PT-3 DE.AE-1
MUD file maker	MUD file maker (https://www.mud-maker.org/)	Yet Another Next Generation (YANG) script graphical user interface (GUI) used to create MUD files	ID.AM-1
MUD file	A YANG model instance that has been serialized in JavaScript Object Notation (JSON) (RFC 7951). The manufacturer of a MUD-capable device creates that device's MUD file. MUD file maker (see previous row) can create MUD files. Each MUD file is also associated with a separate MUD signature file.	Specifies the communications that are permitted to and from a given device	Provides PR.PT-3 Supports ID.AM-1 ID.AM-2 ID.AM-3

Component	Product	Function	Cybersecurity Framework Subcategories
DHCP server	Cisco Internetwork Operating System (IOS) (Catalyst 3850-S)	Dynamically assigns IP addresses; recognizes MUD URL in DHCP DISCOVER message; should notify MUD manager if the device's IP address lease expires or has been released	ID.AM-3 PR.AC-4 PR.AC-5 PR.DS-5 PR.PT-3 DE.AE-1
LLDP	Cisco IOS (Catalyst 3850-S)	Supports capability for devices to advertise their identity and capabilities to neighbors on a local area network segment; provides capability to receive MUD URL in IoT device LLDP type-length-value (TLV) frame as an extension	ID.AM-1
Router or switch	Cisco Catalyst 3850-S (IOS XE software version 16.09.02)	Provides MUD URL to MUD manager; gets configured by the MUD manager to enforce the IoT device's communication profile; performs per-device access control	ID.AM-3 PR.AC-4 PR.AC-5 PR.DS-5 PR.PT-3 DE.AE-1
Certificates	DigiCert certificates (TLS and premium)	Authenticates MUD file server and secures TLS connection between MUD manager and MUD file server; signs MUD files and generates corresponding signature file	PR.AC-1 PR.AC-3 PR.AC-5 PR.AC-7

Component	Product	Function	Cybersecurity Framework Subcategories
MUD-capable IoT device	Raspberry Pi Model 3B (devkit) u-blox C027-G35 (devkit) Samsung ARTIK 520 (devkit) Intel UP Squared Grove (devkit) Molex PoE Gateway and Light Engine	Emits a MUD URL as part of its DHCP DISCOVER message; requests and applies software updates	ID.AM-1
Non-MUD-capable IoT device	Camera Mobile phones Connected lighting devices Connected assistant Printer Baby monitor Wireless access point Digital video recorder	Acts as typical IoT device on a network; creates network connections to cloud services	ID.AM-1
Update server	NCCoE-hosted Apache server Molex update agent	Acts as a device manufacturer's update server that would communicate with IoT devices to provide patches and other software updates	PR.IP-1 PR.IP-3
Unapproved server	NCCoE-hosted Apache server	Acts as an internet host that has not been explicitly approved in a MUD file	DE.DP-3 DE.AM-1
MQTT broker server	NCCoE-hosted MQTT server	Receives and publishes messages to/from clients	ID.AM-3 DE.AE-3
IoT device discovery	Forescout virtual appliances and enterprise manager	Discover IoT devices on network	ID.AM-1 PR.IP-1 DE.AM-1

Each of these components is described more fully in the following sections.

6.2.1 MUD Manager

The MUD manager is a key component of the architecture. It fetches, verifies, and processes MUD files from the MUD file server. It then configures the router or switch with an access list to control communications based on the contents of the MUD files.

The Cisco MUD manager is an open-source implementation. For this project, we used the Cisco MUD manager to support IoT devices that emit their MUD URLs via DHCP messages and to support other IoT devices that emit their MUD URLs via the Institute of Electrical and Electronics Engineers (IEEE) 802.1AB LLDP. The Cisco MUD manager is supported by an open-source implementation of an authentication, authorization, and accounting (AAA) server that communicates by using the Remote Authentication Dial-In User Service (RADIUS) protocol (i.e., a RADIUS server) called FreeRADIUS. When the MUD URL is emitted via DHCP or LLDP, it is extracted from the corresponding message, and the switch thereafter provides these MUD URLs to the MUD manager via RADIUS messages. The MUD manager then retrieves MUD files associated with those URLs and configures the Catalyst 3850-S switch to enforce the IoT devices' communication profiles based on these MUD files. The switch implements an IP access control list-based policy for src-dnsname, dst-dnsname, my-controller, and controller constructs that are specified in the MUD file, and it uses virtual local area networks (VLANs) to enforce same-manufacturer, manufacturer, and local-networks constructs that are specified in the MUD file. The system supports both lateral east/west protection and appropriate access to internet sites (north/south protection).

When supporting MUD URL emission by LLDP TLV, LLDP TLV must be enabled on both the Cisco switch and the IoT device. A policy-map configuration and a corresponding template are used to cause media access control (MAC) authentication bypass to happen. This will trigger an access-session attribute that will cause LLDP TLVs (including the MUD URL) to be forwarded in an accounting message to the RADIUS server.

Some manual preconfiguration of VLANs on the switch is required. The Cisco MUD manager supports a default policy for IPv4. It implements a static mapping between domain names and IP addresses inside a configuration file.

The version of the Cisco MUD manager used in this project is a proof-of-concept implementation that is intended to introduce advanced users and engineers to the MUD concept. It is not a fully automated MUD manager implementation, and some protocol features are not present. These are described in Section 10.1, Findings.

6.2.2 MUD File Server

In the absence of a commercial MUD file server for this project, the NCCoE implemented its own MUD file server by using an Apache web server. This file server signs and stores the MUD files along with their corresponding signature files for the IoT devices used in the project. Upon receiving a GET request for the MUD files and signatures, it serves the request to the MUD manager by using https.

6.2.3 MUD File

Using the MUD file maker component referenced above in Table 6-1, it is possible to create a MUD file with the following contents:

- internet communication class—access to cloud services and other specific internet hosts:
    - host: updateserver (hosted internally at the NCCoE)
        - protocol: TCP
        - direction-initiated: from IoT device
        - source port: any
        - destination port: 80
- controller class—access to **classes** of devices that are known to be controllers (could describe well-known services such as DNS or NTP):
    - host: mqttbroker (hosted internally at the NCCoE)
        - protocol: TCP
        - direction-initiated: from IoT device
        - source port: any
        - destination port: 1883
- local-networks class—access to/from **any** local host for specific services (e.g., Hypertext Transfer Protocol [http] or Hypertext Transfer Protocol Secure [https]):
    - host: any
        - protocol: TCP
        - direction-initiated: from IoT device
        - source port: any
        - destination port: 80
- my-controller class—access to controllers specific to this device:
    - controllers: null (to be filled in by the network administrator)
        - protocol: TCP
        - direction-initiated: from IoT device
        - source port: any
        - destination port: 80
- same-manufacturer class—access to devices of the same manufacturer:

    - same-manufacturer: null (to be filled in by the MUD manager]
        - protocol: TCP
        - direction-initiated: from IoT device
        - source port: any
        - destination port: 80
- manufacturer class—access to devices of a specific manufacturer (identified by MUD URL):
    - manufacturer: devicetype (URL decided by the device manufacturer)
        - protocol: TCP
        - direction-initiated: from IoT device
        - source port: any
        - destination port: 80

6.2.4 Signature File

According to the IETF MUD specification, “a MUD file MUST be signed using Cryptographic Message Syntax (CMS) as an opaque binary object.” The MUD file *(ciscopi2.json)* was signed with the OpenSSL tool by using the command described in the specification (which is in Volume C of this publication). A Premium Certificate, requested from DigiCert, was leveraged to generate the signature file *(ciscopi2.p7s).* Once created, the signature file is stored on the MUD file server.

6.2.5 DHCP Server

The DHCP server in the architecture is MUD-capable. In addition to dynamically assigning IP addresses, it recognizes the DHCP option (161) and extracts the MUD URL from the IoT device’s DHCP message. The MUD URL is provided to the MUD manager. The DHCP server is typically embedded in a router/switch. This project uses the DHCP server that is embedded in the Cisco Catalyst 3850-S.

Cisco IOS provides a basic DHCP server that is useful in small-/medium-business and home network environments, where centralized address management is not required. As described in the previous section, the DHCP server in this case is configured to allocate addresses for the test network, provide a default router, and configure a domain name server. It is **not** used to deliver MUD URLs to the MUD manager.

6.2.6 Link Layer Discovery Protocol

The Cisco Catalyst 3850-S switch also supports a MUD-capable version of the LLDP that provides the MUD URL in the LLDP TLV frame as an extension. When a MUD-capable IoT device uses LLDP to convey its MUD URL, the Cisco Catalyst 3850-S extracts the MUD URL from the LLDP frame and provides it to the MUD manager via a RADIUS message.

6.2.7 Router/Switch

This project uses the Cisco Catalyst 3850-S switch. The Cisco Catalyst 3850-S is an enterprise-class layer 3 switch capable of Universal PoE for digital building solutions. The optional PoE feature means it can be configured to supply power to capable devices over Ethernet through its ports. In addition to providing DHCP services, the switch acts as a broker for connected IoT devices for AAA through the FreeRADIUS server. The LLDP is enabled on ports that MUD-capable devices are plugged into to help facilitate recognition of connected IoT device features, capabilities, and neighbor relationships at layer 2. Additionally, an access session policy is configured on the switch to enable port control for multihost authentication and port monitoring. The combined effect of these switch configurations is a dynamic access list, which has been generated by the MUD manager, being active on the switch to permit or deny access to and from MUD-capable IoT devices. The version of the Cisco Catalyst switch used in this project is a proof-of-concept implementation that is intended to introduce advanced users and engineers to the MUD concept. Some protocol features are not present. These are described in Section 10.1, Findings.

6.2.8 Certificates

DigiCert’s CertCentral web-based platform allows provisioning and managing publicly trusted X.509 certificates for TLS and code signing as well as a variety of other purposes. After establishing an account, clients can log in, request, renew, and revoke certificates by using only a browser. Multiple roles can be assigned within an account, and a discovery tool can inventory all certificates within the enterprise. In addition to certificate-specific features, the platform offers baseline enterprise software-as-a-service capabilities, including role-based access control, Security Assertion Markup Language, single sign-on, and security policy management and enforcement. All account features come with full parity between the web portal and a publicly available API. For this implementation, two certificates were provisioned: a private TLS certificate for the MUD file server to support the https connection from the MUD manager to the MUD file server, and a Premium Certificate for signing the MUD files.

6.2.9 IoT Devices

This section describes the IoT devices used in the laboratory implementation. There are two distinct categories of devices: devices that can emit a MUD URL in compliance with the MUD specification, i.e., MUD-capable IoT devices; and devices that are not capable of emitting a MUD URL in compliance with the MUD specification, i.e., non-MUD-capable IoT devices.

6.2.9.1 MUD-Capable IoT Devices

The project used several MUD-capable IoT devices: NCCoE Raspberry Pi (devkit), u-blox C027-G35 (devkit), Samsung ARTIK 520 (devkit), Intel UP Squared Grove (devkit), Molex PoE Gateway, and Molex Light Engine. The NCCoE modified the devkits to simulate IoT devices. All of the MUD-capable IoT devices demonstrate the ability to emit a MUD URL as part of a DHCP transaction or LLDP message and to request and apply software updates.

6.2.9.1.1 Molex PoE Gateway and Light Engine

Molex developed this set of IoT devices. The PoE Gateway acts as a network end point and manages lights, sensors, and other devices. One of the devices managed by the PoE Gateway is a light engine that Molex provided.

6.2.9.1.2 NCCoE Raspberry Pi (Devkit)

The Raspberry Pi devkit runs the Raspbian 9 operating system. It is configured to include a MUD URL that it emits during a typical DHCP transaction. The NCCoE developed a Python script that allowed the Raspberry Pi to receive and process on and off commands by using the MQTT protocol, which were sent to the light-emitting diode (LED) bulb connected to the Raspberry Pi.

6.2.9.1.3 NCCoE u-blox C027-G35 (Devkit)

The u-blox C027-G35 devkit runs the Arm Mbed operating system. The NCCoE modified several of the Mbed-OS libraries to configure the devkit to include a MUD URL that it emits during a typical DHCP transaction. The u-blox devkit is also configured to initiate network connections to test network traffic throughout the MUD process.

6.2.9.1.4 NCCoE Samsung ARTIK 520 (Devkit)

The Samsung ARTIK 520 devkit runs the Fedora 24 operating system. It is configured to include a MUD URL that it emits during a typical DHCP transaction. The same Python script mentioned earlier was used to simulate a connected lock. This Python script allowed the ARTIK devkit to receive on and off commands by using the MQTT protocol.

6.2.9.1.5 NCCoE Intel UP Squared Grove (Devkit)

The Intel UP Squared Grove devkit runs the Ubuntu 16.04 LTS operating system. It is configured to include a MUD URL that it emits during a typical DHCP transaction. The same Python script mentioned earlier was used to simulate a connected lighting device. This allowed the UP Squared Grove devkit to receive on and off commands by using the MQTT protocol.

6.2.9.2 Non-MUD-Capable IoT Devices

The laboratory implementation also includes a variety of legacy, non-MUD-capable IoT devices that are not capable of emitting a MUD URL. These include cameras, mobile phones, lighting, a connected assistant, a printer, a baby monitor, a wireless access point, and a digital video recorder (DVR).

6.2.9.2.1 Cameras

The three cameras utilized in the laboratory implementation are produced by two different manufacturers. They stream video and audio either to another device on the network or to a cloud service. These cameras are controlled and managed by a mobile phone.

6.2.9.2.2 Mobile Phones

Two types of mobile phones are used for setting up, interacting with, and controlling IoT devices.

6.2.9.2.3 Lighting

Two types of connected lighting devices are used in the laboratory implementation. These connected lighting components are controlled and managed by a mobile phone.

6.2.9.2.4 Connected Assistant

A connected assistant is utilized in the laboratory implementation. The device demonstrates and tests the wide range of network traffic generated by a connected assistant.

6.2.9.2.5 Printer

A connected printer is connected to the laboratory network wirelessly to demonstrate connected printer usage.

6.2.9.2.6 Baby Monitor

A baby monitor with remote control plus video and audio capabilities is connected wirelessly to the laboratory network. This baby monitor is controlled and managed by a mobile phone.

6.2.9.2.7 Wireless Access Point

A connected wireless access point is used in the laboratory implementation to demonstrate the network activity and functionality of this type of device.

6.2.9.2.8 Digital Video Recorder

A connected DVR is connected to the laboratory implementation network. This is also controlled and managed by a mobile phone.

6.2.10 Update Server

The update server is designed to represent a device manufacturer or trusted third-party server that provides patches and other software updates to the IoT devices. This project used an NCCoE-hosted update server that provides faux software update files.

6.2.10.1 NCCoE Update Server

The NCCoE implemented its own update server by using an Apache web server. This file server hosts faux software update files to be served as software updates to the IoT device devkits. When the server receives an http request, it sends the corresponding faux update file.

6.2.10.2 Molex Update Agent

The process for updating the firmware on a Molex PoE Gateway is currently manual, with the firmware update taking place over the Constrained Application Protocol, UDP, and Trivial File transfer Protocol. The update process is initiated by an update agent on the local network connecting to the PoE Gateway and sending the firmware update information.

6.2.11 Unapproved Server

The NCCoE implemented its own unapproved server by using an Apache web server. This web server acts as an unapproved internet host, i.e., an internet host that is not explicitly approved in the MUD file. This was created to test the communication between a MUD-capable IoT device and an internet host that is not included in the MUD file and should thus be denied. To verify that the traffic filters were applied as expected, we tested communication to and from the unapproved server and the MUD-capable IoT device.

6.2.12 MQTT Broker Server

The NCCoE implemented an MQTT broker server by using the open-source tool Mosquitto. The server communicates messages among multiple clients. For this project, it allows mobile devices to set up with the appropriate application to communicate with the MQTT-enabled IoT devices in the build. The messages exchanged by the devices are on and off messages, which allow the mobile device to control the LED light on the IoT device.

6.2.13 IoT Device Discovery

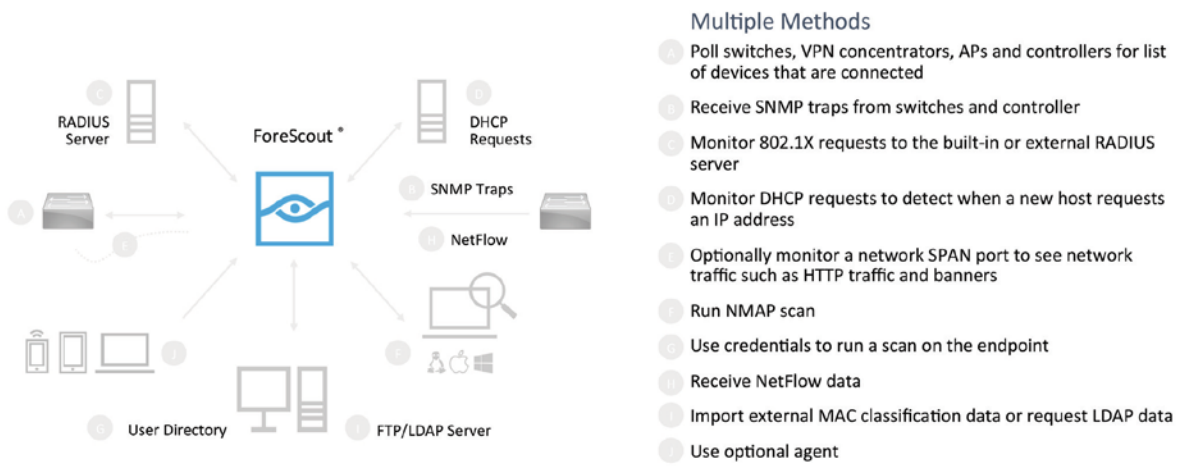
This project uses Forescout appliance and enterprise manager to provide an IoT device discovery service for the demonstration network. The Forescout appliance can discover, inventory, profile, and classify all attached devices to validate that the access that is being granted to each device is consistent with that device's type. Forescout can also continuously monitor the actions of these assets as they join and leave the network. While Forescout provides a wide range of data collection capabilities, items this project focuses on include:

- device information
  - device type
  - manufacturer
  - connection type
  - hardware information
  - MAC and IP addresses
  - operating system

        - network services
  - network configuration
    - wired or wireless

The Forescout appliance detects IoT devices in real time as they connect to the network. It uses both passive monitoring and integration with the network infrastructure. As a device connects to the network, Forescout may learn about that device via a variety of different techniques to discover and classify it without requiring agents, as shown in Figure 6-1. The methods demonstrated in this project include Forescout passive discovery of devices by using switch polling, importation of MAC classification data, and TCP fingerprinting. Due to the passive nature of the device discovery, neither performance nor reliability of the IoT devices is impacted.

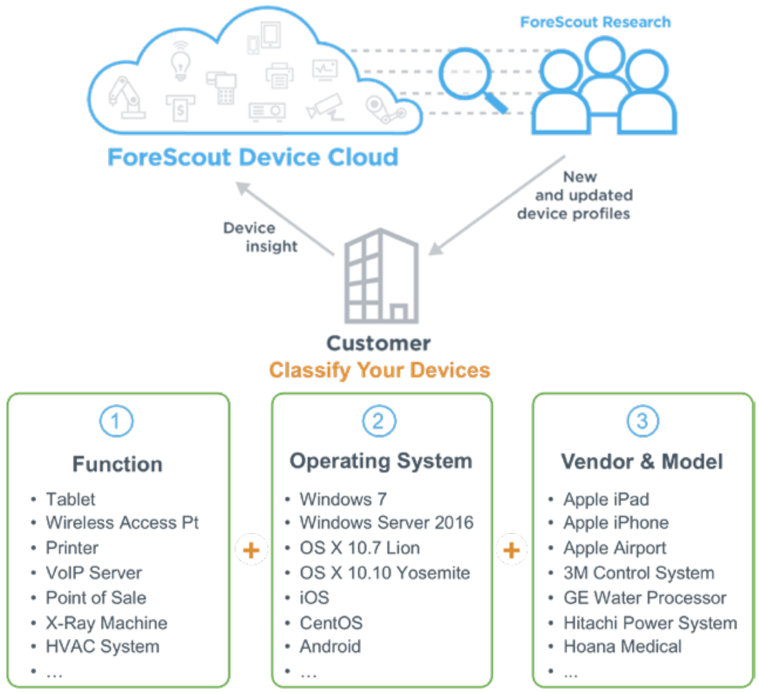
Figure 6-1 Methods the Forescout Platform Can Use to Discover and Classify IP-Connected Devices



Forescout is deployed as virtual appliances on the NCCoE laboratory network and managed by a single enterprise manager. After discovering IoT devices and collecting relevant information, classification is the next step.

To automatically classify discovered devices, the Forescout platform includes Forescout Device Cloud. Device Cloud allows users to benefit from crowdsourced device insight to auto-classify their devices, as shown in Figure 6-2. It also auto-classifies the devices by their type and function, operating system and version, and manufacturer and model. Users can leverage new and updated auto-classification profiles published by Forescout. In addition, they can create custom classification policies to auto-classify devices unique to their environments. At this writing, the Forescout appliance cannot identify whether an IoT device on the network is MUD-capable.

Figure 6-2 Classify IoT Devices by Using the Forescout Platform



6.3 Build Architecture

In this section we present the logical architecture of Build 1 relative to how it instantiates the reference architecture depicted in Figure 4-1. We also describe Build 1's physical architecture and present message flow diagrams for some of its processes.

6.3.1 Logical Architecture

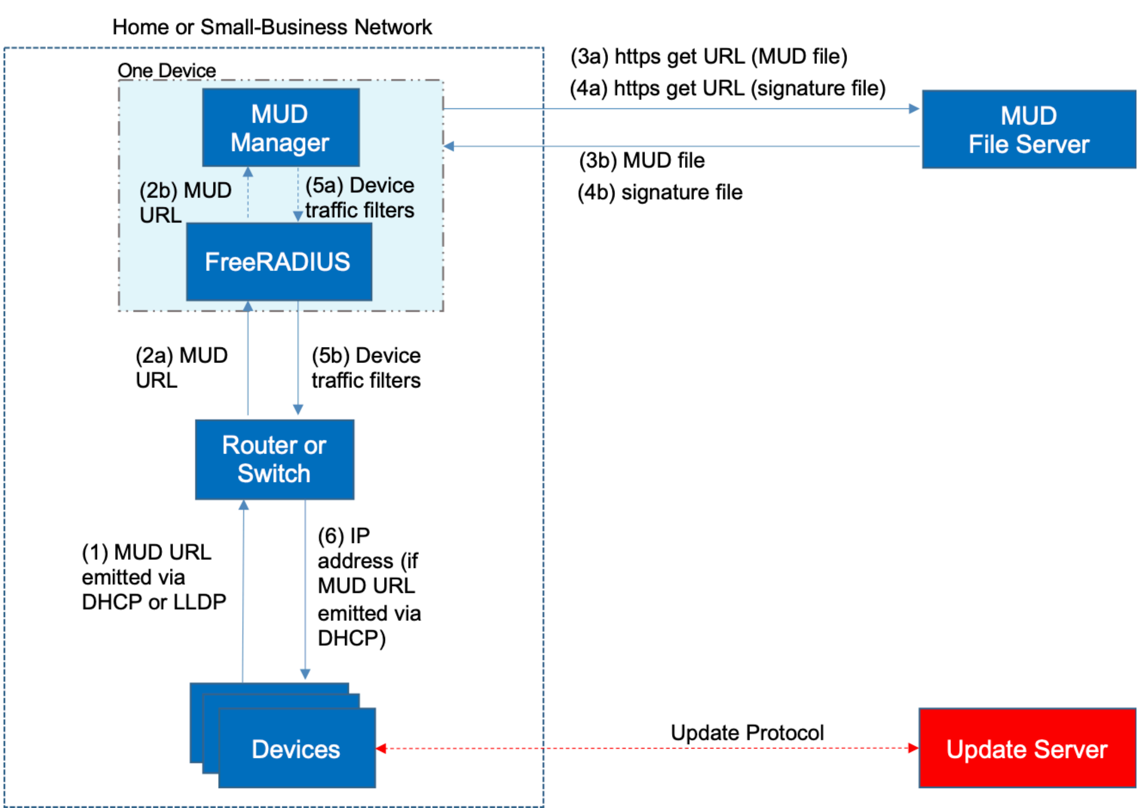
Figure 6-3 depicts the logical architecture of Build 1. Figure 6-3 uses numbered arrows to depict in detail the flow of messages needed to support installation of MUD-based access control rules for a MUD-capable device. Build 1 was designed with a single device serving as the MUD manager and FreeRADIUS server that interfaces with the Catalyst 3850-S switch over TCP/IP. It supports two mechanisms for MUD URL emission: DHCP and LLDP. Figure 6-3 depicts only the steps performed when using DHCP emission. The Catalyst 3850-S switch contains a DHCP server that is configured to extract MUD URLs from IPv4 DHCP transactions.

- Upon connecting a MUD-capable device, the MUD URL is emitted via either DHCP or LLDP (step 1).
- The Catalyst 3850-S switch sends the MUD URL to the FreeRADIUS server (step 2a); this is passed from the FreeRADIUS server to the MUD manager (step 2b).

- Once the MUD URL is received, the MUD manager uses this URL to fetch the MUD file from the MUD file server (step 3a); if successful, the MUD file server at the specified location will serve the MUD file (step 3b).
- Next, the MUD manager requests the signature file associated with the MUD file (step 4a) and upon receipt (step 4b) verifies the MUD file by using its signature file.
- Once the MUD file has been verified successfully, the MUD manager passes the device's traffic filters to the FreeRADIUS server (step 5a), which in turn sends the device's traffic filters to the router or switch, where they are applied (step 5b).
- The device is finally assigned an IP address (step 6).

Once the device's traffic filters are applied to the router or switch, the MUD-capable IoT device will be able to communicate with approved local hosts and internet hosts as defined in the MUD file, and any unapproved communication attempts will be blocked.

Figure 6-3 Logical Architecture–Build 1

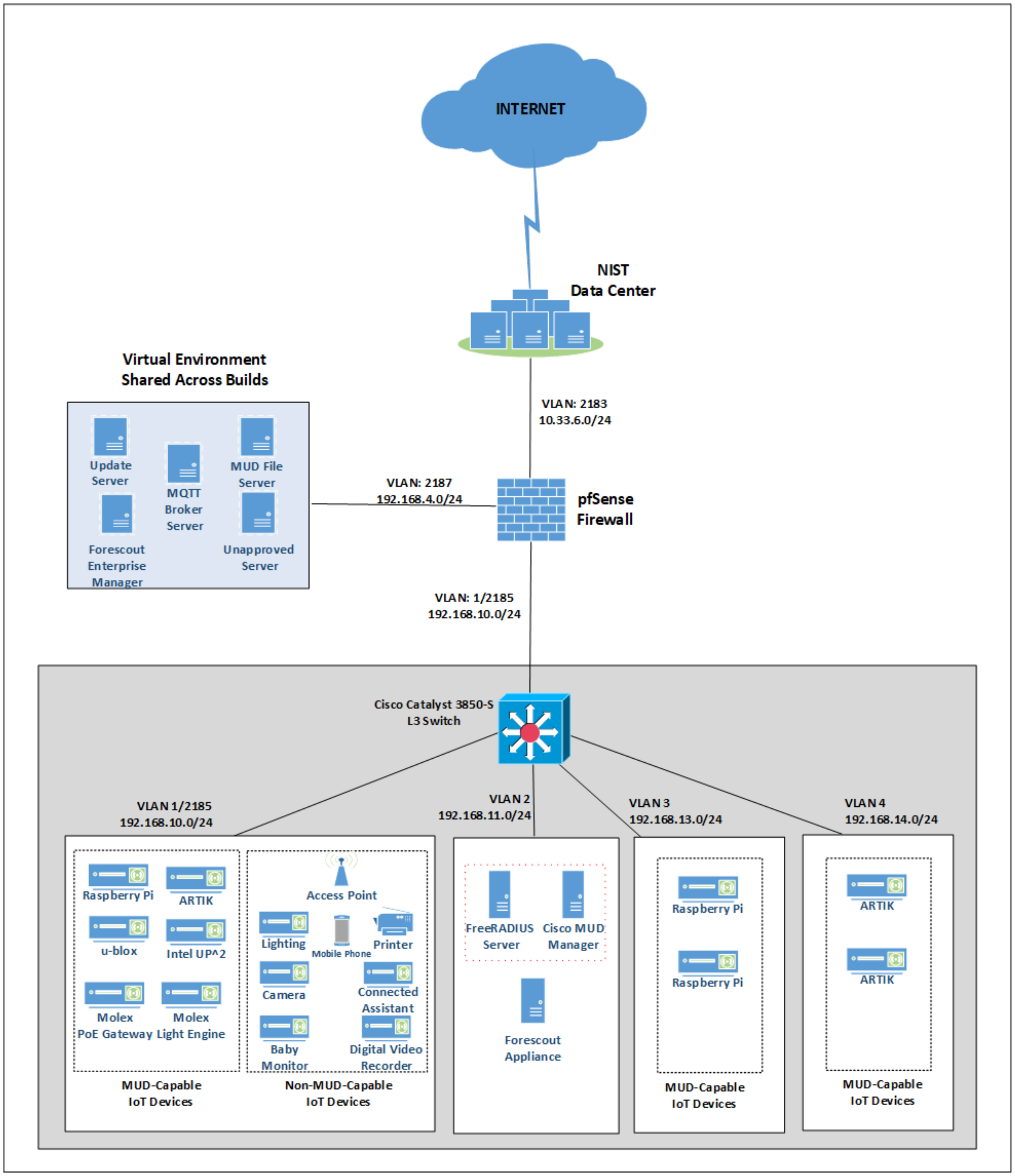


6.3.2 Physical Architecture

Figure 6-4 describes the physical architecture of Build 1. The Catalyst 3850-S switch is configured to host four VLANs. The first VLAN, VLAN 1, hosts many IoT devices. Three separate instances of DHCP servers are configured for VLANs 1, 3, and 4 to dynamically assign IPv4 addresses to each IoT device that connects to the switch on each of these VLANs. VLAN 2 is configured on the Catalyst switch to host the Cisco MUD manager, the FreeRADIUS server, and the Forescout appliance. VLAN 3 and VLAN 4 are configured to host IoT devices from the same manufacturer. Specifically, VLAN 3 hosts two Raspberry Pi devices, while VLAN 4 hosts two u-blox devices. The network infrastructure as configured utilizes the IPv4 protocol for communication both internally and to the internet.

In addition, Build 1 utilized a portion of the virtual environment that was shared across builds. Services hosted in this environment included an update server, MUD file server, MQTT broker, Forescout enterprise manager, and unapproved server.

1312 **Figure 6-4 Physical Architecture–Build 1**



A full description of Cisco's proof-of-concept MUD manager implementation is at https://github.com/CiscoDevNet/MUD-Manager. The Cisco MUD manager is built as a callout from FreeRADIUS and uses MongoDB to store policy information. The MUD manager is configured from a JSON file that will vary slightly based on the installation. This configuration file provides several static bindings and directives as to whether both egress and ingress ACLs should be applied, and it identifies the definition of the local network class on the network.

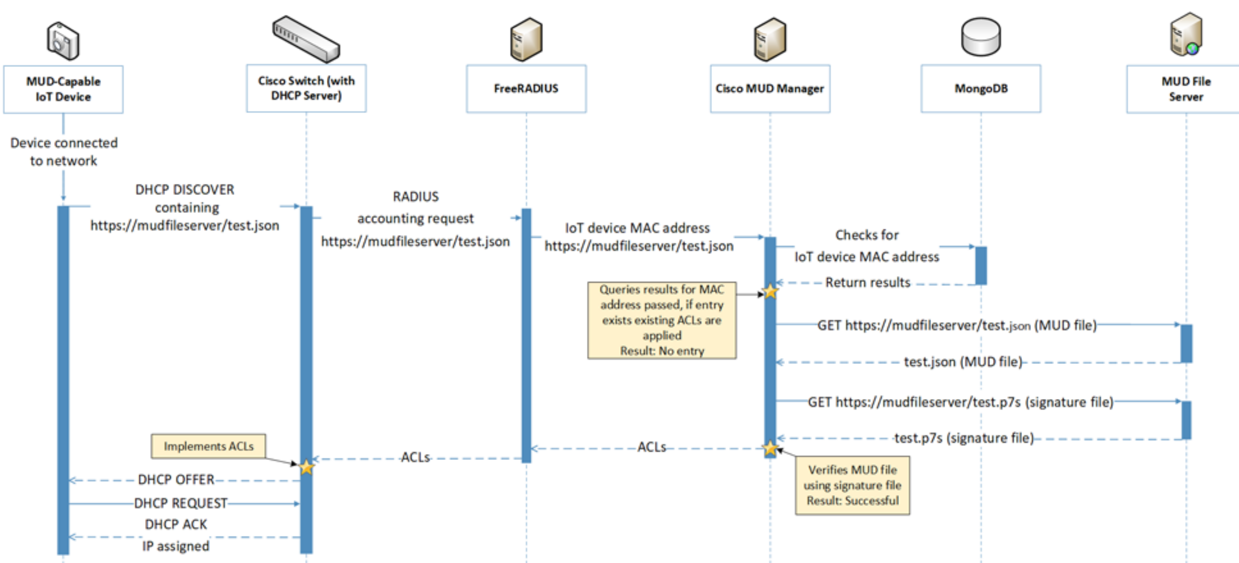
6.3.3 Message Flow

This section presents the message flows used in Build 1 during several different processes of note.

6.3.3.1 Installation of MUD-Based Access Control Rules for MUD-Capable Devices

Figure 6-5 shows the message flow of the process of installing access control rules for a MUD-capable IoT device that emits a MUD URL via DHCPv4.

Figure 6-5 MUD-Capable IoT Device MUD-Based ACL Installation Message Flow–Build 1



As shown in Figure 6-5, the message flow is as follows:

- A MUD-capable IoT device is connected to the network.
- The MUD-capable IoT device begins a DHCPv4 transaction in which DHCP option 161, the Internet Assigned Numbers Authority (IANA)-assigned value for MUD, is transmitted as part of a DHCP DISCOVER message. It is possible to transmit the option in both DISCOVER and REQUEST messages.

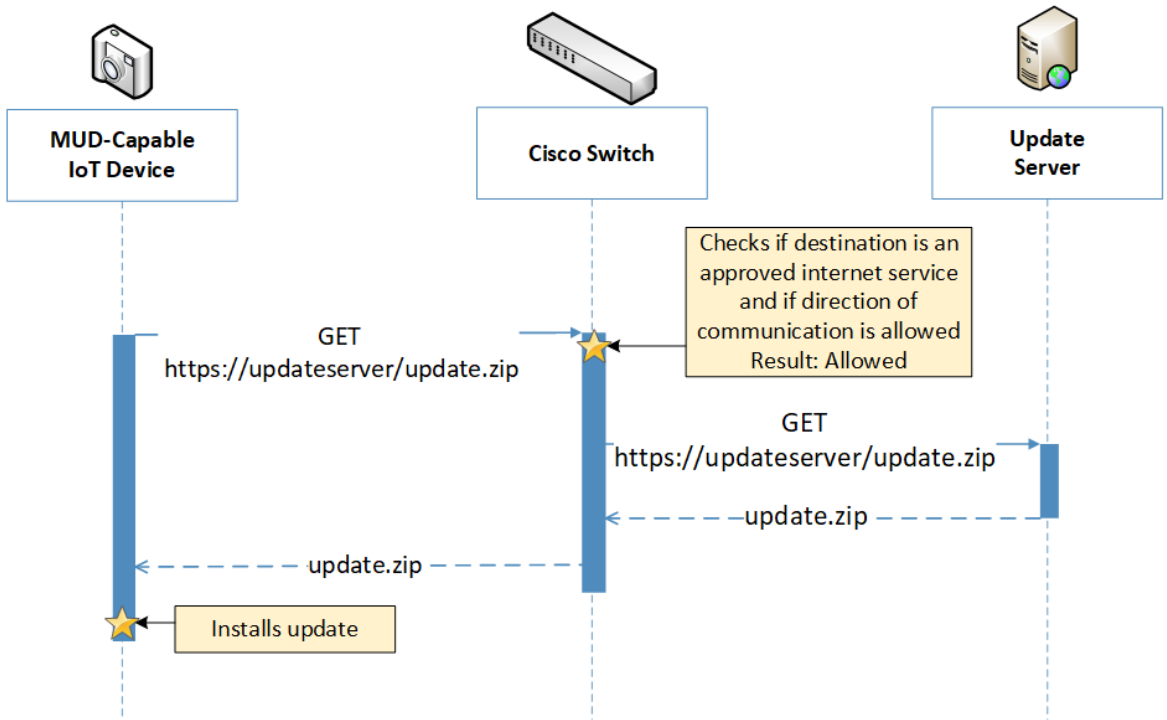
- The DHCP server on the Cisco switch recognizes that option and extracts the MUD URL from the DHCP message, which is sent from the switch to the FreeRADIUS server in the associated accounting request. From this point, the FreeRADIUS server sends the MAC address and MUD URL for the newly connected device to the MUD manager.
- Next, the MUD manager does a query for the MAC address in its database, searching for any cached MUD files associated with the MAC address and MUD URL. If an entry does not exist, as depicted in the figure, the MUD manager fetches the MUD file and signature file from the MUD file server.
- The MUD manager verifies the MUD file with the corresponding signature file and translates the contents into ACLs, which are passed through the FreeRADIUS server to the Cisco switch, where they are applied.
- The MUD-capable IoT device is assigned an IP address and is ready to be used on the network. When the MUD-capable IoT device is in use, access of all traffic to and from the IoT device is controlled by the Cisco switch, which will enforce the MUD ACLs for that device.

As an example, the subsections below address several different types of traffic that might apply to an IoT device. The message flow diagram in each subsection shows how this traffic would interact with Build 1’s infrastructure.

6.3.3.2 Updates

After a device has been permitted to connect to the home/small-business network, it should periodically check for updates. The message flow for updating the IoT device is shown in Figure 6-6 Update Process Message Flow–Build 1.

Figure 6-6 Update Process Message Flow–Build 1



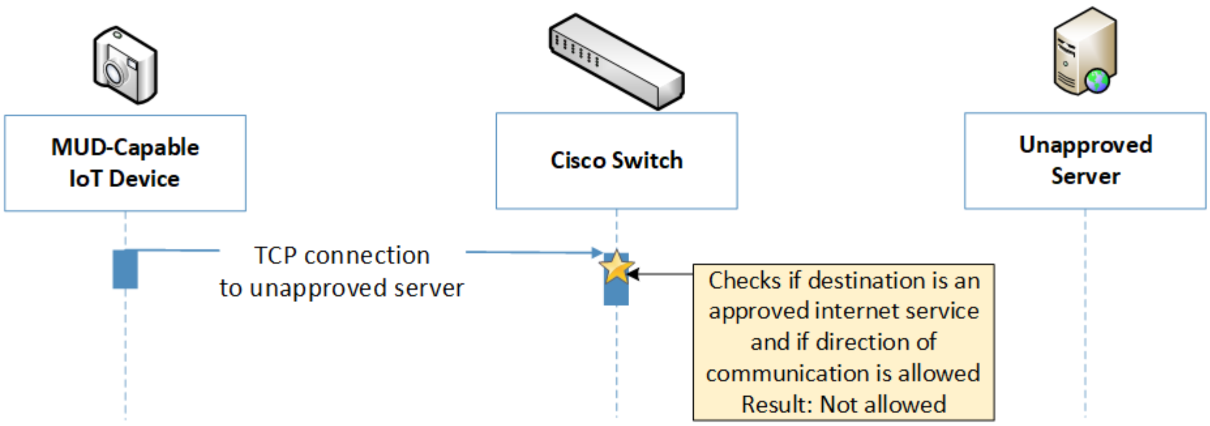
As shown in Figure 6-6 Update Process Message Flow–Build 1, the message flow is as follows:

- A MUD-capable IoT device initiates an https request to the update server.
- The Cisco switch checks its ACLs to determine if the destination and direction of communication should be allowed for the IoT device, and the switch allows the request after verification.
- The update server completes the process by sending the requested update package to the IoT device.

6.3.3.3 Prohibited Traffic

Figure 6-7 shows the message flows used to handle prohibited traffic in Build 1's infrastructure.

Figure 6-7 Prohibited Traffic Message Flow–Build 1



As shown in Figure 6-7, when an IoT device attempts to send traffic to an external domain, the message flow is as follows:

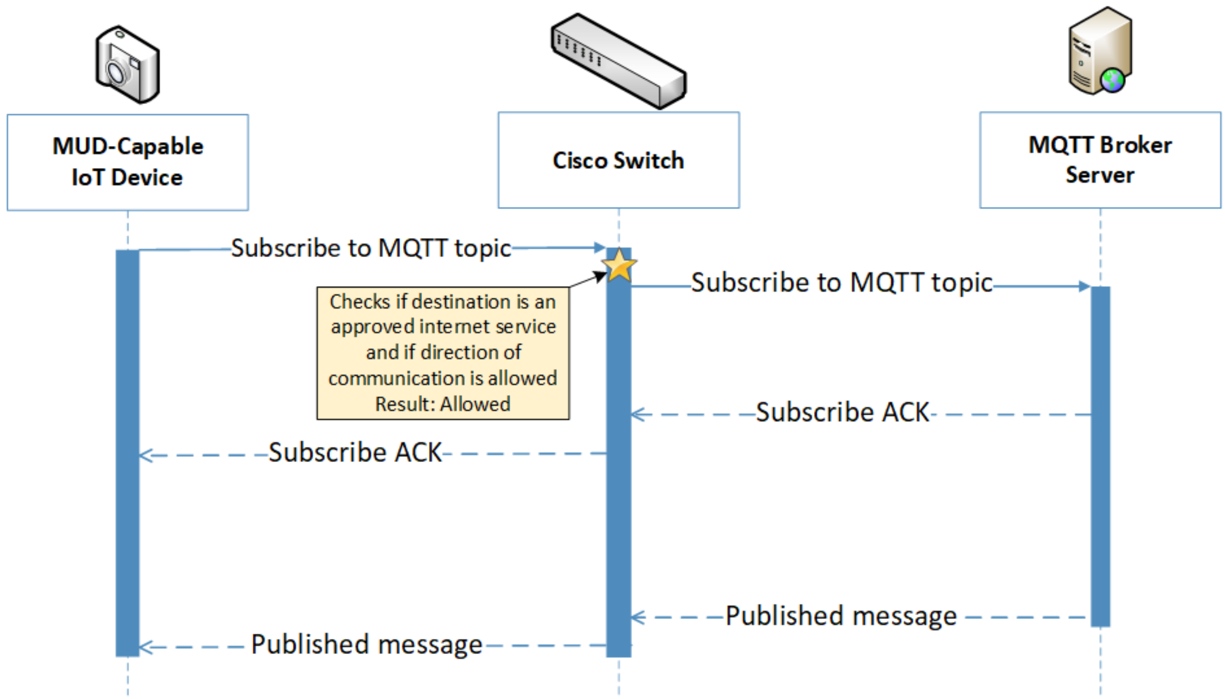
- The MUD-capable IoT device initiates a TCP request to an unapproved server.
- The Cisco switch checks its ACLs to determine if the destination and direction of communication should be allowed for the IoT device, and the switch blocks the unapproved communication.

At publication time, ingress access control was not yet supported in Build 1. That is, if an unapproved server attempts to send traffic to an IoT device on the local network, this traffic will currently not be blocked. However, responses from the IoT device will still be blocked. Specifics are in Section 10.1, Findings.

6.3.3.4 MQTT Protocol Example

Figure 6-8 shows the message flows used to handle MQTT communication in Build 1's infrastructure.

Figure 6-8 MQTT Protocol Process Message Flow–Build 1



As shown in Figure 6-8, the message flow is as follows:

- The MUD-capable IoT device initiates a Subscribe message to the MQTT broker.
- The Cisco switch checks its ACLs to determine if the destination and direction of communication should be allowed for the IoT device, and the switch allows the Subscribe message after verification.
- The MQTT broker server sends a Subscribe Acknowledgement (ACK) to the IoT device.
- The MQTT broker server sends a Published message to the IoT device.

6.4 Functional Demonstration

A functional evaluation and a demonstration of Build 1 were conducted that involved two types of activities:

- Evaluation of conformance to the MUD RFC. We tested Build 1 to determine the extent to which it correctly implements basic functionality defined within the MUD RFC.
- Demonstration of additional (non-MUD-related) capabilities. It did not verify the example implementation's behavior for conformance to a standard or specification or any other expected set of capabilities; rather, it demonstrated advertised capabilities of the example implementation related to its ability to increase device and network security in ways that are

independent of the MUD RFC. These capabilities may provide security for both non-MUD-capable and MUD-capable devices. Examples of this type of activity include device discovery, attribute identification, and monitoring.

Table 6-2 summarizes the tests that we performed to evaluate Build 1's MUD-related capabilities, and Table 6-3 summarizes the exercises that we performed to demonstrate Build 1's non-MUD-related capabilities. Both tables list each test or exercise identifier, the test or exercise's expected and observed outcomes, and the applicable Cybersecurity Framework Subcategories and NIST SP 800-53 controls for which each test or exercise was designed to verify support. We detailed the tests and exercises listed in the table in a separate supplement for functional demonstration results. Boldface text highlights the gist of the information being conveyed.

Table 6-2 Summary of Build 1 MUD-Related Functional Tests

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
IoT-1	**ID.AM-1:** Physical devices and systems within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-2:** Software platforms and applications within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **DE.AE-1:** A baseline of network operations and expected data flows for users and systems is established and managed.	A **MUD-capable IoT device is configured to emit a MUD URL within a DHCP message.** The DHCP server extracts the MUD URL, which is sent to the MUD manager. The MUD manager requests the MUD file and signature from the MUD file server, and the MUD file server serves the MUD file to the MUD manager. The MUD file explicitly permits traffic to/from some internet services and hosts and implicitly denies traffic to/from all other internet services. **The MUD manager translates the MUD file information**	Upon connection to the network, the MUD-capable IoT device has its MUD **policy enforcement point (PEP) router/switch automatically configured according to the MUD file's route-filtering policies.**	Pass

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	PR.AC-4: Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC-5, AC-6, AC-14, AC-16, AC-24 **PR.AC-5:** Network integrity is protected, incorporating network segregation where appropriate. **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7 **PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10 **PR.IP-3:** Configuration change control processes are in place. **NIST SP 800-53 Rev. 4** CM-3, CM-4, SA-10 **PR.PT-3:** The principle of least functionality is incorporated by configuring systems to provide only essential capabilities. **NIST SP 800-53 Rev. 4** AC-3, CM-7 **PR.DS-2:** Data in transit is protected.	**into local network configurations that it installs on the router or switch that is serving as the MUD PEP for the IoT device.**		
IoT-2	**PR.AC-7:** Users, devices, and other assets are authenticated (e.g., single-factor, multifactor)	A MUD-capable IoT device is configured to emit a URL for a MUD	When the MUD-capable IoT device is connected	Pass

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	commensurate with the risk of the transaction (e.g., individuals' security and privacy risks and other organizational risks). **NIST SP 800-53 Rev. 4** AC-7, AC-8, AC-9, AC-11, AC-12, AC-14, IA-1, IA-2, IA-3, IA-4, IA-5, IA-8, IA-9, IA-10, IA-11	file, but the **MUD file server that is hosting that file does not have a valid TLS certificate. Local policy has been configured to ensure that if the MUD file for an IoT device is located on a server with an invalid certificate, the router/switch will be configured to deny all communication to/from the device.**	to the network, the MUD manager sends locally defined policy to the router/switch that handles whether to allow or block traffic to the MUD-capable IoT device. Therefore, the **MUD PEP router/switch will be configured to block all traffic to and from the IoT device.**	
IoT-3	**PR.DS-6:** Integrity-checking mechanisms are used to verify software, firmware, and information integrity. **NIST SP 800-53 Rev. 4** SI-7	A MUD-capable IoT device is configured to emit a URL for a MUD file, but the **certificate that was used to sign the MUD file had already expired at signing. Local policy has been configured to ensure that if the MUD file for a device has a signature that was signed by a certificate that had already expired at the time of signature, the device's MUD PEP router/switch will be configured to deny all**	When the MUD-capable IoT device is connected to the network and the MUD file and signature are fetched, the MUD manager will detect that the MUD file's signature was created by using a certificate that had already expired at signing. According to local policy, the **MUD PEP will be configured to block all traffic**	Pass

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
		communication to/from the device.	**to/from the device.**	
IoT-4	**PR.DS-6:** Integrity-checking mechanisms are used to verify software, firmware, and information integrity. **NIST SP 800-53 Rev. 4** SI-7	A MUD-capable IoT device is configured to emit a URL for a MUD file, but the **signature of the MUD file is invalid. Local policy has been configured to ensure that if the MUD file for a device is invalid, the router/switch will be configured to deny all communication to/from the IoT device.**	When the MUD-capable IoT device is connected to the network, the MUD manager sends locally defined policy to the router/switch that handles whether to allow or block traffic to the MUD-capable IoT device. Therefore, the **MUD PEP router/switch will be configured to block all traffic to and from the IoT device.**	Pass
IoT-5	**ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating	Test IoT-1 has run successfully, meaning that the MUD PEP router/switch has been configured based on a **MUD file that permits traffic to/from some internet locations and implicitly denies traffic to/from all other internet locations.**	When the MUD-capable IoT device is connected to the network, its MUD PEP **router/switch will be configured to enforce the route filtering that is described in the device's MUD file** with respect to traffic being	Pass (for testable procedure, ingress cannot be tested)

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10 **PR.PT-3:** The principle of least functionality is incorporated by configuring systems to provide only essential capabilities. **NIST SP 800-53 Rev. 4** AC-3, CM-7		permitted to/from some internet locations, and traffic being implicitly blocked to/from all remaining internet locations.	
IoT-6	**ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **PR.AC-5:** Network integrity is protected, incorporating network segregation where appropriate. **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7 **PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10 **PR.PT-3:** The principle of least functionality is incorporated by	Test IoT-1 has run successfully, meaning that the MUD PEP router/switch has been configured based on a **MUD file that permits traffic to/from some lateral hosts and implicitly denies traffic to/from all other lateral hosts.** (The MUD file does not explicitly identify the hosts as lateral hosts; it identifies classes of hosts to/from which traffic should be denied, where one or more hosts of this class happen to be lateral hosts.)	When the MUD-capable IoT device is connected to the network, its MUD PEP **router/switch will be configured to enforce the access control information that is described in the device's MUD file** with respect to traffic being permitted to/from some lateral hosts, and traffic being implicitly blocked to/from all remaining lateral hosts.	Pass (for testable procedure, ingress cannot be tested)

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	configuring systems to provide only essential capabilities. **NIST SP 800-53 Rev. 4** AC-3, CM-7 **PR.IP-3:** Configuration change control processes are in place. **PR.DS-3:** Assets are formally managed throughout removal, transfers, and disposition.			
IoT-7	**PR.IP-3:** Configuration change control processes are in place. **NIST SP 800-53 Rev. 4** CM-3, CM-4, SA-10 **PR.DS-3:** Assets are formally managed throughout removal, transfers, and disposition. **NIST SP 800-53 Rev. 4** CM-8, MP-6	Test IoT-1 has run successfully, meaning that the MUD PEP **router/switch has been configured based on the MUD file** for a specific MUD-capable device in question. Next, have **the IoT device change DHCP state by explicitly releasing its IP address lease, causing the device's policy configuration to be removed from the MUD PEP router/switch.**	When the MUD-capable **IoT device explicitly releases its IP address lease,** the MUD-related configuration for that IoT device will be removed from its MUD PEP router/switch.	Failed
IoT-8	**PR.IP-3:** Configuration change control processes are in place. **NIST SP 800-53 Rev. 4** CM-3, CM-4, SA-10 **PR.DS-3:** Assets are formally managed throughout removal, transfers, and disposition. **NIST SP 800-53 Rev. 4** CM-8, MP-6	Test IoT-1 has run successfully, meaning that the MUD PEP **router/switch has been configured based on the MUD file** for a specific MUD-capable device in question. Next, have **the IoT device change DHCP state by**	When the MUD-capable **IoT device's IP address lease expires,** the MUD-related configuration for that IoT device will be removed from its MUD PEP router/switch.	Failed (not supported)

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
		waiting until the IoT device's address lease expires, causing the device's policy configuration to be removed from the MUD PEP router/switch.		
IoT-9	**ID.AM-1:** Physical devices and systems within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-2:** Software platforms and applications within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **DE.AE-1:** A baseline of network operations and expected data flows for users and systems is established and managed. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CM-2, SI-4 **PR.AC-4:** Access permissions and authorizations are managed, incorporating the principles of	Test IoT-1 has run successfully, meaning the MUD PEP **router/switch has been configured based on the MUD file** for a specific MUD-capable device in question. The MUD file contains domains that resolve to multiple IP addresses. The MUD PEP router/switch should be configured to permit communication to or from all IP addresses for the domain.	A domain in the MUD file resolves to two different IP addresses. The MUD manager will create ACLs that permit the MUD-capable device to send traffic to both IP addresses. The MUD-capable device attempts to send traffic to each of the IP addresses, and the MUD PEP router/switch permits the traffic to be sent in both cases.	Pass

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-17, AC-19, AC-20, SC-15 **PR.AC-5:** Network integrity is protected, incorporating network segregation where appropriate. **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7 **PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-8, MP-6 **PR.IP-3:** Configuration change control processes are in place. **NIST SP 800-53 Rev. 4** CM-8, MP-6 **PR.DS-2:** Data in transit is protected. **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10			
IoT-10	**ID.AM-1:** Physical devices and systems within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-2:** Software platforms and applications within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5	A MUD-capable IoT device is configured to emit a MUD URL. Upon being connected to the network, its MUD file is retrieved, and the PEP is configured to enforce the policies specified in that MUD URL for that device. **Within 24**	Upon reconnection of the IoT device to the network, **the MUD manager does not contact the MUD file server. Instead, it uses the cached MUD file.** It translates this	Pass

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	ID.AM-3: Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **DE.AE-1:** A baseline of network operations and expected data flows for users and systems is established and managed. **PR.AC-4:** Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC-5, AC-6, AC-14, AC-16, AC-24 **PR.AC-5:** Network integrity is protected, incorporating network segregation where appropriate. **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7 **PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10	**hours (i.e., within the cache-validity period for that MUD file), the IoT device is reconnected to the network.** After 24 hours have elapsed, the same device is reconnected to the network.	MUD file's contents into appropriate route-filtering rules and installs these rules onto the PEP for the IoT device. Upon reconnection of the IoT device to the network, after 24 hours have elapsed, the MUD manager does fetch a new MUD file.	

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	PR.IP-3: Configuration change control processes are in place. **NIST SP 800-53 Rev. 4** CM-3, CM-4, SA-10 **PR.PT-3:** The principle of least functionality is incorporated by configuring systems to provide only essential capabilities. **NIST SP 800-53 Rev. 4** AC-3, CM-7 **PR.DS-2:** Data in transit is protected.			
IoT-11	**ID.AM-1:** Physical devices and systems within the organization are inventoried.	A **MUD-capable IoT device can emit a MUD URL.** The device should leverage one of the specified manners for emitting a MUD URL.	Upon initialization, the MUD-capable IoT device broadcasts a DHCP message on the network, including at most one **MUD URL, in https scheme, within the DHCP transaction.** OR Upon initialization, the MUD-capable IoT device **emits a MUD URL as an LLDP extension.**	Pass

1401 In addition to supporting MUD, Build 1 demonstrates capabilities with respect to device discovery,
1402 attribute identification, and monitoring, as shown in Table 6-3.

Table 6-3 Non-MUD-Related Functional Capabilities Demonstrated

Exercise	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Exercise Summary	Expected Outcome	Observed Outcome
CnMUD-1	**ID.AM-1:** Physical devices and systems within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-2:** Software platforms and applications within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **DE.AE-1:** A baseline of network operations and expected data flows for users and systems is established and managed. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CM-2, SI-4 **DE.CM-1:** The network is monitored to detect potential cybersecurity events. **NIST SP 800-53 Rev. 4** AC-2, AU-12, CA-7, CM-3, SC-5, SC-7, SI-4	A **visibility/monitoring component** is connected to the local IoT network. It **is configured to detect all devices connected to the network, discover attributes of these devices, categorize the devices, and monitor the devices** for any change of status.	Upon being connected to the network, the **visibility/monitoring component detects all connected devices, identifies their attributes** (e.g., type, IP address, OS), and categorizes them. **When an additional device is powered on, it is also detected, and its attributes identified. When a device is powered off, its change of status is detected.**	As expected

6.5 Observations

We observed the following limitations to Build 1 that are informing improvements to its current proof-of-concept implementation:

- MUD manager (version 3.0.1):
    - In previous versions (version 1.0), DNS resolution of internet host names in the MUD file was performed manually and remained static. Dynamic resolution of Fully Qualified Domain Names has since been added and is currently supported.
    - Translation and implementation of the model construct from the MUD file was not supported at testing time. However, this should be addressed in newer versions.
- Catalyst 3850-S switch (IOS version 16.09.02):
    - The MUD URL cannot be extracted when emitted via DHCPv6. Hence, the switch is only able to support MUD-capable IoT devices that use DHCPv4 and IPv4. This version of the switch does not yet support MUD-capable IoT devices when they are configured to use IPv6. IPv6 functionality is expected to be supported in the future.
    - The DHCP server does not notify the MUD manager of changes in DHCP state for MUD-capable IoT devices on the network. According to the MUD specification, the DHCP server should notify the MUD manager if the MUD-capable IoT device's IP address lease expires or has been released. However, this version of the DHCP server does not do so at testing time. This is expected to be addressed in the future.
    - Ingress dynamic ACLs (DACLs) (i.e., DACLs that pertain to traffic that is received from sources external to the network and directed to local IoT devices) are not supported with this version. Consequently, even if a MUD-capable IoT device's MUD file indicates that the IoT device is not authorized to receive traffic from an external domain, the DACL that is needed to prohibit that ingress traffic will not be configured on the switch. As a result, unless there is some other layer of security in place, such as a firewall that is configured to block this incoming traffic, the IoT device will still be able to receive incoming packets from that unauthorized external domain, which means it will still be vulnerable to attacks originating from that domain, despite the fact that the device's MUD file makes it clear that the device is not authorized to receive traffic from that domain. Because egress DACLs (i.e., DACLs that pertain to traffic that is sent from IoT devices to an external domain) are supported, however, even though packets that are sent from an outside domain are not stopped from being received at the IoT device, return traffic from the device to the external domain will be stopped. This means, for example, that if an attacker is able to get packets to an IoT device from an outside domain, it will not be possible for the attacker to establish a TCP connection with the device from that outside domain, thereby limiting the range of attacks that can be launched against the IoT device. This is expected to be addressed in the future.

7 Build 2

The Build 2 implementation uses a product from MasterPeace Solutions called Yikes! to support MUD. Yikes! is a commercial router/cloud service solution focused on consumer and small-business markets. It consists of a Yikes! router, a cloud service, and a mobile application that interfaces with the cloud service. In addition to supporting MUD, the Yikes! router and cloud service perform device discovery on the network and apply additional traffic rules to both MUD-capable and non-MUD-capable devices based on device manufacturer and model.

Also integrated with the Yikes! router in Build 2 is open-source software called Quad9 Active Threat Response (Q9Thrt), which builds on the Quad9 DNS service provided by Global Cyber Alliance. Q9Thrt enables the Yikes! router to take advantage of threat-signaling intelligence that is available through the Quad9 DNS service. Build 2 can use this information to block access, first to domains and, subsequently, to related IP addresses, that have been determined to be dangerous. This threat-signaling capability can protect both MUD-capable and non-MUD-capable devices. Build 2 also uses certificates from DigiCert.

7.1 Collaborators

Collaborators that participated in this build are described briefly in the subsections below.

7.1.1 MasterPeace Solutions

MasterPeace Solutions, Ltd. is a cybersecurity company in Columbia, Maryland, that focuses on serving federal intelligence community agencies. MasterPeace also operates the MasterPeace LaunchPad start-up studio, chartered with launching cyber-oriented technology product companies. A current LaunchPad start-up portfolio company, Yikes!, has developed a solution that includes both a MUD manager and cloud-based support for non-MUD IoT device security. Yikes! was created to bring automated enterprise-level security to consumer and small-business networks. Those networks are typically flat (unsegmented), predominantly connected via Wi-Fi-enabled devices, and managed by individuals who possess relatively little IT or cyber background compared with enterprise IT and cyber teams. Learn more about MasterPeace at https://www.masterpeaceltd.com.

7.1.2 Global Cyber Alliance

GCA is an international, cross-sector effort dedicated to eradicating cyber risk and improving our connected world. It achieves its mission by uniting global communities, implementing concrete solutions, and measuring the effect. GCA, a 501(c)3, was founded in September 2015 by the Manhattan District Attorney's Office, the City of London Police, and the Center for Internet Security. Learn more about GCA at https://www.globalcyberalliance.org.

7.1.3 DigiCert

See Section 6.1.2 for a description of DigiCert.

7.2 Technologies

Table 7-1 lists all of the products and technologies used in Build 2 and provides a mapping among the generic component term, the specific product used to implement that component, and the security Function Subcategories that the product provides. When applicable, both the Function Subcategories that a component provides directly and those that it supports but does not provide directly are listed and labeled as such. For rows in which the provides/supports distinction is not noted, the component directly provides all listed Categories. Refer to Table 5-1 for an explanation of the NIST Cybersecurity Framework Subcategory codes.

Table 7-1 Products and Technologies

Component	Product	Function	Cybersecurity Framework Subcategories
MUD manager	MasterPeace Yikes! router	Fetches, verifies, and processes MUD files from the MUD file server; configures router or switch with traffic filters to enforce firewall rules based on the MUD file	Provides PR.PT-3 Supports ID.AM-1 ID.AM-2 ID.AM-3 PR.AC-4 PR.AC-5 PR.DS-5 DE.AE-1
MUD file server	MasterPeace-hosted Apache server	Hosts MUD files; serves MUD files to the MUD manager by using https	ID.AM-1 ID.AM-2 ID.AM-3 PR.AC-4 PR.AC-5 PR.DS-5 PR.PT-3 DE.AE-1
MUD file maker	MUD file maker (https://www.mud-maker.org/)	YANG script GUI used to create MUD files	ID.AM-1
MUD file	A YANG model instance that has been serialized in JSON (RFC 7951). The manufacturer of a MUD-capable device creates that device's MUD file.	Specifies the communications that are permitted to and from a given device	Provides PR.PT-3 Supports ID.AM-1

Component	Product	Function	Cybersecurity Framework Subcategories
	MUD file maker (see previous row) can create MUD files. Each MUD file is also associated with a separate MUD signature file.		ID.AM-2 ID.AM-3
DHCP server	MasterPeace Yikes! router (Linksys WRT 3200ACM)	Dynamically assigns IP addresses; recognizes MUD URL in DHCP DISCOVER message; should notify MUD manager if the device's IP address lease expires or has been released	ID.AM-3 PR.AC-4 PR.AC-5 PR.DS-5 PR.PT-3 DE.AE-1
Router or switch	MasterPeace Yikes! router (Linksys WRT 3200ACM)	Provides MUD URL to MUD manager; gets configured by the MUD manager to enforce the IoT device's communication profile; performs per-device firewall rule enforcement	ID.AM-3 PR.AC-4 PR.AC-5 PR.DS-5 PR.PT-3 DE.AE-1
Certificates	DigiCert Premium Certificate	Used to sign MUD files and generate corresponding signature file	PR.AC-1 PR.AC-3 PR.AC-5 PR.AC-7
MUD-capable IoT device	Raspberry Pi Model 3B (devkit) Samsung ARTIK 520 (devkit) BeagleBone Black (devkit) NXP i.MX 8M (devkit)	Emits a MUD URL as part of its DHCP DISCOVER message; requests and applies software updates	ID.AM-1
Non-MUD-capable IoT device	Camera Mobile phones Connected lighting devices Connected assistant Printer	Acts as typical IoT devices on a network; creates network connections to cloud services	ID.AM-1

Component	Product	Function	Cybersecurity Framework Subcategories
	Digital video recorder		
Update server	NCCoE-hosted Apache server	Acts as a device manufacturer's update server that would communicate with IoT devices to provide patches and other software updates	PR.IP-1 PR.IP-3
Unapproved server	NCCoE-hosted Apache server	Acts as an internet host that has not been explicitly approved in a MUD file	DE.DP-3 DE.AM-1
IoT device discovery, categorization, and traffic policy enforcement	MasterPeace Yikes! router (Linksys WRT 3200ACM) and Yikes! cloud service	Discovers, classifies, and constrains traffic to/from IoT devices on network based on information such as DHCP header, MAC address, operating system, manufacturer, and model	ID.AM-1 PR.IP-1 DE.AM-1
Display and configuration of device information and traffic policies	MasterPeace Yikes! mobile application	Interacts with the Yikes! cloud to receive, display, and change information about the Yikes! router traffic policies and identification and categorization information about connected devices	ID.AM-1 PR.IP-1 DE.AM-1
Threat agent	GCA Quad9 threat agent, which is part of the open-source software Q9Thrt and is integrated into the Yikes! router	Monitors DNS traffic to/from devices on the local network and detects when domains are not resolved. When domains are not resolved, it queries the	ID.RA-1 ID.RA-2 ID.RA-3

Component	Product	Function	Cybersecurity Framework Subcategories
		Quad9 threat API regarding whether the domain is dangerous and, if so, what threat intelligence provider has flagged it as such. If a domain is determined to be dangerous, it notifies the Quad9 MUD manager of this threat.	
Threat-signaling MUD manager	GCA Quad9 MUD manager, which is part of the open-source software Q9Thrt and is integrated into the Yikes! router	Requests, receives, and parses the threat MUD file provided by the threat-signaling service's threat MUD file server, and applies its rules to create configurations to the Yikes! router's DNS service and its firewall rules that prohibit all devices from accessing the locations listed in the threat MUD file	ID.RA-1 ID.RA-2 ID.RA-3
Threat-signaling DNS services	GCA Quad9 DNS service	Receives input from several threat intelligence providers (including ThreatSTOP). Receives DNS resolution queries from local DNS service. For domains that are not known to be a threat, it simply resolves those domains to their IP address and provides this address to the request-	ID.RA-1 ID.RA-2 ID.RA-3

Component	Product	Function	Cybersecurity Framework Subcategories
		ing device. For domains that have been flagged as dangerous, it does not perform address resolution and instead returns a NULL response.	
Threat-signaling API	GCA Quad9 threat API	Receives queries from the threat-signaling agent on the local network regarding domains that were not resolved. If a domain was not resolved because it had been flagged as dangerous, it responds with the name of the threat intelligence provider that had flagged the domain as dangerous.	ID.RA-1 ID.RA-2 ID.RA-3
Threat MUD file server	ThreatSTOP threat MUD file server	Receives requests from the threat-signaling MUD manager on the local network for the threat MUD file corresponding to a domain that has been flagged as dangerous. Responds by providing the threat MUD file (and the MUD file's signature file) that is associated with the threat that has made this domain dangerous. This threat file will contain not just the domain and IP address	ID.RA-1 ID.RA-2 ID.RA-3

Component	Product	Function	Cybersecurity Framework Subcategories
		of the domain that the router had tried, unsuccessfully, to resolve; it will also include the list of all domains and IP addresses that are associated with the threat in question, i.e., all domains and IP addresses that are associated with this threat campaign.	
Threat MUD File	Threat file in MUD file format provided by ThreatSTOP listing all dangerous domains and IP addresses associated with any given threat	This is a file that has the exact same format as a MUD file, thus providing a standardized format for conveying the domains and IP addresses of all dangerous sites that are associated with a given threat and should therefore be blocked. Unlike a typical MUD file, however, this file does not contain usage description information regarding the permitted communication profile of some specific type of device. Instead, the information in this file is intended to be applied to the entire network (both MUD-capable and non-MUD-capable devices). Furthermore, it will list only external	ID.RA-1 ID.RA-2 ID.RA-3

Component	Product	Function	Cybersecurity Framework Subcategories
		sites to and from which traffic should be prohibited because the sites are associated with a given threat, not sites with which communication should be permitted, and it will not provide any rules regarding local network traffic that should be permitted or prohibited. Also, any given threat may be associated with a number of different domains and/or IP addresses. This threat file is designed to list all domains and IP addresses that are associated with any given threat that should be blocked. The file will also differ from a typical MUD file insofar as its mfg-name field will contain the name of the threat intelligence provider rather than the name of a device manufacturer, and its model-name field will typically contain the name of the threat that the file is associated with rather than model information about any IoT device.	

Each of these components is described more fully in the following sections.

7.2.1 MUD Manager

The MUD manager is a key component of the architecture. It fetches, verifies, and processes MUD files from the MUD file server. It then configures the router with firewall rules to control communications based on the contents of the MUD files. The Yikes! MUD manager is a logical component within the physical Yikes! router. The Yikes! router supports IoT devices that emit their MUD URLs via DHCP messages. When the MUD URL is emitted via DHCP, it is extracted from the DHCP message and provided to the MUD manager, which then retrieves the MUD file and signature file associated with that URL and configures the Yikes! router to enforce the IoT device's communication profile based on the MUD file. The router implements firewall rules for src-dnsname, dst-dnsname, my-controller, controller, same-manufacturer, manufacturer, and local-networks constructs that are specified in the MUD file. The system supports both lateral east/west protection and appropriate access to internet sites (north/south protection).

By default, Yikes! prohibits each device on the network from communicating with all other devices on the network unless explicitly permitted either by the MUD file or by local policy rules that are configurable within the Yikes! router.

The version of the Yikes! MUD manager used in this project is a prerelease implementation that is intended to introduce home and small-business network users to the MUD concept. It is intended to be a fully automated MUD manager implementation that includes all MUD protocol features.

7.2.2 MUD File Server

In the absence of a commercial MUD file server for use in this project, the NCCoE used a MUD file server hosted by MasterPeace that is accessible via the internet. This file server stores the MUD files along with their corresponding signature files for the IoT devices used in the project. Upon receiving a GET request for the MUD files and signatures, it serves the request to the MUD manager by using https.

7.2.3 MUD File

Using the MUD file maker component referenced above in Table 7-1, it is possible to create a MUD file with the following contents:

- internet communication class—access to cloud services and other specific internet hosts:
    - host: www.osmud.org
        - protocol: TCP
        - direction-initiated: from IoT device
        - source port: any

        - destination port: 443

- controller class—access to **classes** of devices that are known to be controllers (could describe well-known services such as DNS or NTP):
    - host: www.getyikes.com
        - protocol: TCP
        - direction-initiated: from IoT device
        - source port: any
        - destination port: 443
- local-networks class—access to/from **any** local host for specific services (e.g., http or https):
    - host: any
        - protocol: TCP
        - direction-initiated: from IoT device
        - source port: any
        - destination port: 80
- my-controller class—access to controllers specific to this device:
    - controllers: null (to be filled in by the network administrator)
        - protocol: TCP
        - direction-initiated: from IoT device
        - source port: any
        - destination port: 80
- same-manufacturer class—access to devices of the same manufacturer:
    - same-manufacturer: null (to be filled in by the MUD manager)
        - protocol: TCP
        - direction-initiated: from IoT device
        - source port: any
        - destination port: 80
- manufacturer class—access to devices of a specific manufacturer (identified by MUD URL):
    - manufacturer: Google (URL decided by the device manufacturer)
        - protocol: TCP
        - direction-initiated: from IoT device

- source port: any
- destination port: 80

7.2.4 Signature File

According to the IETF MUD specification, “a MUD file MUST be signed using CMS as an opaque binary object.” All the MUD files in use (e.g., *yikesmain.json*) were signed with the OpenSSL tool by using the command described in the specification (detailed in Volume C of this publication). A Premium Certificate, requested from DigiCert, was leveraged to generate the signature file (e.g., *yikesmain.p7s*). Once created, the signature file is stored on the MUD file server.

7.2.5 DHCP Server

The DHCP server in the architecture is MUD-capable and, like the MUD manager, is a logical component within the Yikes! router. In addition to dynamically assigning IP addresses, it recognizes the DHCP option (161) and extracts the MUD URL from the IoT device’s DHCP message. It then provides the MUD URL to the MUD manager. The DHCP server provided by the Yikes! router is useful in small-/medium-business and home network environments where centralized address management is not required.

7.2.6 Router/Switch

This build uses the MasterPeace Yikes! router. The Yikes! router is a customized original equipment manufacturer product, which at the time of this implementation is a preproduction product developed on a Linksys WRT 3200ACM router. It is a self-contained router, Wi-Fi access point, and firewall that communicates locally with Wi-Fi devices and wired devices. The Yikes! router initially isolates all devices connected to the router from one another. When devices connect to the router, the Yikes! router provides the device’s DHCP header, MAC address, operating system, and connection characteristics to the Yikes! cloud service, which attempts to identify and categorize each device based on this information. The Yikes! router receives from the Yikes! cloud service rules for north/south and east/west filtering based on the Yikes! cloud processing (see Section 7.2.11) and any custom user settings that may have been configured in the Yikes! mobile application (see Section 7.2.12). These rules may apply to both MUD-capable and non-MUD-capable devices.

In addition to this category-based traffic policy enforcement that the Yikes! router provides for all devices, the Yikes! router also provides MUD support for MUD-capable IoT devices that emit MUD URLs via DHCP. Future work may be done to support MUD-capable devices that emit MUD URLs via X.509 or LLDP. The Yikes! router receives the MUD URL emitted by the device, retrieves the MUD file associated with that URL, and configures traffic filters (firewall rules) on the router to enforce the communication limitations specified in the MUD file for each device. The Yikes! router requires access to the internet to support secure API access to the Yikes! cloud service.

Last, the Yikes! router also provides integrated support for threat signaling by incorporating GCA Quad9 threat agent (see Section 7.2.13) and GCA Quad9 MUD manager (see Section 7.2.14) capabilities. Both the Quad9 threat agent and the Quad9 MUD manager are components of the open-source software Q9Thrt. See Section 7.3.1.3 for a description of Build 2's threat-signaling architecture and more information on Q9Thrt.

7.2.7 Certificates

DigiCert provisioned a Premium Certificate for signing the MUD files. The Premium Certificate supports the key extensions required to sign and verify Cryptographic Message Syntax (CMS) structures as required in the MUD specification. Further information about DigiCert's CertCentral web-based platform, which allows provisioning and managing publicly trusted X.509 certificates, is in Section 6.2.8.

7.2.8 IoT Devices

This section describes the IoT devices used in the laboratory implementation. There are two distinct categories of devices: devices that can emit a MUD URL in compliance with the MUD specification, i.e., MUD-capable IoT devices; and devices that are not capable of emitting a MUD URL in compliance with the MUD specification, i.e., non-MUD-capable IoT devices.

7.2.8.1 MUD-Capable IoT Devices

The project used several MUD-capable IoT devices: NCCoE Raspberry Pi (devkit), Samsung ARTIK 520 (devkit), BeagleBone Black (devkit), and NXP i.MX 8m (devkit). The NCCoE team modified the devkits to simulate MUD capability within IoT devices. All of the MUD-capable IoT devices demonstrate the ability to emit a MUD URL as part of a DHCP transaction and to request and apply software updates.

7.2.8.1.1 NCCoE Raspberry Pi (Devkit)

The Raspberry Pi devkit runs the Raspbian 9 operating system. It is configured to include a MUD URL that it emits during a typical DHCP transaction.

7.2.8.1.2 NCCoE Samsung ARTIK 520 (Devkit)

The Samsung ARTIK 520 devkit runs the Fedora 24 operating system. It is configured to include a MUD URL that it emits during a typical DHCP transaction.

7.2.8.1.3 NCCoE BeagleBone Black (Devkit)

The BeagleBone Black devkit runs the Debian 9.5 operating system. It is configured to include a MUD URL that it emits during a typical DHCP transaction.

7.2.8.1.4 NCCoE NXP i.MX 8m (Devkit)

The NXP i.MX 8m devkit runs the Yocto Linux operating system. The NCCoE modified a Wi-Fi start-up script on the device to configure it to emit a MUD URL during a typical DHCP transaction.

7.2.8.2 Non-MUD-Capable IoT Devices

The laboratory implementation also includes a variety of legacy, non-MUD-capable IoT devices that are not capable of emitting a MUD URL. These include cameras, mobile phones, connected lighting, a connected assistant, a printer, and a DVR.

7.2.8.2.1 Cameras

The three cameras utilized in the laboratory implementation are produced by two different manufacturers. They stream video and audio either to another device on the network or to a cloud service. These cameras are controlled and managed by a mobile phone.

7.2.8.2.2 Mobile Phones

Two types of mobile phones are used for setting up, interacting with, and controlling IoT devices.

7.2.8.2.3 Lighting

Two types of connected lighting devices are used in the laboratory implementation. These connected lighting components are controlled and managed by a mobile phone.

7.2.8.2.4 Connected Assistant

A connected assistant is utilized in the laboratory implementation. The device demonstrates and tests the wide range of network traffic generated by a connected assistant.

7.2.8.2.5 Printer

A connected printer is connected to the laboratory network wirelessly to demonstrate connected printer usage.

7.2.8.2.6 Digital Video Recorder

A connected DVR is connected to the laboratory implementation network. This is also controlled and managed by a mobile phone.

7.2.9 Update Server

The update server is designed to represent a device manufacturer or trusted third-party server that provides patches and other software updates to the IoT devices. This project used an NCCoE-hosted update server that provides faux software update files.

7.2.9.1 NCCoE Update Server

The NCCoE implemented its own update server by using an Apache web server. This file server hosts faux software update files to be served as software updates to the IoT device devkits. When the server receives an http request, it sends the corresponding faux update file.

7.2.10 Unapproved Server

As with Build 1, the NCCoE implemented and used its own unapproved server for Build 2. Details are in Section 6.2.11.

7.2.11 IoT Device Discovery, Categorization, and Traffic Policy Enforcement–Yikes! Cloud

The Yikes! cloud uses proprietary techniques and machine learning to analyze information about each device that is provided to it by the Yikes! router. The Yikes! cloud uses the DHCP header, MAC address, operating system, and connection characteristics of devices to automatically classify each device, including make, model, and Yikes! device category. Yikes! has a comprehensive list of categories that includes these examples:

- mobile: phone, tablet, e-book, connected watch, wearable, car
- home and office: computer, laptop, printer, IP phone, scanner
- connected home: IP camera, connected device, connected plug, light, voice assistant, thermostat, doorbell, baby monitor
- network: router, Wi-Fi extender
- server: network attached storage, server
- engineering: Raspberry Pi, Arduino

The Yikes! cloud then uses the Yikes! category to define specific east/west rules for that device and every other device on the Yikes! router's network. It also looks up the device in the Yikes! proprietary IoT device library, and, if available, provides specialized north/south filtering rules for that device. The east/west and north/south rules are then configured on the Yikes! router for local enforcement.

The Yikes! cloud also provides information about the device, whether it is MUD-capable, its categorization, and filtering rules to the Yikes! mobile application (see Section 7.2.12). This information is presented to the user in a graphical user interface, and the user can make specific changes. These changes are also configured on the Yikes! router for enforcement.

7.2.12 Display and Configuration of Device Information and Traffic Policies–Yikes! Mobile Application

Yikes! also provides a mobile application for additional capabilities, which at publication time was accessed through a web user interface (UI). The Yikes! mobile application allows users further fine-grained device-filtering control. The Yikes! mobile application interacts with the Yikes! cloud to receive and display information about the traffic policies that are configured on the Yikes! router as well as identification and categorization information about devices connected to the network. The Yikes!

mobile application enables device information that is populated automatically by the Yikes! cloud to be overridden, and it enables users to configure traffic policies to be enforced by the router.

7.2.13 Threat Agent

Build 2 has a threat-signaling agent integrated into the Yikes! router. This threat-signaling agent is part of the open-source software called Q9Thrt, which builds on and extends the Quad9 DNS service provided by GCA. More information on Q9Thrt is at https://github.com/osmud/q9thrt.

7.2.13.1 GCA Quad9 Threat Agent

The GCA Quad9 threat agent monitors DNS traffic to/from devices on the local network and detects when domains are not resolved by the Quad9 DNS service. When a domain is not resolved, it could mean one of two things: Either the domain has been flagged as potentially unsafe, or the domain does not exist (perhaps because it was mistyped, for example). The Quad9 threat agent eavesdrops on DNS responses that are sent from the Quad9 DNS service in the cloud to the Yikes! router's local DNS services. If the Quad9 threat agent detects a null response, it queries the Quad9 threat API to inquire as to whether the domain is dangerous and, if so, what threat intelligence provider has flagged it as such. If it receives a response indicating that a domain has been determined to be unsafe, it informs the Quad9 MUD manager (see Section 7.2.18) component (which is also integrated into the Yikes! router).

7.2.14 Threat-Signaling MUD Manager

Build 2 has a second MUD manager integrated into the Yikes! router that is designed to retrieve and parse the threat MUD file (see Section 7.2.18) retrieved from the threat intelligence provider. This threat-signaling MUD manager is part of the open-source software called GCA Q9Thrt, which builds on and extends the Quad9 DNS service provided by GCA. More information on Q9Thrt may be found at https://github.com/osmud/q9thrt.

7.2.14.1 GCA Quad9 MUD Manager

The GCA Quad9 MUD manager retrieves and parses threat MUD files. Threat MUD files are files that are written in MUD file format that list the domains and IP addresses of locations on the internet that have been determined to be unsafe and should be blocked because they are associated with a known threat. When the Quad9 threat agent (which is also integrated into the Yikes! router) learns that a threat has been found, it informs the Quad9 MUD manager and provides the Quad9 MUD manager with the URL of the threat MUD file. The Quad9 MUD manager uses https to request the threat MUD file and the threat MUD file's signature file. Assuming the signature file indicates that the threat MUD file is valid, the Quad9 MUD manager parses the threat MUD file and uses the threat MUD file rules to configure both the firewall and the local DNS services in the Yikes! router. It configures the firewall to prohibit all devices from accessing the domains and IP addresses listed in the threat MUD file, and it configures the local DNS services to return null responses when asked to resolve domain names listed in the threat MUD file.

7.2.15 Threat-Signaling DNS Services

Build 2 accesses external DNS services that receive input from several internet threat intelligence providers and are thus able to respond to domain name resolution requests for unsafe domains by signaling that the requested domain is potentially unsafe. These DNS services are provided by GCA.

7.2.15.1 GCA Quad9 DNS Service

GCA Quad9 DNS service receives input from several threat intelligence providers, making them aware of which domains have been determined to be unsafe. One of the threat intelligence providers that provides input to Quad9 DNS service is ThreatSTOP. For domains that are not known to be a threat, Quad9 DNS service behaves like any other DNS service would by resolving those domain names to their IP address(es) and providing those addresses to the requesting device. For domains that have been flagged as dangerous, however, Quad9 DNS service does not perform domain name resolution; instead, it returns a null response to the requesting device.

7.2.16 Threat-Signaling API

Build 2 accesses an external threat-signaling API that, when queried regarding specific domain names, responds by indicating whether the domain has been determined to be unsafe and, if so, the name of the threat intelligence provider responsible for the threat information. This threat-signaling API is provided by GCA.

7.2.16.1 GCA Quad9 Threat API

When a device on the local network makes a DNS request for a domain that does not get resolved, this means either that the domain does not exist or that it is unsafe. To determine which is the case for any given domain, the Quad9 threat agent on the Yikes! router queries the Quad 9 Threat API regarding that domain. If the domain is considered unsafe, the Quad9 threat API responds with the name of the threat intelligence provider that had flagged the domain as dangerous and other information that is needed to retrieve the associated threat MUD file.

7.2.17 Threat MUD File Server

Build 2 accesses an external threat MUD file server containing threat MUD files (see Section 7.2.18) for threats that a threat intelligence provider has identified and documented. The threat MUD file server used in Build 2 hosts threat MUD files provided by the threat intelligence provider ThreatSTOP.

7.2.17.1 ThreatSTOP Threat MUD File Server

When the Quad9 MUD manager on the Yikes! router is informed by the Quad9 threat agent that a threat has been found, the Quad9 MUD manager contacts the ThreatSTOP threat MUD file server to retrieve the threat MUD file associated with that threat. This threat MUD file server hosts threat MUD files (see Section 7.2.18) for threats that ThreatSTOP has identified and documented. When it receives a request from the Quad9 MUD manager for a threat file corresponding to a domain, the ThreatSTOP threat MUD file server responds by providing the threat file that is associated with the threat that has made this domain unsafe. This threat file will contain not just the domain and IP address of the domain that the router had tried unsuccessfully to resolve; it will also include all domains and IP addresses that are associated with the threat in question.

7.2.18 Threat MUD File

Build 2 uses threat MUD files provided by the threat intelligence provider ThreatSTOP. Threat MUD files have the same format as MUD files, thus providing a standardized format for conveying the domains and IP addresses of all dangerous sites that are associated with a given threat and should therefore be blocked. Unlike a typical MUD file, however, a threat MUD file does not contain manufacturer usage description information regarding the communication profile of some specific type of device. Instead, the information in this file is intended to be applied to the entire network (both MUD-capable and non-MUD-capable devices). Furthermore, the threat MUD file will list only external sites to and from which traffic should be prohibited because the sites are associated with a given threat, not sites with which communication should be permitted, and it will not provide any rules regarding local network traffic that should be permitted or prohibited. Also, any given threat may be associated with several different domains and/or IP addresses. The threat MUD file is designed to list all domains and IP addresses that are associated with any given threat that should be blocked. The file will also differ from a typical MUD file insofar as its mfg-name field will typically contain the name of the threat intelligence provider rather than the name of a device manufacturer, and its model-name field will typically contain the name of the threat that the file is associated with rather than model information about a particular IoT device.

7.3 Build Architecture

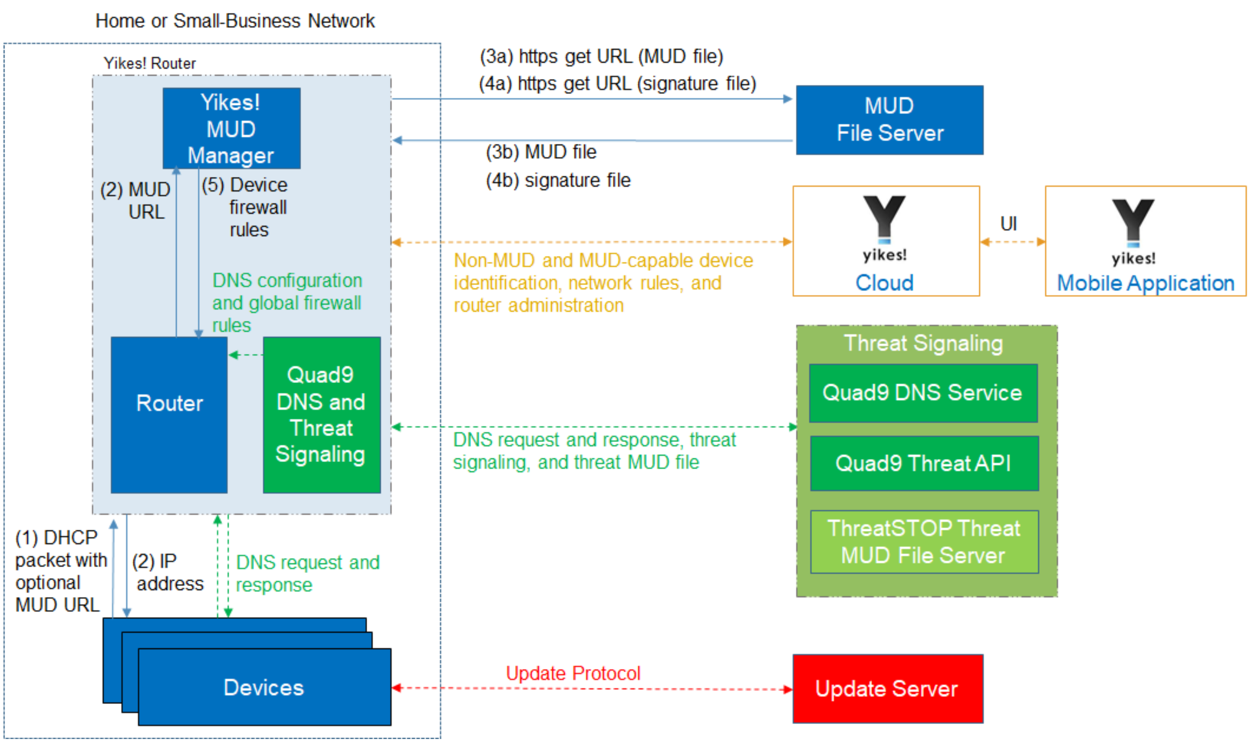
In this section we present the logical architecture of Build 2 relative to how it instantiates the reference architecture depicted in Figure 4-1. We also describe Build 2’s physical architecture and present message flow diagrams for some of its processes.

7.3.1 Logical Architecture

Figure 7-1 depicts the logical architecture of Build 2. Figure 7-1 uses numbered arrows to depict in detail the flow of messages needed to support installation of MUD-based access control rules for a MUD-capable device. The other key aspects of the Build 2 architecture (i.e., the Yikes! cloud, the Yikes! mobile application, threat signaling, and the update server) are depicted but not described in the same depth as MUD.

Yikes! is designed to run as a router with a connection to the Yikes! cloud and to be managed via the Yikes! mobile application. The Yikes! cloud provides traffic rules to the Yikes! router that apply to devices based on device category. The Yikes! router also supports threat-signaling capabilities that enable it to refrain from connecting to domains that threat intelligence services have flagged as potentially dangerous. The logical architecture for Build 2 also includes the notion of ensuring that all IoT devices can access update servers so they can remain up-to-date with the latest security patches. MUD, Yikes! cloud, and threat-signaling support are each described in their respective subsections below.

Figure 7-1 Logical Architecture—Build 2



7.3.1.1 MUD Capability

As shown in Figure 7-1, the Yikes! router includes integrated support for MUD in the form of a Yikes! MUD manager component and a MUD-capable DHCP server (not depicted). Support for MUD also requires access to a MUD file server that hosts MUD files for the MUD-capable IoT devices being connected to the network.

The Yikes! router currently supports DHCP as the mechanism for MUD URL emission. It contains a DHCP server that is configured to extract MUD URLs from IPv4 DHCP transactions.

As shown in Figure 7-1, the flow of messages needed to support installation of MUD-based access control rules for a MUD-capable device is as follows:

- Upon connecting a MUD-capable device, the MUD URL is emitted via DHCP (step 1).
- The Yikes! DHCP server on the router receives the request from the device and assigns it an IP address (step 2).
- At the same time, the DHCP server sends the MUD URL to the Yikes! MUD manager (step 2).
- Once the MUD URL is received, the MUD manager uses it to fetch the MUD file from the MUD file server (step 3a); if successful, the MUD file server at the specified location will serve the MUD file (step 3b).
- Next, the MUD manager requests the signature file associated with the MUD file (step 4a) and upon receipt (step 4b) verifies the MUD file by using its signature file.
- Assuming the MUD file has been verified successfully, the MUD manager translates the traffic rules that are in the MUD file into firewall rules that it installs onto the Yikes! router (step 5). Once the firewall rules are installed on the router, the MUD-capable IoT device will be able to communicate with approved local hosts and internet hosts as defined in the MUD file, and any unapproved communication attempts will be blocked.

7.3.1.2 Yikes! Cloud Capability

The Yikes! cloud includes the ability to identify and categorize both MUD-capable and non-MUD-capable devices that join the network, and it serves as the repository of traffic policies that can be applied to categories of devices regardless of whether those devices are MUD-capable. The Yikes! router communicates with the Yikes! cloud via a secure API. This communication is required for the router to send information related to the network to the Yikes! cloud service as well as to receive network rules and router administration from the Yikes! cloud. Network rules and router administration are configured through the Yikes! mobile application.

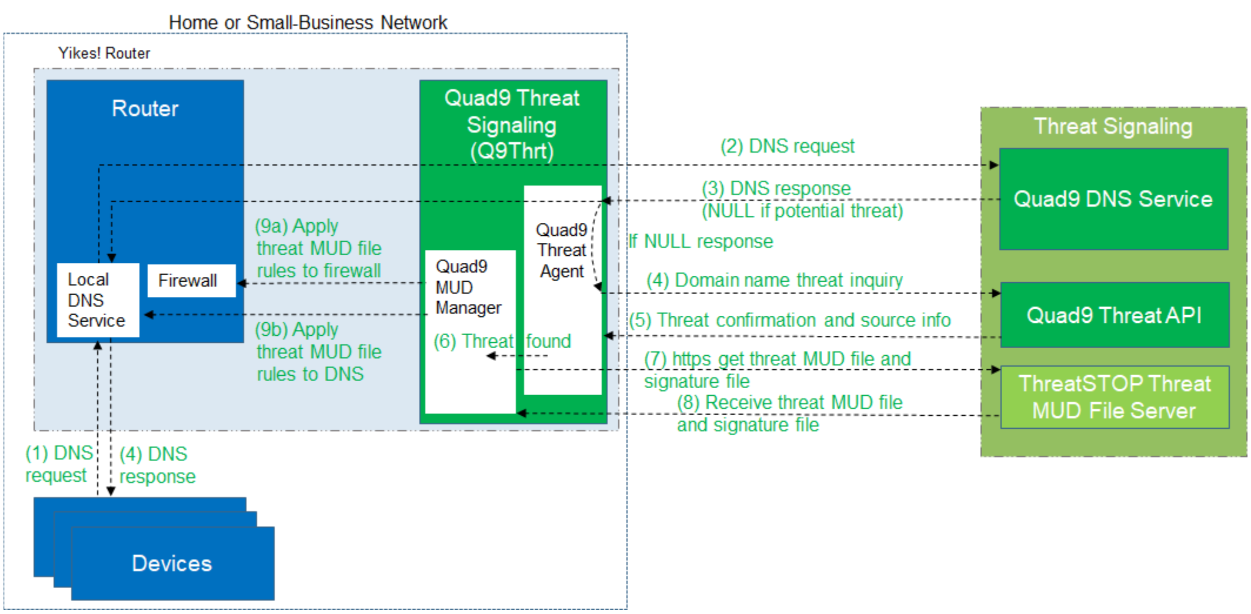
It is possible that both Yikes! cloud traffic policies and MUD file traffic policies could both apply to any given device in the network. For any given device, if these policies conflict, MUD file policies are given precedence over Yikes! traffic policies. If the policies do not conflict, they are both applied to the device. If a device is not MUD-capable, the Yikes! cloud policies that apply to it will be applied. If a device is MUD-capable but its MUD file is not applied (because, for example, the TLS certificate of the MUD file server is not valid or the MUD file is determined to be invalid), the Yikes! cloud rules that apply to the MUD-capable device will still be applied.

7.3.1.3 Threat-Signaling Capability

Build 2 integrates a threat-signaling capability that protects both MUD-capable and non-MUD-capable devices from the latest cybersecurity threats that have been detected by threat intelligence services. It prevents devices from accessing external domains and IP addresses that are associated with known current cybersecurity threats.

Figure 7-2 depicts a detailed view of Build 2's threat-signaling architecture. As shown, GCA's Quad9 threat agent and Quad9 MUD manager (which are both part of Q9Thrt) are integrated into the Yikes! router to support threat signaling. Additionally, the Yikes! router requires the use of several external components to support threat signaling: Quad9 DNS service, which receives threat information feeds from a variety of threat intelligence services; Quad9 threat API, which confirms a threat as well as information regarding how to find the threat MUD file for that threat; and the ThreatSTOP threat MUD file server, which provides the threat MUD file for the threat.

Figure 7-2 Threat-Signaling Logical Architecture–Build 2



The messages that are exchanged among architectural components to support threat signaling are depicted by arrows and numbered in sequence in Figure 7-2. The result of this message flow is to protect a local device from connecting to a domain that has been identified as unsafe by a threat intelligence service from which Quad9 DNS service receives information which, in this case, is ThreatSTOP.

As depicted in Figure 7-2, the steps are as follows:

- A local device (which may or may not be an IoT device and may or may not be MUD-capable) sends a DNS resolution requests to its local DNS service, which is hosted on the Yikes! router (step 1).
- If the local DNS service cannot resolve the request itself, it will forward the request to the Quad9 DNS service (step 2).
- The Quad9 DNS service will return a DNS response to the Yikes! router's local DNS service. The Quad9 DNS service receives input from several threat intelligence providers (not depicted in the diagram), so it is aware of whether the domain in question has been identified to be unsafe. If the domain has not been identified as unsafe, the Quad9 DNS service will respond with the IP address(es) corresponding to the domain (as would any normal DNS service). If the domain has been flagged as unsafe, however, the Quad9 DNS service will not resolve the domain. Instead, it will return an empty (null) DNS response message to the local DNS service (step 3).
- The local DNS service will forward the DNS response to the device that originally made the DNS resolution request (step 4).
- Meanwhile, the Quad9 Threat Agent that is running on the Yikes! router monitors all DNS requests and responses. When it sees a domain that does not get resolved, it sends a query to the Quad9 Threat API asking whether the domain is dangerous and, if so, what threat intelligence provider had flagged it as such and with what threat it is associated (step 4).
- The Quad9 Threat API responds with this information, which, in this case, informs the threat agent that the domain is indeed dangerous and if it wants more information about the blocked domain, it should contact ThreatSTOP (a threat intelligence provider) and request a particular threat MUD file. This threat MUD file will list domains and IP addresses that should be blocked because they are all associated with the same threat campaign as this threat (step 5).
- The Quad9 threat agent provides this information to the Quad9 MUD manager (step 6).
- The Quad9 MUD manager requests the threat MUD file (and the threat MUD file's signature file) from the ThreatSTOP threat MUD file server (step 7).
- The Quad9 MUD manager receives the threat MUD file (and the threat MUD file's signature file) from the ThreatSTOP threat MUD file server and uses the signature file to verify that the threat MUD file is valid (step 8).
- Assuming the threat MUD file is valid, the Quad9 MUD manager uses the threat MUD file to configure the router's firewall to block all domains and IP addresses listed in this threat MUD file (step 9a).
- The Quad9 MUD manager also configures the router's local DNS services to provide empty responses for DNS requests that are made for all domain names that are listed in the threat MUD file (step 9b).

Threat-signaling rules have higher precedence than MUD rules, which, in turn, have higher precedence than Yikes! category rules. This means that if a domain is flagged as dangerous by threat-signaling intelligence, none of the devices on the local network will be permitted to communicate with it—even MUD-capable devices whose MUD files list that domain as permissible.

Threat-signaling rules time out after 24 hours, at which time the firewall rules associated with those rules are removed from the router. If, after 24 hours, a device tries to connect to that domain but is still considered dangerous, the firewall rules will no longer be in place in the router to prevent access to the domain. However, when the device attempts to access the domain, the same DNS resolution process as depicted in Figure 7-2 will be performed all over again: when the device requests resolution of the domain name, the Quad9 DNS service will return an empty DNS response message, and the threat MUD file for that domain will be retrieved and its rules installed on the router firewall for another 24 hours.

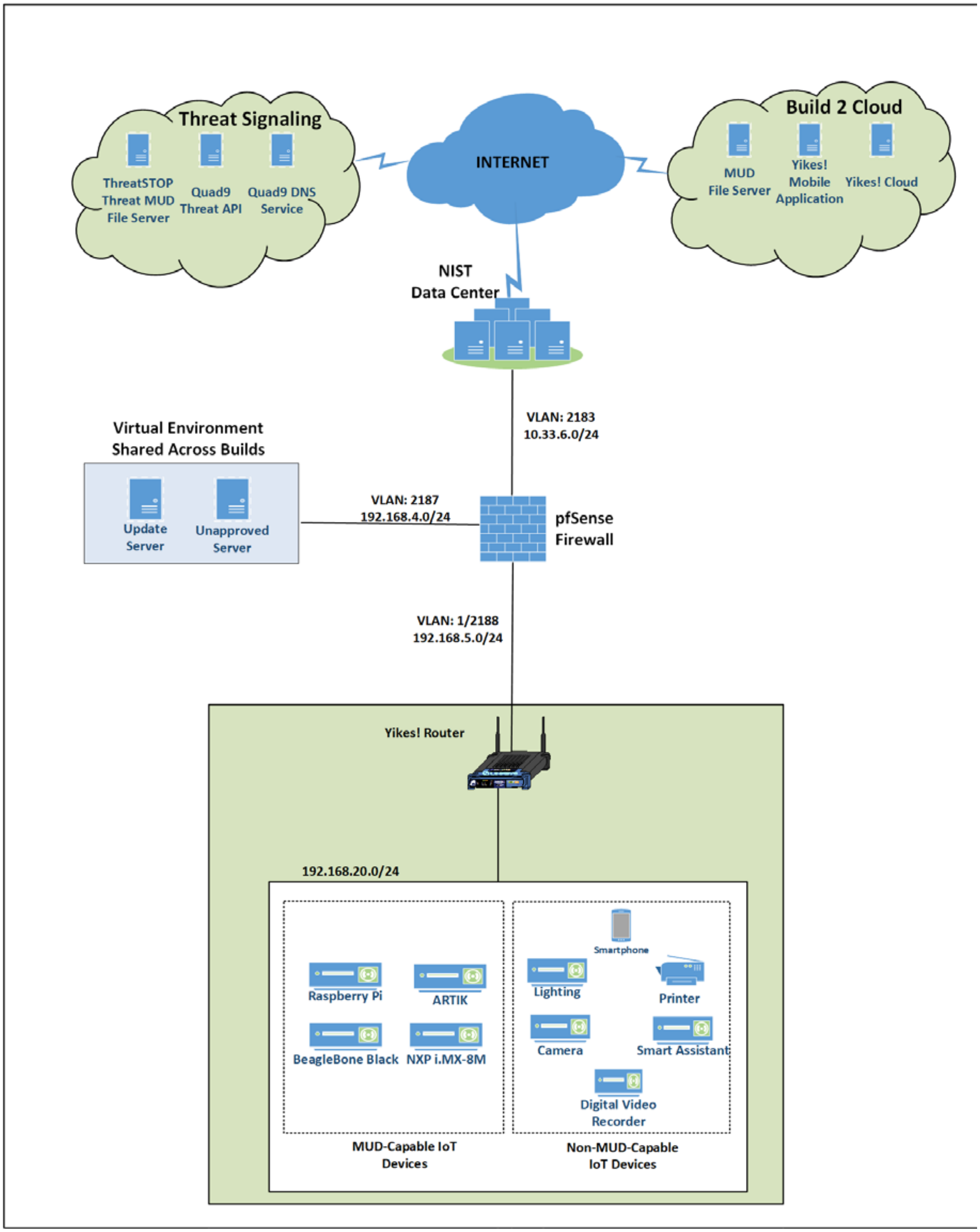
7.3.2 Physical Architecture

Figure 7-3 depicts the physical architecture of Build 2. A single DHCP server instance is configured for the local network to dynamically assign IPv4 addresses to each IoT device that connects to the Yikes! router. This single subnet hosts both MUD-capable and non-MUD-capable IoT devices. The network infrastructure as configured utilizes the IPv4 protocol for communication both internally and to the internet.

In addition, this build uses a portion of the virtual environment that is shared across builds. Services hosted in this environment include an update server and an unapproved server.

Internet-accessible cloud services are also supported in Build 2. This includes a MUD file server and Yikes! cloud services. To support threat-signaling functionality, a ThreatSTOP threat MUD file server, Quad9 threat API, and Quad9 DNS service were utilized.

1894 **Figure 7-3 Physical Architecture—Build 2**



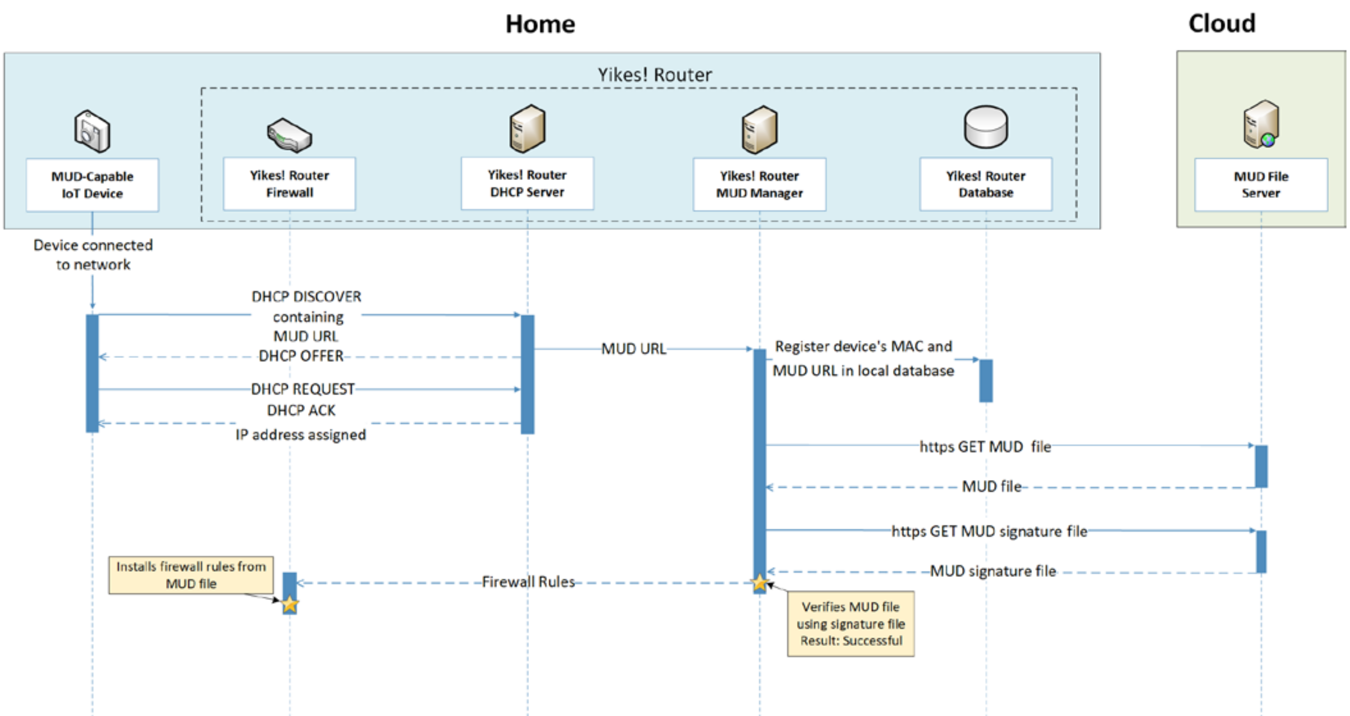
7.3.3 Message Flow

This section presents the message flows used in Build 2 during several different processes of note.

7.3.3.1 Installation of MUD-Based Access Control Rules for MUD-Capable Devices

Figure 7-4 depicts the message flows involved in the process of installing MUD-based access control rules for a MUD-capable IoT device in Build 2.

Figure 7-4 MUD-Capable IoT Device MUD-Based ACL Installation Message Flow—Build 2



The components used to support Build 2 are deployed across the home/small-business network (shown in blue) and the cloud (shown in green). A single device called the Yikes! router on the home/small-business network hosts five logical components: the Yikes! router firewall, the Yikes! router DHCP server, the Yikes! router MUD manager, the Yikes! router database, and the Yikes! router agent. (The Yikes! agent is not depicted in Figure 7-4 because it is not involved in installing MUD-based access control rules for the MUD-capable device.) The MUD file server is in the cloud, as are the device's update server and the Yikes! cloud service. (Again, only the MUD file server is depicted in Figure 7-4 because it is the only cloud component that is involved in installing MUD-based access control rules for the MUD-capable device.)

As shown in Figure 7-4, the message flow is as follows:

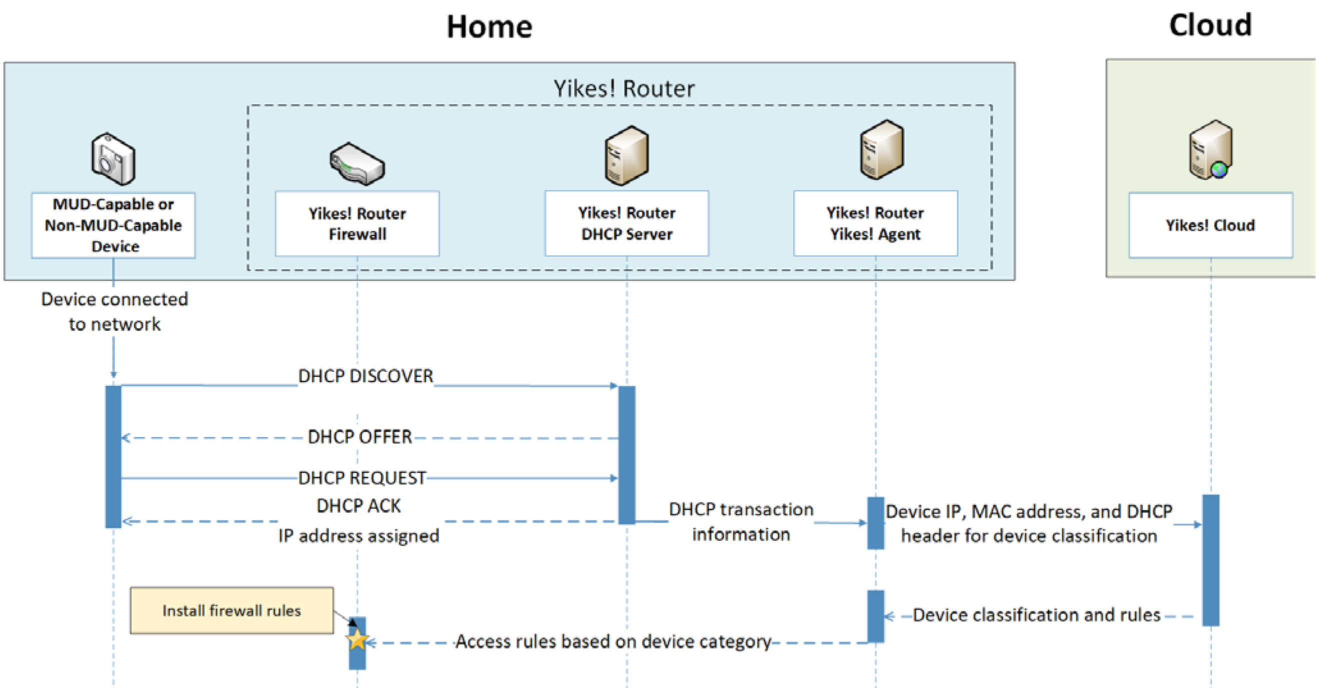
- When a MUD-capable IoT device is connected to the home/small-business network in Build 2, it exchanges DHCP protocol messages with the DHCP server on the router to obtain an IP address. The IoT device provides its MUD file URL within the DHCP DISCOVER message, as specified in the MUD RFC.
- The DHCP server forwards the MUD file URL and the MAC address of the connecting device to the MUD manager.
- The MUD manager registers the MAC address and MUD file URL of the device in the database that is located on the router.
- The MUD manager fetches the MUD file and the MUD file signature file from the MUD file server.
- After verifying that the MUD file is valid, the MUD manager installs the access control rules that correspond to the MUD file rules onto the router's firewall.

7.3.3.2 Installation of Category-Based Access Control Rules for All Devices

Figure 7-5 depicts the message flows involved in the process of installing category-based access control rules for all devices in Build 2 (both MUD-capable and non-MUD-capable devices), which are as follows:

- When a device is connected to the home/small-business network in Build 2, it exchanges DHCP protocol messages with the DHCP server to obtain an IP address. If it is a MUD-capable device, it also includes a MUD URL in this DHCP protocol exchange, and the message flow depicted in Figure 7-4 occurs in addition to the following message flow that is depicted in Figure 7-5. If it is a non-MUD-capable device, it does not include a MUD URL in this DHCP protocol exchange, and only the following message flow occurs.
- The DHCP server forwards information relevant to the connecting device such as IP address, MAC address, and DHCP header to the Yikes! router agent.
- The Yikes! router agent, in turn, forwards this information to the Yikes! cloud so the cloud can try to identify and classify the device.
- The Yikes! cloud sends the Yikes! router agent its determination of the device's category and associated traffic rules.
- The Yikes! router agent then configures the router with firewall rules for the device based on the device's category. Note that for this process to work, it is assumed that the Yikes! cloud has been preconfigured with various categories and traffic profile rules pertaining to each category. These rules can be configured by a user at any time by using the Yikes! mobile application.
- Note that if a device is MUD-capable and its MUD file rules conflict with its Yikes! category rules, both the device MUD rules and Yikes! category rules are installed, but the MUD rules take precedence and are enforced first.

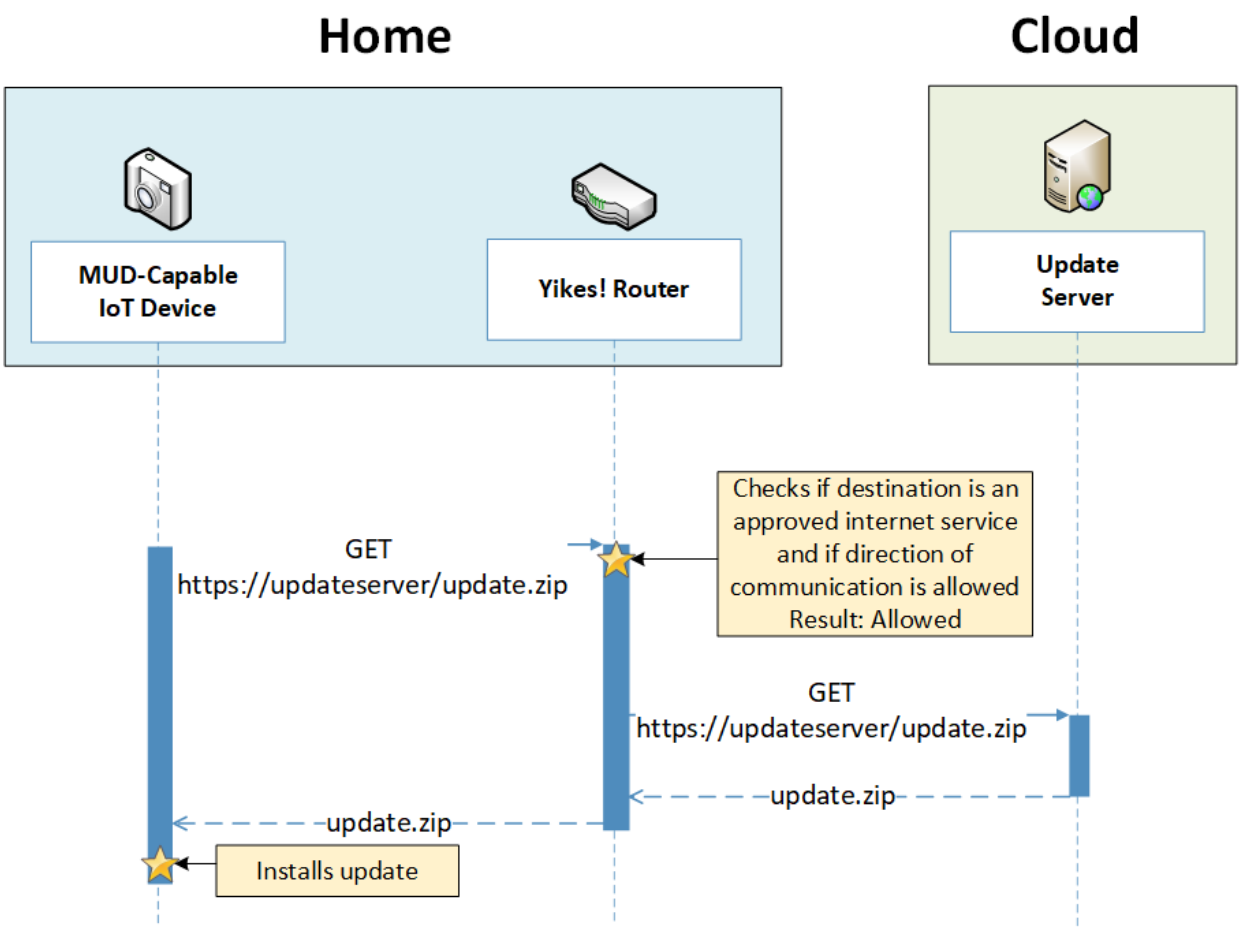
Figure 7-5 All Device Category-Based ACL Installation Message Flow—Build 2



1945 ## 7.3.3.3 Updates

1946 After a device has been permitted to connect to the home/small-business network, it should
1947 periodically check for updates. The message flow for updating the IoT device is shown in Figure 7-6
1948 Update Process Message Flow—Build 2.

1949 **Figure 7-6 Update Process Message Flow—Build 2**



1950 As shown in Figure 7-6 Update Process Message Flow—Build 2, the message flow is as follows:

1951 - The device generates an https GET request to its update server.

1952 - The Yikes! router will consult the firewall rules for this device to verify that it is permitted to
1953 send traffic to the update server. Assuming there were explicit rules in the device's MUD file
1954 enabling it to send messages to this update server, the Yikes! router will forward the request to
1955 the update server.

1956 - The update server will respond with a zip file containing the updates.

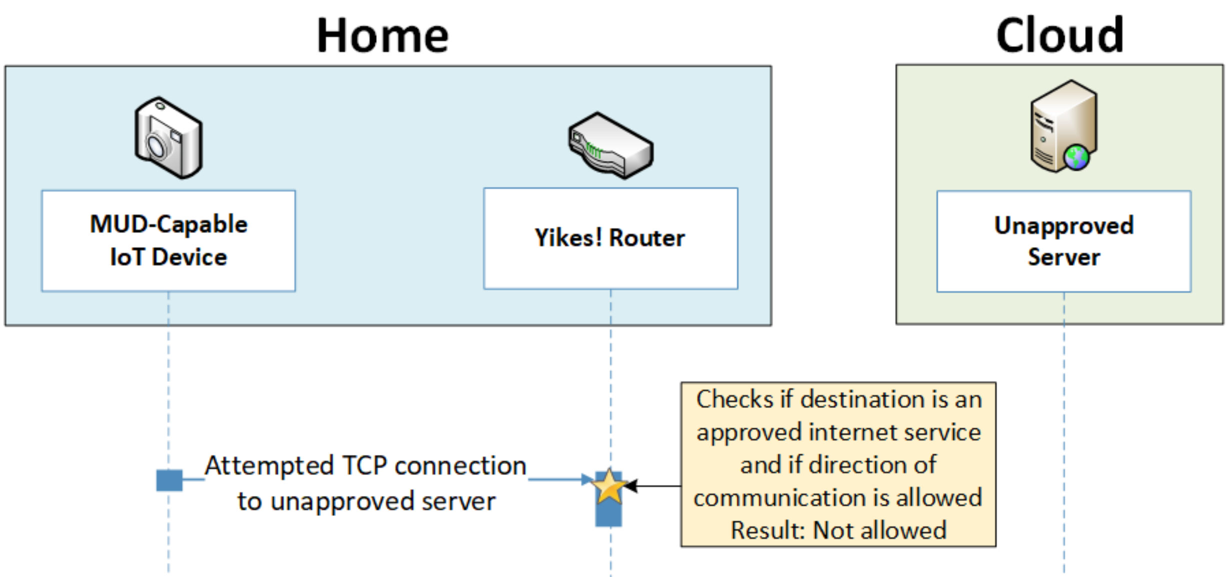
1957 - The Yikes! router will forward this zip file to the device for installation.

7.3.3.4 Prohibited Traffic

Figure 7-7 shows an attempt to send traffic that is prohibited by the MUD file and so is blocked by the Yikes! router.

- A connection attempt is made from a local IoT device to an unapproved server. (The unapproved server is located at a domain to which the MUD file does not explicitly permit the IoT device to send traffic.)
- This connection attempt is blocked because there is no firewall rule in the Yikes! router that permits traffic from the IoT device to the unapproved server.

Figure 7-7 Unapproved Communications Message Flow—Build 2



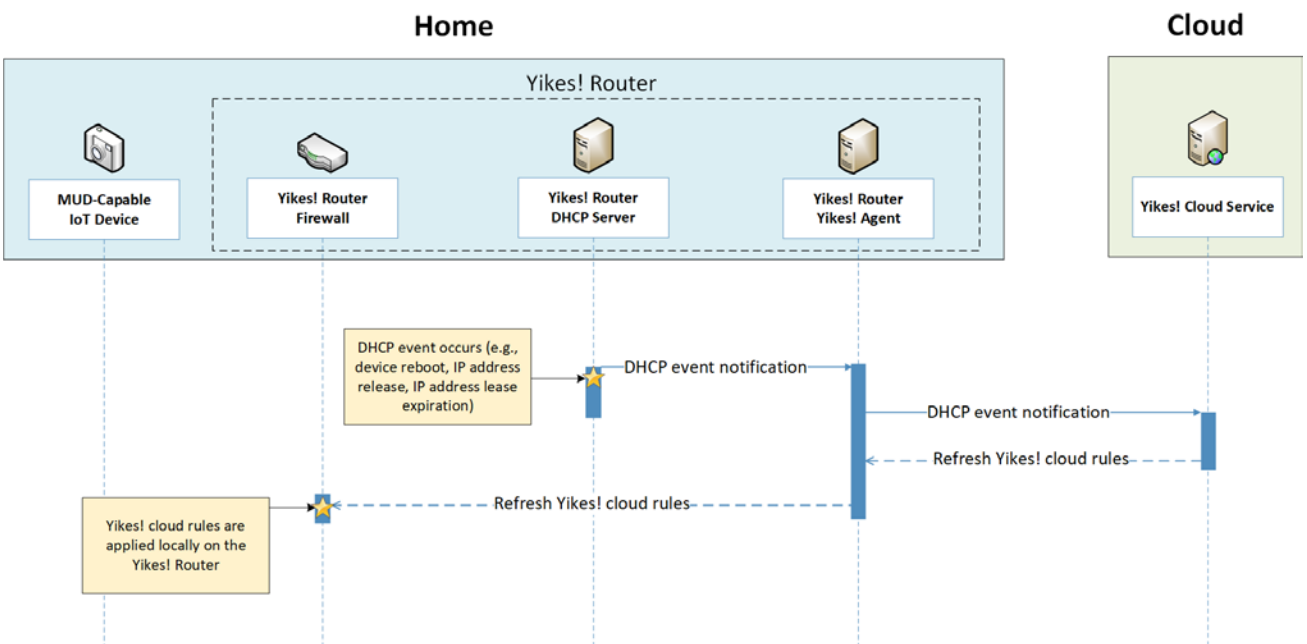
7.3.3.5 DHCP Events

Figure 7-8 shows the message flow when a change of DHCP state occurs, for example, when a device's IP address is assigned to a newly connected device, a lease expires, or a lease is explicitly released by the device. The Yikes! agent is triggered to send a notification to the Yikes! cloud to update or refresh the Yikes! cloud rules on the router when a DHCP event occurs. This update refreshes the firewall rules defined at the device category level that have been configured through the Yikes! cloud to be applied onto the Yikes! router. Figure 7-8 shows the following message flow:

- The DHCP event triggers a notification that is sent to the Yikes! router Yikes! agent.
- The Yikes! router Yikes! agent forwards the notification to the Yikes! cloud service.

- The Yikes! cloud service responds by sending a refresh of all Yikes! cloud rules to the Yikes! router agent.
- The Yikes! router Yikes! agent installs these refreshed rules onto the Yikes! router firewall.

Figure 7-8 DHCP Event Message Flow—Build 2



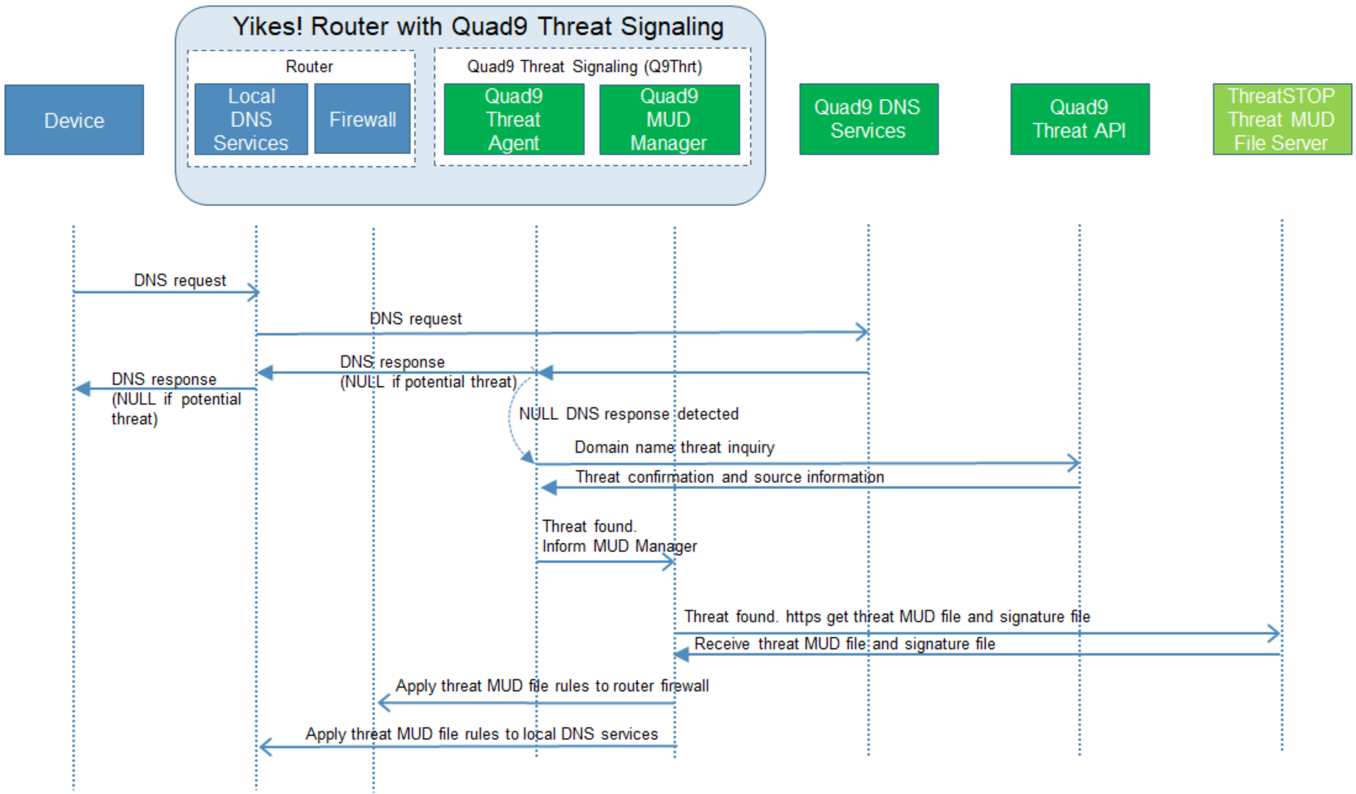
7.3.3.6 Threat Signaling

Figure 7-9 shows the message flow required to support threat signaling in Build 2.

- A local device (which may or may not be an IoT device and may or may not be MUD-capable) sends a DNS resolution request to its local DNS service, which is hosted on the Yikes! router.
- If the local DNS service cannot resolve the request itself, it will forward the request to the Quad9 DNS service.
- The Quad9 DNS service receives input from several threat intelligence providers (not depicted in the diagram) so the providers are aware of whether the domain in question has been identified to be unsafe. If the domain has not been identified as unsafe, the Quad9 DNS service will respond with the IP address(es) corresponding to the domain (as would any normal DNS service). If the domain has been flagged as unsafe, however, the Quad9 DNS service will not resolve the domain. Instead, it will return an empty (null) DNS response message to the local DNS service.

- The local DNS service will forward the DNS response to the device that originally made the DNS resolution request.
- Meanwhile, the Quad9 threat agent that is running on the Yikes! router monitors all DNS requests and responses. When it sees a domain that does not get resolved, it sends a query to the Quad9 threat API asking whether the domain is dangerous and, if so, which threat intelligence provider had flagged it as such and with what threat it is associated (this query is labeled “Domain name threat inquiry” in Figure 7-9).
- The Quad9 threat API responds with this information, which, in this case, informs the threat agent that if it wants more information about the blocked domain, it should contact ThreatSTOP (a threat intelligence provider) and request a threat MUD file. This threat MUD file will list domains and IP addresses that should be blocked because they are all associated with the same threat campaign as this threat.
- Next, the Quad9 threat agent provides this information to the Quad9 MUD manager.
- The Quad9 MUD manager requests and receives this threat MUD file and the threat MUD file signature file from the ThreatSTOP threat MUD file server.
- After ensuring that the threat MUD file is valid, the Quad9 MUD manager uses the threat MUD file to configure the router’s firewall to block all domains and IP addresses listed in this threat MUD file.
- The Quad9 MUD manager also configures the router’s local DNS services to provide empty responses for DNS requests that are made for all domains that are listed in the threat MUD file.

2013 **Figure 7-9 Message Flow for Protecting Local Devices Based on Threat Intelligence—Build 2**



2014 ## 7.4 Functional Demonstration

2015 A functional evaluation and a demonstration of Build 2 were conducted that involved two types of
2016 activities:

2017 - Evaluation of conformance to the MUD RFC—Build 2 was tested to determine the extent to
2018 which it correctly implements basic functionality defined within the MUD RFC.

2019 - Demonstration of additional (non-MUD-related) capabilities—It did not verify the example
2020 implementation's behavior for conformance to a standard or specification; rather, it
2021 demonstrated advertised capabilities of the example implementation related to its ability to
2022 increase device and network security in ways that are independent of the MUD RFC. These
2023 capabilities may provide security for both non-MUD-capable and MUD-capable devices.
2024 Examples of this type of activity include device discovery, identification and classification, and
2025 support for threat signaling.

Table 7-2 summarizes the tests used to evaluate Build 2’s MUD-related capabilities, and Table 7-3 summarizes the exercises used to demonstrate Build 2’s non-MUD-related capabilities. Both tables list each test or exercise identifier, a summary of the test or exercise, the test or exercise’s expected and observed outcomes, and the applicable Cybersecurity Framework Subcategories and NIST SP 800-53 controls for which each test or exercise verifies support. The tests and exercises listed in the table are detailed in a separate supplement for functional demonstration results. Boldface text is used to highlight the gist of the information that is being conveyed.

Table 7-2 Summary of Build 2 MUD-Related Functional Tests

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
IoT-1	**ID.AM-1:** Physical devices and systems within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-2:** Software platforms and applications within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **DE.AE-1:** A baseline of network operations and expected data flows for users and systems is established and managed. **PR.AC-4:** Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties.	A **MUD-capable IoT device is configured to emit a MUD URL within a DHCP message.** The DHCP server assigns its IP address and extracts the MUD URL, which is sent to the MUD manager. The MUD manager requests the MUD file and signature from the MUD file server, and the MUD file server serves the MUD file to the MUD manager. The MUD file explicitly permits traffic to/from some internet services and hosts and implicitly denies traffic to/from all other internet services. **The MUD manager translates the MUD file information into local network configurations that it installs on the router or switch that is serving**	Upon connection to the network, the MUD-capable IoT device has its MUD **PEP router/switch automatically configured according to the MUD file’s route-filtering policies.**	Pass

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	NIST SP 800-53 Rev. 4 AC-1, AC-2, AC-3, AC-5, AC-6, AC-14, AC-16, AC-24 **PR.AC-5:** Network integrity is protected, incorporating network segregation where appropriate. **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7 **PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10 **PR.IP-3:** Configuration change control processes are in place. **NIST SP 800-53 Rev. 4** CM-3, CM-4, SA-10 **PR.PT-3:** The principle of least functionality is incorporated by configuring systems to provide only essential capabilities. **NIST SP 800-53 Rev. 4** AC-3, CM-7 **PR.DS-2:** Data in transit is protected.	**as the MUD PEP for the IoT device.**		
IoT-2	**PR.AC-7:** Users, devices, and other assets are authenticated (e.g., single-factor, multifactor) commensurate with the risk of the transaction (e.g., individuals' security and privacy risks and other organizational risks). **NIST SP 800-53 Rev. 4** AC-7, AC-8, AC-9, AC-11, AC-12, AC-14, IA-1, IA-	A MUD-capable IoT device is configured to emit a URL for a MUD file, but the **MUD file server that is hosting that file does not have a valid TLS certificate. Local policy has been configured to ensure that if the MUD file for**	When the MUD-capable IoT device is connected to the network, the MUD manager sends locally defined policy to the router/switch that handles whether to allow or block traffic to the	Pass

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	2, IA-3, IA-4, IA-5, IA-8, IA-9, IA-10, IA-11	**an IoT device is located on a server with an invalid certificate, the router/switch will be configured by local policy to allow all communication to/from the device.**	MUD-capable IoT device. Therefore, the **MUD PEP router/switch will be configured to allow all traffic to and from the IoT device.**	
IoT-3	**PR.DS-6:** Integrity-checking mechanisms are used to verify software, firmware, and information integrity. **NIST SP 800-53 Rev. 4** SI-7	A MUD-capable IoT device is configured to emit a URL for a MUD file, but the **certificate that was used to sign the MUD file had already expired at signing. Local policy has been configured to ensure that if the MUD file for a device has a signature that was signed by a certificate that had already expired at the time of signature, the device's MUD PEP router/switch will be configured by local policy to either allow or deny all communication to/from the device.**	When the MUD-capable IoT device is connected to the network and the MUD file and signature are fetched, the MUD manager will detect that the MUD file's signature was created by using a certificate that had already expired at signing. According to local policy, the **MUD PEP will be configured to either allow or block all traffic to/from the device.**	Pass
IoT-4	**PR.DS-6:** Integrity-checking mechanisms are used to verify software, firmware, and information integrity. **NIST SP 800-53 Rev. 4** SI-7	A MUD-capable IoT device is configured to emit a URL for a MUD file, but the **signature of the MUD file is invalid. Local policy has**	When the MUD-capable IoT device is connected to the network, the MUD manager sends locally defined policy to the router/switch	Pass

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
		been configured to ensure that if the MUD file for a device is invalid, the router/switch will allow all communication to/from the IoT device.	that handles whether to allow or block traffic to the MUD-capable IoT device. Therefore, the **MUD PEP router/switch will be configured to allow all traffic to and from the IoT device.**	
IoT-5	**ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10 **PR.PT-3:** The principle of least functionality is incorporated by configuring systems to provide only essential capabilities. **NIST SP 800-53 Rev. 4** AC-3, CM-7	Test IoT-1 has run successfully, meaning that the MUD PEP router/switch has been configured based on a **MUD file that permits traffic to/from some internet locations and implicitly denies traffic to/from all other internet locations.**	When the MUD-capable IoT device is connected to the network, its MUD PEP **router/switch will be configured to enforce the route filtering that is described in the device's MUD file** with respect to traffic being permitted to/from some internet locations, and traffic being implicitly blocked to/from all remaining internet locations.	Pass (for testable procedure, ingress cannot be tested due to Network Address Translation [NAT])
IoT-6	**ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8	Test IoT-1 has run successfully, meaning that the MUD PEP router/switch has been	When the MUD-capable IoT device is connected to the network, its MUD	Pass

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	PR.DS-3: Assets are formally managed throughout removal, transfers, and disposition. **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **PR.AC-5:** Network integrity is protected, incorporating network segregation where appropriate. **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7 **PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **PR.IP-3:** Configuration change control processes are in place. **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10 **PR.PT-3:** The principle of least functionality is incorporated by configuring systems to provide only essential capabilities. **NIST SP 800-53 Rev. 4** AC-3, CM-7	configured based on a **MUD file that permits traffic to/from some lateral hosts and implicitly denies traffic to/from all other lateral hosts.** (The MUD file does not explicitly identify the hosts as lateral hosts; it identifies classes of hosts to/from which traffic should be denied, where one or more hosts of this class happen to be lateral hosts.)	PEP **router/switch will be configured to enforce the access control information that is described in the device's MUD file** with respect to traffic being permitted to/from some lateral hosts, and traffic being implicitly blocked to/from all remaining lateral hosts.	
IoT-7	**PR.IP-3:** Configuration change control processes are in place. **NIST SP 800-53 Rev. 4** CM-3, CM-4, SA-10 **PR.DS-3:** Assets are formally managed throughout removal, transfers, and disposition.	Test IoT-1 has run successfully, meaning that the MUD PEP **router/switch has been configured based on the MUD file** for a specific MUD-capable device in question.	When the MUD-capable **IoT device explicitly releases its IP address lease,** the MUD-related configuration for that IoT device will be removed from its	Pass

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
		Next, have **the IoT device change DHCP state by explicitly releasing its IP address lease, causing the device's policy configuration to be removed from the MUD PEP router/switch.**	MUD PEP router/switch.	
IoT-8	**PR.IP-3:** Configuration change control processes are in place. **NIST SP 800-53 Rev. 4** CM-3, CM-4, SA-10 **PR.DS-3:** Assets are formally managed throughout removal, transfers, and disposition.	Test IoT-1 has run successfully, meaning that the MUD PEP **router/switch has been configured based on the MUD file** for a specific MUD-capable device in question. Next, have **the IoT device change DHCP state by waiting until the IoT device's address lease expires, causing the device's policy configuration to be removed from the MUD PEP router/switch.**	When the MUD-capable **IoT device's IP address lease expires,** the MUD-related configuration for that IoT device will be removed from its MUD PEP router/switch.	Pass
IoT-9	**ID.AM-1:** Physical devices and systems within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-2:** Software platforms and applications within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-3:** Organizational communication and data flows are mapped.	Test IoT-1 has run successfully, meaning the MUD PEP **router/switch has been configured based on the MUD file** for a specific MUD-capable device in question. The MUD file contains domains that resolve to multiple IP addresses.	A domain in the MUD file resolves to two different IP addresses. The MUD manager will create firewall rules that permit the MUD-capable device to send traffic to both IP addresses. The MUD-	Pass

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	NIST SP 800-53 Rev. 4 AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **DE.AE-1:** A baseline of network operations and expected data flows for users and systems is established and managed. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CM-2, SI-4 **PR.AC-4:** Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC-5, AC-14, AC-16, AC-24 **PR.AC-5:** Network integrity is protected, incorporating network segregation where appropriate. **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7 **PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10 **PR.IP-3:** Configuration change control processes are in place. **NIST SP 800-53 Rev. 4** CM-2, CM-3, SA-10	The MUD PEP router/switch should be configured to permit communication to or from all IP addresses for the domain.	capable device attempts to send traffic to each of the IP addresses, and the MUD PEP router/switch permits the traffic to be sent in both cases.	

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	PR.DS-2: Data in transit is protected. **NIST SP 800-53 Rev. 4** SC-8, SC-11, SC-12			
IoT-10	**ID.AM-1:** Physical devices and systems within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-2:** Software platforms and applications within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **DE.AE-1:** A baseline of network operations and expected data flows for users and systems is established and managed. **PR.AC-4:** Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC-5, AC-6, AC-14, AC-16, AC-24 **PR.AC-5:** Network integrity is protected, incorporating network segregation where appropriate. **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7	A MUD-capable IoT device is configured to emit a MUD URL. Upon being connected to the network, its MUD file is retrieved, and the PEP is configured to enforce the policies specified in that MUD URL for that device. **Within 24 hours (i.e., within the cache-validity period for that MUD file), the IoT device is reconnected to the network.** After 24 hours have elapsed, the same device is reconnected to the network.	Upon reconnection of the IoT device to the network, **the MUD manager does not contact the MUD file server. Instead, it uses the cached MUD file.** It translates this MUD file's contents into appropriate route-filtering rules and installs these rules onto the PEP for the IoT device. Upon reconnection of the IoT device to the network, after 24 hours have elapsed, the MUD manager does fetch a new MUD file.	Not testable in preproduction implementation

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	PR.IP-1: A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10 **PR.IP-3:** Configuration change control processes are in place. **NIST SP 800-53 Rev. 4** CM-3, CM-4, SA-10 **PR.PT-3:** The principle of least functionality is incorporated by configuring systems to provide only essential capabilities. **NIST SP 800-53 Rev. 4** AC-3, CM-7 **PR.DS-2:** Data in transit is protected.			
IoT-11	**ID.AM-1:** Physical devices and systems within the organization are inventoried.	A **MUD-enabled IoT device can emit a MUD URL.** The device should leverage one of the specified manners for emitting a MUD URL.	Upon initialization, the MUD-enabled IoT device broadcasts a DHCP message on the network, including at most one **MUD URL, in https scheme, within the DHCP transaction.**	Pass

In addition to supporting MUD, Build 2 can identify a device's make (i.e., manufacturer) and model, categorize devices based on their make and model, and associate device categories with traffic policies that affect both internal and external traffic transmissions, as shown in Table 7-3.

Table 7-3 Non-MUD-Related Functional Capabilities Demonstrated

Exercise	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Exercise Summary	Expected Outcome	Observed Outcome
YnMUD-1	**ID.AM-1:** Physical devices and systems within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-2:** Software platforms and applications within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **DE.AE-1:** A baseline of network operations and expected data flows for users and systems is established and managed. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CM-2, SI-4 **DE.CM-1:** The network is monitored to detect potential cybersecurity events. **NIST SP 800-53 Rev. 4** AC-2, AU-12, CA-7, CM-3, SC-5, SC-7, SI-4	A device identification and a categorization capability are supported by the router and cloud services. The **router is designed to detect all devices connected to the network and leverage cloud services to identify the devices using attributes associated with them, as well as categorize the devices by type when possible. If unable to identify and categorize them, devices are designated as uncategorized.**	Upon being connected to the network, the **router detects all connected devices and leverages a cloud service, which identifies each device's make and model using attributes** (e.g., type, IP address, OS), and **categorizes them** (e.g., cell phone, printer, smart appliance).	As expected
YnMUD-2	**ID.AM-1:** Physical devices and systems within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-3:** Organizational communication and data flows are mapped.	After executing YnMUD-1 successfully, the **UI is used to modify make, model, and/or category of connected devices.**	Connected devices have been identified and categorized automatically upon being connected to the network. Using the UI, show that the make and model of	As expected

Exercise	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Exercise Summary	Expected Outcome	Observed Outcome
			a device can be modified, and that the category of the device can be assigned manually.	
YnMUD-3	**ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **ID.AM-4:** External information systems are catalogued. **NIST SP 800-53 Rev. 4** AC-20, SA-9 **PR.AC-1:** Identities and credentials are issued, managed, verified, revoked, and audited for authorized devices, users and processes. **NIST SP 800-53 Rev. 4** AC-1, AC-2, IA-1, IA-2, IA-3, IA-4, IA-5, IA-6, IA-7, IA-8, IA-9, IA-10, IA-11 **NIST SP 800-53 Rev. 4** PE-2, PE-3, PE-4, PE-5, PE-6, PE-8 **PR.AC-3:** Remote access is managed. **NIST SP 800-53 Rev. 4** AC-1, AC-17, AC-19, AC-20, SC-15 **PR.AC-4:** Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC- 5, AC-6, AC-14, AC-16, AC-24 **PR.AC-5:** Network integrity is protected (e.g., network segregation, network segmentation).	**The router can apply traffic policies to categories of devices that restrict initiation of (south-to-north) communications to internet sites** by all devices in the specified category. Communication **can be configured to (a) allow all internet communication, (b) deny all internet communication to devices of a specific make and model, or (c) permit communication only to/from specified internet domains and devices of a specific make and model.**	Through the UI, device category rules can be defined to permit connectivity to every internet location by selecting “Allow All Internet Traffic” or to device-specific sites by selecting “IoT specific sites.” Set rules for the **computer category to permit all internet traffic,** and attempt to initiate communication from laptop to any internet host. **All internet communication from laptop will be approved.** Next, set rules for **Smart Appliance category to permit IoT-specific site,** and attempt to initiate communication to specific sites permitted for the make and model of the device being tested. **All specified sites for device make and model should be**	As expected

Exercise	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Exercise Summary	Expected Outcome	Observed Outcome
	NIST SP 800-53 Rev. 4 AC-4, AC-10, SC-7		**permitted, and any other communication outside these specified hosts should be blocked.** Last, set rules for **a third type of device category (cell phone) to permit IoT-specific sites, but do not specify any sites as permissible. The device should not be permitted to initiate communication with any internet sites.**	
YnMUD-4	**ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **ID.AM-4:** External information systems are catalogued. **NIST SP 800-53 Rev. 4** AC-20, SA-9 **PR.AC-1:** Identities and credentials are issued, managed, verified, revoked, and audited for authorized devices, users, and processes. **NIST SP 800-53 Rev. 4** AC-1, AC-2, IA-1, IA-2, IA-3, IA-4, IA-5, IA-6, IA-7, IA-8, IA-9, IA-10, IA-11 **PR.AC-3:** Remote access is managed. **NIST SP 800-53 Rev. 4** AC-1, AC-17, AC-19, AC-20, SC-15	The router can apply **policies to categories of devices** (as defined by a user through the UI) to **specify rules regarding initiation of lateral (east/west) communications to other categories of devices on the local network. All traffic is enforced according to** rules associated with **the device's category.**	**Through the UI,** device category rules can be defined to **permit connectivity between categories of devices.** Set rules for **category x to permit communication with category y but not to category z.** After rules have been set, **attempt to communicate from a device in category x to a device in category y;** the router will **permit this communication** to occur. Next, **attempt to communicate from**	As expected

Exercise	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Exercise Summary	Expected Outcome	Observed Outcome
	PR.AC-4: Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC- 5, AC-6, AC-14, AC-16, AC-24 **PR.AC-5:** Network integrity is protected (e.g., network segregation, network segmentation). **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7		**a device in category x to a device in category z;** the router **will not permit this communication** to occur.	
YnMUD-5	**ID.RA-2:** Cyber threat intelligence is received from information-sharing forums and sources. **NIST SP 800-53 Rev. 4** SI-5, PM-15, PM-16 **ID.RA-3:** Threats, both internal and external, are identified and documented. **NIST SP 800-53 Rev. 4** RA-3, SI-5, PM-12, PM-16 **ID.RA-5:** Threats, vulnerabilities, likelihoods, and impacts are used to determine risk. **NIST SP 800-53 Rev. 4** RA-2, RA-3, PM-16 **PR.AC-4:** Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties.	The router can query a threat intelligence provider and receiving threat information related to domains that devices on the network are attempting to access. In **response to threat information, all devices on the local network are prohibited from visiting specific domains and IP addresses.**	A **device on the network sends a DNS request for a malicious domain** to which it is attempting to navigate. The **router receives a response indicating that the domain is potentially malicious. The router queries** threat **services** regarding the domain and receives back the URL for the threat MUD file that is associated with the domain. The router retrieves the threat MUD file and installs its rules as global firewall rules. As a result, the **device that attempted to communicate with the dangerous**	As expected

Exercise	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Exercise Summary	Expected Outcome	Observed Outcome
			domain is blocked from communicating with that domain as well as all other domains associated with that same threat.	
YnMUD-6	**PR.AC-3:** Remote access is managed. **NIST SP 800-53 Rev. 4** AC-1, AC-17, AC-19, AC-20, SC-15 **PR.AC-4:** Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC- 5, AC-6, AC-14, AC-16, AC-24 **PR.AC-5:** Network integrity is protected (e.g., network segregation, network segmentation). **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7 **ID.RA-2:** Cyber threat intelligence is received from information-sharing forums and sources. **NIST SP 800-53 Rev. 4** SI-5, PM-15, PM-16 **ID.RA-3:** Threats, both internal and external, are identified and documented. **NIST SP 800-53 Rev. 4** RA-3, SI-5, PM-12, PM-16	YnMUD-5 was successfully completed, i.e., **in response to threat information received in YnMUD-5, all devices on the local network are prohibited from visiting not only the domains that are associated with the identified threat but also with all IP addresses associated with these domains.**	A **different device on the network attempts to communicate with the malicious domain identified in test YnMUD-5 via its IP address instead of its domain.** Router firewall rules prohibiting access to this IP address should already be present as a result of test **YnMUD-5. As** a result, the **device that attempted to communicate to the IP address is prevented from initiating communication**.	As expected
YnMUD-7	**PR.AC-3:** Remote access is managed. **NIST SP 800-53 Rev. 4** AC-1, AC-17, AC-19, AC-20, SC-15	YnMUD-5 was successfully completed, resulting in the router being configured with threat	Log in to the router and verify that the firewall rules that	As expected

Exercise	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Exercise Summary	Expected Outcome	Observed Outcome
	PR.AC-4: Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC- 5, AC-6, AC-14, AC-16, AC-24 **PR.AC-5:** Network integrity is protected (e.g., network segregation, network segmentation). **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7 **ID.RA-2:** Cyber threat intelligence is received from information-sharing forums and sources. **NIST SP 800-53 Rev. 4** SI-5, PM-15, PM-16 **ID.RA-3:** Threats, both internal and external, are identified and documented. **NIST SP 800-53 Rev. 4** RA-3, SI-5, PM-12, PM-16	intelligence rules**. The threat intelligence was received more than 24 hours earlier.** It indicated domains and IP addresses that should not be trusted, and those domains and IP addresses were blocked by firewall rules installed on the router. **After 24 hours, these firewall rules have been removed from the router.**	prohibited communication to malicious domains (and that were verified as present in the previous two tests) are no longer present.	

7.5 Observations

Build 2 was able to successfully permit and block traffic to and from MUD-capable IoT devices as specified in the MUD files for the devices. It was also able to constrain communications to and from all devices (both MUD-capable and non-MUD-capable) based on the traffic profile associated with the device's category in the Yikes! cloud.

We observed the following limitations to Build 2 that are informing improvements to its current proof-of-concept implementation:

- MUD manager (version 1.1.3):
    - MUD file caching is not supported in this version of the MUD manager. The MUD manager fetches a new MUD file for every MUD request that occurs, regardless of the cache-validity of the current MUD file.

- Yikes! cloud:
    - Yikes! performs device identification using data available at the time a device requests an IP address during the network connection process. Future versions of the product may collect additional information about a device to improve the specificity of device identification.
- Yikes! mobile application:
    - At demonstration time, the Yikes! mobile application was under development. For this reason, Yikes! provided a web-hosted replica of the mobile application under development. This was accessible via web browsers on both mobile and computer platforms.
- Yikes! router (version 1.1.3):
    - At demonstration time, DHCP was the only MUD URL emission method supported. LLDP and X.509 MUD URL emission methods are not supported by the current version of the Yikes! router.
    - When MUD-capable devices are first connected and introduced to the network, the default policy in this version of the Yikes! router is to allow communications while the MUD file is being requested and processed. This results in a short period of time during which the device has received an IP address and is able to communicate unconstrained on the network before the MUD rules related to the device are applied.
    - In some situations, when a MUD-capable IoT device is connected to the network, the base router configurations may contend with the MUD rules. This can result in the initial instances of unapproved attempted communication from the MUD-capable device to other devices on the local network being permitted until the router reconciles the configuration. Traffic to or from locations outside the local network is not impacted and only approved traffic is ever allowed.
    - At demonstration time, the automated process to associate the Yikes! router with the Yikes! cloud service was still under development, and association had to be done manually by MasterPeace.
- threat signaling (version 0.4.0):
    - Access to threat-signaling information is triggered when a device on the local network makes a DNS resolution request for a domain that has been flagged as dangerous because it is associated with some known threat. If a device attempts to connect to a dangerous site using that site's IP address rather than its domain name without first attempting to resolve a domain name that is associated with the same threat that is associated with the dangerous site, the threat-signaling mechanism provided in Build 2 will not block access to that IP address. Therefore, users are cautioned to use domain names rather than IP addresses when attempting outbound communication to ensure that they can take full advantage of the threat-signaling protections offered by Build 2.

8 Build 3

The Build 3 implementation uses equipment supplied by CableLabs to onboard devices and to support MUD. Build 3 leverages the Wi-Fi Alliance's Wi-Fi Easy Connect specification to securely onboard devices to the network (i.e., to provision each device with the unique network credentials that it needs to connect to the network). It also uses SDN to create separate trust zones (e.g., network segments) to which devices are assigned according to their intended network function. The Build 3 network platform is called Micronets, and there is an open-source reference implementation of the Micronets platform available on the Micronets project site as well as on GitHub. The Micronets platform is continually evolving with new features and functionality being added to its open-source reference implementation.

Micronets consists of:

- an on-premises Micronets-capable gateway that resides on the home/small-business network. A micronet is a trust zone that is implemented as a network segment and is used to group devices together into trust domains that isolate devices based on their function and access policy. The Micronets Gateway manages and enforces service-specific micronets and customer-defined micronets. Cloud-based microservices layer that hosts various Micronets services (e.g., SDN controller, Micronets Manager, MUD Manager, Configuration microservice, identity server (optional), DHCP/DNS configuration services) that interact with the on-premises Micronets Gateway to manage local devices and network connectivity. The most important of these is the Micronets Manager, which interacts with all of the other microservices to coordinate the state of the Micronets-enabled on-premises network.
- Cloud-based Intelligent Services and Business Logic layer (e.g., machine-learning-based services) that is operated by the service provider.
- Micronets APIs, which allow partners and service providers to interface with a customer's micro-networks environment to provision and deliver specific customer-requested services.
- Micronets Mobile App that supports device onboarding using the Wi-Fi Easy Connect protocol

These various components may be used in combination to onboard devices and leverage MUD, if supported by the device. The on-premises Micronets Gateway supports the Wi-Fi Easy Connect protocol for IoT device onboarding and leverages it to provision the device in the right trust domain. This Micronets Gateway can enforce policy-based flows where the policies can be derived from MUD-based traffic constraints or other policy sources. It also supports dynamic micro-segmentation.
CableLabs provided prototype Micronets platform components in the NCCoE lab based on the open-source reference implementation available on GitHub. A more detailed description of Micronets can be found in CableLabs' Micronets white paper, and the various Micronets components listed above are each described more fully in Section 8.3.1.

8.1 Collaborators

Collaborators that participated in this build are described briefly in the subsections below.

8.1.1 CableLabs

CableLabs is an innovation and R&D laboratory for the cable industry. CableLabs is a not-for-profit global network technologies organization with member companies around the world. CableLabs offers state-of-the-art research and innovation facilities with collaborative ecosystem made up of thousands of vendors. In November 2018, CableLabs publicly announced Micronets, a next-generation on-premise network platform. Micronets provides adaptive security for all devices connecting to residential or small-business networks through dynamic micro-segmentation and management of connectivity to those devices. Micronets is designed to provide seamless and transparent security to users without burdening them with the technical aspects of configuring the network. Micronets incorporates and leverages MUD as one technology component to help identify and manage the connectivity of devices, in support of the broader Micronets platform. It also leverages the Wi-Fi Easy Connect protocol to enable IoT devices to be onboarded easily and securely, and to provide each IoT device with unique network credentials. In addition, Micronets can provide enhanced security for high-value or sensitive devices, further reducing the risk of compromise for these devices and their applications. Learn more about CableLabs at https://www.cablelabs.com.

8.1.2 DigiCert

See Section 6.1.2 for a description of DigiCert.

8.2 Technologies

Table 8-1 lists all the products and technologies used in Build 3 and provides a mapping among the generic component term, the specific product used to implement that component, and the function subcategories that the product provides. When applicable, both the function subcategories that a component provides directly and those that it supports, but does not provide directly, are listed and labeled as such. For rows in which the provides/supports distinction is not noted, all listed categories are directly provided by the component. Refer to Table 5-1 for an explanation of the Cybersecurity Framework’s Subcategory codes.

2148 **Table 8-1 Products and Technologies**

Component	Product	Function	Cybersecurity Framework Subcategories
MUD manager	Service provider's cloud infrastructure MUD Manager component	Fetches, verifies, caches, and processes MUD files from the MUD file server; provides parsed MUD rules as ACLs to the Micronets Manager that is on the service provider cloud, which will send these ACLs to the home/small-business network Micronets Gateway, which will convert them into traffic flow rules.	Provides: PR.PT-3 Supports: ID.AM-1 ID.AM-2 ID.AM-3 PR.AC-4 PR.AC-5 PR.DS-5 DE.AE-1
MUD file server	A web server that hosts the device's MUD file	Hosts MUD files; serves MUD files to the MUD manager over https.	ID.AM-1 ID.AM-2 ID.AM-3 PR.AC-4 PR.AC-5 PR.DS-5 PR.PT-3 DE.AE-1
MUD file maker	MUD file maker (https://www.mud-maker.org/)	YANG script GUI used to create MUD files	ID.AM-1

Component	Product	Function	Cybersecurity Framework Subcategories
MUD file	A YANG model instance that has been serialized in JSON (RFC 7951). The manufacturer of a MUD-capable device creates that device's MUD file. MUD file maker (see previous row) can create MUD files. Each MUD file is also associated with a separate MUD signature file.	Specifies the communications that are permitted to and from a given device	Provides: PR.PT-3 Supports: ID.AM-1 ID.AM-2 ID.AM-3 PR.DS-5
Router/Switch	Micronets Gateway and access point	An integrated SDN-capable switch/router and Wi-Fi access point that routes traffic on the home/small-business network. During onboarding, receives ACLs that enforce the IoT device's MUD file rules from the Micronets Manager; creates flow rules to enforce these ACLs. Creates micronets (subnetworks) to separate devices into trust zones.	ID.AM-3 PR.AC-4 PR.AC-5 PR.DS-5 PR.PT-3 DE.AE-1

Component	Product	Function	Cybersecurity Framework Sub-categories
Certificates	DigiCert certificates (TLS and Premium)	Authenticate the MUD file server and secure the TLS connection between the MUD manager and the MUD file server and other components of the Micronets platform; sign MUD files and generate a corresponding signature file	PR.AC-1 PR.AC-3 PR.AC-5 PR.AC-7
MUD-capable IoT device	Raspberry Pi Model 3 B+ (devkit)	When put into onboarding mode, it displays a QR code that contains its Wi-Fi Easy Connect bootstrapping information (including elements that identify its MUD file) and begins listening for Wi-Fi Easy Connect protocol messages. After being authenticated by the Easy Connect-capable Micronets Gateway, the gateway provides it with its unique network credentials. Also requests and applies software updates	ID.AM-1

Component	Product	Function	Cybersecurity Framework Subcategories
Update server	NCCoE-hosted Apache server	Acts as a device manufacturer's update server that would communicate with IoT devices to provide patches and other software updates	PR.IP-1 PR.IP-3
Unapproved server	NCCoE-hosted Apache server	Acts as an internet host that has not been explicitly approved in a MUD file	DE.DP-3 DE.AM-1
MUD registry	Micronets MUD Registry	Provides a service that looks up each MUD-capable device's MUD file URL based on the contents of the information element field and the public key field in the device's Wi-Fi Easy Connect bootstrapping information	Provides: ID.AM-1 Supports: ID.AM-3 PR.DS-5 PR.IP-1

Component	Product	Function	Cybersecurity Framework Subcategories
SDN controller	Micronets Manager	Although the Micronets Manager does not use the OpenFlow protocol, it functions as an SDN controller by conveying to the Micronets Gateway the ACLs and micronet topology that it wants the gateway to create and enforce. During the onboarding process, it provides the gateway with device-specific configuration, including ACLs to enforce the communications profile specified in the device's MUD file; it also indicates the micronet (trust zone) to which each IoT device should be assigned (as directed by user input to the Micronets mobile application).	Supports: PR.AC-3 PR.AC-5 PR.AC-4 PR.DS-1 PR.DS-2 PR.DS-5 PR-PT-3

Component	Product	Function	Cybersecurity Framework Subcategories
onboarding manager	Micronets Configuration Microservice and Micronets Manager	Resides in the service provider cloud and manages the onboarding process. Receives the onboarding request and device bootstrapping information from the Micronets mobile phone application (via the multiple-system operator (MSO) portal) and provides it to the Micronets Gateway. Looks up the device's MUD file URL in the MUD registry, sends the MUD file URL to the MUD manager, receives back ACLs corresponding to the parsed MUD rules from the MUD manager, and provides these to the Micronets Gateway for enforcement	Supports: PR.AC-3 PR.AC-5 PR.AC-4 PR.DS-1 PR.DS-2 PR.DS-5 PR-PT-3

Component	Product	Function	Cybersecurity Framework Subcategories
User and device interface to the onboarding manager	Micronets mobile Phone application	Acts as both the Micronets Configuration Microservice's user interface and its device bootstrapping information reader. Collects device bootstrapping information from both the QR code and user input and sends this information with the onboarding request to the Micronets Manager's Configuration Microservice via the service provider's MSO portal	Supports: ID.AM-1 ID.AM-3 PR.AC-3 PR.AC-4 PR.AC-5 PR.DS-1 PR.DS-2 PR.IP-1 PR.PT-3
Bootstrapping interface to the onboarding manager	MSO portal	Receives the onboarding request from the Micronets mobile application and forwards it to the Micronets Configuration Microservice that is associated with the specific user/owner of the network and the device	Supports: ID.AM-1 ID.AM-3 PR.AC-3 PR.AC-4 PR.AC-5 PR.DS-1 PR.DS-2 PR.IP-1 PR.PT-3

Component	Product	Function	Cybersecurity Framework Subcategories
Network onboarding component	Wi-Fi Easy Connect-Capable Micronets Gateway	Based on bootstrapping and other information it receives from the onboarding manager (i.e., the Micronets Manager), interacts directly with each IoT device via the Wi-Fi Easy Connect protocol to authenticate the device, establish a secure channel with it, and provide it with its unique network credentials	Provides: PR.AC-3 PR.AC-4 PR.AC-7 Supports: PR.AC-5 PR.DS-1 PR.DS-2 PR.DS-5 PR.DS-6 PR.PT-3

Each of these components is described more fully in the following sections.

8.2.1 MUD Manager

The Micronets MUD manager is a component within the service provider cloud. During the onboarding process, the MUD manager receives the device's MUD URL from the Micronets Manager and checks its cache for the MUD file corresponding to the MUD URL. If the file is not cached or if it is cached but has been there too long, the MUD manager fetches the MUD file that is at this URL and the MUD file's signature file, verifies the MUD file based on this signature file, parses the MUD file, and generates ACLs for the device based on the MUD file. The MUD manager sends these ACLs to the Micronets Manager, which forwards them to the Micronets Gateway so it can create and install traffic flow rules to enforce the MUD file rules. The MUD manager generates ACLs for src-dnsname, dst-dnsname, my-controller, controller, same-manufacturer, manufacturer, and local-networks constructs that are specified in the MUD file. It supports both lateral east/west protection and appropriate access to internet sites (north/south protection).

8.2.2 MUD File Server

In the absence of a commercial MUD file server for this project, the NCCoE used a MUD file server that is hosted on a Linode server that is accessible via the internet. This file server stores the MUD files along with their corresponding signature files for the IoT devices used in the project. Upon receiving a GET request for the MUD files and signatures, it serves the request to the MUD manager by using https.

8.2.3 MUD File

Using the MUD file maker component referenced above in Table 8-1, it is possible to create a MUD file with the following contents:

- Internet communication class–access to cloud services and other specific internet hosts:
    - Host: updateserver (hosted internally at the NCCoE)
        - Protocol: TCP
        - Direction-initiated: from IoT device
        - Source port: any
        - Destination port: 80
- Controller class–access to **classes** of devices that are known to be controllers (could describe well-known services such as DNS or NTP):
    - Host: nccoe-server1.micronets.net
        - Protocol: TCP
        - Direction-initiated: from IoT device
        - Source port: any
        - Destination port: 1883
- Local-networks class–access to/from **any** local host for specific services (e.g., http or https):
    - Host: any
        - Protocol: TCP
        - Direction-initiated: from IoT device
        - Source port: any
        - Destination port: 80
- My-controller class–access to controllers specific to this device:
    - Controllers: mm.micronets.in
        - Protocol: TCP
        - Direction-initiated: from IoT device
        - Source port: any

    - Destination port: 80
- Same-manufacturer class–access to devices of the same manufacturer:
  - Same-manufacturer: null (to be filled in by the MUD manager]
    - Protocol: TCP
    - Direction-initiated: from IoT device
    - Source port: any
    - Destination port: 80
- Manufacturer class–access to devices of a specific manufacturer (identified by MUD URL):
  - Manufacturer: devicetype (URL decided by the device manufacturer)
    - Protocol: TCP
    - Direction-initiated: from IoT device
    - Source port: any
    - Destination port: 80

8.2.4 Signature file

According to the IETF MUD specification, "a MUD file MUST be signed using CMS as an opaque binary object." The MUD file (e.g., *nist-model-fe_northsouth.json*) was signed with the OpenSSL tool by using the command described in the specification (detailed in Volume C of this publication). A Premium certificate, requested from DigiCert, was leveraged to generate the signature file (e.g., *nist-model-fe_northsouth.p7s*). Once created, the signature file is stored on the MUD file server.

8.2.5 Router/Switch

This build uses the Micronets Gateway as the router/switch on the home/small-business network. The Micronets Gateway is an SDN-capable switch that interfaces with the Micronets Manager in the service provider cloud via a RESTful interface. The gateway receives ACLs and micronet topology information from the Micronets Manager that the gateway converts to traffic flow rules that it enforces. The gateway is also integrated with a Wi-Fi access point and supports connectivity for both wired and wireless components. In support of MUD, this gateway serves as the policy enforcement point for the access control rules that are defined in each device's MUD file. These access control rules are instantiated on the switch as traffic flow rules.

In support of MUD, the gateway implements north/south IP access control protection based on src-dnsname, dst-dnsname, my-controller, and controller constructs that are specified in the MUD file. The gateway is also responsible for creating and enforcing micronets, which segregate devices. Each micronet represents a distinct trust domain and, at a minimum, represents a distinct IP subnetwork. By definition, devices in the same micronet are permitted to communicate with one another without restrictions. However, devices in different micronets are not permitted to communicate with one another unless such communication is explicitly permitted by local communications rules in the devices' MUD files.

During the onboarding process, devices are manually assigned to micronets by user input that is provided to the Micronets mobile application after the device's QR code is scanned. If devices are assigned to micronets in a way that is consistent with the local communications rules that are in the devices' MUD files, then Micronets can serve as the mechanism to enforce those local communications restrictions for the devices. By default, devices that were onboarded in Build 3 were manually assigned to separate micronets to ensure that only local communications that were explicitly permitted in the devices' MUD files would be permitted.

Devices can talk to other devices in the same micronet without restrictions. (Cross-device communication can be enabled between micronets as needed.) Sorting devices into specific micronets for enforcing local communications restrictions defined in the MUD file cannot currently be performed automatically. However, future versions of the Micronets implementation may support automatic placement of devices into specific micronets based on the local communications rules defined in their MUD files, thereby using the communications constraints imposed by each micronet to enforce same-manufacturer, manufacturer, and local-networks constructs.

8.2.6 Certificates

DigiCert provisioned a Premium Certificate for signing the MUD files. The Premium Certificate supports the key extensions required to sign and verify CMS structures as required in the MUD specification. DigiCert certificates also authenticate the MUD file server; secure the TLS connection between the MUD manager and the MUD file server; and mutually authenticate the connection between the Micronets Manager and the micronets. All of the web services also use web certificates. Further information about DigiCert's CertCentral web-based platform, which allows provisioning and managing publicly trusted X.509 certificates, is in Section 6.2.8.

8.2.7 IoT Devices

This section describes the IoT devices used in the laboratory implementation. There are two distinct categories of devices: devices that support MUD and have a vendor code value in the information element field of their onboarding QR code to indicate the location of the device's MUD file server, i.e., MUD-capable IoT devices; and devices that do not support MUD and do not have a value in the information element field of their onboarding QR code, i.e., non-MUD-capable IoT devices. For more information regarding how the information element field value is used to locate the device's MUD file, see Section 8.3.1.1.

8.2.7.1 MUD-Capable IoT Devices

The project used several MUD-capable IoT devices, all of which were Micronets Raspberry Pi devkits.

8.2.7.1.1 Micronets Raspberry Pi (Devkit)

The Raspberry Pi devkit runs the Raspbian 9 operating system. It was provisioned with one Wi-Fi Easy Connect bootstrapping public/private key pair before it initiates onboarding. This device is capable of being placed in Easy Connect onboarding mode, at which point it begins displaying a QR code and listening for Wi-Fi Easy Connect protocol messages. The QR code that the device displays contain the device's bootstrapping information, which includes:

- the public bootstrapping key of the device. (i.e., the public key from the public/private key pair that has already been stored on the device)
- MAC address of device
- Wi-Fi-channel the device will use
- information element (i.e., a code that identifies a device vendor)

Note that if the information element field is empty, the device is not considered to be MUD-capable. If the information element field contains a value, however, this value identifies the device's manufacturer. It is assumed that each manufacturer has a well-known location for serving MUD files; therefore, the value in the information element field indicates the location of the device's MUD file server. The value in the public key field, in addition to serving as the device's public key, is also used to indicate which of the files on the device's MUD file server is the device's MUD file.

8.2.7.2 Non-MUD-Capable IoT Devices

The laboratory implementation also includes non-MUD-capable IoT devices. In this case, several Raspberry Pi devices running the Raspbian 9 operating system are utilized.

8.2.8 Update Server

The update server is designed to represent a device manufacturer or trusted third-party server that provides patches and other software updates to the IoT devices. This project used an NCCoE-hosted update server that provides faux software update files.

8.2.8.1 NCCoE Update Server

The NCCoE implemented its own update server by using an Apache web server. This file server hosts software update files to be served as software updates to the IoT device devkits. When the server receives an http request, it sends the corresponding update file.

8.2.9 Unapproved Server

As with Builds 1 and 2, the NCCoE implemented and used its own unapproved server for Build 3. Details are in Section 6.2.11.

8.2.10 MUD Registry

The Micronets MUD Registry resides in the service provider cloud. Its purpose is to provide a lookup service for determining the URL of each device's MUD file. Currently, the Wi-Fi Easy Connect bootstrapping information in the QR code does not include an information field that has been designated to explicitly carry the device's MUD file URL. Instead, the device's MUD file URL is determined indirectly by using two elements of the device's bootstrapping information:

- The information element field contains a code that identifies the device's manufacturer, and it is assumed that each manufacturer has a well-known location for serving MUD files.
- The public key field locates the device's MUD file on that manufacturer's well-known MUD file server.

The Micronets Manager extracts these two items from the device's bootstrapping information, sends them to the MUD registry, and, in response, receives back the URL of the device's MUD file. The Micronets Manager then provides this MUD file URL to the MUD manager.

MUD-based ACLs are enforced only for IoT devices that have bootstrapping information with a vendor code value in the information element field to indicate the location of the device's MUD file server. If the information element field in an IoT device's bootstrapping information is empty, it is assumed that the device does not have a MUD file, and the device will be onboarded without any restraints on its communications other than those imposed by its location on a given micronet.

8.2.11 SDN Controller

The Micronets Manager resides in the service provider cloud. It is responsible for coordinating and managing a collection of micro-services, one of which helps manage the traffic flow rules on the home/small-business network's Micronets Gateway. The Micronets Manager does not use the OpenFlow protocol to configure traffic flow rules to the Micronets Gateway. However, the Micronets Manager effectively functions as an SDN controller insofar as it uses a RESTful interface to the Micronets Gateway to convey the micronets topology and express the ACLs that it wants the gateway to convert to traffic flow rules that the gateway will enforce.

During the onboarding process, the Micronets Manager sends ACLs to the Micronets Gateway to enforce the communications profile specified in the device's MUD file. It also tells the gateway what trust zones (e.g., micronets) to create on the Micronets Gateway and assigns IoT devices to these micronets as directed by user input. The intention is for devices to be assigned to micronets according to policies for that device class, the device functionality, and the desired level of device protection.

Although the Micronets Manager is not currently capable of automatically assigning devices to micronets based on the local communications rules in the device's MUD file, the goal is to be able to automate such assignment in the future.

8.2.12 Onboarding Manager

The Micronets Manager resides in the service provider cloud. It is responsible for a collection of micro-services, one of which is the Micronets Configuration Microservice. The Micronets Configuration Microservice is the managing and controlling element of the Micronets onboarding process, and it effectively serves as the device onboarding manager. During the onboarding process, the Micronets Manager receives the onboarding request and bootstrapping information from the Micronets mobile phone application (via the MSO portal), looks up the device's MUD file URL in the MUD registry, sends the MUD file URL to the MUD manager, and receives back the parsed MUD rules as ACLs from the MUD manager that it sends to the Micronets Gateway. The Micronets Manager provisions the device by providing network configuration for the device (e.g., IP address, assigned micronet, Wi-Fi credentials) and also provides the device's bootstrapping information (e.g., the device's public key, its MAC address, the Wi-Fi channel it will use) to the Micronets Gateway so the Wi-Fi Easy Connect capabilities in the Micronets Gateway can interact with the device to onboard it and place it in the appropriate micronet.

8.2.13 User and Device Interface to the Onboarding Manager

The Micronets mobile phone application acts as both the Micronets Configuration Microservice's user interface and its IoT device bootstrapping information reader. When a device is put into onboarding mode, it displays a QR code that contains its bootstrapping information. A user positions the Micronets mobile phone application so the phone's camera can scan the device's QR code, thereby providing the application with the device's bootstrapping information. The application also requests additional user input regarding the device, including its Micronets class and a device name. The application then sends an onboarding request containing this bootstrapping and user-supplied device information to the Micronets Manager's Configuration Microservice via the service provider's MSO portal.

8.2.14 Bootstrapping Interface to the Onboarding Manager

The MSO portal is part of the service provider's general-purpose cloud infrastructure. It serves as the interface through which the Micronets mobile application can send a device bootstrapping request to the configuration micro-service in the cloud. This service request will include the device bootstrapping information that the Micronets mobile application collects from both the device QR code and the user who is performing the onboarding. The MSO portal forwards this onboarding request and its associated bootstrapping information to the configuration micro-service, which manages the onboarding process in the service provider cloud.

8.2.15 Network Onboarding Component

The Micronets Gateway is Wi-Fi Easy Connect-capable and serves as the network onboarding component. The Wi-Fi Easy Connect onboarding capabilities that reside in the Micronets Gateway are responsible for interacting directly with IoT devices to perform device onboarding. The gateway receives the IoT device's bootstrapping information (e.g., MAC address, public key, Wi-Fi frequency the device will use, MUD ACLs [if any], micronet class, and device name) from the Micronets Manager. After creating and installing any necessary MUD-based flow rules pertaining to the device (if the device is MUD-capable), the gateway initiates the onboarding process with the device by using the Wi-Fi Easy Connect protocol. The gateway authenticates the device, establishes a secure channel with the device, and then, using this secure channel, provides the device with the unique credentials that it needs to connect to the network (e.g., the network service set identifier [SSID] and the device's unique pre-shared key [PSK]). Once the device has been provisioned with its network credentials, it can use those credentials in a standard Wi-Fi handshake to connect to the network via the network access point.

8.3 Build Architecture

In this section we present the logical architecture of Build 3 relative to how it instantiates the reference architecture depicted in Figure 4-1. We also describe Build 3's physical architecture and present message flow diagrams for some of its processes.

8.3.1 Logical Architecture

Figure 8-1 depicts the logical architecture of Build 3. Figure 8-1 uses numbered arrows to depict in detail the flow of messages needed to support installation of MUD-based access control rules for a MUD-capable device. In contrast to Builds 1, 2, and 4, installation of the MUD ACLs in Build 3 occurs when the device is onboarded, prior to the device's connection to the network. This onboarding process is accomplished using the Wi-Fi Easy Connect protocol, the general steps of which are also depicted in Figure 8-1. The other key aspects of the Build 3 architecture (i.e., the Micronets micro-services layer, on-premises Micronets components, Intelligent Services and Business Logic layer, and the update server) are depicted but not described in the same depth as MUD-capable onboarding. To avoid excessive complexity in the depiction, the Micronets APIs are not depicted.

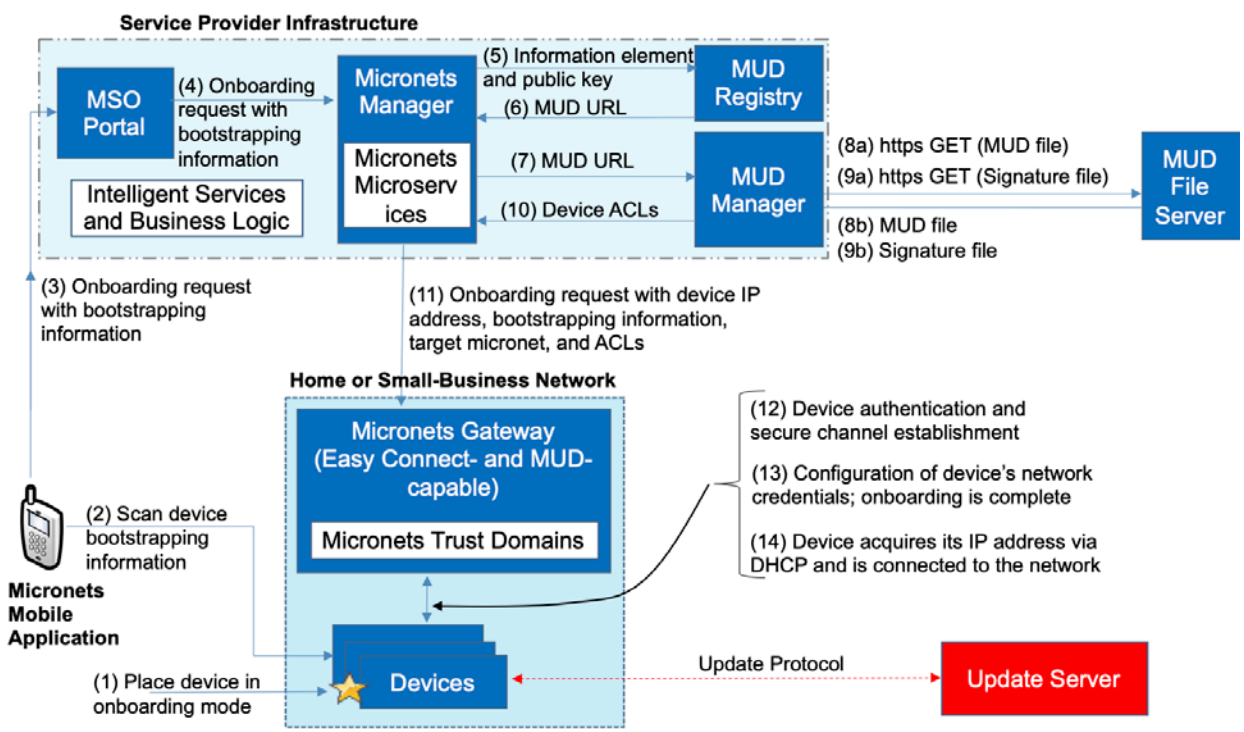
Micronets' logical architecture consists of the following components:

- Micronets mobile application, which supports device onboarding using the Wi-Fi Easy Connect protocol
- On-premises Micronets components, which reside on the home/small-business network. These include the Micronets Gateway, managed services micronets (i.e., micro-networks), and customer micronets. The micro-networks can group devices together into trust domains and isolate them from other devices.

- Cloud-based micro-services layer that hosts various Micronets services. The most important component of this layer is the Micronets Manager, which coordinates the state of the Micronets-enabled on-premises network.
- cloud-based Intelligent Services and Business Logic layer (e.g., machine-learning-based services) that is operated by the service provider
- Micronets API framework, which allows partners and service providers to interface with a customer's micro-networks environment to provision and deliver specific customer-requested services

The logical architecture for Build 3 also includes the notion of ensuring that all IoT devices can access update servers so they can remain up-to-date with the latest security patches. Wi-Fi Easy Connect onboarding of a MUD-capable device using the Micronets mobile application, on-premises Micronets components, the Micronets Microservices layer, the Intelligent Services and Business Logic layer, and the Micronets API framework are each described in their respective subsections below.

Figure 8-1 Logical Architecture—Build 3



8.3.1.1 MUD Capability

As shown in Figure 8-1, Build 3 includes integrated support for MUD in the form of a MUD registry, a MUD manager, a MUD-capable Micronets Manager, and a MUD-capable Micronets Gateway. Support for MUD also requires access to a MUD file server that hosts MUD files for the MUD-capable IoT devices being onboarded.

Build 3 is based on Release 1 of Wi-Fi Easy Connect, which does not include a mechanism for explicitly conveying the device's MUD file URL as part of the device bootstrapping information. To work around this deficiency, Build 3 uses both the information element field and the public key field in the device bootstrapping information to determine the device's MUD file URL. These two fields are used in the following manner:

- The information element field indicates the device's MUD file server. The value in the information element field identifies the device's manufacturer, and it is assumed that each manufacturer has a well-known location for serving MUD files.
- The public key field both conveys the device's public key and identifies the specific file on the manufacturer's MUD file server that is the device's MUD file.
- The Micronets Manager extracts these two values from the bootstrapping information and provides them to the MUD registry lookup service, which in turn responds with the URL of the MUD file associated with an onboarded device. This MUD file URL is then provided to the MUD manager so it can fetch and verify the MUD file.

Future versions of Micronets, subsequent to Build 3, are expected to implement a later version of Wi-Fi Easy Connect (Release 2 or later), which will include a mechanism to optionally and explicitly convey the device's MUD file URL as part of the device onboarding process. Having such a field will greatly simplify the process of conveying the device's MUD file URL to the MUD manager and will obviate the need for a MUD registry.

As shown in Figure 8-1, the flow of messages needed to support installation of MUD-based access control rules for a MUD-capable device in Build 3 is as follows:

- The device must be put into onboarding mode to cause it to display its QR code (which contains its bootstrapping information) and to listen for Wi-Fi Easy Connect protocol messages (step 1).
- The Micronets mobile application is opened and scans the device's QR code. The user also inputs the micronet class to which the device should be assigned, and a device name (step 2).
- The user clicks "onboard," and the application sends the bootstrapping request with the device bootstrapping and other information to the service provider's MSO portal. The Micronets mobile application and the Micronets Manager are each associated with a specific user/subscriber. The onboarding request is sent to the MSO portal so that the portal can ensure that the appropriate Micronets Manager (i.e., the one that is associated with the Micronets mobile application that generated the onboarding request) receives the onboarding request (step 3).
- The MSO portal sends the onboarding request and bootstrapping information to the Micronets Manager (step 4).

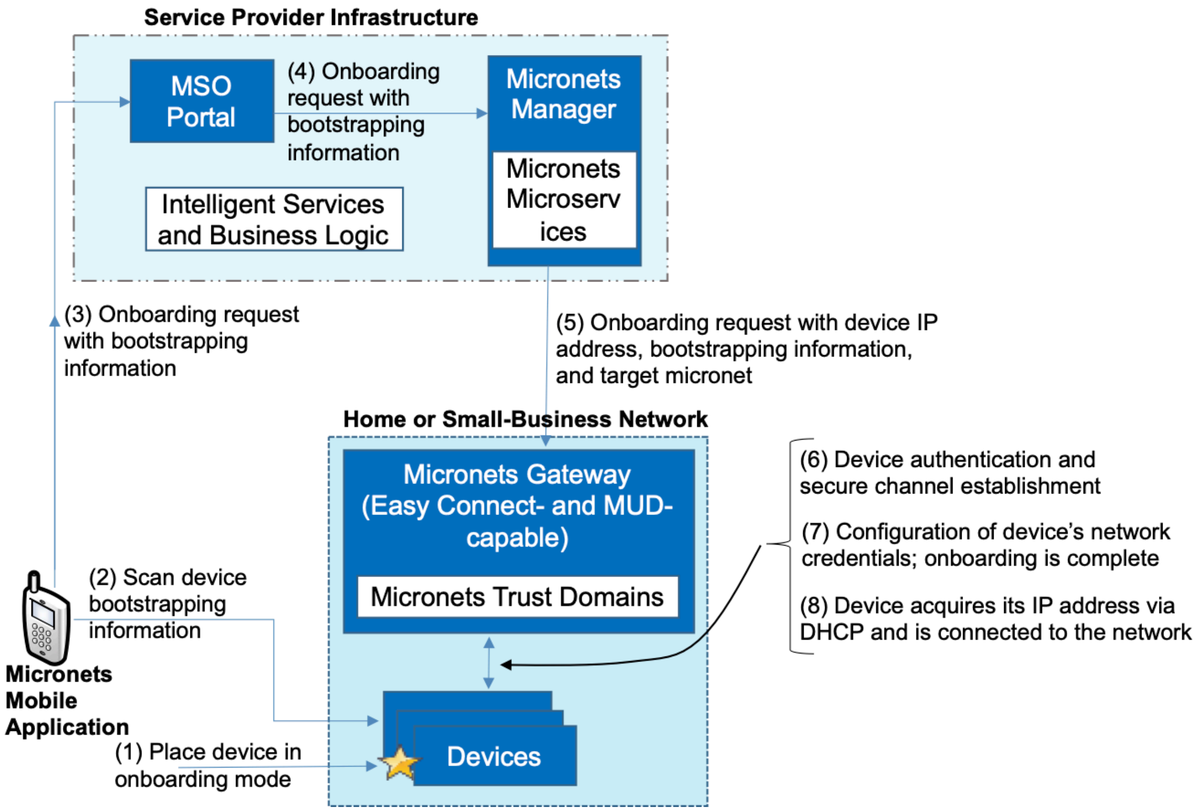
- The Micronets Manager extracts the information element and public key from the bootstrapping information and provides it to the MUD registry (step 5).
- The MUD registry responds with the URL of the device's MUD file (step 6).
- The Micronets Manager provides the MUD file URL to the MUD manager (step 7).
- Once the MUD URL is received, the MUD manager checks its cache to determine if the MUD file is there and, if so, makes sure it has not been there so long that it has exceeded the MUD file caching policy time-out period. If the MUD file is not there or if the file is there but was retrieved too long ago, the MUD manager uses the MUD URL to fetch the MUD file from the MUD file server (step 8a); if successful, the MUD file server at the specified location will serve the MUD file (step 8b).
- Next, the MUD manager requests the signature file associated with the MUD file (step 9a) and upon receipt (step 9b) verifies the MUD file by using its signature file.
- Assuming the MUD file has been verified successfully, the MUD manager parses the MUD file into ACLs that apply to the device and provides these to the Micronets Manager (step 10).
- The Micronets Manager sends these MUD-based ACLs to the on-premises Micronets Gateway, which converts them to traffic flow rules that it installs. These rules ensure that if and when the device connects to the network, it will be subject to the communications policies specified in its MUD file. The device will be permitted only to communicate with local and internet hosts that are explicitly approved in its MUD file, and any other attempts to communicate to or from that device will be blocked (step 11).

  The Micronets Manager also provisions the Micronets Gateway with the device's network configuration and bootstrapping information (e.g., its MAC address, public key, the Wi-Fi channel the device is listening on, the micronet and IP subnet/address to which the device should be assigned, and the device's name) (step 11).
- The Micronets Gateway briefly switches to using the Wi-Fi frequency on which the device is listening (as indicated in the device's bootstrapping information). The Micronets Gateway completes a three-way handshake with the device that constitutes the authentication phase of the Wi-Fi Easy Connect protocol. This protocol exchange serves to both authenticate the device and establish a secure channel with the device (step 12).
- The Micronets Gateway switches back to using its original Wi-Fi frequency. The device switches to using the gateway's frequency and completes a three-way protocol handshake with the device that constitutes the configuration phase of the Wi-Fi Easy Connect protocol. This protocol exchange provisions the device with the credentials that it needs to connect to the network (e.g., the network SSID and the device's unique PSK). At this point, onboarding is complete (step 13).
- The device is now able to connect to the network by presenting its credentials in a standard Wi-Fi handshake (step 14).

8.3.1.2 Wi-Fi Easy Connect Device Onboarding

Figure 8-2 is a simplified version of Figure 8-1. It depicts only the flow of messages needed to support device onboarding in Build 3 (i.e., the message flow needed to support onboarding non-MUD-capable devices).

Figure 8-2 Wi-Fi Easy Connect Onboarding Architecture—Build 3



As shown in Figure 8-2, the flow of messages needed to support onboarding a non-MUD-capable device in Build 3 is as follows:

- The device must be put into onboarding mode to cause it to display its QR code (which contains its bootstrapping information) and to listen for Wi-Fi Easy Connect protocol messages (step 1).
- The Micronets mobile application is opened and scans the device's QR code. The user also inputs the micronet class to which the device should be assigned, and a device name (step 2).
- The user clicks "onboard," and the application sends the bootstrapping request with the device bootstrapping and other information to the service provider's MSO portal (step 3).
- The MSO portal sends the onboarding request and bootstrapping information to the Micronets Manager (step 4).

- The Micronets Manager extracts the public key from the bootstrapping information (because there is no information element, no MUD lookup is performed).
- The Micronets Manager provisions the Micronets Gateway with the device's network configuration and bootstrapping information (e.g., its MAC address, public key, the Wi-Fi channel the device is listening on, the micronet to which the device should be assigned, and the device's name). It also allocates an IP address compatible with the device's target micronet (step 5).
- The Micronets Gateway briefly switches to using the Wi-Fi frequency on which the device is listening (as indicated in the device's bootstrapping information). The Micronets Gateway completes a three-way handshake with the device, which constitutes the authentication phase of the Wi-Fi Easy Connect protocol. This protocol exchange both authenticates the device and establishes a secure channel with the device (step 6).
- The Micronets Gateway switches back to using its original Wi-Fi frequency. The device switches to using the gateway's frequency and completes a three-way protocol handshake with the device, which constitutes the configuration phase of the Wi-Fi Easy Connect protocol. This protocol exchange provisions the device with the credentials that it needs to connect to the network (e.g., the network SSID and the device's unique PSK). At this point, onboarding is complete (step 7).
- The device acquires an IP address via DHCP and is connected to the network (step 8).

8.3.1.3 On-Premises Micronets

The on-premises Micronets consists of the Micronets Gateway, micronets managed by the service provider, and customer micronets, all of which are located on the home/small-business network. The Micronets Gateway is responsible for configuring and enforcing the micronets, which segregate devices. Each micronet represents a distinct trust domain and, at a minimum, represents a distinct IP subnet. IoT devices that are not permitted to exchange traffic with other IoT devices must be placed in separate micronets to isolate them from one another. The Micronets Gateway receives instructions regarding what micronets to set up and assignment of devices to micronets from the Micronets Manager that is in the service provider cloud. The Micronets Gateway is integrated with a Wi-Fi access point, but it supports both wired and wireless connectivity.

8.3.1.3.1 MUD-Driven Policies

The Micronets definition and the placement of devices within a given micronet are governed by the Micronets Manager and are driven by specific policies. Note that the Micronets Manager is associated with the specific user/subscriber who has the on-premises gateway and who is associated with the mobile application. In Build 3, devices are assigned to micronets manually; user input to the Micronets mobile application determines the micronet to which each device will be assigned. Future implementations of Micronets are expected to use MUD-based policies to automatically assign devices to specific micronets.

8.3.1.3.2 Customer Micronets

Customers acquire and connect their own devices. They may even integrate entire service-oriented networks, such as a connected home lighting system. In the future, customer-networked devices may be fingerprinted or authenticated by using an ecosystem certificate (e.g., an Open Connectivity Foundation–certified device) and automatically placed into an appropriate micronet.

8.3.1.4 Micronets Microservices

The Micronets Microservices layer in the service provider cloud hosts several network management-related micro-services that interact with the on-premises Micronets Gateway to manage local devices and network connectivity. One of the core micro-services, the Micronets Manager, coordinates the entire state of the Micronets-enabled on-premises network. It orchestrates the overall delivery of services to the IoT devices and ultimately to the user. The Micronets Manager engages and manages numerous micro-services, including the DHCP/DNS manager, identity server, MUD manager, and MUD registry.

8.3.1.5 Intelligent Services and Business Logic

The Intelligent Services and Business Logic layer resides in the service provider cloud. This architectural component is the interface for the Micronets platform to interact with the rest of the world. It functions as a receiver of the user's intent and business rules from the user's services and is designed to use machine-learning-based services to combine the user's intent and business rules into operational decisions that are handed over to the Micronets micro-services for execution. This layer has not been fully implemented in Build 3. However, it is envisioned that in future versions of Micronets, this layer may receive information from various Micronets micro-services and in turn use that information to dynamically update the access rules for connected IoT devices. For example, to support devices that do not have a MUD file, a "synthetic" MUD file generator and MUD file server could be provided that can host crowdsourced MUD files that are provided to the Micronets micro-services. Other examples include an IoT fingerprinting service that could detect classify devices on the network or an artificial intelligence/machine-learning-based malware detection service that could provide updated MUD files or access policies based on actively detected threats in the network.

8.3.1.6 Micronets API Framework

Each Micronets component (the micro-services as well as the gateway services) exposes a set of APIs that form the Micronets API framework. Some of the APIs can be exposed to allow partners and service providers to interface with the customer's Micronets environment to securely provision and deliver specific services that the customer has requested.

8.3.2 Physical Architecture

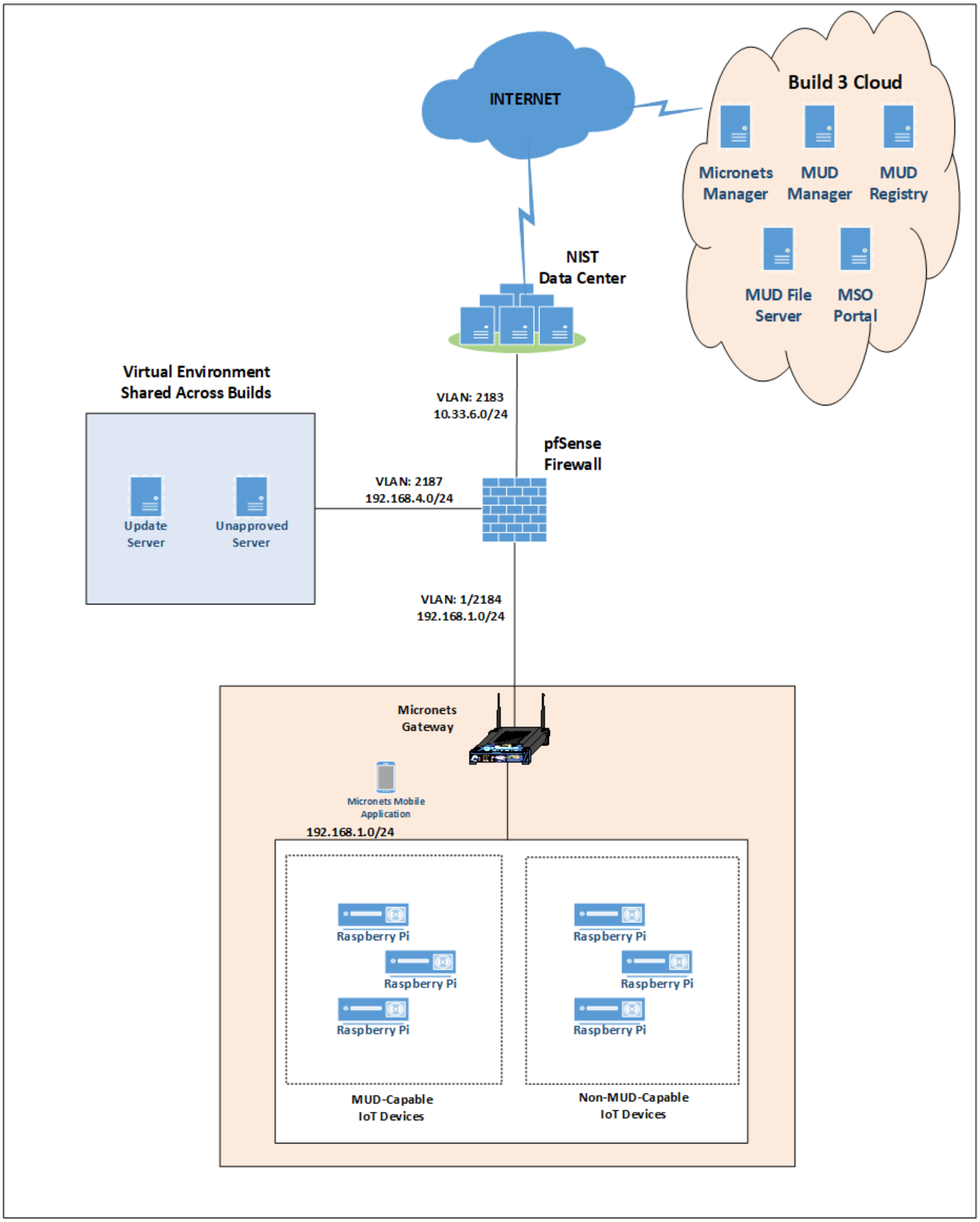
Figure 8-3 depicts the physical architecture of Build 3. The Micronets Gateway that is depicted is an SDN-capable switch. This switch receives instructions from the Micronets Manager in the Build 3 cloud via a RESTful interface. The Micronets Gateway creates and manages various subnetworks (i.e., micronets) on the local network. It allocates an IP address to each MUD-capable and non-MUD-capable IoT device and assigns each device to a specific micronet, which serves as a mechanism to group together devices that are permitted to communicate with one another and to isolate devices that are not. This gateway is also a router configured to enforce the communication constraints of each MUD-capable device as defined in its MUD file. Lastly, the gateway is also Wi-Fi Easy Connect-capable. It uses the Wi-Fi Easy Connect protocol to authenticate devices and provision them with device-specific network credentials. The network infrastructure as configured utilizes the IPv4 protocol for communication both internally and to the internet.

Build 3 also uses a portion of the virtual environment that is shared across builds. Services hosted in this environment include an update server and an unapproved server.

Internet-accessible cloud services are also supported in Build 3. Those depicted include a Micronets Manager, a MUD registry, a MUD manager, and a MUD file server. The Micronets Manager manages a number of different micro-services that are also hosted in the cloud but not depicted, including a configuration micro-service that manages the onboarding process in the service provider cloud.

The Micronets mobile application is also used within the NCCoE laboratory. It runs on a mobile phone and uses that phone's camera to scan in the QR code of IoT devices. This application serves as the user and device bootstrapping interface for the Wi-Fi Easy Connect onboarding process, requesting user input such as the micronet class and name of each device. This application obtains each device's bootstrapping information from the device's QR code and sends it and the user-provided information, along with the onboarding request, to the Micronets Configuration Microservice via the MSO portal. The MSO portal is the fifth cloud service depicted.

2590 **Figure 8-3 Physical Architecture—Build 3**



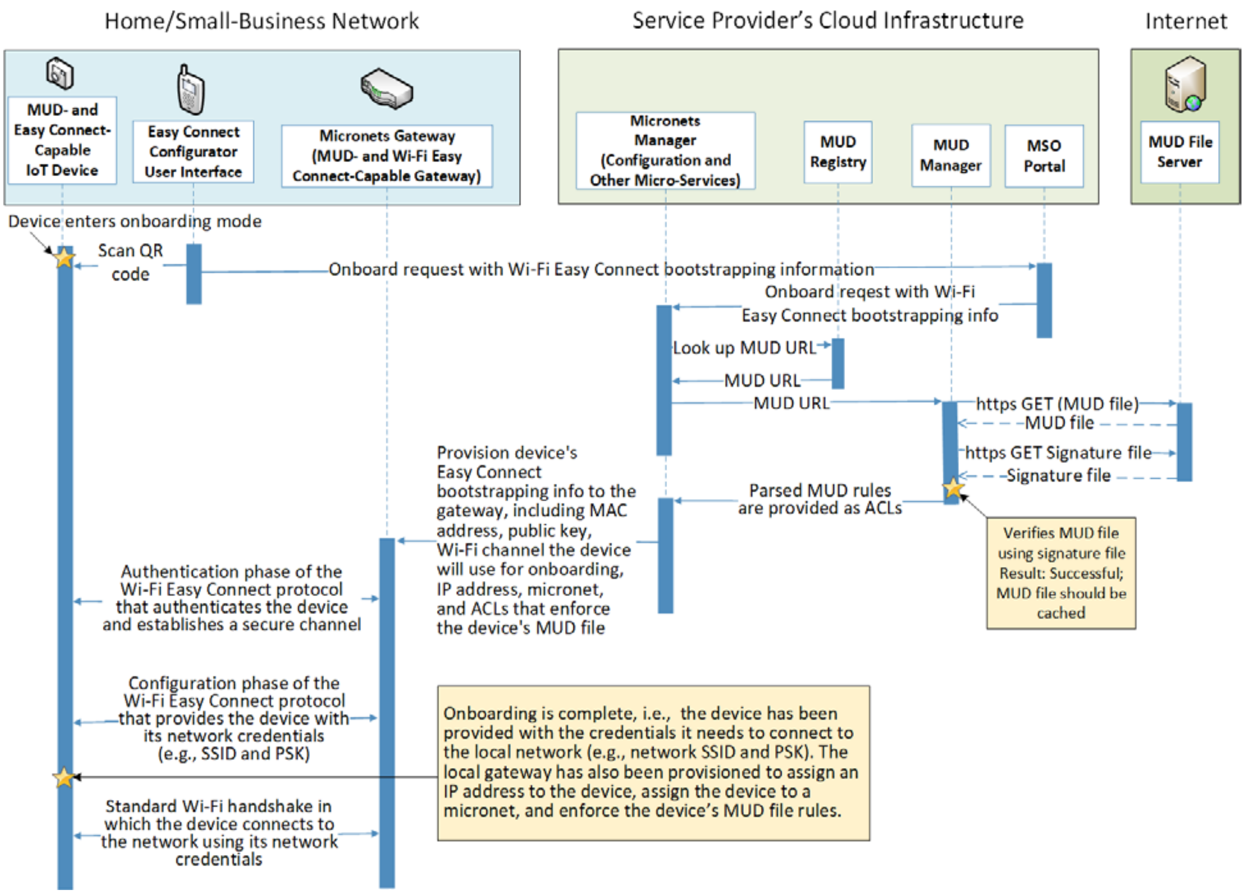
8.3.3 Message Flow

This section presents the message flows used in Build 3 during several different processes of note.

8.3.3.1 Onboarding MUD-Capable Devices

Figure 8-4 depicts the message flows involved in the process of onboarding a MUD-capable device in Build 3, which is accomplished using the Wi-Fi Alliance's Wi-Fi Easy Connect protocol.

Figure 8-4 MUD-Capable IoT Device Onboarding Message Flow—Build 3



The components used to support Build 3 are deployed across the home/small-business network, the service provider cloud, and the internet in general. As shown in Figure 8-4, the onboarding message flow for MUD-capable devices is as follows:

- The IoT device must be placed in onboarding mode. This causes it to display a QR code and to listen for Wi-Fi Easy Connect protocol messages.

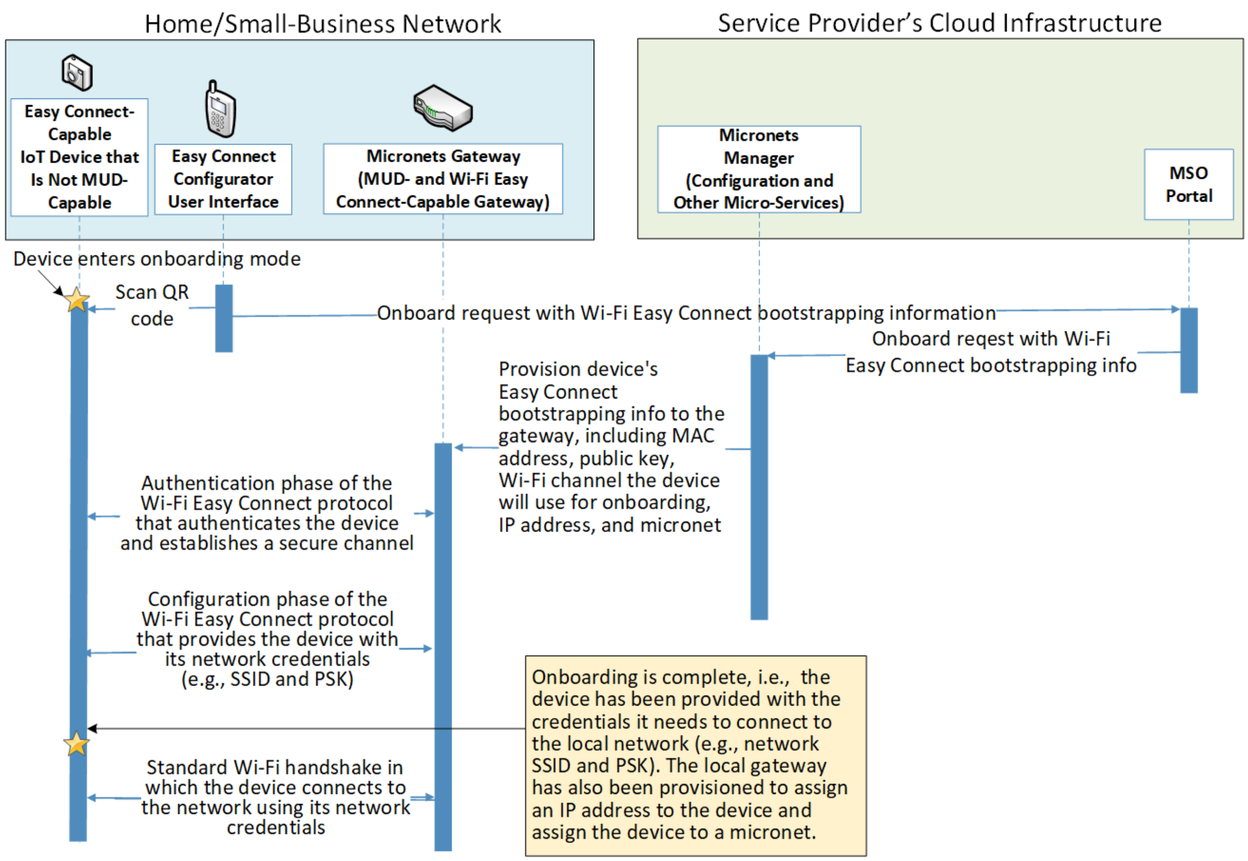
- The Micronets mobile application is opened, “onboard” is selected, and the phone is positioned so that its camera can read the device’s QR code. This provides the device’s bootstrapping information to the configuration element running in the mobile application. The user is also required to input additional device-specific information to the mobile phone application, such as the micronet class of the device and a human-friendly name for the device.
- The mobile phone application sends an onboarding request, along with the device’s bootstrapping and other information, to the service provider’s MSO portal.
- The MSO portal forwards this request and information to the Micronets Manager that is running in the service provider cloud.
- The Micronets Manager sends the information element and the public key field values from the device’s bootstrapping information to the MUD registry.
- The MUD registry responds with the URL for the device’s MUD file.
- The Micronets Manager sends the MUD file URL to the MUD manager.
- The MUD manager fetches the MUD file and the MUD file signature file from the MUD file server.
- After verifying that the MUD file is valid, the MUD manager sends the access control rules that correspond to the MUD file rules to the Micronets Manager.
- The Micronets Manager provisions the device’s bootstrapping information to the Micronets Gateway that is running on the home/small-business network. Specifically, the Micronets Manager provides the gateway with the device’s MAC address, its public key, the Wi-Fi channel on which it will listen for onboarding messages, its micronet, its IP subnet/address, its name, and ACLs needed to enforce the communications profile specified by the device’s MUD file.
- The Micronets Gateway initiates the authentication phase of the Wi-Fi Easy Connect protocol: It sends an authentication request to the IoT device, receives an authentication response from the device, and responds by sending an authentication confirmation to the device. As a result of this exchange, the device has been authenticated, and there is now a secure channel between the Micronets Gateway and the IoT device.
- The IoT device initiates the configuration phase of the Wi-Fi Easy Connect protocol: It sends a configuration request to the Micronets Gateway, receives a configuration response from the Micronets Gateway, and responds by sending a configuration result to the Micronets Gateway. As noted earlier, configuration may happen on a frequency different from the one used for authentication. This completes the onboarding process. As a result of the configuration message it received, the device has learned the SSID and the unique credential that it needs to connect to the home/small-business network. In addition, the Micronets Gateway has been provided with both the micronet to which the device will be assigned upon connection to the network and ACLs that express the device’s communications profile, as specified in its MUD file.

- With onboarding complete, the device initiates a standard Wi-Fi handshake and presents its newly provisioned credentials to connect to the network. It will be assigned its provisioned IP address, it will be located in a micronet that had been specified by the user of the Micronets mobile application at onboarding time, and it will be able to send and receive messages in accordance with both its micronet and the rules specified in its MUD file (i.e., it will not be permitted to communicate with any local devices that are in a different micronet unless such communication is explicitly permitted by its MUD file).

8.3.3.2 Onboarding Non-MUD-Capable Devices

Figure 8-5 depicts the message flows involved in the process of onboarding devices that are Wi-Fi Easy Connect-capable but not MUD-capable in Build 3.

Figure 8-5 Non-MUD-Capable IoT Device Onboarding Message Flow—Build 3



As shown in Figure 8-5, the onboarding message flow for non-MUD-capable devices is as follows:

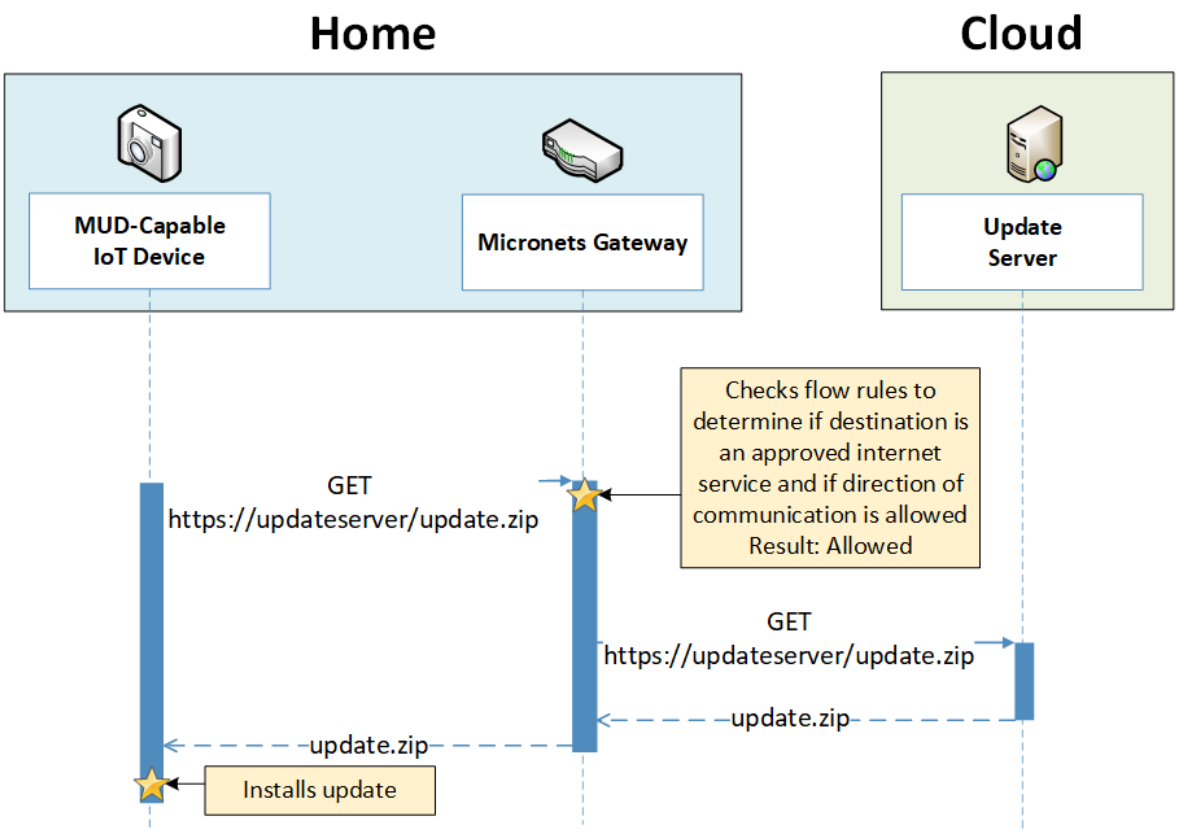
- The IoT device must be placed in onboarding mode. This causes it to display a QR code and to listen for Wi-Fi Easy Connect protocol messages.

- The Micronets mobile application is opened, “onboard” is selected, and the phone is positioned so that its camera can read the device’s QR code. This provides the device’s bootstrapping information to the configuration element running in the mobile application. The user is also required to input additional device-specific information to the mobile phone application, such as the micronet class of the device and a human-friendly name for the device.
- The mobile phone application sends an onboarding request, along with the device’s bootstrapping and other information, to the service provider’s MSO portal.
- The MSO portal forwards this request and information to the Micronets Manager that is running in the service provider cloud.
- The Micronets Manager extracts the public key from the bootstrapping information (because there is no information element, no MUD lookup is performed).
- The Micronets Manager provisions the device’s bootstrapping information to the Micronets Gateway that is running on the home/small-business network. Specifically, the Micronets Manager provides the gateway with the device’s MAC address, its public key, the Wi-Fi channel it will use, its micronet, and its name.
- The Micronets Gateway initiates the authentication phase of the Wi-Fi Easy Connect protocol: It sends an authentication request to the IoT device, receives an authentication response from the device, and responds by sending an authentication confirmation to the device. As a result of this exchange, the device has been authenticated, and there is now a secure channel between the Micronets Gateway and the IoT device.
- The IoT device initiates the configuration phase of the Wi-Fi Easy Connect protocol: It sends a configuration request to the Micronets Gateway, receives a configuration response from the Micronets Gateway, and responds by sending a configuration result to the Micronets Gateway. This completes the onboarding process. As a result of the configuration message it received, the device has learned the SSID and the unique credential that it needs to connect to the home/small-business network. In addition, the Micronets Gateway has been provided with the micronet class to which the device will be assigned upon connection to the network.
- With onboarding complete, the device initiates a standard Wi-Fi handshake and presents its newly provisioned credentials to connect to the network. It will be assigned an IP address, it will be located on the micronet that had been specified by the user of the Micronets mobile application at onboarding time, and it will be able to send and receive messages in accordance with its micronet class (i.e., it will not be permitted to communicate with any local devices that are in a different micronet).

8.3.3.3 Updates

After a device has connected to the home/small-business network, it should periodically check for updates. The message flow for updating the IoT device is shown in Figure 8-6.

2688 **Figure 8-6 Update Process Message Flow—Build 3**



2689 As shown in Figure 8-6, the message flow is as follows:

2690 - The device generates an https GET request to its update server.
2691 - The Micronets Gateway will consult the flow rules for this device to verify that it is permitted to
2692 send traffic to the update server. Assuming there were explicit rules in the device's MUD file
2693 enabling it to send messages to this update server, the Micronets Gateway will forward the
2694 request to the update server.
2695 - The update server will respond with a zip file containing the updates.
2696 - The Micronets Gateway will forward this zip file to the device for installation.

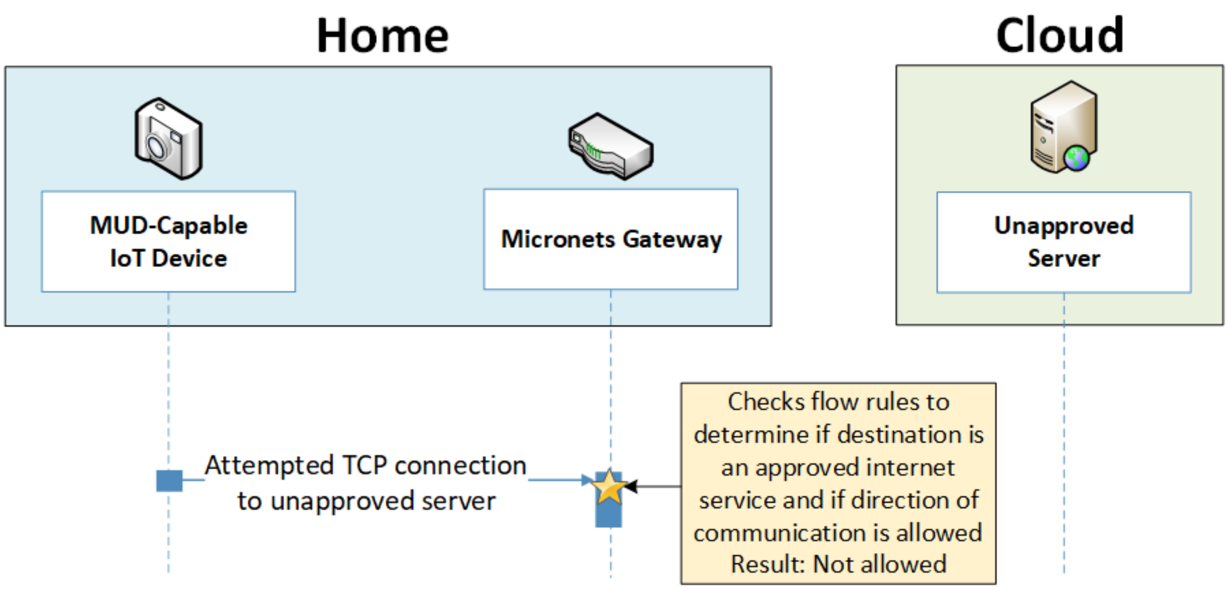
2697 ### 8.3.3.4 Prohibited Traffic

2698 Figure 8-7 shows an attempt to send traffic that is prohibited by the MUD file and so is blocked by the
2699 Micronets Gateway.

2700 - A connection attempt is made from a local IoT device to an unapproved server. (The
2701 unapproved server is located at a domain to which the MUD file does not explicitly permit the
2702 IoT device to send traffic.)

2703 - This connection attempt is blocked because there is no flow rule in the Micronets Gateway that
2704 permits traffic from the IoT device to the unapproved server.

2705 **Figure 8-7 Unapproved Communications Message Flow—Build 3**



2706 ## 8.4 Functional Demonstration

2707 A functional evaluation and a demonstration of Build 3 were conducted that involved two types of
2708 activities:

2709 - evaluation of conformance to the MUD RFC—Build 3 was tested to determine the extent to
2710 which it correctly implements basic functionality defined within the MUD RFC.

2711 - demonstration of additional capabilities—Build 3 supports onboarding via the Wi-Fi Easy
2712 Connect protocol and provides the capability to segregate devices onto specific micronets upon
2713 connection to the network. Both capabilities were demonstrated.

2714 Table 8-2 summarizes the tests used to evaluate Build 3's MUD-related capabilities, and Table 8-3
2715 summarizes the exercises used to demonstrate Build 3's non-MUD-related capabilities (i.e., its
2716 onboarding and Micronets-related functionality). Both tables list each test or exercise identifier, a
2717 summary of the test or exercise, the test or exercise's expected and observed outcomes, and the
2718 applicable Cybersecurity Framework Subcategories and NIST SP 800-53 controls for which each test or
2719 exercise verifies support. The tests and exercises listed in the table are detailed in a separate

supplement for functional demonstration results. Boldface text highlights the gist of the information that is being conveyed.

Table 8-2 Summary of Build 3 MUD-Related Functional Tests

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
IoT-1	**ID.AM-1:** Physical devices and systems within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-2:** Software platforms and applications within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **DE.AE-1:** A baseline of network operations and expected data flows for users and systems is established and managed. **PR.AC-4:** Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC-5, AC-6, AC-14, AC-16, AC-24 **PR.AC-5:** Network integrity is protected, incorporating network segregation where appropriate.	**A MUD-capable IoT device that is also Wi-Fi Easy Connect-capable is onboarded** as follows: The device is put into onboarding mode, causing it to display a QR code containing its bootstrapping information and to listen for Wi-Fi Easy Connect messages on the frequency indicated by the QR code. The Micronets mobile onboarding application is opened and scans the QR code. The user provides additional information and clicks “onboard.” This causes the device bootstrapping information to be sent to the Micronets Manager via the operator’s MSO portal in the service provider cloud. The following operations are then performed automatically: The Micronets Manager looks up the device’s MUD file URL in the MUD registry and	The Micronets Gateway will be configured to enforce the policies specified in **the IoT device’s MUD file. ACLs will be installed on the gateway** to reflect MUD-filtering rules.	Pass

	NIST SP 800-53 Rev. 4 AC-4, AC-10, SC-7 **PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10 **PR.IP-3:** Configuration change control processes are in place. **NIST SP 800-53 Rev. 4** CM-3, CM-4, SA-10 **PR.PT-3:** The principle of least functionality is incorporated by configuring systems to provide only essential capabilities. **NIST SP 800-53 Rev. 4** AC-3, CM-7 **PR.DS-2:** Data in transit is protected.	provides the URL to the MUD manager. The MUD manager contacts the MUD file server and verifies that it has a valid TLS certificate. The MUD manager requests the MUD file and the MUD signature file and validates the MUD file. The MUD manager parses the MUD rules and translates these to ACLs (route-filtering rules) that it sends to the Micronets Manager. The Micronets Manager provides the device's bootstrapping information and its MUD ACLs to the Micronets Gateway so that the gateway is now configured to enforce the policies specified in the device's MUD file. The gateway briefly switches to the device's frequency and initiates Wi-Fi Easy Connect authentication. The device switches to the gateway's frequency and receives network credentials via Wi-Fi Easy Connect. The device connects to the network.		
IoT-2	**PR.AC-7:** Users, devices, and other assets are authenticated (e.g., sin-	A MUD-capable IoT device initiates the Wi-Fi	The MUD manager will detect that the MUD file server	Pass

	gle-factor, multifactor) commensurate with the risk of the transaction (e.g., individuals' security and privacy risks and other organizational risks). **NIST SP 800-53 Rev. 4** AC-7, AC-8, AC-9, AC-11, AC-12, AC-14, IA-1, IA-2, IA-3, IA-4, IA-5, IA-8, IA-9, IA-10, IA-11	Easy Connect onboarding process, but the **MUD file server that is hosting the device's MUD file does not have a valid TLS certificate.** Therefore, the device's MUD manager drops the connection to the MUD file server. The Micronets Manager provisions the device on the Micronets Gateway as if the device had not been associated with a MUD file. **The device does not have any MUD-related restrictions imposed on its communications.** (Note that it is a local policy decision as to whether an implementation will fail "closed" and restrict all communications or fail "open" and not impose any communications restrictions. Build 3 fails open. In theory, it could also act such as assigning the device to a more restricted micronet.)	does not have a valid TLS certificate and will drop the connection to the MUD file server. According to local policy, **the Micronets Gateway will be configured to permit the device to connect to the network and communicate without any MUD-based restrictions**.	
IoT-3	**PR.DS-6:** Integrity-checking mechanisms verify software, firmware, and information integrity. **NIST SP 800-53 Rev. 4** SI-7	A MUD-capable IoT device initiates the Wi-Fi Easy Connect onboarding process, but the **certificate that was used to sign the MUD file for this device had already expired at**	The MUD manager will detect that the MUD file's signature was created by using a certificate that had already expired at signing. According to local policy, **the**	Pass

		signing. Therefore, the Micronets Manager provisions the device on the Micronets Gateway as if the device had not been associated with a MUD file. **The device does not have any MUD-related restrictions imposed on its communications.** (Note that it is a local policy decision as to whether the implementation will fail "closed" and restrict all communications or fail "open" and not impose any communications restrictions. Build 3 fails open. In theory, it could also act such as assigning the device to a more restricted micronet.)	**Micronets Gateway will be configured to permit the device to connect to the network and communicate without any MUD-based restrictions.**	
IoT-4	**PR.DS-6:** Integrity-checking mechanisms verify software, firmware, and information integrity. **NIST SP 800-53 Rev. 4** SI-7	A MUD-capable IoT device initiates the Wi-Fi Easy Connect onboarding process, but the **signature of the device's MUD file is invalid.** Therefore, the Micronets Manager provisions the device on the Micronets Gateway as if the device had not been associated with a MUD file. **The device does not have any MUD-related restrictions imposed on its communications.**	The MUD manager will detect that the MUD file's signature is invalid. According to local policy, **the Micronets Gateway will be configured to permit the device to connect to the network and communicate without any MUD-based restrictions.**	Pass

		(Note that it is a local policy decision as to whether the implementation will fail "closed" and restrict all communications or fail "open" and not impose any communications restrictions. Build 3 fails open. In theory, it could also act such as assigning the device to a more restricted micronet.)		
IoT-5	**ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10 **PR.PT-3:** The principle of least functionality is incorporated by configuring systems to provide only essential capabilities. **NIST SP 800-53 Rev. 4** AC-3, CM-7	Test IoT-1 has run successfully, meaning that the Micronets Gateway has been configured based on a **MUD file that permits traffic to/from some internet locations and implicitly denies traffic to/from all other internet locations.**	When the MUD-capable IoT device is connected to the network, its **Micronets Gateway will be configured to enforce the route filtering that is described in the device's MUD file with respect to** traffic being permitted to/from some **internet locations,** and traffic being implicitly blocked to/from all remaining internet locations.	Pass (for testable procedure –ingress cannot be tested)
IoT-6	**ID.AM-3:** Organizational communication and data flows are mapped.	Test IoT-1 has run successfully, meaning that the Micronets Gateway has been configured	When the MUD-capable IoT device is connected to the	Partial pass. The Micronets Gateway

	NIST SP 800-53 Rev. 4 AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **PR.AC-5:** Network integrity is protected, incorporating network segregation where appropriate. **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7 **PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10 **PR.PT-3:** The principle of least functionality is incorporated by configuring systems to provide only essential capabilities. **NIST SP 800-53 Rev. 4** AC-3, CM-7	based on a **MUD file that permits traffic to/from some lateral hosts and implicitly denies traffic to/from all other lateral hosts.** (The MUD file does not explicitly identify the hosts as lateral hosts; it identifies classes of hosts to/from which traffic should be denied, where one or more hosts of this class happen to be lateral hosts.)	network, its Micronets Gateway **will be configured to enforce the access control information that is described in the device's MUD file with respect to** traffic being permitted to/from some **lateral (local) hosts,** and traffic being implicitly blocked to/from all remaining lateral (local) hosts.	does support protocol-based traffic enforcement for local traffic, but it does not yet support port-level traffic enforcement. Also, as is the case for traffic that originates at internet locations and is inbound toward a MUD-capable IoT device, the gateway does not enforce inbound rules for local communications.
IoT-9	**ID.AM-1:** Physical devices and systems within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5	Test IoT-1 has run successfully, meaning the Micronets Gateway has been configured based on the MUD file for a	A domain in the MUD file resolves to two different IP addresses. **The Micronets Manager**	Pass

	ID.AM-2: Software platforms and applications within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **DE.AE-1:** A baseline of network operations and expected data flows for users and systems is established and managed. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CM-2, SI-4 **PR.AC-4:** Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC-5, AC-6, AC-14, AC-16, AC-24 **PR.AC-5:** Network integrity is protected, incorporating network segregation where appropriate. **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7 **PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM- 5, CM-6, CM-7, CM-9, SA-10 **PR.IP-3:** Configuration change control processes are in place.	specific MUD-capable device in question. **The MUD file contains domains that resolve to multiple IP addresses. The Micronets Gateway should be configured to permit communication to or from all IP addresses for the domain.**	**will create flow rules on the Micronets Gateway that permit the MUD-capable device to send traffic to both IP addresses.** The MUD-capable device attempts to send traffic to each of the IP addresses, and the Micronets Gateway permits the traffic to be sent in both cases.	

	NIST SP 800-53 Rev. 4 CM-3, CM-4, SA-10 **PR.DS-2:** Data in transit is protected. **NIST SP 800-53 Rev. 4** SC-8, SC-11, SC-12			
IoT-10	**ID.AM-1:** Physical devices and systems within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-2:** Software platforms and applications within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **DE.AE-1:** A baseline of network operations and expected data flows for users and systems is established and managed. **PR.AC-4:** Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC-5, AC-6, AC-14, AC-16, AC-24 **PR.AC-5:** Network integrity is protected, incorporating network segregation where appropriate. **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7 **PR.IP-1:** A baseline configuration of information technology/industrial	A MUD-capable IoT device is onboarded as described in test IoT-1. As part of this onboarding process, the device's MUD file is retrieved, and the Micronets Gateway is configured to enforce the policies specified in the MUD file for that device. **Within 24 hours (i.e., within the cache-validity period for that MUD file), a second IoT device that is associated with the same MUD file is connected to the network.** After 24 hours have elapsed, a third IoT device that is associated with the same MUD file is connected to the network.	Upon connection of the second IoT device to the network, **the MUD manager does not contact the MUD file server. Instead, it uses the cached MUD file.** It translates this MUD file's contents into appropriate route-filtering rules and provides these to the Micronets Manager for installation onto the Micronets Gateway for the second IoT device. Upon connection of the third IoT device to the network, the MUD manager does fetch a new MUD file.	Pass

	control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10 **PR.IP-3:** Configuration change control processes are in place. **NIST SP 800-53 Rev. 4** CM-3, CM-4, SA-10 **PR.PT-3:** The principle of least functionality is incorporated by configuring systems to provide only essential capabilities. **NIST SP 800-53 Rev. 4** AC-3, CM-7 **PR.DS-2:** Data in transit is protected.			
IoT-11	**ID.AM-1:** Physical devices and systems within the organization are inventoried.	**A MUD-capable IoT device conveys its MUD file URL via two fields in its bootstrapping information (information element and public key), which are encoded in its QR code. The information element contains a code that identifies the device vendor. It is assumed that each manufacturer has a well-known location for serving MUD files. The public key locates the device's MUD file. A MUD registry is deployed on the service provider cloud that, when provided with the information element and public key**	During the onboarding process, the Micronets Manager extracts the information element and public key field values from the device's bootstrapping information and provides these to the MUD registry. The MUD registry responds with the URL of the device's MUD file. The Micronets Manager provides this URL to the MUD manager, and the appropriate MUD file for the device is fetched and used as the basis for the flow rules that are configured on	Pass

		field values from a device's bootstrapping information, responds with the URL of the device's MUD file.	the Micronets Gateway for the device.	

In addition to supporting MUD, Build 3 supports onboarding via the Wi-Fi Easy Connect protocol and provides the capability to place devices onto specific micronets when they are provisioned on the network. Wi-Fi Easy Connect supports easy onboarding of both MUD-capable and non-MUD-capable devices. Micronets are subnetworks that serve to isolate devices. Devices that are on one micronet are not able to exchange traffic with devices on other micronets unless this restriction is overridden by their MUD files. Different micronet classes have been predefined. When a device is onboarded using the Micronets mobile application, the user is asked to input or confirm the class of micronet to which the device should be assigned.

Table 8-3 lists the non-MUD-related (e.g., the Wi-Fi Easy Connect onboarding- and Micronet-related) capabilities that were demonstrated for Build 3.

Table 8-3 Wi-Fi Easy Connect Onboarding- and Micronets-Related Functional Capabilities Demonstrated

Exercise	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Exercise Summary	Expected Outcome	Observed Outcome
MnMUD-1	**PR.AC-1:** Identities and credentials are issued, managed, verified, revoked, and audited for authorized devices, users, and processes. **NIST SP 800-53 Rev. 4** AC-1, AC-2, IA-1, IA-2, IA-3, IA-4, IA-5, IA-6, IA-7, IA-8, IA-9, IA-10, IA-11 **NIST SP 800-53 Rev. 4** PE-2, PE-3, PE-4, PE-5, PE-6, PE-8 **PR.AC-3:** Remote access is managed. **NIST SP 800-53 Rev. 4** AC-1, AC-17, AC-19, AC-20, SC-15	Demonstrates that **non-MUD-capable devices that are Wi-Fi Easy Connect-capable can be onboarded using the Wi-Fi Easy Connect protocol and that, once onboarded, can successfully connect to the network with the credentials they were provided during onboarding; and that they are assigned to the correct micronet.** Specifically, the follow-	**Both devices can successfully connect to the network, and they can send and receive messages to and from each other**.	As expected

Exercise	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Exercise Summary	Expected Outcome	Observed Outcome
	PR.AC-4: Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC- 5, AC-6, AC-14, AC-16, AC-24 **PR.AC-5:** Network integrity is protected (e.g., network segregation, network segmentation). **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7	ing steps are performed for two separate IoT devices: The device is put into onboarding mode, causing it to display a QR code containing its bootstrapping information and to listen for Wi-Fi Easy Connect messages on the frequency indicated by the QR code. The Micronets mobile onboarding application is opened and scans the QR code. The user assigns the device to a particular micronet and clicks “onboard.” This causes the device bootstrapping information to be sent to the Micronets Manager via the operator’s MSO portal in the service provider cloud. The following operations are then performed automatically: The Micronets Manager provides the device’s bootstrapping information and its MUD ACLs to the Micronets Gateway. The gateway briefly switches to the device’s frequency and initiates Wi-Fi Easy		

Exercise	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Exercise Summary	Expected Outcome	Observed Outcome
		Connect authentication to authenticate the device and establish a secure channel with it. The device switches to the gateway's frequency and initiates the Wi-Fi Easy Connect configuration phase to receive its network credentials from the gateway. The device connects to the network. Note that both IoT devices are assigned to the same micronet class.		
MnMUD-2	**PR.AC-1:** Identities and credentials are issued, managed, verified, revoked, and audited for authorized devices, users, and processes. **NIST SP 800-53 Rev. 4** AC-1, AC-2, IA-1, IA-2, IA-3, IA-4, IA-5, IA-6, IA-7, IA-8, IA-9, IA-10, IA-11 **NIST SP 800-53 Rev. 4** PE-2, PE-3, PE-4, PE-5, PE-6, PE-8 **PR.AC-3:** Remote access is managed. **NIST SP 800-53 Rev. 4** AC-1, AC-17, AC-19, AC-20, SC-15 **PR.AC-4:** Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC- 5, AC-6, AC-14, AC-16, AC-24	Demonstrates that **devices that are assigned to the same micronet can communicate with each other but not with devices in a different micronet**. Run exercise MnMUD-1, with the result that there are two devices connected to the correct network (Device 1 and Device 2), and they are on the same micronet. Run exercise MnMUD-1 for a third device, but this time assign the device to a different micronet class in step 7a and name it Device 3 in step 7b.	Non-MUD-capable devices can be onboarded with the network credentials necessary to ensure that they connect to the correct network and, once connected, are assigned to the correct micronet. **Devices in the same micronet can communicate with one another, but devices in different micronets cannot.**	As expected

Exercise	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Exercise Summary	Expected Outcome	Observed Outcome
	PR.AC-5: Network integrity is protected (e.g., network segregation, network segmentation). **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7	Verify that Device 1 and Device 2 (which are both on micronet CLASS 1) can send and receive messages to and from each other. Verify that neither Device 1 nor Device 2 can send or receive messages to or from Device 3 (which is on micronet CLASS 2).		
MnMUD-3	**PR.AC-1:** Identities and credentials are issued, managed, verified, revoked, and audited for authorized devices, users, and processes. **NIST SP 800-53 Rev. 4** AC-1, AC-2, IA-1, IA-2, IA-3, IA-4, IA-5, IA-6, IA-7, IA-8, IA-9, IA-10, IA-11 **NIST SP 800-53 Rev. 4** PE-2, PE-3, PE-4, PE-5, PE-6, PE-8 **PR.AC-3:** Remote access is managed. **NIST SP 800-53 Rev. 4** AC-1, AC-17, AC-19, AC-20, SC-15 **PR.AC-4:** Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC- 5, AC-6, AC-14, AC-16, AC-24 **PR.AC-5:** Network integrity is protected (e.g., network segregation, network segmentation). **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7	Run exercise MnMUD-1, with the result that there are two devices connected to the correct network (Device 1 and Device 2), and they are on the same micronet. Run exercise MnMUD-1 for a third device (Device 3), and assign this device to the same micronet class as the first two devices. Verify that all three devices are connected to the correct network and can exchange messages with one another. Then configure the gateway to **revoke the credentials of Device 2. Verify that Device 2 cannot send messages to or receive messages from Device 1 or Device 3. Verify**	After multiple IoT devices have been onboarded and connected to the network, **the credentials of one of these devices can be revoked at the Micronets Gateway, causing that device to be disconnected. The other devices, which have their own unique credentials, remain connected**.	As expected

Exercise	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Exercise Summary	Expected Outcome	Observed Outcome
		that Device 1 and Device 3 can send messages to and from each other.		

8.5 Observations

Build 3 was able to successfully onboard IoT devices using the Wi-Fi Easy Connect protocol, assign those devices to the appropriate micronet class based on user input, and, if the devices are MUD-capable, permit and block traffic to and from the devices as specified in the devices' MUD files. Build 3 was also able to constrain communications to and from local devices (both MUD-capable and non-MUD-capable) based on the micronet class to which the devices were assigned.

We observed the following limitations to Build 3 that are informing improvements to its current proof-of-concept implementation:

- MUD manager:
    - Port/protocol-level traffic filtering is not supported in this version of the MUD manager. If a MUD file rule permits some type of communication between two local devices using a specific port or protocol, Build 3 erroneously permits this communication between those two local devices using all ports. It does not matter whether the MUD rule is specified using port numbers (e.g., 80/443) or protocols (UDP/TCP); neither level of traffic filtering is supported.
- micronets assignment:
    - Within a micronet, all devices can communicate with one another. To enforce the lateral communications rules specified in a device's MUD file, only devices whose MUD files explicitly permit them to communicate with one another should be assigned to the same micronet. Build 3 currently requires assignment of devices to micronets to be performed manually by the user who operates the Micronets mobile application during onboarding. It may not be realistic to expect this user to be familiar with the contents of the device's MUD file and know how to assign devices to micronets accordingly. Ideally, the assignment of devices to micronets should be performed automatically, with the Micronets Manager examining the MUD file rules for the device and, based on those rules, automatically assigning the device to micronets that will enforce the device's local communications profile. Such automatic assignment of devices to micronets, however, is not yet supported. Currently, the only way to ensure that only local communications that are explicitly permitted by the MUD file will be permitted is for the user who is performing the

onboarding to manually assign each device to its own separate micronet. Future implementations of the Micronets Manager may be capable of automatically adding devices with similar local-network restrictions into discrete micronets.

- conveyance of the device's MUD file URL:
    - Build 3 implements Wi-Fi Easy Connect protocol Release 1, which was the current version at the time. Wi-Fi Easy Connect Release 1 does not have a mechanism for conveying the device's MUD file URL in the device bootstrapping information. As a result, Build 3 relies on a workaround to indicate the URL of the MUD file associated with a device. As described previously, this workaround uses the information element field and the public key field in the device bootstrapping information. It also relies on a MUD registry lookup service and an assumption that every manufacturer has a well-known location for serving MUD files. On the other hand, the most recent version of Wi-Fi Easy Connect, Release 2, as specified in the Wi-Fi Alliance's DRAFT Device Provisioning Protocol Specification Version 1.2, does define a mechanism for optionally including the device's MUD file URL in the device bootstrapping information that is conveyed. Future versions of Micronets, subsequent to Build 3, are expected to simply implement the latest Wi-Fi Easy Connect release (Release 2 or later) and will thereby greatly simplify the process of conveying the device's MUD file URL to the MUD manager. Anyone desiring to duplicate the Build 3 implementation in their own environment must either provide their own MUD registry or use the MUD registry created by CableLabs, which CableLabs has offered to make available for this purpose.
- authenticating the association between a device and its MUD file URL

    It is worth noting that the MUD registry that is implemented in Build 3 serves not just as a mechanism for locating each device's MUD file. Assuming that the registry is trusted, it also serves to authenticate the association between the device and its MUD file. When using Build 3, the assumption is that the central registry is a trusted and reliable entity with which each vendor has registered the location of its MUD file server (or the location of a secondary registry that can be used to locate that vendor's MUD file servers). Therefore, this central registry can be trusted to provide a valid association between each device and its MUD file or between each device and the vendor-specific registry that will point to the particular MUD file. The MUD registry architecture that is in place to support the central registry and vendor-specific subregistries in Build 3 is nontrivial; there are no shortcuts when it comes to providing an authenticated association between a device and its MUD file.

    Once Easy Connect Release 2 is implemented, the MUD registry will no longer be necessary. The association between the device and its MUD file will be provided by inclusion of the MUD URL in the device bootstrapping information. Trust in this association will rely on the manufacturer's root of trust, i.e., on the trustworthiness of the certificate authority that signed the certificate for the manufacturer that signed the MUD file. Hence, to be able to trust that a MUD file is in fact correctly associated with a particular device, either:

- The certificate authority that signed the device manufacturer's certificate must be trusted, (as will be the case when Easy Connect Release 2 is implemented) or
- The association between the device and its MUD file must be provided by a central registry that everyone trusts (as is the case in Build 3).

We observed the following benefit of Build 3:

- MUD configuration during onboarding avoids periods during which connected MUD-capable devices are permitted to communicate unrestrained.
  - In implementations other than Build 3 that configure the MUD-related traffic flow rules during device connection, there may be small windows of time during which a device is permitted unrestricted communications while its MUD file is being requested and processed, before the MUD rules related to the device are applied. Because Build 3 configures the MUD-related traffic flow rules on the Micronets Gateway during onboarding, before the device is provisioned with its network credentials, it is not possible for there to be a time period during which the device is connected to the network before its MUD traffic flow rules are provisioned on the gateway.
- Use of Wi-Fi Easy Connect in Build 3 enables each device to be provisioned with its own unique network credentials.
  - Per-device credentialing ensures that even if the credentials of one device are known, these credentials cannot be presented by other devices (e.g., devices that are not authorized to connect to the network) to gain access to the network.
  - Per-device credentialing enables the credentials of some devices to be revoked or changed without interfering with the ability of other devices to connect to the network.
- Network credentials are provisioned to each device via an automated protocol, thereby minimizing the opportunity for human error.
- Network credentials are provisioned to each device over a secure channel, minimizing the possibility of their disclosure. No human being has an opportunity to be privy to the credentials of any device.

9 Build 4

The Build 4 implementation uses software developed at the NIST Advanced Networking Technologies laboratory that is called NIST-MUD. The purpose of this implementation is to serve as a working prototype of the MUD RFC to demonstrate feasibility and scalability. NIST-MUD is intended to provide a platform for research and development by industry and academia. It is released as a simple, minimal, open-source reference implementation of an SDN controller/MUD manager on GitHub.

The NIST MUD manager is implemented as a feature that is running on an OpenDaylight SDN controller. The SDN controller/MUD manager uses the OpenFlow (1.3) protocol to configure the MUD rules on an SDN-capable switch that is deployed on the home or small-business network. Build 4 also uses certificates from DigiCert.

9.1 Collaborators

Collaborators that participated in this build are described briefly in the subsections below.

9.1.1 NIST Advanced Networking Technologies Laboratory

The NIST Advanced Networking Technologies lab mission is networking research and advanced prototyping of emerging standards.

9.1.2 DigiCert

See Section 6.1.2 for a description of DigiCert.

9.2 Technologies

Table 9-1 lists all of the products and technologies used in Build 4 and provides a mapping among the generic component term, the specific product used to implement that component, and the security control(s) that the product provides. When applicable, both the Function Subcategories that a component provides directly and those that it supports but does not provide directly are listed and labeled as such. For rows in which the provides/supports distinction is not noted, all listed Categories are directly provided by the component. Some functional Subcategories are described as being directly provided by a component. Others are supported but not directly provided by a component. Refer to Table 5-1 for an explanation of the NIST Cybersecurity Framework Subcategory codes.

Table 9-1 Products and Technologies

Component	Product	Function	Cybersecurity Framework Subcategories
SDN controller	OpenDaylight SDN Controller	Used to manage the SDN switch on the home/small-business network. Provides a protocol stack on top of which the MUD manager is built; includes an OpenFlow plug-in that is used	Provides ID.AM-3 PR.PT-3

Component	Product	Function	Cybersecurity Framework Subcategories
		to send flow rules to the SDN switch.	
MUD manager	NIST-MUD SDN controller/MUD manager (implemented as a feature on an OpenDaylight open-source SDN controller)	Fetches, verifies, and processes MUD files from the MUD file server maintained by the manufacturer; can also receive MUD files through a Representational State Transfer (REST) API if a manufacturer does not provide a MUD file server. Parses MUD files and converts them to flow rules. Eavesdrops on IoT device DNS requests to obtain the IP address values to insert into flow rules when instantiating MUD file access control entries (ACEs).	Provides PR.PT-3 Supports ID.AM-1 ID.AM-2 ID.AM-3 PR.AC-4 PR.AC-5 PR.DS-5 DE.AE-1
MUD file server	NCCoE-hosted Python (requests)-based https server	Hosts MUD files and signature files; serves MUD files to the MUD manager by using https	ID.AM-1 ID.AM-2 ID.AM-3 PR.AC-4 PR.AC-5 PR.DS-5 PR.PT-3 DE.AE-1
MUD file maker	MUD file maker (https://www.mud-maker.org/)	GUI used to create example MUD files	ID.AM-1

Component	Product	Function	Cybersecurity Framework Subcategories
MUD file	A YANG model instance that has been serialized in JSON (RFC 7951). The manufacturer of a MUD-capable device creates that device's MUD file. MUD file maker (see previous row) can be used to create MUD files. Each MUD file is also associated with a separate MUD signature file.	Specifies the communications that are permitted to and from a given device	Provides PR.PT-3 Supports ID.AM-1 ID.AM-2 ID.AM-3
DHCP server	dnsmasq DHCP server	Functions as a generic DHCP server; does not provide any MUD-specific functions	ID.AM-3 PR.AC-4 PR.AC-5 PR.DS-5 PR.PT-3 DE.AE-1
Router or switch	Northbound Networks wireless SDN switch	Routes traffic on the home/small-business network. Gets configured with OpenFlow 1.3 flow rules that enforce MUD file ACEs.	ID.AM-3 PR.AC-4 PR.AC-5 PR.DS-5 PR.PT-3 DE.AE-1
Certificates	DigiCert Premium Certificate	Used to sign MUD files and generate corresponding signature file	PR.AC-1 PR.AC-3 PR.AC-5 PR.AC-7
MUD-capable IoT device 1 (has MUD file profile1)	Raspberry Pi Model 3	Emits a MUD URL as part of its DHCP REQUEST	ID.AM-1
Second MUD-capable IoT device (has MUD file profile1)	Raspberry Pi model 3	Emits a MUD URL as part of the DHCP REQUEST. Acts as the second device made by the same manufacturer as device 1.	ID.AM-1

Component	Product	Function	Cybersecurity Framework Subcategories
Third MUD-capable IoT device (has MUD file profile2)	Raspberry Pi Model 3	Emits a MUD URL as part of the DHCP REQUEST. Acts as a device made by another manufacturer (so we can test interactions between the first type of device and the second type of device).	ID.AM-1
Non-MUD-capable IoT device	Raspberry Pi without a MUD profile	Acts as a typical IoT device on the home/small-business network; does not emit a MUD URL and does not have an associated MUD file. Its traffic is unrestricted.	ID.AM-1
Controller	Raspberry Pi without a MUD profile	Acts as a device controller for the first MUD-enabled device	
Update server	NCCoE-hosted Raspberry Pi Python (request)-based servers (two are used)	Acts as a device manufacturer's update server that would communicate with IoT devices to provide patches and other software updates	PR.IP-1 PR.IP-3
Unapproved server	Raspberry Pi running a web server	Acts as an internet host that has not been explicitly approved in a MUD file	DE.DP-3 DE.AM-1

9.2.1 SDN Controller

The switch on the home/small-business network is an SDN switch that is managed by an OpenDaylight SDN controller. OpenDaylight provides protocol stacks on top of which the MUD manager is built. In Build 4, the protocol stack used is a southbound protocol plug-in for the OpenFlow 1.3 protocol that is used by OpenDaylight applications (e.g., the MUD manager) to send flow rules to the OpenFlow-enabled SDN switch on the home/small-business network. OpenDaylight also allows applications to export “northbound” RESTCONF/YANG model APIs that are primarily used for configuration purposes.

9.2.2 MUD Manager

The MUD manager is an OpenDaylight application written in Java. OpenDaylight uses the Apache Karaf Open Service Gateway Initiative container. The MUD manager is a Karaf feature that uses OpenDaylight libraries and bundles. The IETF-published YANG model for MUD is imported into OpenDaylight directly for the MUD manager implementation.

The MUD manager receives the MUD URL for an IoT device, fetches that MUD file and its corresponding signature file, and uses the signature file to verify the validity of the MUD file. If signature verification succeeds, the MUD manager generates SDN flow rules corresponding to the ACEs that are in the MUD file and pushes them to the SDN switch on the home/small-business network by using the OpenFlow protocol. The instantiation of some flow rules (i.e., those relating to DNS names that have not yet been resolved) may have to be deferred because the IP addresses to be inserted into the flow rules corresponding to these ACEs depend on domain name resolution as seen by the IOT device, which may not yet have been performed. If domain name resolution is performed by a device on the home/small-business network for any domain name that is referenced by a flow rule, the flow rule will be instantiated and sent to the SDN switch.

If signature verification fails or if the MUD file is not retrievable (for example, if the manufacturer website is down or does not have a valid TLS certificate), the MUD manager sends packet classification flow rules to the SDN switch that cause the device to be blocked. In a blocked state, the device may only access DHCP, DNS, and NTP services on the network. This effectively quarantines the device until the MUD file may be verified.

The MUD manager can manage multiple switches. The system achieves memory scalability by a multiple flow table design that uses O(N) flow rules for N distinct MAC addresses seen at the switch.

9.2.3 MUD File Server

In the absence of a commercial MUD file server for use in this project, the NCCoE implemented its own MUD file server by using a Python (requests)-based web server. This file server serves the MUD files along with their corresponding signature files for the IoT devices used in the project. Upon receiving a GET request for the MUD files and signatures, it serves the request to the MUD manager by using https.

9.2.4 MUD File

We test interactions between two manufacturers and between two devices made by the same manufacturer. To accomplish this, two MUD files are defined (referred to as "profile1" and "profile2" in the table above).

9.2.5 Signature File

According to the IETF MUD specification, "a MUD file MUST be signed using CMS as an opaque binary object." The MUD files were signed with the OpenSSL tool by using the command described in the specification (as detailed in Volume C of this guide). A Premium Certificate, requested from DigiCert, was leveraged to generate the signature files. Once created, the signature files are stored on the MUD file server along with the MUD files. The certificate is added to the trust store of the Java Virtual Machine running the MUD manager to enable signature verification.

9.2.6 DHCP Server

NIST-MUD is a Layer-2 implementation. Devices are identified by MAC addresses. NIST-MUD is designed to work with devices that join the network by issuing a DHCP request.

DHCP requests for MUD-enabled devices may contain a MUD URL. The DHCP request (with embedded MUD URL) is sent to the SDN switch, which forwards it simultaneously to the SDN controller/MUD manager and the DHCP server. This is accomplished via an SDN flow rule that is inserted by the MUD manager into the switch flow table when the switch connects to the MUD manager. After extracting the MUD URL from the DHCP packet, the MUD manager proceeds to retrieve the MUD file that is pointed to by the MUD URL.

Because the SDN switch forwards the DHCP request to the MUD manager rather than the DHCP server forwarding the DHCP request to the MUD manager, no modifications to the DHCP server are needed. The MUD manager instead of the DHCP server is responsible for stripping the MUD URL out of the DHCP request. Therefore, Build 4 can use a generic DHCP server that is not required to support any MUD-specific capabilities.

9.2.7 Router/Switch

The switch used on the home/small-business network is a wireless SDN switch that comes bundled with the Northbound Networks Wireless Access Point. The access point bundles a NAT router, DNS server, and DHCP server. The SDN controller/MUD manager is connected to the public-facing side of the switch's NAT component. The switch is OpenFlow-enabled and interacts with its SDN controller/MUD manager via the OpenFlow 1.3 protocol. The SDN switch serves as the enforcement point for MUD policy. Packets sent between devices, between devices and controllers referenced in MUD files, and between devices and the internet must pass through the switch, which is where enforcement occurs.

9.2.8 Certificates

DigiCert provisioned a Premium Certificate for signing the MUD files. The Premium Certificate supports the key extensions required to sign and verify CMS structures as required in the MUD specification. Further information about DigiCert's CertCentral web-based platform, which allows for provisioning and managing publicly trusted X.509 certificates, can be found in Section 6.2.8.

9.2.9 IoT Devices

This section describes the IoT devices used in the laboratory implementation. There are two distinct categories of devices: devices that can emit a MUD URL in compliance with the MUD specification, i.e., MUD-capable IoT devices; and devices that are not capable of emitting a MUD URL in compliance with the MUD specification, i.e., non-MUD-capable IoT devices.

9.2.9.1 MUD-Capable IoT Devices

Three Raspberry Pi devkits used on the home/small-business network are designated as MUD-capable. Two emit the same MUD URL (corresponding to profile1) and the third emits a different MUD URL (corresponding to profile2).

9.2.9.2 Non-MUD-Capable IoT Devices

A fourth Raspberry Pi on the home/small-business network functions as a non-MUD-capable IoT device. Because it does not have an associated MUD file, its communications are not restricted.

9.2.10 Controller and My-Controller

A fifth Raspberry Pi device on the home/small-business network is designated as controller and my-controller. Note that a host cannot simultaneously be designated as a controller and be part of the local network. Hence, the Raspberry Pi that performs this function is not part of the local network category.

9.2.11 Update Server

The update server is designed to represent a device manufacturer or trusted third-party server that provides patches and other software updates to the IoT devices. This project used an NCCoE-hosted update server that provides faux software update files.

9.2.11.1 NCCoE Update Server

The NCCoE implemented its own update server by using an Apache web server. This file server hosts faux software update files to be served as software updates to the IoT device devkits. When the server receives an http request, it sends the corresponding faux update file.

In Build 4, there are two update servers, both of which are Raspberry Pi hosts on the public side of the switch. The DNS server on the switch is configured to return two addresses corresponding to the DNS name of the update server (e.g., www.nist.local maps to two IP addresses). This enables us to test access control when multiple addresses are returned from a DNS lookup.

9.2.12 Unapproved Server

A Raspberry Pi running a web server acts as an unapproved internet host and is used to test the communication between a MUD-capable IoT device and an internet host that is not included in the device's MUD file, so the IoT device should not be permitted to send traffic to it. To verify that the traffic filters were applied as expected, communication to and from the unapproved server and the first MUD-capable IoT device (with profile1) was tested. This unapproved server (www.antd.local) maps to a single IP address and is set up on the public side of the switch.

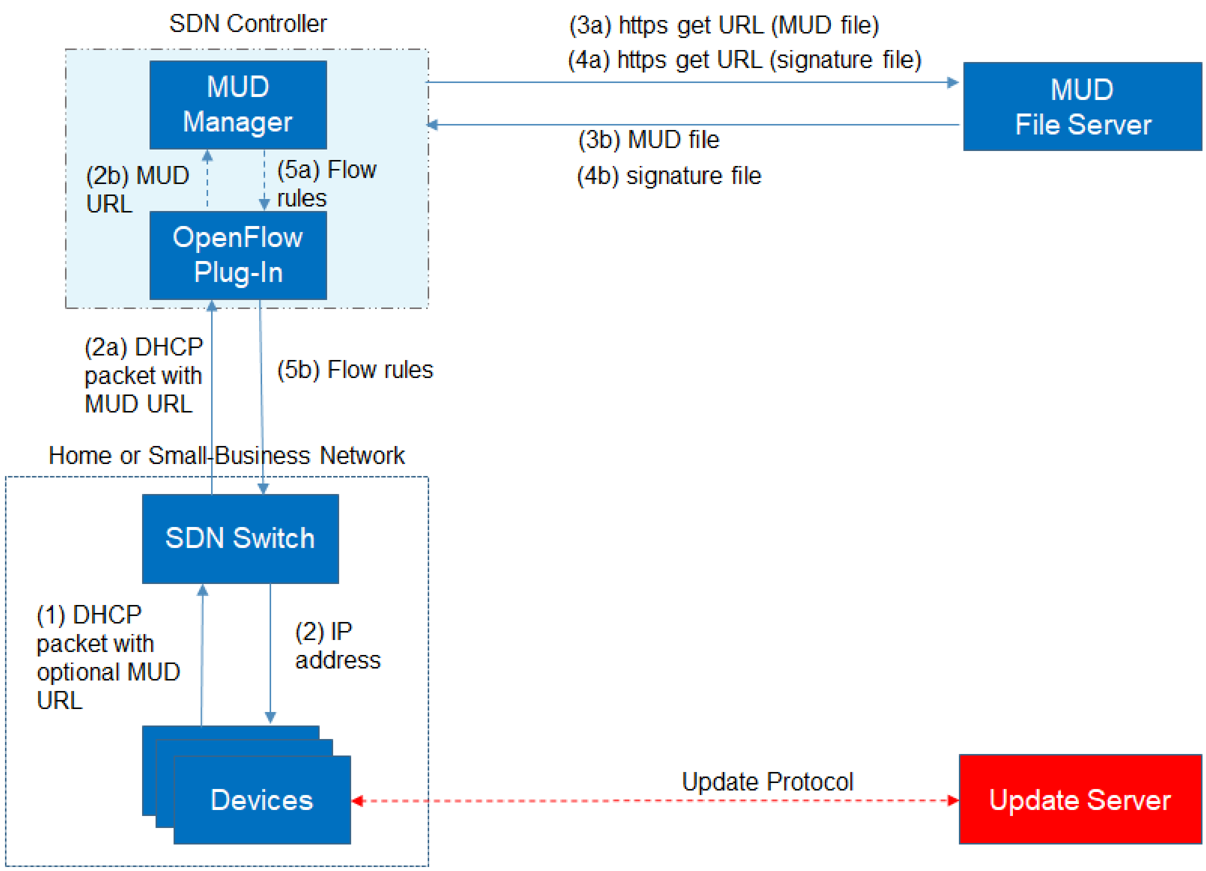
9.3 Build Architecture

In this section we present the logical architecture of Build 4 relative to how it instantiates the reference architecture depicted in Figure 4-1. We also describe Build 4's physical architecture and present message flow diagrams for some of its processes.

9.3.1 Logical Architecture

Figure 9-1 depicts the logical architecture of Build 4. It includes a single device that serves as the SDN controller/MUD manager, which is assumed to be cloud-resident. This SDN controller/MUD manager controls and manages an OpenFlow-enabled SDN switch on the home/small-business network. The SDN switch serves as the MUD policy enforcement point for MUD-capable IoT devices that connect to the home/small-business network. The only automatic MUD URL discovery capability that Build 4 supports is emission of the MUD URL via DHCP. Build 4 does not support LLDP-based or certificate-based MUD URL discovery. However, it is also possible to associate a MUD file with a device that is not capable of emitting a MUD URL by manually associating that device's MAC address with a MUD file URL when using Build 4.

Figure 9-1 Logical Architecture—Build 4



As shown in Figure 9-1, the steps that occur when a MUD-capable IoT device connects to the home/small-business network using Build 4 are as follows:

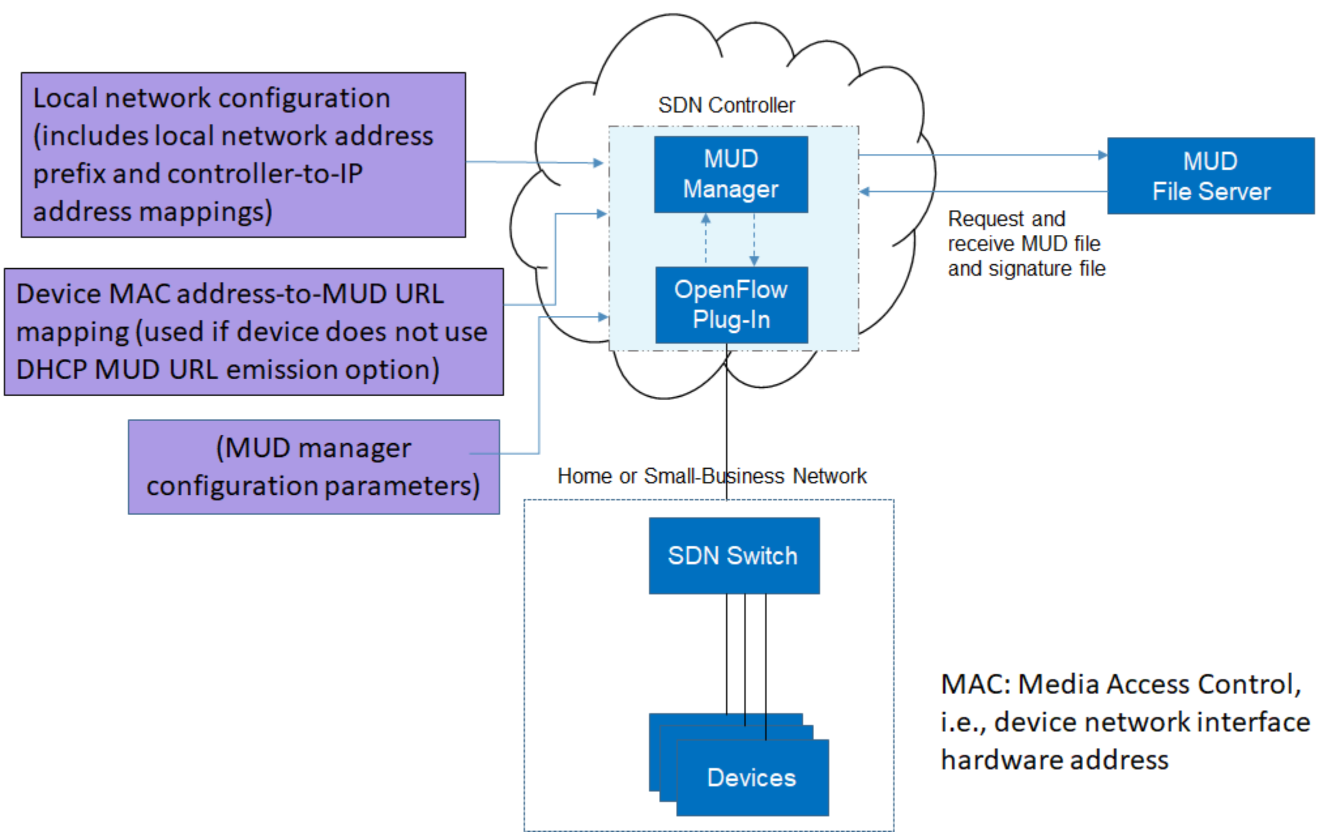
- Upon connecting a MUD-capable device, the MUD URL is emitted via DHCP (step 1).
- The SDN switch sends the DHCP packet containing the MUD URL to the SDN controller/MUD manager via the OpenFlow protocol (step 2a); this is passed from the OpenFlow plug-in to the MUD manager (step 2b).
- Simultaneously, the device is assigned an IP address (step 2).
- Once the DHCP packet is received at the MUD manager, the MUD manager extracts the MUD URL from the DHCP packet and requests the MUD file from the MUD file server by using the MUD URL (step 3a); if successful, the MUD file server at the specified location will serve the MUD file (step 3b).

- Next, the MUD manager requests the signature file associated with the MUD file (step 4a) and upon receipt (step 4b) verifies the MUD file by using its signature file.
- After the MUD file has been verified successfully, the MUD manager creates flow rules corresponding to the MUD file ACEs and provides these to the OpenFlow plug-in (step 5a), which in turn sends the flow rules to the SDN switch, where they are applied (step 5b).

Once the device's flow rules are installed at the SDN switch, the MUD-capable IoT device will be able to communicate with approved local hosts and internet hosts as defined in the MUD file, and any unapproved communication attempts will be blocked. Devices that are not MUD-capable will not have their communications restricted in any way by the MUD manager, assuming they have not been manually associated with a MUD file.

Figure 9-2 depicts some configuration information that can be provided to the Build 4 SDN controller/MUD manager via its REST API.

Figure 9-2 Example Configuration Information for Build 4



As shown in Figure 9-2, the MUD manager exports a YANG-based REST API to allow administrators to configure the SDN controller/MUD manager. This API is not exposed to the network users. It provides the following capabilities:

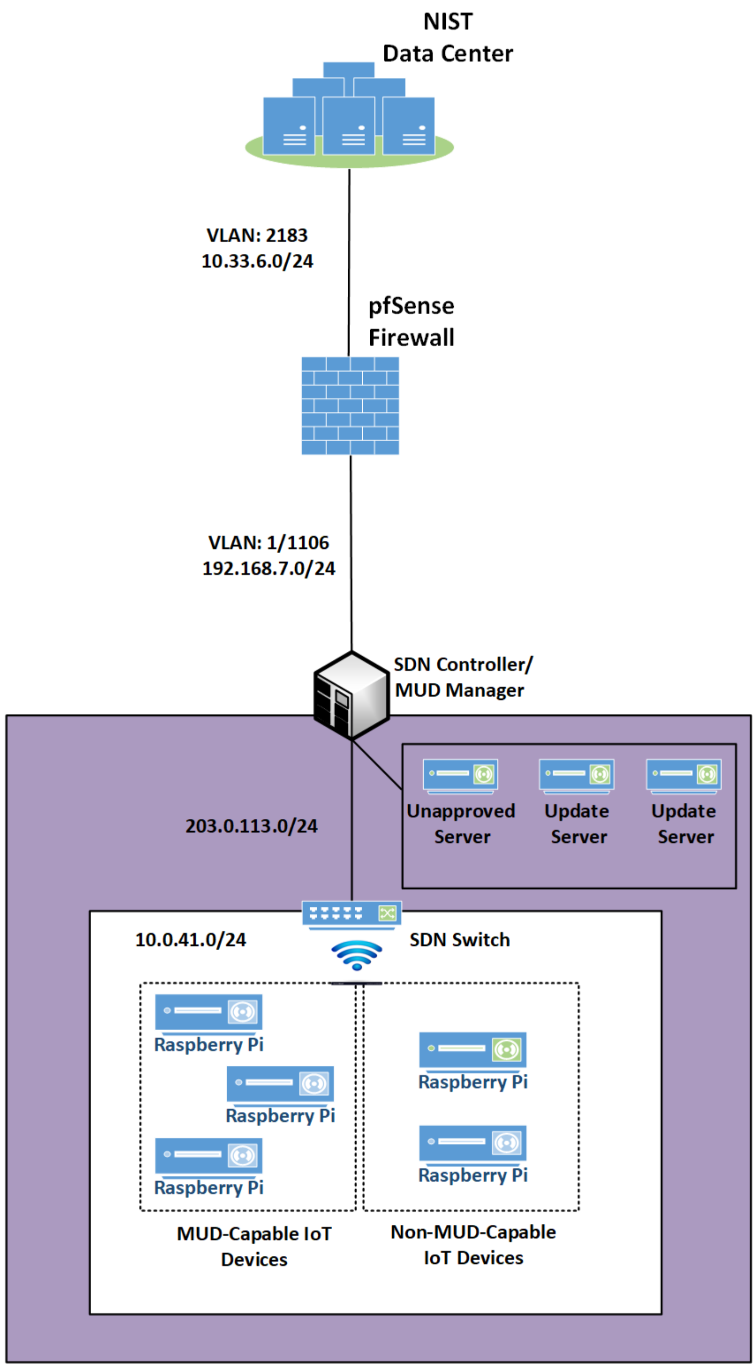
- application configuration—This allows the network administrator to define parameters for the application. The SDN controller/MUD manager must be provided with configuration information for the home and small-business networks that it manages. In addition, configuration parameters for the MUD manager must be supplied.
- controller-class mapping API—This allows the network administrator to define “well-known” network services such as DNS, NTP, and DHCP on the local network and the address prefix used for “local networks.”
- device-association—In Build 4, the MUD file URL can be provided to the MUD manager by using the normal DHCP-based MUD URL emission mechanism that is depicted in Figure 9-1. Alternatively, to support devices that are not able to emit a MUD URL, the network administrator can use the REST API to optionally define an association between a device MAC address and a MUD URL.
- MUD file supplied directly—A network administrator can optionally provide a MUD file to the MUD manager by copying it directly into the controller cache in case the manufacturer does not provide a MUD file server.

9.3.2 Physical Architecture

Figure 9-3 depicts the physical architecture of Build 4. A single DHCP server instance is configured for the local network to dynamically assign IPv4 addresses to each IoT device that connects to the SDN switch. This single subnet hosts both MUD-capable and non-MUD-capable IoT devices. The network infrastructure as configured utilizes the IPv4 protocol for communication both internally and to the internet.

The SDN switch is connected across a Wide Area Network (WAN) to the SDN controller/MUD manager. This connection allows the SDN switch to be managed by the SDN controller/MUD manager and enables network flow rules to be updated appropriately. The update servers and unapproved server for Build 4 are also located in this WAN.

3032 **Figure 9-3 Physical Architecture—Build 4**



9.3.3 Message Flow

This section presents the message flows used in Build 4 during several different processes of note.

NIST MUD works by using six flow tables containing flow rules that are applied to each packet in the following order:

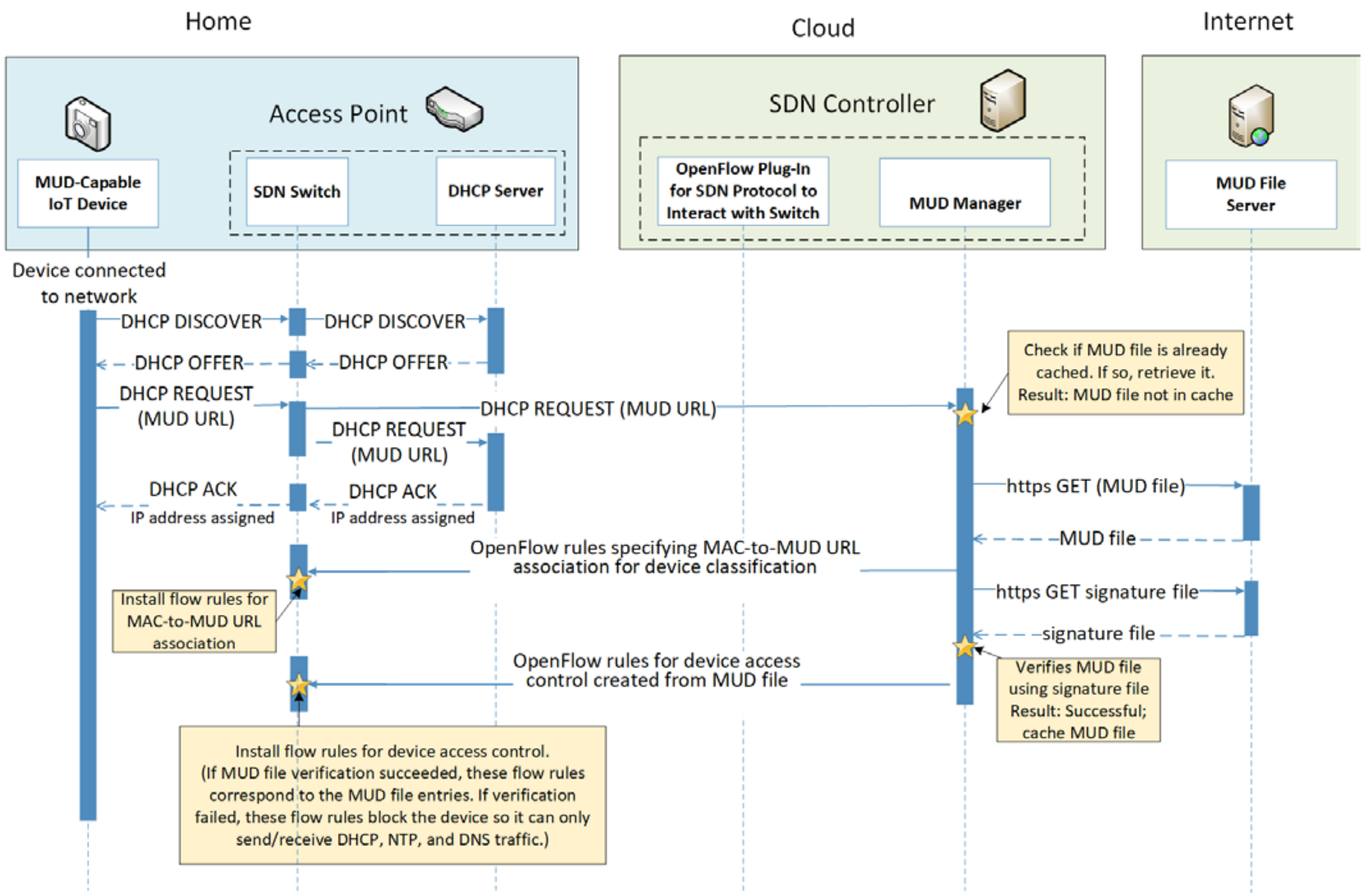
- Table 0, Source MAC address classification table, classifies a packet based on its source IP/MAC address.
- Table 1, Destination MAC address classification table, classifies a packet based on its destination IP/MAC address.
- Table 2, From-Device flow rules table, associates ACEs with the packet based on the packet's source classification if such ACEs exist. ACEs in this table correspond to the From-Device policy in the MUD file. The MUD-specific ACEs that are applied in this table are matched to the packet based on metadata assigned in the first two tables.
- Table 3, To-Device flow rules table, associates ACEs with the packet based on the packet's destination classification if such ACEs exist. ACEs in this table correspond to the To-Device policies in the MUD file. The MUD-specific ACEs that are applied in this table are matched to the packet based on metadata assigned in the first two tables.
- Table 4, Pass-Through table—If a packet has an ACE associated with it (i.e., if it has had a MUD-specific ACE applied to it by table 2 or by table 3 that indicates that it should be permitted), it will be sent to this table and the SDN switch will forward it. (For device-to-device communication based on the manufacturer, model, or local network constructs, there must be both a From-Device rule (in table 2) and a To-Device rule (in table 3) for the communication to be allowed. Otherwise the packet is dropped.)
- Table 5, Drop table—All packets from MUD-enabled devices are by default sent to the Drop table unless there is a MUD rule (and therefore a MUD-specific ACE) that applies to the packet indicating that the packet should be permitted (in which case the packet would have been sent to the Pass-Through table). Unprotected devices are metadata-associated with the reserved MUD URL "UNCLASSIFIED," which allows all packets to and from these devices to be permitted (i.e., there are rules in tables 2 and 3 that permit all traffic to these unprotected devices).

Note that a packet may have just one classification based on source and destination MAC/IP address. Packets originating from devices with assigned MUD URLs are not considered to be part of the local network. Hosts with controller classifications (including those with "well-known" controller classifications such as DHCP, DNS, and NTP servers) are not considered to be part of the local network.

9.3.3.1 Installing MUD-Based Access Control Rules for MUD-Capable Devices

Figure 9-4 shows the message flow that occurs when a MUD-capable device connects to the home/small-business network in Build 4.

Figure 9-4 MUD-Based Flow Rules Installation Message Flow—Build 4



As shown in Figure 9-4, the message flow is as follows:

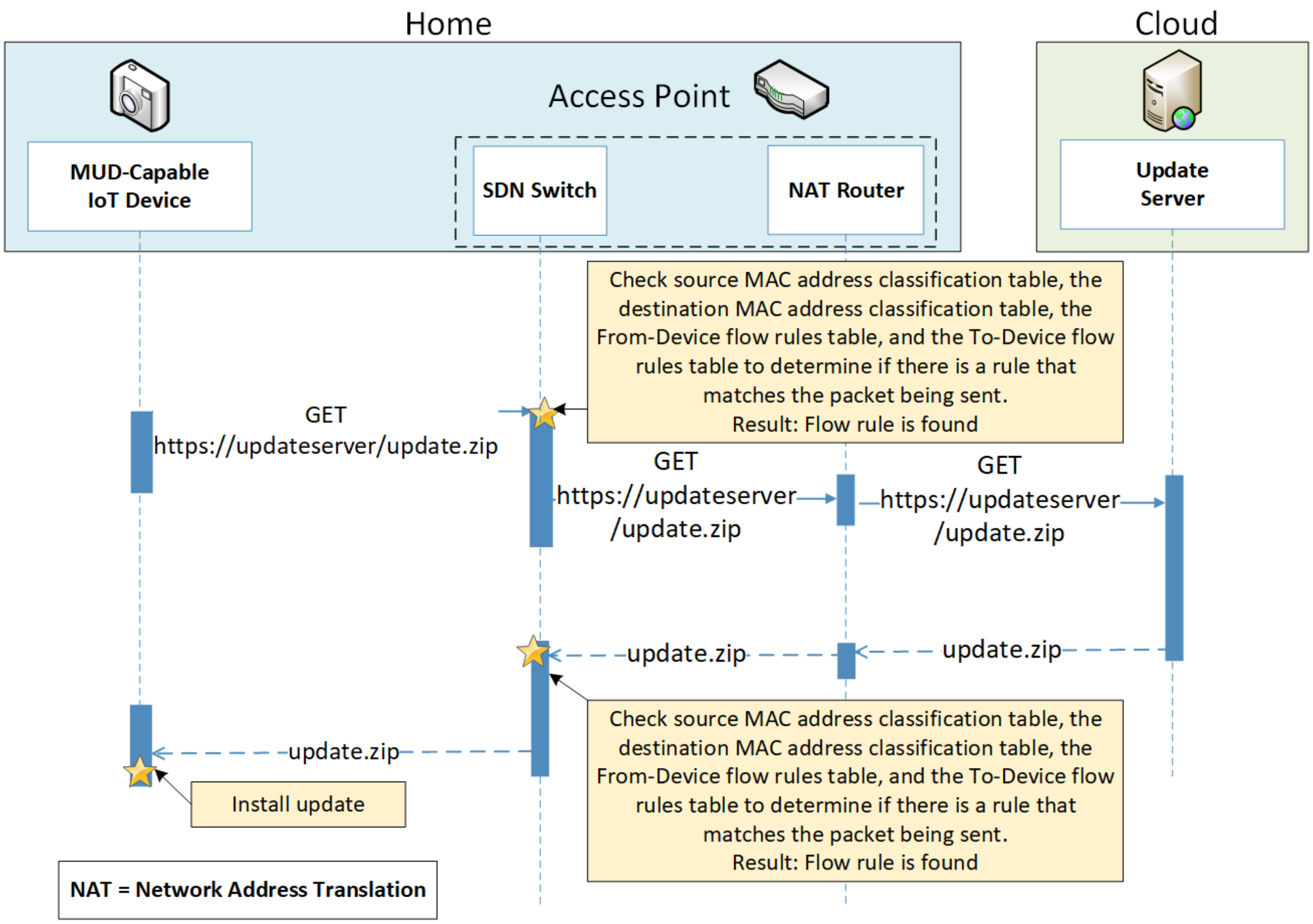
- The IoT device sends out a DHCP DISCOVER message to the SDN switch.
- The AP resident DHCP server sends back a DHCP offer that gets sent back to the device via the SDN switch.
- The device then sends out a DHCP request containing the MUD URL, which gets sent simultaneously to the AP resident DHCP server by the SDN switch and to the MUD manager.
- The AP resident DHCP server sends an IP address to the device in a DHCP ACK message via the switch.
- Based on the MUD URL presented in the DHCP request, the MUD manager checks to see if the corresponding MUD file is already cached. In the example depicted, the MUD file is not in the cache.
- The MUD manager retrieves the MUD file from the manufacturer server.

- The MUD manager installs packet classification flow rules into flow tables 0 and 1 (see Section 9.3.3.4) on the SDN switch. These classification rules associate the MAC address of the device interface with the MUD URL. Other classification information such as whether the packet belongs to the local network is also assigned in the first two tables. Table 0 is for source classification and table 1 is for destination classification. If the device had previously sent out packets, i.e., before it was associated with a MUD file, they would have been classified as UNCLASSIFIED in tables 0 and 1. Hence, the entries in tables 0 and 1 that correspond to the device must be cleared at this point and repopulated so subsequent packets are associated with the MUD URL.
- The MUD manager installs the MUD file ACEs as a set of flow rules in tables 2 and 3 (see Section 9.3.3.4).

9.3.3.2 Updates

After a device has been permitted to connect to the home/small-business network, it should periodically check for updates. The message flow for updating the IoT device is shown in Figure 9-5.

3095 **Figure 9-5 Update Process Message Flow—Build 4**



3096 As shown in Figure 9-5, the message flow is as follows:

3097 - The device generates an https GET request to its update server.

3098 - The SDN switch will consult its flow rules for this device to verify that it is permitted to send
3099 traffic to the update server. Assuming there were explicit rules in the device's MUD file enabling
3100 it to send messages to this update server, the SDN switch will forward the request to the NAT
3101 router, which will then forward it to the update server.

3102 - The update server will respond with a zip file containing the updates.

3103 - The return traffic will be sent via the NAT router to the switch.

3104 - The destination MAC address of the packet identifies the device, and appropriate metadata is
3105 assigned in table 1.

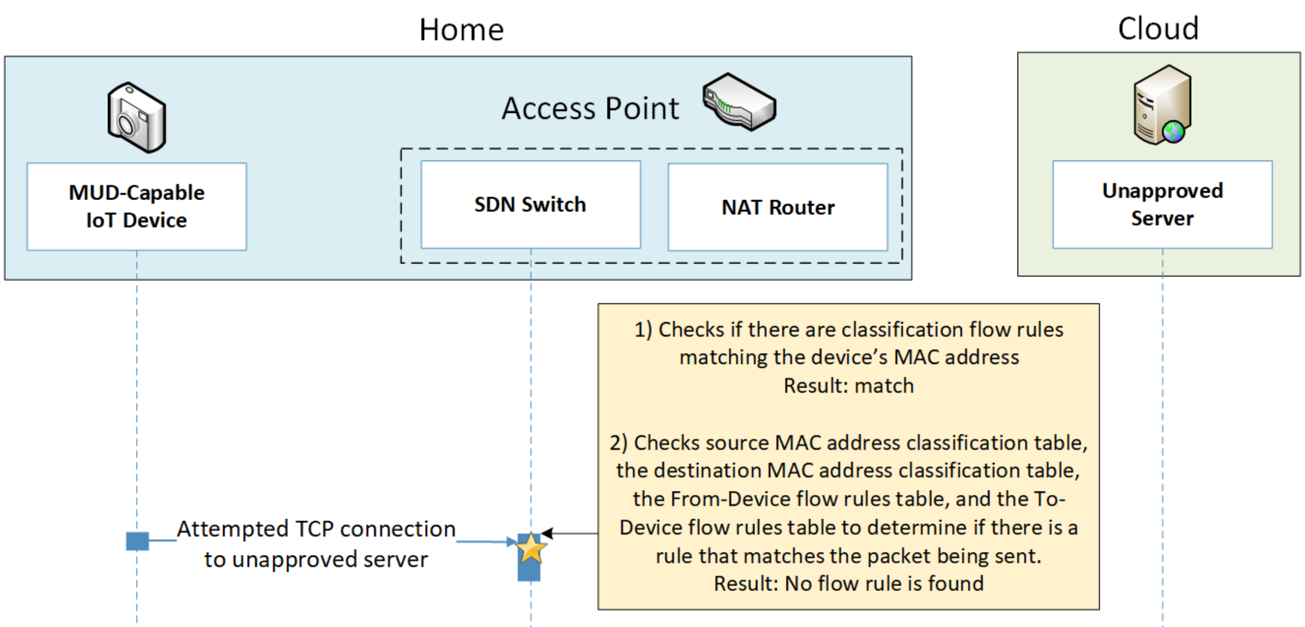
3106 - The source MAC and IP are UNCLASSIFIED, and appropriate metadata is assigned in table 0.

- The packet is forwarded through table 2 and finds a matching flow rule in table 3 from where it is forwarded to the Pass-Through table (4). Two-way communication is thus established.
- The SDN switch will forward this zip file to the device for installation.

9.3.3.3 Prohibited Traffic

Figure 9-6 shows the message flow that occurs when an IoT device attempts to send traffic that is not permitted by its MUD file.

Figure 9-6 Unapproved Communications Message Flow—Build 4



As shown in Figure 9-6, the message flow is as follows:

- A TCP packet is originated from the IoT device with a source MAC address of the device's switch-facing interface and a destination MAC address that is set to the AP-resident router's switch-facing interface. The source IP address is set to the device IP address and destination IP address is set to the unapproved server IP address.
- The packet arrives at the SDN switch, at which point it:
  - enters flow tables 0 and 1, where it is classified and receives the following metadata assignment as a result:
    - <<source-manufacturer, source-model, is-local> <dest-manufacturer, dest-model, is-local>> is assigned in tables 0 and 1

The <source-manufacturer, source-model> are obtained from the MUD URL assigned to the packet. The is-local flag will be set to False because devices with MUD URLs assigned are not considered to be part of the local network.

The destination manufacturer and model assignments will be UNCLASSIFIED, UNCLASSIFIED and is-local is false because the router MAC address is UNCLASSIFIED, and the destination IP address is not part of the local network. Thus, the metadata assignment after table 0 and 1 are traversed will be

<<source-manufacturer,source-model,False><UNCLASSIFIED,UNCLASSIFIED,False>>

- enters flow table 2, where source metadata-based flow rules have been previously inserted
    - If there is a flow rule that allows the communication, the packet is sent to table 4 (the Pass-Through table), which allows the communication. In the example scenario that is depicted in Figure 9-6, there is no flow rule in table 3 that allows the communications.
    - However, there is a flow rule in table 2 that matches the <source-manufacturer, source-model> that sends the packet to the Drop table (table 5).

- In the example scenario depicted, there is no flow rule found that matches the packet that the IoT device is attempting to send. Therefore, the SDN switch sends the packet to table 5 where there is a single rule that drops the packet.

9.3.3.4 Installation of Timed-Out Flow Rules and Eventual Consistency

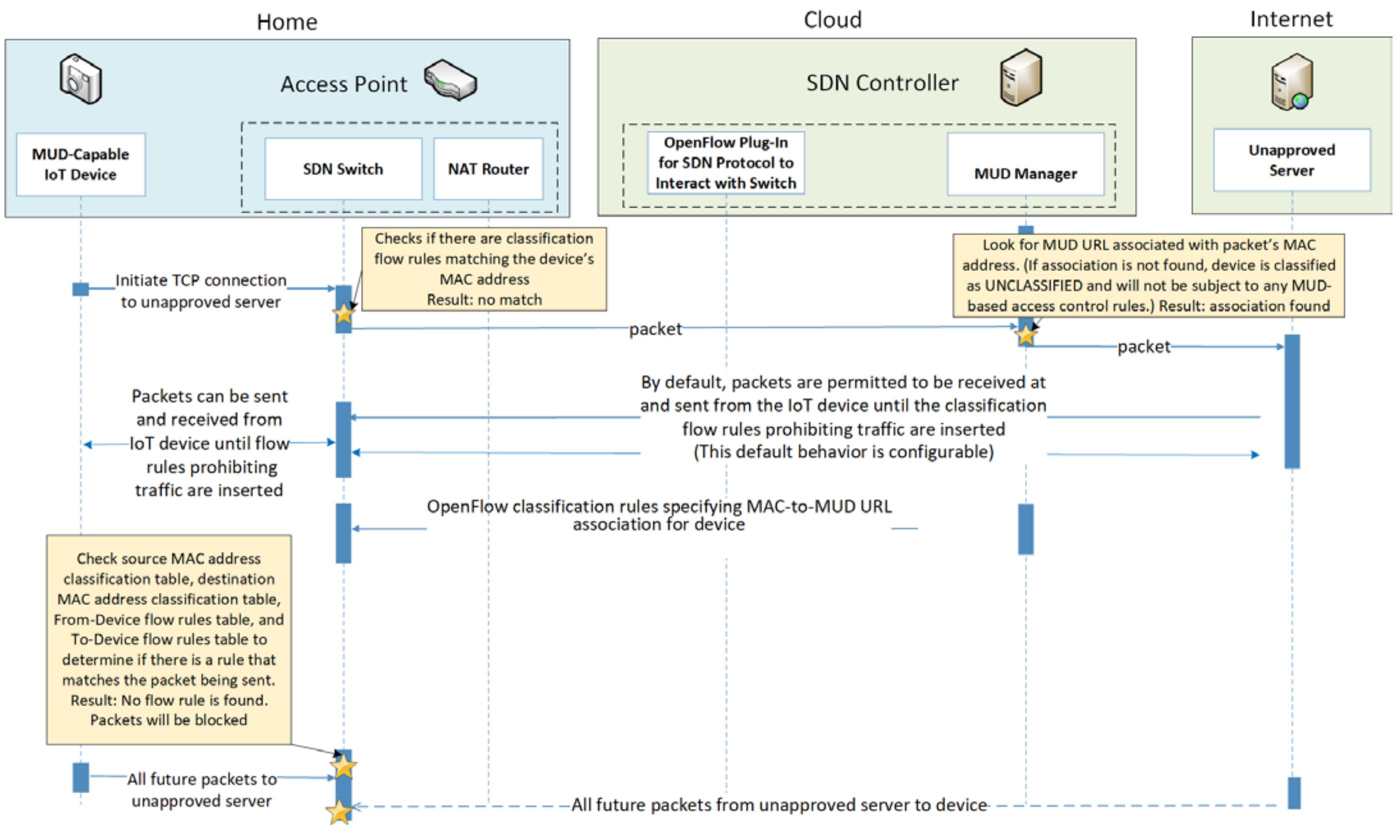
Insertion of flow rules onto the SDN switch on the home/small-business network is dynamic. Rules are computed at the SDN controller/MUD manager and installed on the SDN switch. Flow rules are configured to time out on inactivity to avoid having the SDN switch's flow table fill up. (If an IoT device disconnects from the home/small-business network, there is no need to continue to maintain flow rules for that device on the switch. However, if a device's IP address lease times out, the DHCP server, which has not been modified at all, will not alert the SDN controller/MUD manager of this event. Thus, having the rules time out is an alternative to ensure that rules for disconnected devices will eventually be removed from the switch.)

If an IoT device tries to send a packet, if a packet intended for that device is received at the switch and the source or destination MAC address of the packet does not yet have classification flow rules on the switch, or if the classification flow rules for one or both of those MAC addresses have timed out, the flow rules will need to be sent from the SDN controller/MUD manager to the switch. In this situation, the default OpenFlow rule at the switch (which is inserted in tables 0 and 1 when the switch connects) sends the packet to the MUD manager, and consequently a packet-in event encapsulating the packet is generated at the MUD manager. The packet classification flow rules are then computed and pushed to the switch by the MUD manager during processing of the packet-in event. During this period, additional packets may arrive at the switch.

A design decision had to be made regarding whether to permit the IoT device to send and receive traffic during the window of time while its flow rules are being computed and pushed to the switch. The decision was made to allow an “eventually consistent” model. That is, packets sent by or intended for the IoT device are permitted to proceed through the switch while the SDN flow rules for packet classification are being computed at the SDN controller/MUD manager and sent to the switch. This may result in a few packets that are prohibited by the MUD file ACEs getting through before such violating flows are eventually blocked. This can happen the first time a device sends a packet and every time the flow rules time out due to inactivity. Thus, a misbehaving device or an attacker can have small windows of time during which packets that the MUD file intends to prohibit will be permitted to be exchanged with the device. The alternative is to block the packets while flow rules are computed and inserted. While this alternative behavior can be configured in NIST-MUD, it is not a recommended configuration because it blocks the processing pipeline (resulting in packet drops) while the flow rules are being computed and pushed.

Figure 9-7 shows the message flow that occurs when a device whose flow rules have timed out attempts to initiate communications with an unapproved external server, i.e., a server that is not explicitly listed as a permissible destination in the device’s MUD file.

3176 **Figure 9-7 Installation of Timed-Out Flow Rules and Eventual Consistency Message Flow—Build 4**



3177 As shown in Figure 9-7, the message flow is as follows:

- 3178 The MUD-capable IoT device sends a packet attempting to initiate a TCP connection to an
3179 unapproved server.
- 3180 The SDN switch checks to see if it has packet classification flow rules for this device (which it
3181 determines by looking for rules that match the device's MAC address in tables 0 and 1). In this
3182 case, no flow rules are found for this device.
- 3183 The SDN switch sends the packet to the SDN controller/MUD manager as a result of the default
3184 rule. This is delivered in a packet-in event at the MUD manager.
- 3185 The MUD manager receives the packet-in event and looks to see if there is a MUD URL
3186 associated with the device's MAC address. (If the device does not have an associated MUD file,
3187 it will not be subject to any MUD-based access control rules and will be assigned a reserved
3188 MUD URL of UNCLASSIFIED.) In the example scenario depicted in Figure 9-7, the device was
3189 found to be associated with a MUD file.

- Even though the flow rules corresponding to the sending device's MUD file are not currently installed on the switch, the SDN controller/MUD manager forwards the packet to the unapproved server.
- The unapproved server responds with an acknowledgment packet.
- The IoT device and the unapproved server are permitted to exchange packets for the time being.
- Meanwhile, the MUD manager computes the SDN flow rules that correspond to the device's MUD file and installs them on the SDN switch.
- After the flow rules have been installed on the switch, when the IoT device attempts to send a packet to the unapproved server, the switch will check each of its flow tables in order (i.e., it will check the Source MAC address classification table [table 0], Destination MAC address classification table [table 1], From-Device flow rules table [table 2], and To-Device flow rules table [table 3]) to determine if there is an ACE that matches the packet being sent. In the example scenario depicted, the switch will find packet classification flow rules for the device in tables 0 and 1, but it will not find any matching flow rules in table 2, indicating that the IoT device's MUD file did not contain an ACE that permits the packet to be sent. As a result, the switch will drop the packet.
- In addition, any subsequent packets that may be sent by the unapproved server and received at the SDN switch will be similarly blocked as a result of the switch consulting its flow rules and determining that there are no ACEs that permit the unapproved server to send packets to the IoT device.

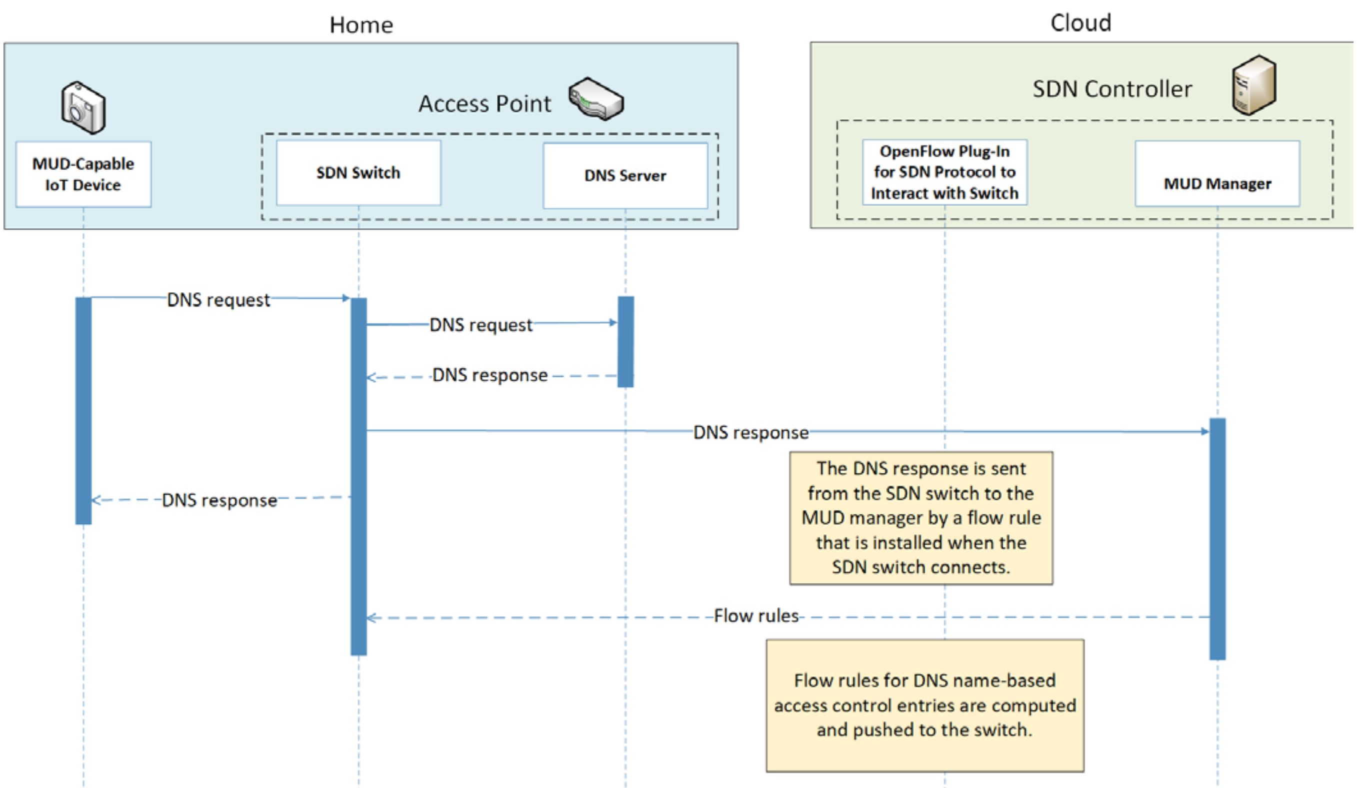
9.3.3.5 DNS Events

MUD allows traffic flow rules to be based on domain names. However, the corresponding SDN flow rules configured in the SDN switch must be based on IP addresses rather than domain names. The MUD manager needs to resolve each host name that is in a MUD file ACE rule to the same value to which it would be resolved by the MUD-enabled IoT device. NIST-MUD is built on the assumption that the SDN controller/MUD manager, which is assumed to be in the cloud, does not necessarily have access to the same DNS resolver as the home/small-business network. Therefore, the SDN controller/MUD manager cannot simply issue DNS queries to resolve domain names that are in MUD files and populate the SDN switch's flow table with the IP addresses that it receives back because the IP addresses that the SDN controller/MUD manager would receive back may not be the same as those that the IoT device would receive back. Instead, as DNS packets are sent from the IoT devices through the SDN-enabled switch, they are also sent to the SDN controller/MUD manager, enabling the SDN controller/MUD manager to snoop on DNS queries and responses that occur on the home/small-business network. The SDN controller/MUD manager extracts the IP address resolution information from each DNS response and uses that information to populate the flow table with the appropriate IP address for rules in the MUD file.

Each time a domain name is resolved for a device on the home/small-business network, the MUD manager must check to determine if there are any flow rules that use that domain name that had previously been deferred (i.e., that have not yet been instantiated and sent to the switch) because the IP address corresponding to that domain name had not yet been known. If so, the MUD manager must instantiate those flow rules by inserting the IP address that corresponds to that domain name in place of that domain name and sending the flow rules to the SDN switch.

Figure 9-8 shows the message flow that occurs when the MUD-capable device does a DNS name lookup and the SDN controller/MUD manager uses the IP address returned in the DNS response to instantiate deferred flow rules for installation on the SDN switch.

Figure 9-8 DNS Event Message Flow—Build 4



As shown in Figure 9-8, the message flow is as follows:

- The IoT device (or any device on the network managed by the switch) does a name lookup by sending a DNS request to the SDN switch, which has a default rule that allows access to DNS.

- The SDN switch forwards the DNS request to a DNS server. In our experiment, this DNS server is resident on the access point.
- The DNS server sends a DNS response back to the SDN switch. The response contains a domain name resolution. Note that if the access point were configured to use an upstream DNS server, the response would be returned from that server and routed back to the device via the switch. For simplicity and control of our experimental setup, we use the AP-resident DNS server so there is no routing of DNS request and response.
- The SDN switch sends the DNS response to the MUD manager, which caches the name resolution information for the switch and updates any DNS-name-based ACEs for MUD files that it manages.
- Concurrently with the previous step, the SDN switch also sends the DNS response to the device that originally generated the DNS request.
- The MUD manager instantiates flow rules corresponding to these DNS-name-based ACEs by substituting each domain's IP address for its domain name and installing the flow rules into flow tables 2 and 3 on the SDN switch.

9.4 Functional Demonstration

A functional evaluation and a demonstration of Build 4 were conducted that involved evaluation of conformance to the MUD RFC. Build 4 was tested to determine the extent to which it correctly implements basic functionality defined within the MUD RFC.

Table 9-2 summarizes the tests that were performed to evaluate Build 4's MUD-related capabilities. It lists each test identifier, the test's expected and observed outcomes, and the applicable Cybersecurity Framework Subcategories and NIST SP 800-53 controls for which each test is designed to verify support. The tests that are listed in the table are detailed in a separate supplement for functional demonstration results. Boldface text is used to highlight the gist of the information that is being conveyed.

Table 9-2 Summary of Build 4 MUD-Related Functional Tests

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
IoT-1	**ID.AM-1:** Physical devices and systems within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-2:** Software platforms and applications within the organization are inventoried.	A **MUD-enabled IoT device is configured to emit a MUD URL.** The MUD manager requests the MUD file and signature from the MUD file server, and the MUD file server	Upon connection to the network, the MUD-enabled IoT device has its MUD **PEP router/switch automatically configured according to the MUD file's**	Pass

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	NIST SP 800-53 Rev. 4 CM-8, PM-5 **ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **DE.AE-1:** A baseline of network operations and expected data flows for users and systems is established and managed. **PR.AC-4:** Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC-5, AC-6, AC-14, AC-16, AC-24 **PR.AC-5:** Network integrity is protected, incorporating network segregation where appropriate. **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7 **PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10 **PR.IP-3:** Configuration change control processes are in place.	serves the MUD file to the MUD manager. The MUD file explicitly permits traffic to/from some internet services and hosts, and implicitly denies traffic to/from all other internet services. **The MUD manager translates the MUD file information into local network configurations that it installs on the router or switch that is serving as the MUD PEP for the IoT device.**	**route-filtering policies.**	

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	NIST SP 800-53 Rev. 4 CM-3, CM-4, SA-10 **PR.PT-3:** The principle of least functionality is incorporated by configuring systems to provide only essential capabilities. **NIST SP 800-53 Rev. 4** AC-3, CM-7 **PR.DS-2:** Data in transit is protected.			
IoT-2	**PR.AC-7:** Users, devices, and other assets are authenticated (e.g., single-factor, multifactor) commensurate with the risk of the transaction (e.g., individuals' security and privacy risks and other organizational risks). **NIST SP 800-53 Rev. 4** AC-7, AC-8, AC-9, AC-11, AC-12, AC-14, IA-1, IA-2, IA-3, IA-4, IA-5, IA-8, IA-9, IA-10, IA-11	A MUD-enabled IoT device is configured to emit a URL for a MUD file, but the **MUD file server that is hosting that file does not have a valid TLS certificate. Local policy has been configured to ensure that if the MUD file for an IoT device is located on a server with an invalid certificate, the router/switch will be configured to deny all communication to/from the device.**	When the MUD-enabled IoT device is connected to the network, the MUD manager sends locally defined policy to the router/switch that handles whether to allow or block traffic to the MUD-enabled IoT device. Therefore, the **MUD PEP router/switch will be configured to block all traffic to and from the IoT device.**	Pass
IoT-3	**PR.DS-6:** Integrity-checking mechanisms are used to verify software, firmware, and information integrity. **NIST SP 800-53 Rev. 4** SI-7	A MUD-enabled IoT device is configured to emit a URL for a MUD file, but the **certificate that was used to sign the MUD file had already expired at signing. Local policy has been configured to ensure that if the MUD file for a device has a signature that was**	When the MUD-enabled IoT device is connected to the network and the MUD file and signature are fetched, the MUD manager will detect that the MUD file's signature was created by using a certificate that had already expired	Pass

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
		signed by a certificate that had already expired at the time of signature, the device's MUD PEP router/switch will be configured to deny all communication to/from the device.	at signing. According to local policy, the **MUD PEP will be configured to block all traffic to/from the device.**	
IoT-4	**PR.DS-6:** Integrity-checking mechanisms are used to verify software, firmware, and information integrity. **NIST SP 800-53 Rev. 4** SI-7	A MUD-enabled IoT device is configured to emit a URL for a MUD file, but the **signature of the MUD file is invalid. Local policy has been configured to ensure that if the MUD file for a device is invalid, the router/switch will be configured to deny all communication to/from the IoT device.**	When the MUD-enabled IoT device is connected to the network, the MUD manager sends locally defined policy to the router/switch that handles whether to allow or block traffic to the MUD-enabled IoT device. Therefore, the **MUD PEP router/switch will be configured to block all traffic to and from the IoT device.**	Pass
IoT-5	**ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **PR.IP-1:** A baseline configuration of information technology/industrial	Test IoT-1 has run successfully, meaning that the MUD PEP router/switch has been configured based on a **MUD file that permits traffic to/from some internet locations and implicitly denies traffic to/from all other internet locations.**	When the MUD-enabled IoT device is connected to the network, its MUD PEP **router/switch will be configured to enforce the route filtering that is described in the device's MUD file** with	Pass

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10 **PR.PT-3:** The principle of least functionality is incorporated by configuring systems to provide only essential capabilities. **NIST SP 800-53 Rev. 4** AC-3, CM-7		respect to traffic being permitted to/from some internet locations, and traffic being implicitly blocked to/from all remaining internet locations.	
IoT-6	**ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **PR.AC-5:** Network integrity is protected, incorporating network segregation where appropriate. **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7 **PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10 **PR.IP-3:** Configuration change control processes are in place.	Test IoT-1 has run successfully, meaning that the MUD PEP router/switch has been configured based on a **MUD file that permits traffic to/from some lateral hosts and implicitly denies traffic to/from all other lateral hosts.** (The MUD file does not explicitly identify the hosts as lateral hosts; it identifies classes of hosts to/from which traffic should be denied, where one or more hosts of this class happen to be lateral hosts.)	When the MUD-enabled IoT device is connected to the network, its MUD PEP **router/switch will be configured to enforce the access control information that is described in the device's MUD file** with respect to traffic being permitted to/from some lateral hosts, and traffic being implicitly blocked to/from all remaining lateral hosts.	Pass

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	PR.PT-3: The principle of least functionality is incorporated by configuring systems to provide only essential capabilities. **NIST SP 800-53 Rev. 4** AC-3, CM-7 **PR.DS-3:** Assets are formally managed throughout removal, transfers, and disposition.			
IoT-9	**ID.AM-1:** Physical devices and systems within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-2:** Software platforms and applications within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4 **DE.AE-1:** A baseline of network operations and expected data flows for users and systems is established and managed. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CM-2, SI-4 **PR.AC-4:** Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC-5, AC-6, AC-14, AC-16, AC-24	Test IoT-1 has run successfully, meaning the MUD PEP **router/switch has been configured based on the MUD file** for a specific MUD-capable device in question. The MUD file contains domains that resolve to multiple IP addresses. The MUD PEP router/switch should be configured to permit communication to or from all IP addresses for the domain.	A domain in the MUD file resolves to two different IP addresses. The MUD manager will create firewall rules that permit the MUD-capable device to send traffic to both IP addresses. The MUD-capable device attempts to send traffic to each of the IP addresses, and the MUD PEP router/switch permits the traffic to be sent in both cases.	Pass

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	PR.AC-5: Network integrity is protected, incorporating network segregation where appropriate. **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7 **PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM- 5, CM-6, CM-7, CM-9, SA-10 **PR.IP-3:** Configuration change control processes are in place. **NIST SP 800-53 Rev. 4** CM-3, CM-4, SA-10 **PR.DS-2:** Data in transit is protected. **NIST SP 800-53 Rev. 4** SC-8, SC-11, SC-12			
IoT-10	**ID.AM-1:** Physical devices and systems within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-2:** Software platforms and applications within the organization are inventoried. **NIST SP 800-53 Rev. 4** CM-8, PM-5 **ID.AM-3:** Organizational communication and data flows are mapped. **NIST SP 800-53 Rev. 4** AC-4, CA-3, CA-9, PL-8 **PR.DS-5:** Protections against data leaks are implemented. **NIST SP 800-53 Rev. 4** AC-4, AC-5, AC-6, PE-19, PS-3, PS-6, SC-7, SC-8, SC-13, SC-31, SI-4	A MUD-capable IoT device is configured to emit a MUD URL. Upon being connected to the network, its MUD file is retrieved, and the PEP is configured to enforce the policies specified in that MUD URL for that device. **Within 24 hours (i.e., within the cache-validity period for that MUD file), the IoT device is reconnected to the network.** After 24 hours have	Upon reconnection of the IoT device to the network, **the MUD manager does not contact the MUD file server. Instead, it uses the cached MUD file.** It translates this MUD file's contents into appropriate route-filtering rules and installs these rules onto the PEP for the IoT device. Upon reconnection of the IoT device to the network, after 24	Pass

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
	DE.AE-1: A baseline of network operations and expected data flows for users and systems is established and managed. **PR.AC-4:** Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties. **NIST SP 800-53 Rev. 4** AC-1, AC-2, AC-3, AC-5, AC-6, AC-14, AC-16, AC-24 **PR.AC-5:** Network integrity is protected, incorporating network segregation where appropriate. **NIST SP 800-53 Rev. 4** AC-4, AC-10, SC-7 **PR.IP-1:** A baseline configuration of information technology/industrial control systems is created and maintained, incorporating security principles (e.g., concept of least functionality). **NIST SP 800-53 Rev. 4** CM-2, CM-3, CM-4, CM-5, CM-6, CM-7, CM-9, SA-10 **PR.IP-3:** Configuration change control processes are in place. **NIST SP 800-53 Rev. 4** CM-3, CM-4, SA-10 **PR.PT-3:** The principle of least functionality is incorporated by configuring systems to provide only essential capabilities. **NIST SP 800-53 Rev. 4** AC-3, CM-7 **PR.DS-2:** Data in transit is protected.	elapsed, the same device is reconnected to the network.	hours have elapsed, the MUD manager does fetch a new MUD file.	

Test	Applicable Cybersecurity Framework Subcategories and NIST SP 800-53 Controls	Test Summary	Expected Outcome	Observed Outcome
IoT-11	**ID.AM-1:** Physical devices and systems within the organization are inventoried.	A **MUD-enabled IoT device can emit a MUD URL.** The device should leverage one of the specified manners for emitting a MUD URL.	Upon initialization, the MUD-enabled IoT device broadcasts a DHCP message on the network, including at most one **MUD URL, in https scheme, within the DHCP transaction OR as an LLDP extension.**	Pass

9.5 Observations

NIST-MUD was able to successfully permit and block traffic to and from MUD-capable IoT devices as specified in the MUD files for the devices.

NIST-MUD does not implement LLDP extensions or certificate-based device authentication. (An authentication server can, however, inform the MUD manager of the MAC to MUD URL association using the API provided by NIST-MUD.) The current implementation supports devices that emit their MUD URL using the MUD DHCP extension or that are associated with their MUD URL by the provided API (i.e., the administrator or network authentication server configures the association).

NIST-MUD does not implement secure conveyance of the device's MUD URL. A device may "lie" about its identity by issuing a spurious DHCP request with a MUD URL embedded. There are no certificate-based checks to verify that the MUD URL that the device emits is in fact that device's MUD URL.

As was discussed in Section 9.3.3.4, a misbehaving device or an attacker can have small windows of time where illegal packets can be exchanged with a device the first time the device sends or receives packets after its flow rules have timed out. This is because the design decision was made to permit packets sent by or intended for the IoT device to proceed through the switch while the SDN flow rules for packet classification are being computed at the SDN controller/MUD manager and pushed to the switch. The alternative is to block the packets while classification rules are inserted. While this can be configured, it is not a recommended configuration because it disrupts correct behavior.

10 General Findings, Security Considerations, and Recommendations

This section introduces findings based on the build implementations and demonstrations, security considerations, and recommendations.

10.1 Findings

Based on our experiences with the various builds considered and demonstrated in this project, we offer the following findings:

- It is possible to achieve significantly better security than is typically achieved in today's (non-MUD-capable) home and small-business networks by deploying and using MUD on those networks to constrain the communications of IoT devices.
- MUD is designed to protect limited-purpose devices whose communication needs can be clearly defined. These communication needs are defined in terms of not only the ports and protocols with which the IoT devices are permitted to communicate but also the destinations with which the IoT devices can communicate. If a device is not a limited-purpose device but instead has very general communication requirements that cannot be clearly defined (e.g., a laptop or a phone), then the device does not lend itself to protection by MUD.
- The demonstrated approach, as implemented in each of the builds, shows that by using MUD-capable IoT devices on networks where support for MUD has been deployed, it is possible to manage access to MUD-capable IoT devices in a manner that maintains device functionality while
    - preventing access to the MUD-capable IoT device from other devices on the internal network that are not from manufacturers or device classes explicitly permitted by the MUD-capable device's MUD file
    - preventing the MUD-capable IoT device from being used to access unauthorized external domains
    - preventing the MUD-capable IoT device from accessing other devices on the internal network that are not from manufacturers or device classes explicitly permitted by the MUD-capable device's MUD file
- MUD can help prevent MUD-capable IoT devices from being used to launch DDoS and other network-based attacks that are typically made possible by commandeering IoT devices found on today's home and small-business networks. For MUD to provide this protection, it must be deployed correctly, networks must use MUD-capable IoT devices, and MUD files must be written and available for these devices so that the files authorize only the outgoing communications that each MUD-capable IoT device needs to maintain its intended functionality.

- There are commercially available network visibility/monitoring technologies that can detect connected devices and identify certain device attributes (e.g., type, IP address, OS) throughout the duration of a device's connection to the network. These "fingerprinting" technologies are also able to detect when the devices leave the network or are powered off and to note their change of status accordingly.
- Setup and configuration of the components needed to deploy MUD on a network (MUD-capable router/switch and MUD manager) should ideally be able to be performed easily, right out of the box, to enable typical home or small-business users to deploy MUD successfully. While Build 2 and Build 3 are plug-and-play solutions designed to be easily deployable, setup and configuration of the other builds are not currently sufficiently user-friendly to enable the typical, nontechnical user to deploy these implementations easily and seamlessly. For MUD to be widely deployed on home/small-business networks, emphasis on ease of use will be crucial.
- MUD has the potential to help with the security of even those IoT devices that have been deprecated and are no longer receiving regular updates. Eventually, most IoT devices will reach a point at which they will no longer be updated by their manufacturer. This is a dangerous point in any device's life cycle because it means that any of its security vulnerabilities that become known after this point will not be protected against, leaving the device open to attack. For MUD-capable devices that reach this end-of-life stage, however, the use of MUD provides additional protection that is not available to non-MUD-capable devices. Even if a MUD-capable device can no longer be updated, its MUD file will still limit the other devices with which that MUD-capable device is able to communicate, thereby limiting what other devices could be used to attack it and what other devices it could be used to attack. In the future, there are expected to be many IoT devices that are no longer being updated by their manufacturers but will continue to be used. The ability to leverage MUD to limit the communication profiles of such unsupported devices will be important for protecting these highly vulnerable devices from attack by unauthorized endpoints and for protecting the internet from attack by these vulnerable devices.
- Even when using components that are fully conformant to the MUD specification, there are still some behaviors that will be determined by local policy. If the default policy that is provided by a specific product out of the box is not sufficient, user action will be required to configure the device according to a different and desired policy. User-friendly interfaces will be needed to enable the typical, nontechnical user of a home or small-business network to interact with the MUD components to modify their default settings when needed. For example, the MUD specification does not dictate what action to take (e.g., block or permit traffic to the IoT device) if the MUD manager is not able to validate the device's MUD file server's TLS certificate or if the MUD manager is not able to validate the device's MUD file signature. In either of these cases, if the default behavior that the device is configured to perform is not acceptable to the user, the user would need to reconfigure the device to perform the desired behavior. Ideally the device would provide a user-friendly interface through which to do so.
- In the absence of mechanisms that enable users to configure the specific local policy that is enforced when encountering certain error situations, MUD manager implementers may want to

give additional thought to the local policies that the MUD manager enforces by default. There is a trade-off to be made between security and availability. Enforcing default local policies that are nuanced may enable an implementation to achieve a more desirable balance between security and availability in some situations. For example, the MUD RFC does not specify what behavior an implementation should exhibit when errors are experienced during retrieval or validation of a device's MUD file. A MUD file server could be found to have an invalid TLS certificate, which is highly suspicious and therefore concerning; or it could be found to have an otherwise valid TLS certificate that has simply expired, which may be less concerning. Similarly, the MUD file itself could be found to have an invalid signature (concerning) or a signature that is otherwise valid but whose associated certificate had expired at the time it was used to sign the MUD file (perhaps less concerning). Given the absence of guidance in the RFC regarding how an implementation should behave in such situations, the implementation is expected to behave according to local policy.

The implementation can fail closed, as do Builds 1 and 4, meaning that the device will not be permitted to send or receive any traffic. While such a policy is extremely secure, it also renders the devices unreachable and effectively useless. Alternatively, the implementation can fail open, as it does in the case of Builds 2 and 3, meaning that the device is permitted to communicate freely, as if it does not have an associated MUD file. Builds 2 and 3 enable MUD-capable devices that have invalid MUD files or that have MUD file servers with invalid TLS certificates to connect to the network and communicate without being subject to any MUD-related traffic constraints. While this behavior is not erroneous, some users may be surprised to learn that a device that purports to be MUD-capable may not actually be subject to any of the rules in its MUD file in these situations.

There is merit in the argument that devices should be able to communicate unconstrained (rather than not being able to communicate at all) when their MUD file or MUD file server certificates are otherwise valid but have expired. However, it is more difficult to make the case that these devices should be able to be communicate unconstrained if their MUD file signature or MUD file server certificate has not expired but is invalid. It may be desirable, therefore, to consider implementing a default local policy that determines whether to fail open or fail closed depending on the reason that the MUD file signature or MUD file server certificate cannot be validated. Alternatively, an implementer may want to take advantage of unique product features in its response to error situations such as these and consider classifying devices as being in a specific category (in the case of Build 2) or placing devices in a specific micronet (in the case of Build 3) that results in the devices being subjected to appropriate communication constraints. An implementer utilizing Easy Connect onboarding could even prevent a wireless device from being provisioned with network credentials if the MUD manager were not able to validate the device's MUD file.

- The MUD specification (RFC 8520) states that the mud-signature element in the MUD file is optional, but it does not specify what the behavior of the MUD manager should be in the event that the mud-signature element is not present in a MUD file. MUD manager implementers should give careful thought to the behavior that their MUD manager implementations enforce by default. They should make this behavior clear so that users who are deploying MUD on their networks understand whether their MUD manager will automatically process a MUD file that does not have a mud-signature element or whether it will cease processing such a MUD file and wait for administrator input. MUD manager implementers should also make it possible for users to configure this MUD manager behavior as needed by local policy. A MUD manager that automatically processes MUD files that do not include a mud-signature element is vulnerable to accepting and processing as valid MUD files that have been modified by attackers if those attackers have deleted the mud-signature element from the MUD file.

- There is still a dearth of MUD-capable IoT devices. Users wanting to deploy MUD do not yet have the option to do so because of a lack of availability of MUD-capable IoT devices. More vendor buy-in is required to encourage IoT device manufacturers to implement support for MUD in their devices.
- To encourage further adoption of MUD, early adopters should tell their organizational story of change: who in the organization is responsible for understanding what goes into the MUD file, building the MUD file, making the MUD file available on a server, modifying the device to emit a URL, testing MUD-related features, and determining if a MUD file needs to be updated, among other functions.
- Communications between the MUD manager and the router/switch, between the threat-signaling server and the MUD manager/router, and between the IoT devices and their corresponding update servers are not standardized. This lack of standardization has the potential to inhibit interoperability of components that are obtained from different manufacturers, thereby limiting the choice that consumers have to mix architectural components from different vendors in their MUD deployments.
- RFC 8520 states clearly that if the cache-validity timer has not expired, the MUD manager must not check for a new MUD file and should use the cached file instead. It also clearly states that expiration of the cache-validity timer does not require the MUD manager to discard the MUD file. It does not, however, state that if the cache-validity timer has expired, the MUD manager should check for a new MUD file, even though this is the behavior that the RFC authors had intended to specify. It is our understanding that this will be submitted as an erratum for clarification. In the meantime, implementations wishing to conform to the desired behavior should be designed such that if the cache-validity timer has expired, the MUD manager checks for a new MUD file.
- MUD rules are defined in terms of domain names, but when MUD rules are instantiated on routers, IP addresses, rather than domain names, are used. However, the IP address to which any given domain resolves may change. So, if a domain is listed in a MUD file rule and device traffic filters that instantiate this MUD file rule have been installed on the router, when the domain begins resolving to a different address, the device will initially not behave as intended. If the device attempts to communicate with this new IP address, it will not be permitted to do so because there will not yet be device traffic filters in its router that permit it to access this new IP address. The device traffic filters in the router will still be permitting access to the old IP

address. In other words, the device will not be permitted to communicate with the desired domain, despite this communication being permitted by the device's MUD file. This undesirable situation will persist until the device traffic filters in the router are updated to use the new IP address to which the domain now resolves.

To minimize the effect of such a situation, the MUD implementation (e.g., the MUD manager) should periodically generate DNS resolution requests for each of the domains listed in the MUD file and, if any of these domains now resolve to different IP addresses than previously, the device traffic filters using the old IP address should be deleted from the router or switch, and the device traffic filters using the new IP address should be installed. Regarding how often a MUD implementation might want to perform this periodic checking of domain name resolution values, one suggestion is to do so at intervals of TTL+V, where TTL is the time to live value in the A record of the domain's DNS entry, and V might be as long as 86,400 seconds (i.e., 24 hours). (The TTL value specifies how long a resolver is supposed to cache the DNS query before the query expires and the domain should be resolved again. If a DNS record for a domain changes, a new lookup will not be done until the cache expires.) Users should be cautioned that if the IP address to which a domain name resolves changes, the IoT device may be prohibited from communicating with that domain for some period (i.e., V) after the TTL for the domain's DNS entry has expired.

- When a MUD-capable IoT device performs a domain name lookup, it is important that the IP addresses to which the domain name gets resolved match the IP addresses that that domain name got resolved to when the MUD rule containing that domain was installed at the router or switch. If they do not match, then the device could be prohibited from communicating with the desired domain despite the existence of a MUD rule explicitly permitting the device to do so.

  If the router or switch itself does a domain name lookup when the MUD rule is installed on it, and if the device and the router or switch are co-located, then the device and the router or switch will be in the same region and would be expected to have their domain name lookups resolved to the same IP addresses. Therefore, if the router or switch itself performs the domain name lookup when translating a MUD rule to device traffic filters, the IP address(es) that are returned to the IoT device when it performs a domain name lookup should be the same as the IP address(es) that were configured in the device traffic filters.

  However, if some other component, such as a MUD manager or controller that is in the cloud, performs a domain name lookup and sends the resulting device traffic filters to the router or switch for installation, then it is possible that the controller/MUD manager and the router or switch could be in a different region, which could mean that their domain name lookups for a given domain do not resolve to the same IP addresses. For MUD rules to be enforced as expected, measures need to be taken to ensure that the IP addresses that are used in the device traffic filters match the IP addresses that the IoT device would in fact use. Some possible ways of ensuring address alignment include:

  - requiring that the IoT device and the entity that is instantiating the MUD rules as device traffic filters use the same DNS server

    - having the entity that is instantiating the MUD rules as device traffic filters eavesdrop on the DNS queries made by the IoT device so it can learn what IP addresses the IoT device receives back in the DNS responses
    - having the router or switch occasionally send DNS queries for the list of domains it used in MUD files and updating the device traffic filters based on those queries

- In working with project collaborators, the NCCoE determined that MUD is only one of several foundational elements that are important to IoT security. First and foremost, it is imperative that IoT device manufacturers follow best practices for security when designing, building, and supporting their devices. Manufacturers should, for example, understand and manage the security and privacy risks posed by their devices as discussed in NISTIR 8228, *Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks,* as well as the more general guidelines for identifying, assessing, and managing security risks that are discussed in the *Framework for Improving Critical Infrastructure Cybersecurity* (Cybersecurity Framework). In addition, they should continue to support their devices throughout their full life cycle, from initial availability through eventual decommissioning, with regular patches and updates. Cisco has proposed the following four elements as necessary for IoT security:
    - device security by design: certifiable device capabilities
    - device intent: MUD
    - device network onboarding: secure, scalable, automated—bootstrapping remote secure key infrastructure/autonomic networking integrated model approach
    - life-cycle management: behavior, software patches/updates

  All four builds in this project support the second security element listed above (device intent: MUD). Build 3 also supports the third security element (secure, scalable, and automated onboarding of devices to the network) through use of the Wi-Fi Easy Connect protocol.

- When devices are onboarded using the Wi-Fi Easy Connect R1 protocol (as in Build 3), network security is enhanced because:
    - Each IoT device is assigned unique network credentials, which ensures that
        - even if the credentials of one device are known, these credentials cannot be presented by other devices (e.g., devices that are not authorized to connect to the network) to gain access to the network.
        - Credentials of some devices may be revoked or changed without interfering with the ability of other devices to connect to the network.
    - Network credentials are provisioned to each device via an automated protocol, thereby minimizing the opportunity for human error and exposure.
    - Network credentials are provisioned to each device over a secure channel, minimizing the possibility of their disclosure because

- The credentials are never displayed to the user, so presentation of the device's network credentials to the network does not pose any risk that the credentials will be viewed and thereby disclosed.
- No human being has an opportunity to be privy to the credentials of any device.

- While the Wi-Fi Easy Connect protocol onboarding performed in Build 3 (R1) is largely automated, it does require an individual to perform the manual operations of putting the IoT device into onboarding mode (assuming the device does not come out of the box ready to onboard) and scanning the device's QR code. Use of the Wi-Fi Easy Connect protocol relies on trust that the individual who scans the QR code will select the correct network to which to onboard the device. An individual who onboards a device to a network other than the device's intended network effectively executes a takeover attack on that IoT device, thereby preventing the device's intended network from taking control of the device. Such a takeover attack could be executed, in theory, by a rogue individual by:
    - positioning an alternative network within Wi-Fi range of the device
    - obtaining access to the device's QR code
    - putting the device into onboarding mode (or waiting until someone else puts the device into onboarding mode) and onboarding the device to the alternative network before the device is onboarded to its intended network

  By onboarding a device to a network other than its intended network, the owner of the alternative network can take control of the device, thereby denying the owner of the device the ability to use it on its intended network. Even more maliciously, such an attack could allow the owner of the alternative network to access and tamper with the device before eventually allowing it to be onboarded to the intended network, thus enabling a compromised device to be onboarded to the intended network.

- There are numerous ways in which support for MUD can be provided within a home/small-business network. Build 3 demonstrates support for MUD in residential gateway equipment and service provider infrastructure. However, this does not imply any requirement that service providers bear the responsibility for implementing MUD. Builds 1, 2, and 4 simply require that customers acquire and use third-party routers and other related components that are MUD-capable. Integrating MUD capability into residential gateway equipment supplied by service providers, along with strong advocacy and education of customers to explain the benefits of using MUD, represents one approach to encouraging widespread adoption of MUD in home and small-business environments. Factors affecting determination of how and where MUD should be supported include infrastructure and support requirements, cost, and privacy. These are some issues that should be considered:
    - Upgrading all existing internet gateways to be MUD-capable would be a large undertaking, so service providers might perform cost-benefit analyses to determine whether it makes economic sense for them to provide and support MUD-capable internet gateways in homes and small businesses.

- Providing and supporting MUD-capable internet gateways could potentially cast service providers into a situation in which they might be perceived as responsible for troubleshooting problems with the IoT devices themselves. This is a function that is generally outside the service provider's control.
- In addition to upgrading internet gateways to be MUD capable, service providers might choose to make changes to other aspects of the service provider network to support MUD. A service provider's analysis regarding whether it should integrate support for MUD into the residential gateway or simply encourage its customers to use MUD-capable third-party routers should consider any additional network changes that may be needed.
- The MUD manager, by its very nature, is aware of all MUD-capable IoT devices that are attached to the network and of what domains and other types of local devices they are permitted to communicate with. Such information could have privacy ramifications. Whatever organizational controls the MUD manager will have access to this information. If this organization is a service provider, as in the Build 3 implementation, the service provider will be privy to this personal information.

10.2 Security Considerations

Use of MUD, when implemented correctly, allows manufacturers to constrain communications to and from IoT devices to only those sources and destinations intended by the device's manufacturer. By restricting an IoT device's communications to only those that it needs to fulfill its intended function, MUD reduces both the communication vectors that can be used to attack a vulnerable IoT device and the communication vectors that a compromised IoT device can use to attack other devices. MUD does not, however, provide any inherent security protections to IoT devices themselves. If a device's MUD file permits an IoT device to receive communications from a malicious domain, traffic from that domain can be used to attack the IoT device. Similarly, if the MUD file permits an IoT device to send communications to other domains, and if the IoT device is compromised, it can be used to attack those other domains. Users deploying MUD are advised to keep the following security considerations in mind.

- It is important to ensure that the MUD implementation itself is secure and not vulnerable to attack. If the MUD implementation itself were to be compromised, the compromised MUD infrastructure would serve as a venue for attack. As stated in the Security Considerations section of the MUD specification (RFC 8520), "The basic purpose of MUD is to configure access, so by its very nature, it can be disruptive if used by unauthorized parties." Protecting the MUD infrastructure includes ensuring the integrity and security of the MUD file location (e.g., the IoT device MUD URL emission), the MUD manager, the DHCP server (when used for MUD URL emission), the MUD file server, the router, and the private key used to sign the MUD file. If the MUD implementation itself is compromised—e.g., if an IoT device emits an incorrect MUD file URL; if a different MUD file URL is sent to the MUD manager than that provided by the IoT device; if a well-formed, signed MUD file is malicious; if a malicious actor creates a compromised MUD manager; or if a router is compromised so that it does not enforce its device traffic filters—then MUD can be used to enable rather than prevent potentially damaging communications between affected IoT devices and other domains.

- If a malicious actor can create a well-formed, signed, malicious MUD file, the undesirable communications that will be permitted by that MUD file will be readily visible by reading the MUD file. Therefore, for added protection, users implementing MUD should review the MUD files for their IoT devices to ensure that they specify communications that are appropriate for each device. Unfortunately, on home and small-business networks, where users are not likely to have the technical expertise to understand how to read MUD files, users will be required to trust that the MUD files specify communications appropriate for the device or to rely on a third party to perform this review for them.
- MUD implementation depends on the existence and secure operation of a MUD file server from which a device's MUD file can be retrieved. If the manufacturer goes out of business or does not conform to best common practices for patching, the MUD file server domain would be vulnerable to having malware deployed on it and thereby being transformed into an attack vector. To safeguard against such a scenario, a mechanism needs to be defined to enable the domain of the manufacturer to be invalidated so that the MUD manager can be protected from connecting to the compromised MUD file server, despite the fact that IoT devices may continue to emit the URL of the compromised domain. Use of threat-signaling information is one example of such a mechanism.
- To protect all IoT devices on a network, both MUD-capable and non-MUD-capable, users may want to consider investigating mechanisms for supplying MUD files for legacy (non-MUD-capable) devices.
- By emitting or otherwise conveying a MUD URL, a device reveals information about itself, thereby potentially providing an attacker with guidance on what vulnerabilities it might have and how it might be attacked.
- An attacker could spy on the MUD manager to determine what devices are connected to the network and then use this information to plan an attack.
- If an attacker can gain access to the local network, they may be able to use the MUD manager in a reflected denial of service attack by emitting a large amount of MUD URLs (e.g., from spoofed MAC addresses) and forcing the MUD manager to make connection attempts to retrieve files from those MUD URLs. Safeguards to counter this, such as throttling connection attempts of the MUD manager, should be considered.
- MUD users should understand that the main benefit of MUD is its ability to limit an IoT device's communication profile; it does not necessarily permit owners to find, identify, and correct already-compromised IoT devices.
    - If a system is compromised but it is still emitting the correct MUD URL, MUD can detect and stop any unauthorized communications that the device attempts. Such attempts may also indicate potential compromises.

    - On the other hand, a system could be compromised so that it emits a new URL referencing a MUD file that a malicious actor has created to allow the compromised device to engage in communications that should be prohibited. In this case, whether the compromised system will be detected depends on how the MUD manager is configured to react to such a change in MUD URL. According to the MUD specification, if a MUD manager determines that an IoT device is sending a different MUD URL, the MUD manager should not use this new URL without some additional validation, such as a review by a network administrator.
        - If the MUD manager requires an administrator to accept the new URL but the administrator does not accept it, MUD would help owners detect the compromised system and limit the ability of the compromised system to be used in an attack.
        - However, if the MUD manager does not require an administrator to accept the new URL or if it requires an administrator to accept the new URL and the administrator does accept the new URL, MUD would not help owners detect the compromised system, nor would it limit the ability of the compromised system to be used in an attack.
        - As a third possibility, a compromised system could be subjected to a more sophisticated attack that enables it to dynamically change its identity (e.g., its MAC address) along with emitting a new URL. In this case, the compromised system would not be detected unless the MUD manager were configured to require the administrator to explicitly add each new identity to the network.
- The following security considerations are specific to the MUD deployment and configuration process:
    - When an IoT device emits its MUD URL by using DHCP or LLDP rather than using an X.509 certificate that can provide strong authentication of the device or by using some other mechanism that provides a trusted association between the MUD URL and the device, the device may be able to lie about its identity and thereby gain network access it should not have. If a network includes IoT devices that emit their MUD URL by using one of these insecure mechanisms, as do some of the builds implemented in this project, network administrators should take additional precautions to try to improve security. For example, the MUD implementation should be configured to:
        - prevent devices that have not been authenticated from being in the same class as devices that have been strongly authenticated to prevent the non-authenticated devices from getting possibly elevated permissions that are granted to the authenticated devices
        - prevent devices that have not been authenticated from being able to use the same MUD URL as devices that have been strongly authenticated
        - whenever possible, bind communications to the authentication that has been used, e.g., IEEE 802.1X, 802.1AE (MACsec), 802.11i (WPA2), WFA Easy Connect, or future authentication types

    - remove state if an unauthenticated method of MUD URL emission is being used and any form of break in that session is detected
    - not include unauthenticated devices into the manufacturer grouping of any specific manufacturer without additional validation
    - use additional discovery and classification components that may be on the network to try to fingerprint devices that have not been authenticated to try to verify that they are of the type they are asserting to be by their MUD URLs
    - raise an alert and require administrator approval if the MUD manager detects that the signer of a MUD file has changed, to protect against rogue Certificate Authorities
    - raise an alert and require administrator approval if the MUD manager detects that a device's MUD file has changed, to protect compromised IoT devices that seek to be associated with malevolent MUD files
- To protect against domain name ownership changes that would permit a malicious actor to provide MUD files for a device, MUD managers should be configured to cache certificates used by the MUD file server. If a new certificate is retrieved, the MUD manager should check to see if ownership of the domain has changed and, if so, it should raise an alert and require administrator approval.

The points above provide only a summary of the security considerations discussed in the MUD specification (RFC 8520). Users deploying a MUD implementation are encouraged to consult that document directly for more detailed discussion.

Additionally, please refer to NISTIR 8228, *Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks,* for more details related to IoT cybersecurity and privacy considerations.

10.3 Recommendations

The following are recommendations for using MUD:

- Home and small-business network owners should make clear to vendors that both IoT devices and network components need to be MUD-capable. They should use MUD-capable IoT devices on their networks and enable MUD on their networks by deploying all of the MUD-capable network components needed to compose a MUD-capable infrastructure.
- Service providers should consider either providing and supporting or encouraging their customers to use MUD-capable routers on their home and small-business networks. (Note: MUD requires the use of a MUD-capable router; this router could be either standalone equipment provided by a third-party network equipment vendor or integrated with the service provider's residential gateway equipment. While service providers are not required to do so, some may choose to make their residential gateway equipment MUD-capable.)

- IoT device manufacturers should configure their devices to emit or otherwise convey a MUD URL.
- IoT device manufacturers should write MUD files for their devices. By doing so, they will be able to provide network administrators the confidence to know what sort of access their device needs (and what sort of access it does not need), and they will do so in a way that someone trained to operate and install the device does not need to understand network administration.
- IoT device manufacturers should ensure that the MUD files for their devices remain continuously available by hosting these MUD files at their specified MUD URLs throughout the devices' life cycles.
- IoT device manufacturers should update each of their MUD files over the course of their devices' life cycles, as needed, if the communication profiles for their devices evolve.
- Even after an IoT device manufacturer deprecates an IoT device so that it will no longer be supported, the manufacturer should continue to make the device's MUD file available so the device's communication profile can continue to be enforced. This will be especially important for deprecated IoT devices that have unpatched vulnerabilities.
- IoT device manufacturers should provide regular updates to patch security vulnerabilities and other bugs that are discovered throughout the life cycle of their devices, and they should make these updates available at a designated URL that is explicitly named in the device's MUD file as being a permissible endpoint with which the device may communicate.
- Manufacturers of MUD managers, MUD-capable DHCP servers, MUD-capable routers, device onboarding equipment, components for supporting threat signaling, components for supporting device discovery, and other networking equipment that is targeted for use on home and small-business networks should strive to make deployment and configuration of these devices as easy to understand and as user-friendly as possible to increase the probability that they will be deployed and configured correctly and securely, even when the person performing the deployment has limited understanding of network administration.
- Home and small-business network owners should use the information presented in the Security Considerations section of the MUD specification (RFC 8520) to enhance protection of MUD deployments.
- Standards development organizations should standardize communications between the MUD manager and the router, between the threat-signaling server and the MUD manager/router, and between the IoT devices and their corresponding update servers.

The following are recommendations for improving the security of home and small-business networks and IoT devices in general:

- Home and small-business network owners should deploy and use equipment and services that apply policies based on threat, thereby benefitting them with available information on known threats.

- Home and small-business network owners should perform periodic updates to all IoT devices to ensure that the devices will be protected with up-to-date software patches.
- IoT device manufacturers should provide ongoing support for the devices that they sell by making regular software updates and patches available on an ongoing basis.
- Home and small-business network owners should have visibility into every device on their network. Any device is a potential attack or reconnaissance point that must be discovered and secured. Non-MUD-capable devices are inviting targets.
- Home and small-business network owners should segment their networks where possible. Where there are IoT devices with known security risks, e.g., non-MUD-capable devices, these devices should be kept on a separate network segment from the everyday computing devices that are afforded a higher level of cybersecurity protection via regular updates and security software. This is an important step to contain any threats that may emerge from the IoT devices.
- Home and small-business network owners should deploy network components that are needed to support a secure, automated, and easy-to-use onboarding protocol, and they should use IoT devices that are capable of being onboarded via this protocol.
- Manufacturers of network equipment that is targeted for use on home and small-business networks should offer components that support secure, automated, and user-friendly IoT device onboarding, threat signaling, and device discovery.
- Service providers should either provide residential gateway equipment that supports secure, automated, and easy-to-use IoT device onboarding; threat signaling; and device discovery, or they should encourage their customers to use third-party equipment with these capabilities on their home and small-business networks.
- IoT device manufacturers should design their devices to be capable of being onboarded via a secure, automated, and easy-to-use process.
- Home and small-business network owners should consider their deployment of MUD to be only one pillar in the overall security of their network and IoT devices. Deployment of MUD is not a substitute for performing best practices to ensure overall, comprehensive security for their network.
- Manufacturers of MUD-capable network components and MUD-capable IoT devices should consider MUD to be only one pillar in helping users secure their networks and IoT devices. Manufacturers should, for example, understand the security and privacy risks posed by their devices as discussed in NISTIR 8228, *Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks,* as well as the guidelines for identifying, assessing, and managing security risks that are discussed in the *Framework for Improving Critical Infrastructure Cybersecurity* (Cybersecurity Framework). They should use this information as they make decisions regarding both how they design their MUD-capable components and the default configurations with which they provide these components, being mindful of the fact that home

and small-business network users of their components may have only a limited understanding of network administration and security.

The following recommendations are for the MUD RFC editors:

- Consider revising the MUD specification (RFC 8520) to be explicit about the fact that it is deliberately not specifying what the behavior of the MUD manager should be in the event that the mud-signature element is not present in a MUD file. As currently written, it is reasonable to interpret the RFC in several different ways. It could be interpreted as implying that if the mud-signature element is not present, then:
    - The MUD file has not been signed, so the MUD manager may process the MUD file without attempting to validate its signature. This interpretation is vulnerable to hackers modifying the MUD file and deleting the MUD file’s mud-signature element to prevent modification of the MUD file from being detected. Unless all MUD files are required to be signed and to have their signatures validated before processing, it will not be possible for a MUD manager to distinguish between a MUD file that has not been signed and a MUD file that was originally signed but has been modified by an attacker so that its mud-signature element has been deleted.
    - The MUD manager should cease processing the MUD file and wait for administrator input.
    - The MUD manager should attempt to locate and validate the MUD file’s signature via some alternative means. However, no such alternative means is mentioned in the RFC. RFC editors may want to consider including suggestions for potential alternative mechanisms for locating MUD file signatures if the mud-signature element (which has been defined as optional) is not present in the MUD file.
- Consider revising Section 16 (Security Considerations) of the MUD specification (RFC 8520) to make readers aware of the security vulnerability that results from using a MUD manager that is configured to automatically process a MUD file that does not have a mud-signature element.
- Consider revising the MUD specification (RFC 8520) to be explicit about the fact that it is deliberately not dictating what action to take (e.g., block or permit traffic to/from the IoT device) if the MUD manager is not able to validate the device’s MUD file server’s TLS certificate or if the MUD manager is not able to validate the device’s MUD file signature. The RFC indicates that the MUD manager should cease processing the MUD file and await administrator approval, but it may be helpful to readers if the RFC were explicit about the fact that it is remaining silent and leaving up to local policy whether the device should be prevented from sending and receiving all traffic (thereby rendering the devices unreachable and effectively useless), whether the device should be permitted to communicate freely (thereby enabling the device to operate as if it did not have an associated MUD file), or whether the device should be subject to some other local policy.
- Consider revising Section 3.5 (Cache-Validity) of the MUD specification (RFC 8520) to explicitly state that if the cache-validity timer has expired, the MUD manager should check for a new

MUD file. We understand that this is the desired behavior; however, it is not currently made clear in the specification.

The following recommendations are suggestions for continuing activity with the collaboration team:

- Continue work with collaborators to enhance MUD capabilities in their commercial products (see Section 10.1).
- Perform additional work that builds on the broader set of security controls identified in Section 5.2.
- Work with collaborators to demonstrate MUD deployments that are configured to address the security considerations that are raised in the MUD specification, such as
  - configuring IoT devices to emit their MUD URLs in a secure fashion by providing the IoT devices with credentials and binding the device's MUD URLs with their identities
  - restricting the access control permissions of IoT devices that do not emit their MUD URLs in a secure fashion, so they are not elevated beyond those of devices that do not present a MUD policy
  - configuring the MUD manager to raise an exception and seek administrator approval if the signer of a MUD file or the MUD file itself changes
  - for IoT devices that do not emit their MUD URLs in a secure fashion, if their MUD files include rules based on the "manufacturer" construct, performing additional validation measures before admitting the devices to that manufacturer class. For example, look up each device's MAC address and verify that the manufacturer associated with that MAC address is the same as the manufacturer specified in the "manufacturer" construct in that device's MUD file
  - incorporating MUD URL discovery and policy into the secure device onboarding process
- Explore the possibility of using crowdsourcing and analytics to perform traffic flow analysis and thereby adapt and evolve traffic profiles of MUD-capable devices over the course of their use. Instead of simply dropping traffic that is received at the router if that traffic is not within the IoT device's profile, this traffic could be quarantined, recorded, and analyzed for further study. An analytics application that receives such traffic from many sources would be able to analyze the traffic and determine whether there may be valid reasons to expand the device's communication profile.
- Work with collaborators to define a blueprint to guide IoT device manufacturers as they build MUD support into their devices, from initial device availability to eventual decommissioning. Provide guidance on required and recommended manufacturer activities and considerations.
- Execute performance studies to inform manufacturers of consumer routers how MUD impacts performance. Such studies may address concerns that some manufacturers may have regarding the potential performance impacts of MUD.

11 Future Build Considerations

The number of network components that support the MUD protocol continues to grow rapidly. As more MUD-capable IoT devices become available, these too should be demonstrated. In addition, IPv6, for which no MUD-capable products were available for the initial demonstration sequences, adds a new dimension to using MUD to help mitigate IoT-based DDoS and other network-based attacks. As discussed in Section 11.2, inclusion of IPv6-capability should be considered for future builds.

In addition, operationalization, IoT device onboarding, and IoT device life-cycle issues in general are promising areas for further work. With respect to onboarding, mechanisms for devices to securely provide their MUD URL (in addition to using the Wi-Fi Easy Connect protocol) can be investigated and developed as proof-of-concept implementations.

The following features, which are enhancements that are being implemented in Build 4, are potential candidates for inclusion in future IETF MUD drafts:

- The MUD manager implements device quarantine. A device may enter a "quarantine" state when a packet originating from the device triggers an access violation (i.e., does not match any MUD rules). When the device is in a quarantine state, its access is limited to only those ACEs that are allowable under quarantine.
- The MUD manager implements a MUD reporting capability for manufacturers to be able to get feedback on how their MUD-capable devices are doing in the field. To protect privacy, no identifying information about the device or network is included.

11.1 Extension to Demonstrate the Growing Set of Available Components

Arm, CableLabs, Cisco, CTIA, DigiCert, Forescout, Global Cyber Alliance, MasterPeace Solutions, Molex, Patton Electronics, and Symantec have signed CRADAs and are collaborating in the project. There is also strong interest from additional industry collaborators to participate in future builds, particularly if we expand the project scope to include onboarding. Some collaborators have also expressed interest in our demonstrating the enterprise use case. Several of these new potential collaborators may submit letters of interest leading to CRADAs for participation in tackling the challenge of integrating MUD and other security features into enterprise or industrial IoT use cases.

11.2 Recommended Demonstration of IPv6 Implementation

Due to product limitations, the initial phases of this project involved support for only IPv4 and did not include investigation of IPv6 issues. Additionally, due to the absence of NAT in IPv6, all IPv6 devices are directly addressable. Hence, the potential for DDoS and other attacks against IPv6 networks could potentially be worse than it is against IPv4 networks. Consequently, we recommend that demonstration of MUD in an IPv6 environment be performed as part of follow-on work.

Appendix A List of Acronyms

AAA	Authentication, Authorization, and Accounting
ACE	Access Control Entry
ACK	Acknowledgement
ACL	Access Control List
AP	Access Point
API	Application Programming Interface
CIS	Center for Internet Security
CMS	Cryptographic Message Syntax
COBIT	Control Objectives for Information and Related Technology
CRADA	Cooperative Research and Development Agreement
DACL	Dynamic Access Control List
DB	Database
DDoS	Distributed Denial of Service
Devkit	Development Kit
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DVR	Digital Video Recorder
FIPS	Federal Information Processing Standard
GCA	Global Cyber Alliance
GUI	Graphical User Interface
http	Hypertext Transfer Protocol
https	Hypertext Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IOS	Cisco's Internetwork Operating System
IoT	Internet of Things
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
ISA	International Society of Automation
ISO/IEC	International Organization for Standardization/International Electrotechnical Commission
ISP	Internet Service Provider
IT	Information Technology
ITL	National Institute of Standards and Technology's Information Technology Laboratory
JSON	JavaScript Object Notation
LED	Light-Emitting Diode
LLDP	Link Layer Discovery Protocol (Institute of Electrical and Electronics Engineers 802.1AB)

MAC	Media Access Control
MQTT	Message Queuing Telemetry Transport
MSO	Multiple System Operators
MUD	Manufacturer Usage Description
NAT	Network Address Translation
NCCoE	National Cybersecurity Center of Excellence
NIST	National Institute of Standards and Technology
NISTIR	NIST Interagency or Internal Report
NTP	Network Time Protocol
OS	Operating System
PEP	Policy Enforcement Point
PoE	Power over Ethernet
PSK	pre-shared key
QR	Quick Response
RADIUS	Remote Authentication Dial-In User Service
REST	Representational State Transfer
RFC	Request for Comments
RMF	Risk Management Framework
SDN	Software Defined Networking
SP	Special Publication
SSID	service set identifier
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
TLV	Type Length Value
UDP	User Datagram Protocol
UI	User Interface
URL	Uniform Resource Locator
VLAN	Virtual Local Area Network
WAN	Wide Area Network
YANG	Yet Another Next Generation

Appendix B Glossary

Audit	Independent review and examination of records and activities to assess the adequacy of system controls, to ensure compliance with established policies and operational procedures (National Institute of Standards and Technology (NIST) Special Publication (SP) 800-12 Rev. 1).
Best Practice	A procedure that has been shown by research and experience to produce optimal results and that is established or proposed as a standard suitable for widespread adoption (Merriam-Webster).
Botnet	The word “botnet” is formed from the words “robot” and ”network.” Cyber criminals use special Trojan viruses to breach the security of several users’ computers, take control of each computer, and organize all the infected machines into a network of “bots” that the criminal can remotely manage. (https://usa.kaspersky.com/resource-center/threats/botnet-attacks)
Control	A measure that is modifying risk (Note: Controls include any process, policy, device, practice, or other actions that modify risk.) (NIST Interagency or Internal Report [NISTIR] 8053).
Denial of Service	The prevention of authorized access to a system resource or the delaying of system operations and functions (NIST SP 800-82 Rev. 2)
Distributed Denial of Service (DDoS)	A denial of service technique that uses numerous hosts to perform the attack (NISTIR 7711).
Managed Devices	Personal computers, laptops, mobile devices, virtual machines, and infrastructure components require management agents, allowing information technology staff to discover, maintain, and control them. Those with broken or missing agents cannot be seen or managed by agent-based security products.
Mapping	Depiction of how data from one information source maps to data from another information source.
Mitigate	To make less severe or painful or to cause to become less harsh or hostile (Merriam-Webster).

Manufacturer Usage Description (MUD)	A component-based architecture specified in Request for Comments (RFC) 8250 that is designed to provide a means for end devices to signal to the network what sort of access and network functionality they require to properly function.
MUD-Capable	An Internet of Things (IoT) device that can emit a MUD uniform resource locator in compliance with the MUD specification.
Network Address Translation (NAT)	A function by which internet protocol addresses within a packet are replaced with different IP addresses. This function is most commonly performed by either **routers** or firewalls. It enables private IP networks that **use** unregistered IP addresses to connect to the internet. **NAT** operates on a router, usually connecting two networks together, and translates the private (not globally unique) addresses in the internal network into legal addresses before packets are forwarded to another network.
Non-MUD-Capable	An IoT device that is not capable of emitting a MUD URL in compliance with the MUD specification (RFC 8250).
Onboarding	The process by which a device obtains the credentials (e.g., network SSID and password) that it needs in order to gain access to a wired or wireless network.
Operationalization	Putting MUD implementations into operational service in a manner that is both practical and effective.
Policy	Statements, rules, or assertions that specify the correct or expected behavior of an entity. For example, an authorization policy might specify the correct access control rules for a software component. (NIST SP 800-95 and NISTIR 7621 Rev. 1)
Policy Enforcement Point	A network device on which policy decisions are carried out or enforced.
Risk	The net negative impact of the exercise of a vulnerability, considering both the probability and the impact of occurrence. Risk management is the process of identifying risk, assessing risk, and taking steps to reduce risk to an acceptable level. (NIST SP 800-30)
Router	A computer that is a gateway between two networks at open system interconnection layer 3 and that relays and directs data packets through that

	internetwork. The most common form of router operates on IP packets (NIST SP 800-82 Rev. 2).
Server	A computer or device on a network that manages network resources. Examples include file servers (to store files), print servers (to manage one or more printers), network servers (to manage network traffic), and database servers (to process database queries). (NIST SP 800-47)
Security Control	A safeguard or countermeasure prescribed for an information system or an organization designed to protect the confidentiality, integrity, and availability of its information and to meet a set of defined security requirements (NIST SP 800-53 Rev. 4).
Shall	A requirement that must be met unless a justification of why it cannot be met is given and accepted (NISTIR 5153).
Should	This term is used to indicate an important recommendation. Ignoring the recommendation could result in undesirable results. (NIST SP 800-108)
Threat	Any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service. Also, the potential for a threat-source to successfully exploit a particular information system vulnerability (Federal Information Processing Standards 200).
Threat Signaling	Real-time signaling of DDoS-related telemetry and threat-handling requests and data between elements concerned with DDoS attack detection, classification, trace back, and mitigation (https://joinup.ec.europa.eu/collection/rolling-plan-ict-standardisation/cybersecurity-network-and-information-security).
Traffic Filter	An entry in an access control list that is installed on the router or switch to enforce access controls on the network.
Uniform Resource Locator (URL)	A reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it. A typical URL could have the form http://www.example.com/index.html, which indicates a protocol (http), a host name (www.example.com), and a file name (index.html). Also sometimes referred to as a web address.

Update New, improved, or fixed software, which replaces older versions of the same software. For example, updating an operating system brings it up-to-date with the latest drivers, system utilities, and security software. The software publisher often provides updates free of charge. (https://www.computerhope.com/jargon/u/update.htm)

Update Server A server that provides patches and other software updates to IoT devices.

VLAN A broadcast domain that is partitioned and isolated within a network at the data link layer. A single physical local area network (LAN) can be logically partitioned into multiple, independent VLANs; a group of devices on one or more physical LANs can be configured to communicate within the same VLAN, as if they were attached to the same physical LAN.

Vulnerability Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source (NIST SP 800-37 Rev. 2).

Appendix C Bibliography

FIDO Alliance. Specifications Overview [Website]. Available: https://fidoalliance.org/specifications/overview/.

Internet-Draft draft-srich-opsawg-mud-manu-lifecycle-01. (2017, Mar.) “MUD Lifecyle: A Manufacturer's Perspective” [Online]. Available: https://tools.ietf.org/html/draft-srich-opsawg-mud-manu-lifecycle-01.

Internet-Draft draft-srich-opsawg-mud-net-lifecycle-01. (2017, Sept.) “MUD Lifecyle: A Network Operator’s Perspective” [Online]. Available: https://tools.ietf.org/html/draft-srich-opsawg-mud-net-lifecycle-01.

Internet Policy Task Force, National Telecommunications Information Administration. Multistakeholder Working Group for Secure Update of IoT Devices [Website]. Available: https://www.ntia.doc.gov/category/internet-things.

National Institute of Standards and Technology (NIST). (2018, Apr.) Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1 [Online]. Available: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf.

National Institute of Standards and Technology (NIST) Draft Interagency or Internal Report 7823. (2012, Jul.) Advanced Metering Infrastructure Smart Meter Upgradeability Test Framework [Online]. Available: http://csrc.nist.gov/publications/drafts/nistir-7823/draft_nistir-7823.pdf.

National Institute of Standards and Technology (NIST) Interagency or Internal Report 8228. (2018, Sept.) Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks [Online]. Available: https://doi.org/10.6028/NIST.IR.8228.

National Institute of Standards and Technology (NIST). NIST Computer Security Resource Center Risk Management Framework guidance [Website]. Available: https://csrc.nist.gov/projects/risk-management/risk-management-framework-quick-start-guides.

National Institute of Standards and Technology (NIST) Special Publication (SP) 800-30. (2002, Jul.) Risk Management Guide for Information Technology Systems [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-30r1.pdf.

National Institute of Standards and Technology (NIST) Special Publication (SP) 800-30 Revision 1. (2012, Sept.) Guide for Conducting Risk Assessments [Online]. Available: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-30r1.pdf.

National Institute of Standards and Technology (NIST) Special Publication (SP) 800-37 Revision 2. (2018, Dec.) Risk Management Framework for Information Systems and Organizations [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-37r2.pdf.

National Institute of Standards and Technology (NIST) Special Publication (SP) 800-40 Rev. 3. (2013, Jul.) Guide to Enterprise Patch Management Technologies [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-40/rev-3/final.

National Institute of Standards and Technology (NIST) Special Publication (SP) 800-52 Revision 2. (2019, Aug.) Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations [Online]. Available: https://doi.org/10.6028/NIST.SP.800-52r2.

National Institute of Standards and Technology (NIST) Draft Special Publication (SP) 800-53 Rev. 5. (2017, Aug.) Security and Privacy Controls for Information Systems and Organizations (Draft) [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/draft.

National Institute of Standards and Technology (NIST) Special Publication (SP) 800-57 Part 1 Revision 4. (2016, Jan.) Recommendation for Key Management [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf.

National Institute of Standards and Technology (NIST) Special Publication (SP) 800-63-3. (2017, Jun.) Digital Identity Guidelines [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-63/3/final.

National Institute of Standards and Technology (NIST) Special Publication (SP) 800-63-B. (2017, Jun.) Digital Identity Guidelines: Authentication and Lifecycle Management [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-63b/final.

National Institute of Standards and Technology (NIST) Special Publication (SP) 800-147. (2011, Apr.) BIOS Protection Guidelines [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-147/final.

National Institute of Standards and Technology (NIST) Special Publication (SP) 800-147B. (2014, Aug.) BIOS Protection Guidelines for Servers [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-147B.pdf.

National Institute of Standards and Technology (NIST) Special Publication (SP) 800-193. (2018, May.) Platform Firmware Resiliency Guidelines [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-193.pdf.

Office of Management and Budget (OMB) Circular A-130 Revised. (2016, Jul.) Managing Information as a Strategic Resource [Online]. Available: https://obamawhitehouse.archives.gov/omb/circulars_a130_a130trans4/.

Request for Comments (RFC) 2131. (1997, Mar.) “Dynamic Host Configuration Protocol” [Online]. Available: https://tools.ietf.org/html/rfc2131.

Request for Comments (RFC) 2818. (2000, May.) “HTTP Over TLS” [Online]. Available: https://tools.ietf.org/html/rfc2818.

Request for Comments (RFC) 5280. (2008, May.) “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile” [Online]. Available: https://tools.ietf.org/html/rfc5280.

Request for Comments (RFC) 5652. (2009, Sept.) “Cryptographic Message Syntax (CMS)” [Online]. Available: https://tools.ietf.org/html/rfc5652.

Request for Comments (RFC) 6020. (2010, Oct.) “YANG—A Data Modeling Language for the Network Configuration Protocol (NETCONF)” [Online]. Available: https://tools.ietf.org/html/rfc6020.

Request for Comments (RFC) 8520. (2019, Mar.). “Manufacturer Usage Description Specification” [Online]. Available: https://tools.ietf.org/html/rfc8520.

SANS Institute. CWE/SANS Top 25 Most Dangerous Software Errors [Website]. Available: https://www.sans.org/top25-software-errors/.

Wi-Fi Alliance. DRAFT Device Provisioning Protocol Specification Version 1.2, 2020. Available: https://www.wi-fi.org/file/device-provisioning-protocol-draft-specification.