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

Tim Polk
Murugiah Souppaya
NIST

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

Jaideep Singh
Molex

William C. Barker
Dakota Consulting

Dean Coclin
Clint Wilson
DigiCert

Darshak Thakore
Mark Walker
CableLabs

Eliot Lear
Brian Weis
Cisco

Tim Jones
ForeScout

Drew Cohen
MasterPeace

April 2019

PRELIMINARY DRAFT

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

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

NATIONAL CYBERSECURITY CENTER OF EXCELLENCE

DISCLAIMER

National Institute of Standards and Technology Special Publication 1800-15B, Natl. Inst. Stand. Technol. Spec. Publ. 1800-15B, 196 pages, (April 2019), 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: April 24, 2019 through June 24, 2019

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, Maryland 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) architecture is for Internet of Things (IoT) devices to behave as intended by the manufacturer of the devices. This is done by providing a standard way for manufacturers to identify each device's type and to indicate the network communications that it requires to perform its intended function. 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 it will prohibit all other communications with the device.

The NCCoE has demonstrated for IoT product developers and implementers the ability to ensure that when an IoT device connects to a home or small-business network, MUD can be used to automatically permit the device to send and receive only the traffic it requires to perform its intended function.

A distributed denial of service (DDoS) attack can cause significant negative impact to an organization that is dependent on the internet to conduct business. 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. Recently, IoT devices have been exploited to launch DDoS attacks. IoT devices may have unpatched or easily discoverable software flaws, and many have minimal security, are unprotected, or are difficult to secure. A DDoS 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

- communications service providers who may suffer service degradation that affects their customers
- 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 being exploited
- users of IoT devices who may suffer service degradation and potentially incur extra costs due to increased activity by their captured machines

Use of MUD combats these IoT-based DDoS attacks by prohibiting unauthorized traffic to and from IoT devices. 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. MUD provides a standard method for access control information to be available to network control devices. This NIST Cybersecurity Practice Guide shows IoT product and system providers how to integrate and use MUD to help make home and small-business networks more secure. It also shows what users should expect from IoT device manufacturers.

KEYWORDS

botnets; internet of things; IoT; manufacturer usage description; MUD; router; server; software update server; threat signaling.

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.

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
Adnan Baykal	Global Cyber Alliance
Nate Lesser	MasterPeace Solutions
Tom Martz	MasterPeace Solutions
Daniel Weller	MasterPeace Solutions
Kevin Yeich	MasterPeace Solutions
Mo Alhroub	Molex
Bill Haag	NIST
Bryan Dubois	Patton Electronics

Name	Organization
Karen Scarfone	Scarfone Cybersecurity
Matt Boucher	Symantec
Bruce McCorkendale	Symantec
Yun Shen	Symantec
Pierre-Antoine Vervier	Symantec
Sarah Kinling	The MITRE Corporation
Mary Raguso	The MITRE Corporation
Allen Tan	The MITRE Corporation
Paul Watrobski	University of Maryland
Russ Housley	Vigilsec

The Technology Partners/Collaborators who participated in this build 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 this example solution. We worked with:

Technology Partner/Collaborator	Build Involvement
Arm	Subject Matter Expertise
CableLabs	Micronets Gateway Service Provider Server Partner and Service Provider Server Prototype Medical Devices–Raspberry Pi

Technology Partner/Collaborator	Build Involvement
Cisco	Cisco Catalyst 3850S MUD Manager
CTIA	Subject Matter Expertise
DigiCert	Private Transport Layer Security (TLS) Certificate Premium Certificate
ForeScout	CounterACT Appliance–VCT-R Enterprise Manager–VCEM-05
Global Cyber Alliance	Subject Matter Expertise
MasterPeace Solutions	Yikes! Router Yikes! Cloud Yikes! Mobile Application
Molex	Molex light emitting diode (LED) Light Bar Molex power over ethernet (PoE) Gateway
Patton Electronics	Session Border Controller—SN5301/4B/EUI
Symantec	Subject Matter Expertise

Contents

List of Figures

List of Tables

1 Summary

The Manufacturer Usage Description (MUD) Specification (Request for Comments [RFC] 8520) provides a means for IoT devices to signal to networks what sort of access and network functionality they require to properly function. The objective of this project is to show how IoT product and system manufacturers can use MUD to reduce the vulnerability of Internet of Things (IoT) devices to botnets and other automated distributed threats while limiting the utility of any compromised IoT devices to malicious actors. This volume describes the approach adopted for the project, the laboratory architecture demonstrated by the project, and the security characteristics demonstrated in the laboratory environment. The primary technical elements of this project include MUD-capable network gateways/routers supporting wired and wireless network access, MUD managers, MUD file servers, MUD-capable Dynamic Host Configuration Protocol (DHCP) servers, update servers, and threat signaling servers. We used personal computing devices, business computing devices, and both MUD-capable and non-MUD-capable IoT devices to demonstrate the security benefits provided by MUD. MUD will not provide perfect security, but it will significantly increase the effort required by malicious actors to compromise and exploit IoT devices on a home or small-business network. The scenarios examined by this National Cybersecurity Center of Excellence (NCCoE) project involve IoT devices being onboarded and used on home and small-business networks, where plug-and-play deployment is required. The example solution network includes MUD-capable IoT devices that interact with external systems to access secure updates and various cloud services, in addition to interacting with traditional personal computing devices, as permitted by their MUD files. The IoT devices used include smart lighting controllers, cameras, smartphones, printers, baby monitors, digital video recorders, and smart assistants.

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 smart television sets. The IoT is experiencing what some might describe as hypergrowth. Gartner predicts there will be 20.4 billion connected IoT devices by 2020 compared with 8.4 billion in 2017, while Forbes forecasts the market to be $457 billion by 2020 (a 28.5 percent compounded annual growth rate). As connected devices become more commonplace in homes and businesses, security concerns are also increasing. Many full-featured devices such as web servers, personal or business computers, and mobile devices often have state-of-the-art security software protecting them from most known threats. Conversely, many IoT devices are challenging to secure because they are designed to be inexpensive and to perform a single function—resulting in processing, timing, memory, and power constraints. Nevertheless, the consequences of not addressing security concerns of connected devices can be catastrophic. For instance, in typical networking environments, malicious actors can detect an IoT device within minutes of it being connected and then launch an attack on that same device from any system on the internet, unbeknownst to the user. They can also commandeer a group of compromised devices, called a botnet, to launch large-scale attacks.

1.2 Solution

This project demonstrates an approach to significantly strengthen security while deploying IoT devices in home and small-business networks. This approach can help bolster the resiliency of IoT devices and prevent them from being used as platforms from which to mount DDoS attacks across the internet.

The NCCoE sought existing technologies that use the MUD Specification (Request for Comments [RFC] 8520) to permit an IoT device to signal to the network what sort of access and network functionality it requires to properly operate. Constraining the communication abilities of exploited IoT devices reduces the potential for the devices to be used in attacks—both DDoS attacks that could be launched across the internet and attacks on the IoT device's local network that could have security consequences. This practice guide explains how to effectively implement the MUD specification for MUD-capable IoT devices and envisions methods for preventing non-MUD-capable IoT devices from connecting to potentially malicious entities that use threat signaling technology.

1.3 Benefits

This project provides benefits to several different types of stakeholders:

- Communications service providers will benefit from reduction of the set of IoT devices that can be easily used by "bad actors" to participate in DDoS attacks against their networks and thereby degrade service for their customers.
- Organizations and others who use the internet, including businesses that rely on their customers being able to reach them over the internet, as well as critical infrastructures and other public and private sector institutions, will benefit from improved confidence in internet availability and performance due to reductions in network-based attacks.
- IoT device manufacturers will benefit by avoiding reputational damage that they might suffer if their devices could be easily exploited to conduct DDoS attacks.
- Users of IoT devices, including small businesses and homeowners, will benefit from improved understanding of how to find and use the set of tools available to protect their internal networks from being subverted by bad actors and of how to reduce the threats to their businesses 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 National Institute of Standards and Technology (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 IoT-based DDoS threats. This reference design is modular and can be deployed in whole or in parts.

This guide contains three volumes:

- NIST Special Publication (SP) 1800-15A: *Executive Summary*
- NIST SP 1800-15B: *Approach, Architecture, and Security Characteristics*–what we built and why **(you are here)**
- NIST SP 1800-15C: *How-To Guides*–instructions for building the example solution

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:

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

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 example solution to cybersecurity standards and best practices.

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

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 build created in our lab. The how-to guide provides specific product installation, configuration, and integration instructions for implementing the example solution. 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 an example solution.

This guide assumes that information technology (IT) professionals have experience implementing security products within the enterprise. While we have used a suite of commercial products to address this challenge, this guide does not endorse these particular products. Your organization can adopt this solution 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 4.3, Technologies, lists the products we used, and Section 5.2 maps them 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 pathnames; references to documents that are not hyperlinks; new terms; and placeholders	For detailed definitions of terms, see the *NCCoE Glossary.*
Bold	names of menus, options, command buttons, and fields	Choose **File > Edit.**
`Monospace`	command-line input, onscreen computer output, sample code examples, status codes	`Mkdir`
`Monospace Bold`	command-line user input contrasted with computer output	**`service sshd start`**

Typeface/ Symbol	Meaning	Example
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 invited technology providers to participate in demonstrating a proposed approach for deployment of consumer and commercial IoT devices in home and small-business networks in a manner that provides significantly higher security than is typically achieved in today's environments. In this project, current and emerging network standards are applied to home and business networks that are composed of both IoT and fully featured devices (e.g., personal computers and mobile devices) to constrain communications-based malware exploits. Network gateway components and security-aware IoT devices leverage the MUD Specification (RFC 8520) to permit a MUD-capable IoT device to signal to the network what sort of access and network functionality it requires to properly operate. The resulting access control capability reduces the potential for exploited MUD-capable IoT devices to be used in a DDoS attack by constraining their communication abilities. In addition, network components could in the future implement 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). Automatic secure update controls are implemented on all devices used in this project, and they support secure administrative access. (Note that software update formats for IoT devices are not currently standardized. NCCoE experiences with software update strategies will be contributed to emerging standardization activities.)

The NCCoE prepared a *Federal Register* Notice seeking technology providers to provide products and/or expertise to compose prototypes that include MUD-capable routers or switches; MUD managers; MUD file servers; MUD-capable DHCP servers; IoT devices capable of both inserting the MUD uniform resource locator (URL) into DHCP address requests and requesting, verifying, and applying software updates; update servers; and threat signaling servers. Cooperative Research and Development Agreements (CRADAs) were established with qualified respondents, and build teams were assembled. The build teams fleshed out the initial architecture, and the collaborators' components were composed into example implementations. The build team documented the architecture and design implementation. As the build progressed, the team documented the steps taken to install and configure each component of the demonstration environment. The team then conducted functional testing of the demonstration environment, including demonstrating software update processes and responses to attempts to perform prohibited communications. The team conducted and documented the results of a risk assessment and a security characteristics analysis, including mapping the security contributions of the demonstrated capability to the *Framework for Improving Critical Infrastructure Cybersecurity* (Cybersecurity Framework) and other relevant standards. Finally, the NCCoE worked with industry collaborators to suggest future considerations for mitigating IoT-based DDoS threats.

3.1 Audience

The focus of this project is on home and small-business deployments. This guide is intended for

- IoT device manufacturers, sensor manufacturers, networking companies, and industry groups
- internet service providers (ISPs), venture capitalists, and Smart Cities interests
- standards development organizations such as the Internet Engineering Task Force (IETF), foreign government organizations, and state/local governments having IoT authority and standards

3.2 Scope

The objective of this project is to demonstrate a proposed approach for deployment of IoT devices in home and small-business networks in a manner that provides significantly higher security than is typically achieved in today's IoT environments. The scope of this NCCoE project includes both home and small-business applications where plug-and-play deployment is required. The demonstration prototype network includes MUD-capable IoT devices that interact with external systems to access secure updates and various cloud services, in addition to interacting with traditional personal computing devices, as permitted by their MUD files. It employs both MUD-capable and non-MUD-capable IoT devices, such as smart lighting controllers, cameras, smartphones, printers, baby monitors, digital video recorders, and smart assistants.

3.3 Assumptions

The primary technical elements of a MUD-capable home and small-business IoT system include

- MUD managers
- MUD file servers
- MUD file and corresponding signature file
- MUD-capable DHCP servers
- MUD-capable routers or switches supporting wired and wireless network access
- MUD-capable IoT devices
- non-MUD-capable (legacy) IoT devices
- personal computing devices (personal computers, tablets, and phones)
- business computing devices

- update servers

Cost is a major factor affecting consumer purchasing decisions and consequent product development decisions.

MUD-capable IoT devices deployed in environments that incorporate the networking and best practice controls included in this project should be able to only send traffic to and receive traffic from preapproved devices, such as associated cloud-based services or update servers. A malicious actor would need to compromise the professionally operated cloud service or update server to detect or launch an attack, and each compromise would apply only to devices that are designed to communicate with the compromised service or update server. Best practices for administrative access and security updates would reduce the success rate for attempted compromises. Previously long-lived vulnerabilities (global administrative passwords) or short-lived vulnerabilities (known vulnerabilities subject to security updates) would be unavailable. As a result, the malicious actor would be forced to use expensive zero-day attacks or socially engineered administrative passwords, which are not scalable. If an IoT device is compromised despite these controls, virtual network segmentation can prevent lateral movement within the home/enterprise or prevent attacking systems outside the preapproved list; in this situation, control of the IoT device would be of dubious value. Obtaining value from a compromised device would demand the additional step of integrity attacks on the list of approved communicating devices. That is, attacking *www.example.com* with a botnet of thermostats would require modifying the product vendor's list of approved communicating devices to indicate that thermostats should be allowed to communicate with *www.example.com*.

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: A System Life Cycle Approach for Security and Privacy*, material that is available to the public. The risk management framework (RMF) guidance as a whole proved invaluable in giving us a baseline to assess risks, from which we developed the project, the security characteristics of the build, and this guide.

According to CNSSI No. 4009*, Committee on National Security Systems (CNSS) Glossary,* risk management is "the program and supporting processes to manage information security risk to organizational operations (including mission, functions, image, reputation), organizational assets, individuals, other organizations, and the Nation, and includes: (i) establishing the context for risk-related activities; (ii) assessing risk; (iii) responding to risk once determined; and (iv) monitoring risk over time." *Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks,* NIST Interagency/Internal Report (NISTIR) 8228, identified security and privacy considerations and expectations that, together with the *Framework for Improving Critical Infrastructure Cybersecurity* (Cybersecurity Framework) and *Security and Privacy Controls for Federal Information Systems and Organizations* (NIST SP 800-53), informed our risk assessment and subsequent recommendations from which we developed the security characteristics of the build, 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 whitelisting device capabilities and blacklisting 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 the 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 the compromise of 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 move laterally 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 today's complex software systems. Currently accepted coding practices result in approximately one software bug for every one thousand lines of code, and many of these bugs create security vulnerabilities. Modern systems ship with millions of lines of code, creating a target-rich environment for malicious actors. Although some vendors provide patches for security vulnerabilities and an efficient means for securely updating their products, patches are often unavailable or nearly impossible to install on many other products, including many IoT devices. Poorly 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. Once again, IoT devices are particularly vulnerable. The Mirai malware relied heavily on hard-coded administrative access to assemble botnets consisting of more than 100,000 devices.

3.4.3 Risk

The demonstrated capability implements a set of protocols designed to permit users and product support staff to constrain access to IoT devices. Implementation for some but not all IoT components in a system mitigates only the threat based on subversion of those devices. The system as a whole remains vulnerable. A residual risk is that the implementation of the demonstrated capability may be prophylactic only. It does not necessarily permit owners to find, identify, and correct already-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 files by connecting the devices to specific ports and associating each port with a MUD file appropriate to the device. If the device is compromised and attempts unauthorized communication, the attempt should be detected. That is, the device would still 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 bad internet protocol (IP) address can be detected, and the connection can be refused.

4 Architecture

The project architecture is intended for home and business networks that are composed of both IoT (e.g., single-purpose) and fully featured devices (e.g., personal computers and mobile devices) to constrain communications-based malware exploits. The architecture is designed to provide three forms of protection:

- use of the MUD specification to permit a MUD-capable IoT device to signal to the network what sort of access and network functionality it requires to properly operate, thereby reducing the potential for the device to be used in a DDoS 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 devices
- automatic secure software updates to all devices to ensure that operating system (OS) patches are installed promptly

4.1 Logical Architecture

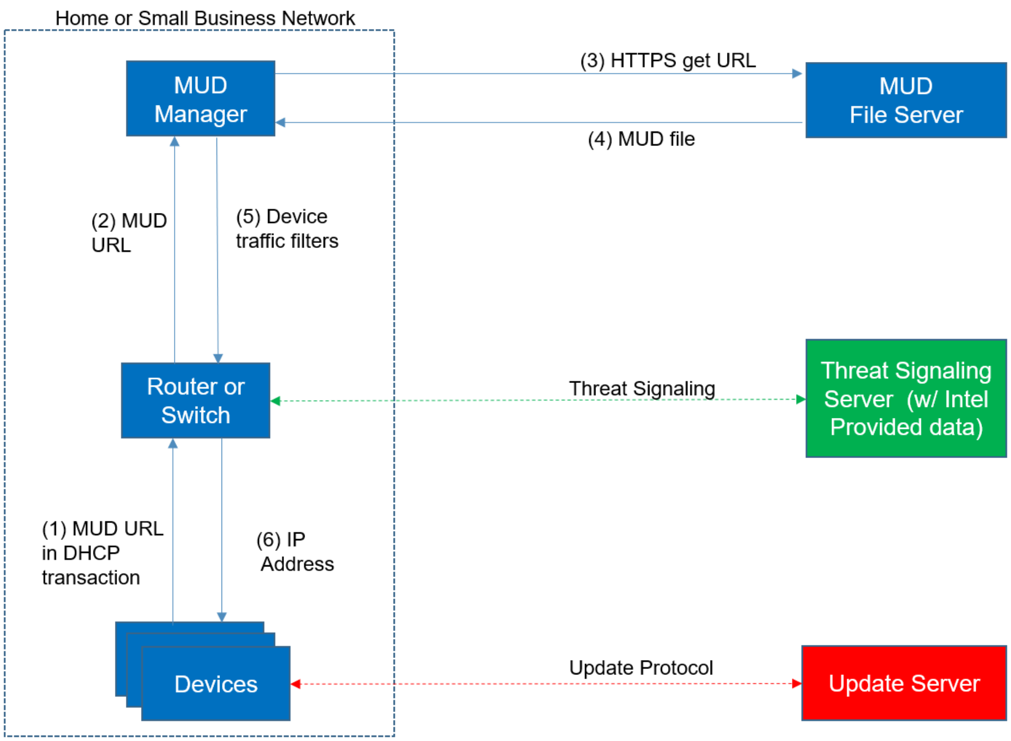
Figure 4-1 depicts the logical architecture. A new functional component, the MUD manager, is introduced into the home or enterprise network to augment the existing networking functionality offered by the router or switch: address assignment and control of access to devices.

IoT devices insert the MUD URL into DHCP address requests that they generate when they attach to the network (e.g., when powered on). The MUD URL is passed to the MUD manager, which retrieves a MUD file from the designated website (denoted as the MUD file server) using https. The MUD file describes the communications requirements for this device; the MUD manager converts the requirements into traffic filters (e.g., access control lists—ACLs) that are installed on the router or switch to enforce access controls on the network. This enables the router or switch to deny traffic sent to or from the IoT device that is outside the device's communications profile.

To provide further security, periodic updates are incorporated into the architecture. IoT devices periodically contact the appropriate update server to download and apply security patches. To ensure that such updates are possible, the IoT device's MUD file must explicitly permit the IoT device to receive traffic from the update server.

The router or switch could also periodically receive threat feeds from the threat signaling server to use as a basis for restricting certain types of network traffic. For example, malicious traffic can be denied access to a device by a cloud-based or infrastructure service like domain name system (DNS), with detailed threat information, including type, severity, and mitigation available to the router or switch on demand. (Note that although threat signaling is part of the logical architecture, it is not part of the current build. Threat signaling is planned for inclusion in a later phase of the project.)

Figure 4-1 Logical Architecture



Note that communications between the MUD manager and router/switch, between the threat signaling server and router/switch, and between IoT devices and the corresponding update server are not standardized.

The components of this architecture will not provide perfect security, but they will significantly increase the effort required by malicious actors to compromise and exploit IoT devices on a home or small-business network.

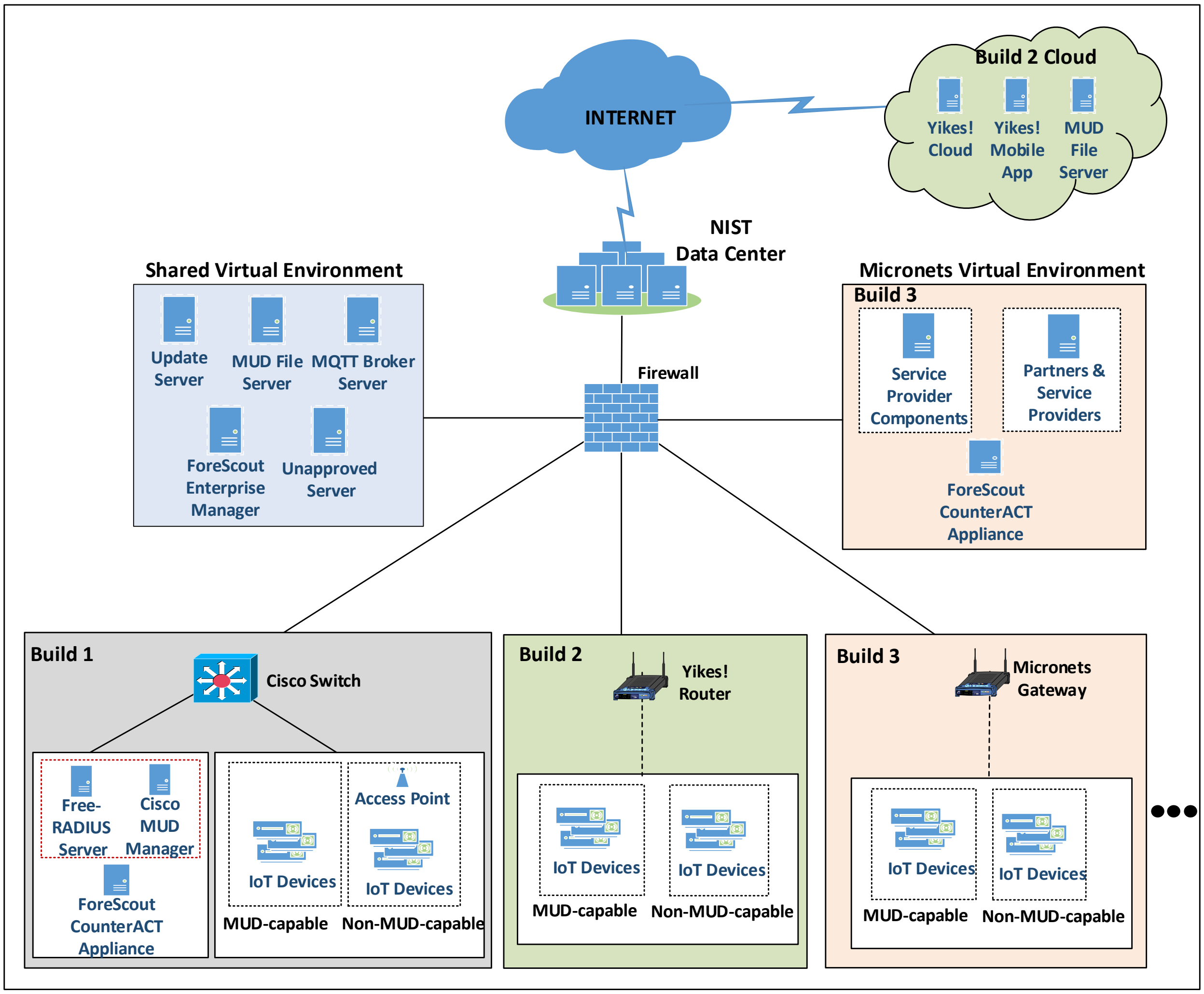
The components shown in the high-level architecture are described in Section 4.3 below.

4.2 Physical Architecture

Figure 4-2 depicts the high-level physical architecture of the NCCoE laboratory implementation. This implementation supports the flexibility to implement additional builds in the future. 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 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 ForeScout Enterprise Manager. (Note that although threat signaling is part of the logical architecture, there is currently no threat signaling server included in the laboratory network's virtual environment; threat signaling is planned for inclusion in a later phase of the project.) These components are hosted at NCCoE and will be used across builds. The Transport Layer Security (TLS) certificate and Premium certificate used by the MUD file server are provided by DigiCert.

Only Build 1, as depicted in the diagram, has been implemented during this phase of the project. Build 2 and Build 3 will be part of the next phase of the project. Build 1 network components consist of a Cisco Catalyst 3850-S switch, a Cisco MUD Manager, a FreeRADIUS Server, and a virtualized ForeScout CounterACT appliance. IoT devices used in this architecture include both MUD-capable and non-MUD-capable IoT devices. The MUD-capable IoT devices for Build 1 include Raspberry Pi, Artik, u-blox, Intel UP Squared, and the Molex Light Engine controlled by Power Over Ethernet (PoE) Gateway. Non-MUD-capable devices chosen for Build 1 include a wireless access point, cameras, a printer, smartphones, lighting devices, a smart assistant device, a baby monitor, and a digital video recorder. Build 1 and the role that each of its components plays in the architecture are explained in more detail in Section 4.3 and Section 4.4.

Figure 4-2 Physical Architecture



4.3 Technologies

Table 4-1 lists all the products and technologies used in this project 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. Some functional Subcategories are described as being directly provided by a component. Others are described as Subcategories, the provision of which is supported by a component but not directly provided by a component. Refer to Table 5-1 for an explanation of the Cybersecurity Framework's Subcategory codes.

Table 4-1 Products and Technologies

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/)	YANG script GUI used to create MUD files	ID.AM-1
MUD file	A YANG model instance that has been serialized in javascript object notation (JSON) [RFC7951]. 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

Component	Product	Function	Cybersecurity Framework Subcategories
DHCP server	Cisco IOS (Catalyst 3850-S)	Dynamically assigns IP addresses; recognizes MUD URL in DHCP DISCOVER; 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
Link Layer Discovery Protocol (LLDP)	Cisco IOS (Catalyst 3850-S)	Supports capability for devices to advertise their identity and capabilities to neighbors on a local area network (LAN) 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; used to sign MUD files and generate 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; requests and applies software updates	ID.AM-1
Non-MUD-capable IoT device	Cameras Smartphones Smart lighting devices Smart assistant Printer Baby monitor Wireless access point Digital video recorder	Acts as typical IoT devices on a network; creates network connections to cloud services	ID.AM-1
Update server	NCCoE-hosted Apache server Molex Update Agent	Provides 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

Component	Product	Function	Cybersecurity Framework Subcategories
MQTT Broker Server	NCCoE-hosted MQTT server	Receives and publishes messages to/from clients	ID.AM-3 DE.AE-3
IoT Device Discovery	ForeScout CounterACT Virtual Appliances and Enterprise Manager	Discovers 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.

4.3.1 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.

4.3.1.1 Cisco MUD Manager

The Cisco MUD Manager is an open-source implementation. For this project, the Cisco MUD Manager was used to support IoT devices that emit their MUD URLs via DHCP messages and other IoT devices that emit their MUD URLs via the 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 network (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 MAC Authentication Bypass (MAB) 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 6.1, Findings.

4.3.2 MUD File Server

In the absence of a commercial MUD manager for use in 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.

4.3.3 MUD File

Using the MUD file maker component referenced above in Table 4-1, 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 Network Time Protocol—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., 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: devicetype (URL decided by the device manufacturer)
        - Protocol: TCP
        - Direction-initiated: from IoT device
        - Source port: any
        - Destination port: 80

4.3.3.1 Signature file

According to the IETF MUD specification, “a MUD file MUST be signed using 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 (this will be detailed 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.

4.3.4 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.

4.3.4.1 Cisco DHCP Server

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.

4.3.5 Router/Switch

This project uses the Cisco Catalyst 3850-S switch.

4.3.5.1 Cisco Catalyst 3850-S

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 also 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 6.1, Findings.

4.3.6 Certificates

DigiCert's CertCentral™ web-based platform allows for 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 using only a browser. Multiple roles can be assigned within an account, and a discovery tool can be used to inventory all certificates within the enterprise. In addition to certificate-specific features, the platform also offers baseline enterprise software as a service (SaaS) capabilities, including role-based access control (RBAC), security assertion markup language (SAML), single sign-on (SSO), and security policy management and enforcement. All account features come with full parity between the web portal and a publicly available application programming interface (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.

4.3.7 IoT Devices

This section describes the IoT devices used in the laboratory implementation. There are two distinct categories of devices: devices that are capable of emitting 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.

4.3.7.1 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 devkits were modified by the NCCoE 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.

4.3.7.1.1 Molex PoE Gateway and Light Engine

This set of IoT devices was developed by Molex. 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 was provided by Molex.

4.3.7.1.2 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.

4.3.7.1.3 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.

4.3.7.1.4 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 smart lock. This Python script allowed the ARTIK devkit to receive on and off commands by using the MQTT protocol.

4.3.7.1.5 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 smart lighting device. This allowed the UP Squared Grove devkit to receive on and off commands by using the MQTT protocol.

4.3.7.2 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, smartphones, lighting, a smart assistant, a printer, a baby monitor, a wireless access point, and a digital video recorder (DVR).

4.3.7.2.1 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 smartphone.

4.3.7.2.2 Smartphones

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

4.3.7.2.3 Lighting

Two types of smart lighting devices are used in the laboratory implementation. These smart lighting components are controlled and managed by a smartphone.

4.3.7.2.4 Smart Assistant

A smart assistant is utilized in the laboratory implementation. The device is used to demonstrate and test the wide range of network traffic generated by a smart assistant.

4.3.7.2.5 Printer

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

4.3.7.2.6 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 smartphone.

4.3.7.2.7 Wireless Access Point

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

4.3.7.2.8 Digital Video Recorder

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

4.3.8 Update Server

The update server provides patches and other software updates to the IoT devices. This project used an NCCoE-hosted update server.

4.3.8.1 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.

4.3.8.2 Molex Update Agent

The process for updating the firmware on a Molex PoE Gateway is currently a manual process, with the firmware update taking place over the CoAP, UDP, and trivial file transfer protocol (TFTP) protocols. The update process is initiated by an update agent on the local network connecting to the PoE Gateway and sending the firmware update information.

4.3.9 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-enabled 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, communication to and from the unapproved server and the MUD-enabled IoT device was tested.

4.3.10 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 provides the ability for mobile devices 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.

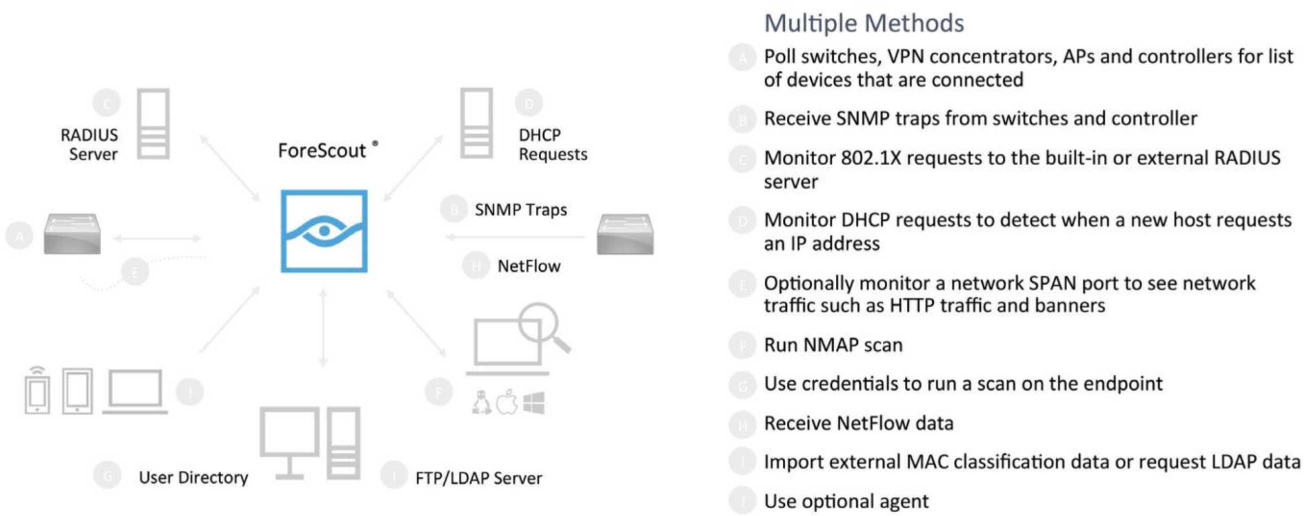
4.3.11 IoT Device Discovery

This project uses ForeScout CounterACT appliance and Enterprise Manager to provide an IoT device discovery service for the demonstration network. CounterACT is able to 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 CounterACT 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

CounterACT 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, CounterACT may learn about that device via a variety of different techniques to discover and classify it without requiring agents, as shown in Figure 4-3. The methods demonstrated in this project included the following: CounterACT passive discovery of devices 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.

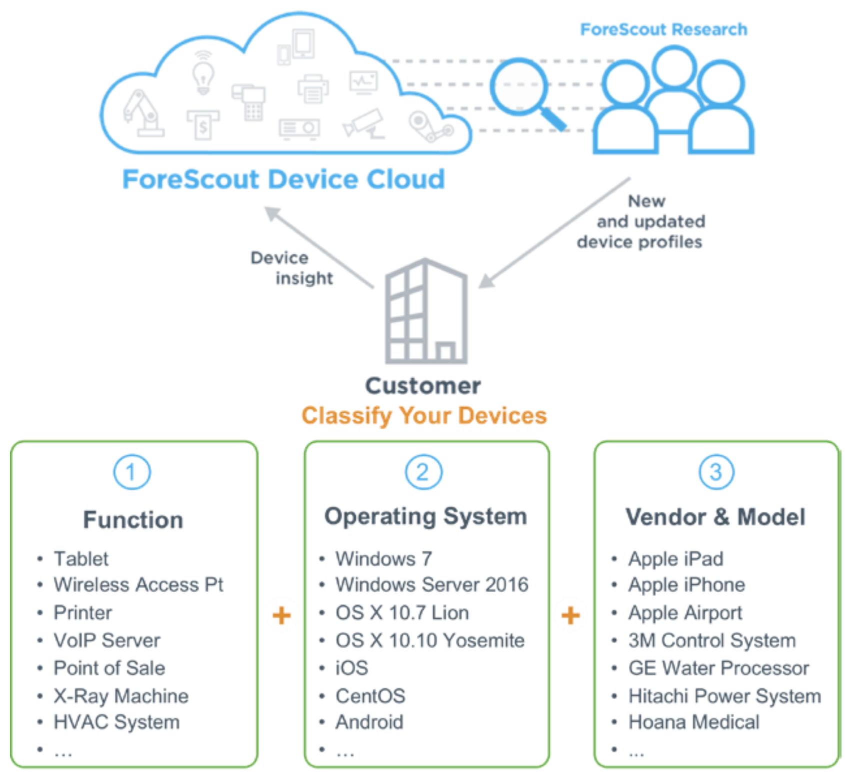
Figure 4-3 Methods the ForeScout Platform Can Use to Discover and Classify IP-Connected Devices



ForeScout CounterACT 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 4-4. 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 the time of this writing, the ForeScout CounterACT appliance did not have the ability to identify whether an IoT device on the network was MUD-enabled.

Figure 4-4 Classify IoT Devices by Using the ForeScout Platform



4.4 Build Demonstration

The MUD manager used in the capability demonstration was provided by Cisco. Cisco is a provider of enterprise, telecommunications, and industrial networking solutions. The work in this project is being undertaken within Cisco's Enterprise Central Software Group, with an eye toward improving the product offering over time. Cisco has provided a proof-of-concept MUD manager as well as a Catalyst 3850-S switch with Power-over-Ethernet.

Figure 4-5 describes the logical architecture of the first build. The example implementation is designed with a single device serving as the MUD manager and FreeRADIUS server that interfaces with the Catalyst 3850-S switch over TCP/IP. The Catalyst 3850-S switch contains a DHCP server that is configured to extract MUD URLs from IPv4 DHCP transactions. Upon connecting a MUD-enabled device, the MUD URL will be emitted in some approved method (LLDP, X.509, or DHCP)–for this example implementation, DHCP and LLDP were leveraged (step 1). The Catalyst 3850-S switch will send 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 will fetch the MUD file 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 will request the signature file associated with the MUD file (step 4a) and upon receipt (step 4b) will verify the MUD file with the respective 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).

Figure 4-5 Logical Architecture–Build 1

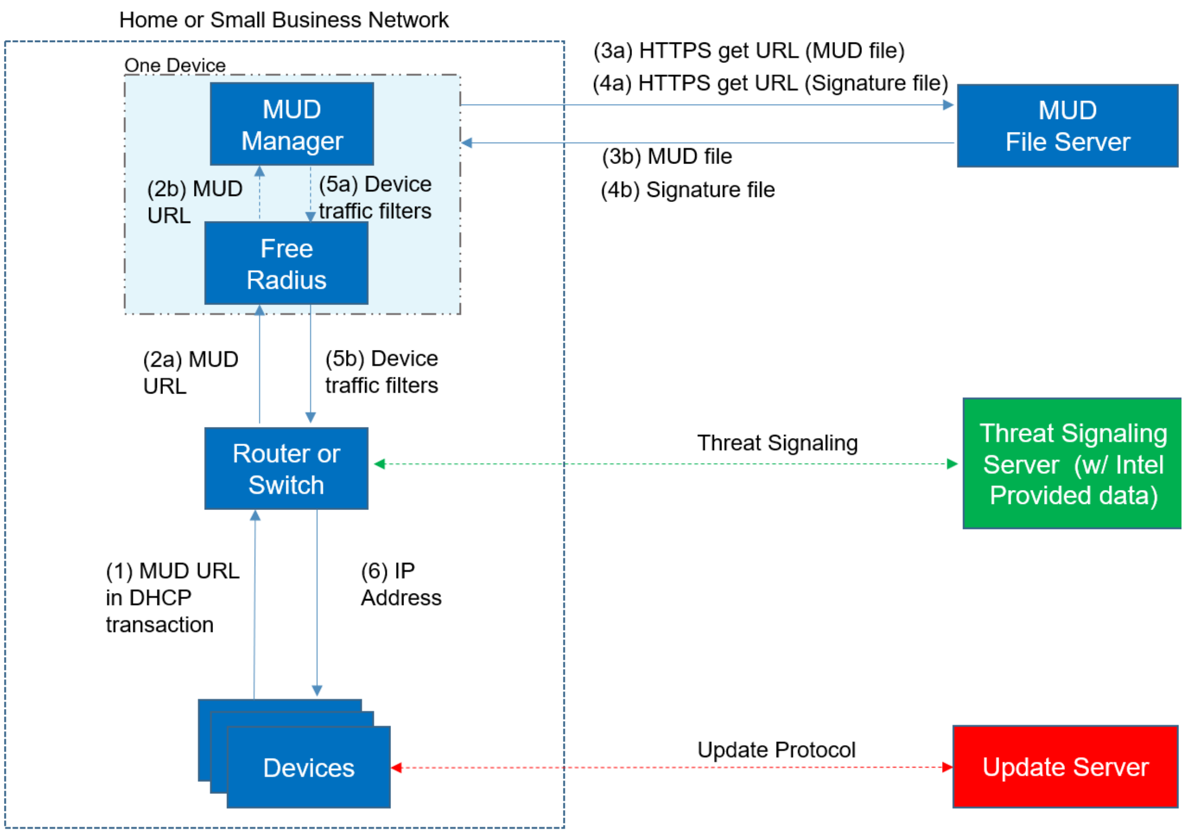
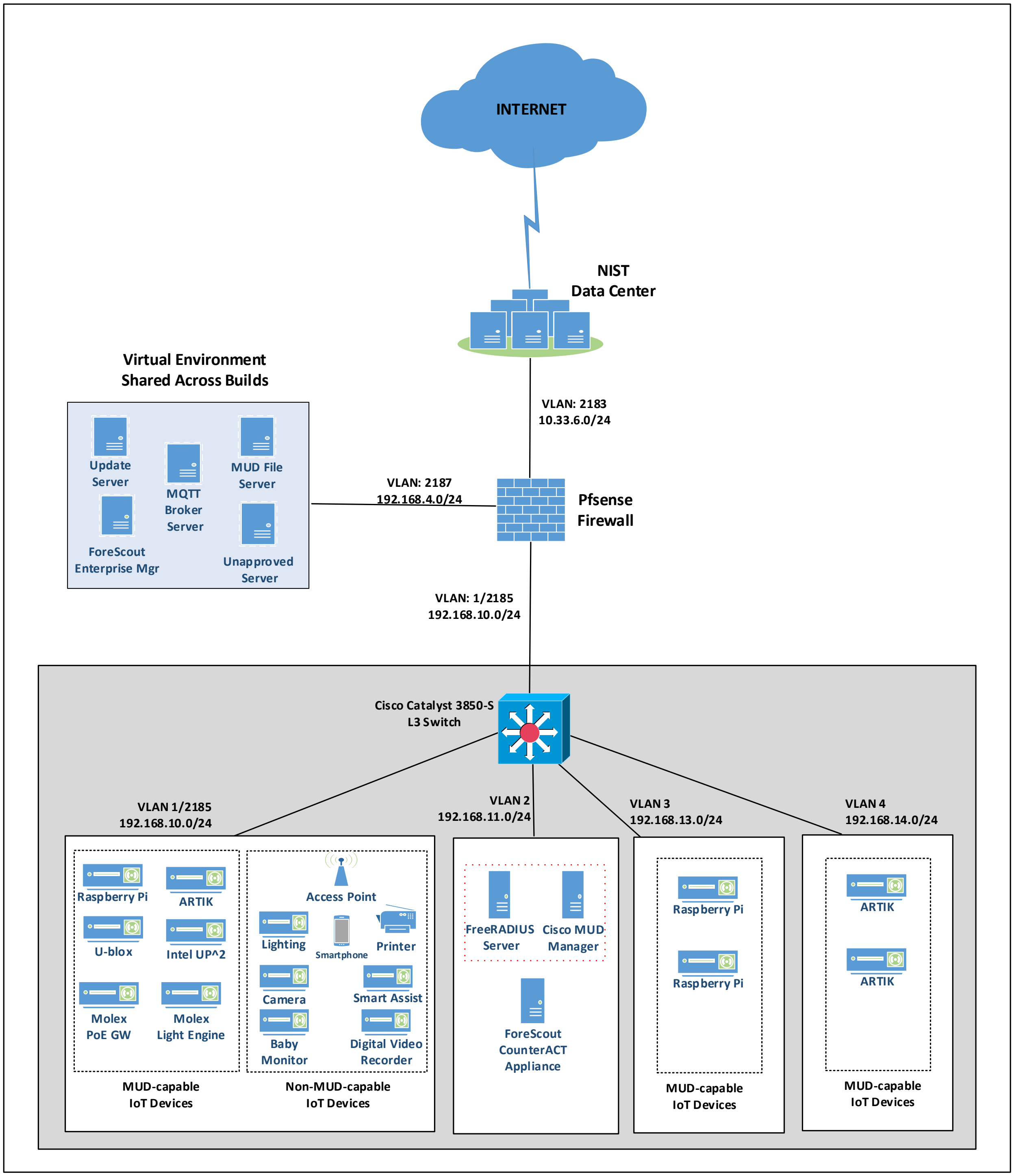


Figure 4-6 describes the physical architecture of laboratory build 1. The Catalyst 3850-S switch is configured to host four VLANs. The first VLAN, VLAN 1 hosts a large number of IoT devices. Three separate instances of DHCP servers are configured for VLANs 1, 3, and 4 to dynamically assign IPv4 addresses 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 CounterACT 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.

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

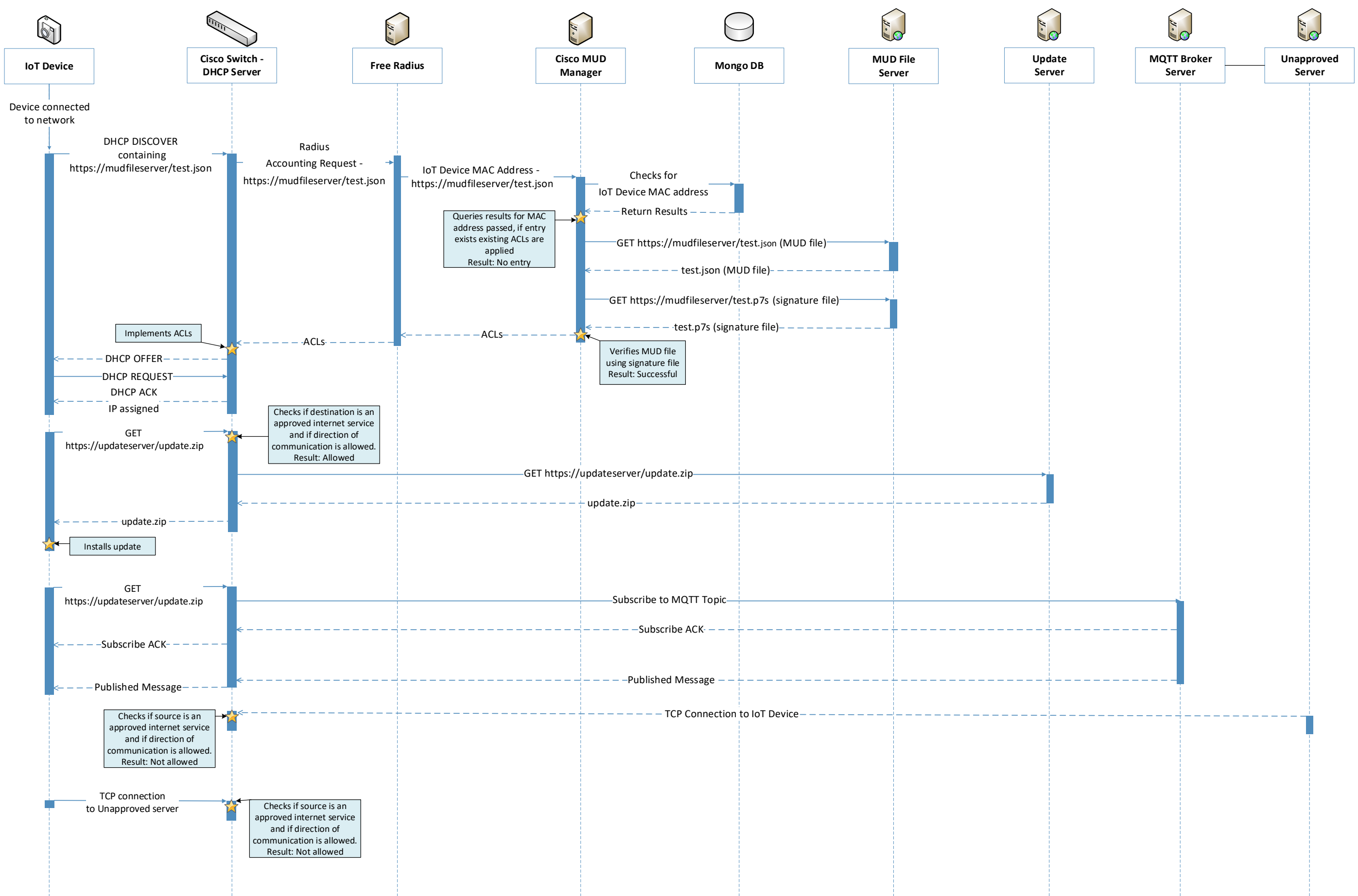


834

A full description of Cisco's proof of concept can be found 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 a number of 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.

Figure 4-7 shows the process flow of onboarding a MUD-enabled IoT device that emits a MUD URL via DHCPv4.

Figure 4-7 Process Flow–Build 1



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

- A MUD-enabled IoT device is connected to the network.

- The MUD-enabled 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 request. It is possible to transmit the option in both DISCOVERY 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 onboarded 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-enabled IoT device is assigned an IP address and is ready to be used on the network.
- Finally, when the MUD-enabled IoT device is in use, access of all traffic to and from the IoT device is controlled by the Cisco switch.

Communications that are allowed by the MUD file include egress north/south and east/west traffic. At the time of publication, ingress access control was not yet supported. Specifics can be found in Section 6.1, Findings. The version of the Cisco MUD Manager implemented in this build leverages a JSON configuration file that is responsible for translating many of the abstractions that are defined by the MUD file. East/west constructs as described in the MUD specification are

- Controller–class of devices known to be controllers (could describe well-known services such as DNS or NTP)
- My-controller–class of devices that the local network administrator admits to the particular 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

In addition to the components required for the MUD solution to function as defined, the example implementation also includes the CounterACT appliance. This appliance was leveraged in order to discover IoT devices on network.

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 in a manner that maintains component functionality while limiting unauthorized access to and from the components. In addition, it seeks to understand the security benefits and drawbacks of the example solution.

5.1 Assumptions and Limitations

The security characteristic analysis has the following limitations:

- It is not a comprehensive test of all security components, nor is it 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 the security characteristic analysis involved assessing how well the reference design addresses the security characteristics that it was intended to support. The NIST Cybersecurity Framework Subcategories were used to provide structure to the security assessment. We consulted the specific sections of each standard that are cited in reference to a Subcategory. The cited sections provide validation points that the example implementation 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 characteristics analysis was conducted in the context of home network and small-business usage scenarios. Use in large enterprise environments may be included in future project extensions but is not considered in this analysis.

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* (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/Internal Report 8228). NIST Interagency/Internal Report 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. As described in the Mitigating IoT-Based Distributed Denial of Service (DDoS) project description, the example implementation directly addresses the PR.AC-1, PR.AC-2, 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 security platform directly addresses NIST SP 800-53 controls AC-3, AC-18, CM-7, SC-5, SC-7, SC-28, and SI-2, and it supports activities addressing NIST SP 800-53 controls AC-4, AC-6, AC-24, CM-8, IA-2, IA-5, IA-8, PA-4, PM-5, RA-5, SC-8, and SI-5. In addition, seven of the NIST Interagency/Internal Report 8228 expectations are addressed by the example implementation. Table 5-1 describes how example implementation characteristics address NIST Interagency/Internal Report 8228 expectations, NIST SP 800-53 controls, and Cybersecurity Framework Subcategories.

Table 5-1 Mapping Demonstration Platform Characteristics to NIST Interagency/Internal Report 8228 Expectations, NIST SP 800-53 Controls, and Cybersecurity Framework Subcategories

	Applicable Project Description Element That Addresses the Expectation	Applicable NISTIR 8228 Expectations	Draft NIST SP 800-53 Rev 5 Controls Supported	Cybersecurity Framework Subcategories Supported
1	IoT devices insert the MUD extension into DHCP address requests when they attach to the network (e.g., power on).	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.
2	The contents of the MUD extension are 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.	Device can interface with enterprise asset management systems.	Provides: AC-3 Access Enforcement AC-18 Wireless Access CM-7 Least Functionality SC-5 Denial of Service Protection SC-7 Boundary Protection	Provides: PR.PT-3 The principle of least functionality 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

			Supports: AC-4 Information Flow Enforcement AC-6 Least Privilege AC-24 Access Control Decisions CM-8 System Component Inventory PM-5 System Inventory	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., 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.
5	IoT devices periodically contact the appropriate update server to download and apply security patches.	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.
7	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. (Note that although threat signaling is included as part of the reference architecture, it has not yet been implemented in the build.)	The device either supports the use of vulnerability scanners or provides built-in vulnerability identification and reporting capabilities.	Supports: AC-24 Access Control Decisions RA-5 Vulnerability Scanning SI-5 Security Alerts, Advisories, and Directives	Supports: ID.RA-2 Cyber threat intelligence is received 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.
11	The contents of the MUD extension are 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 control information for enforcement by the router or switch.	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.
21	IoT devices insert the MUD extension into DHCP address requests when they attach to the network (e.g., power on). The contents of the MUD extension are passed to the MUD manager, which retrieves a MUD file from the designated website (denoted as the MUD file server) by using https. The	Device can prevent unauthorized access to all sensitive	Provides: SC-23 Session Authenticity Supports:	Provides: PR.PT-3 The principle of least functionality is incorpo-

	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.	data transmitted from it over networks.	AC-18 Wireless Access SC-8 Transmission Confidentiality and Integrity	rated 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.
24	IoT devices insert the MUD extension into DHCP address requests when they attach to the network (e.g., power on). The contents of the MUD extension are 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. 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. (As mentioned earlier, although a part of the logical architecture, threat signaling has not yet been implemented in the build.)	There is sufficient centralized control to apply policy or regulatory requirements to personally identifiable information.	Supports: PA-4 Information Sharing with External Parties	None

Table 5-2 details Cybersecurity Framework Identify, Protect, and Detect Categories and Subcategories that the example implementation directly addresses or for which the example implementation may serve a supporting role. Those Subcategories that are directly addressed are highlighted in green. 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 **SA 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 cybersecurity risk to organizational operations (including mission, functions, image, or reputation), organizational assets, and individuals.	**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, 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.	**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 **CIS CSC** 1, 5, 15, 16
	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, 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
E.g., 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-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, 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 and references required to develop this solution are identified in 7.2Appendix E. 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 in the Internet Engineering Task Force.

5.3 Scenarios

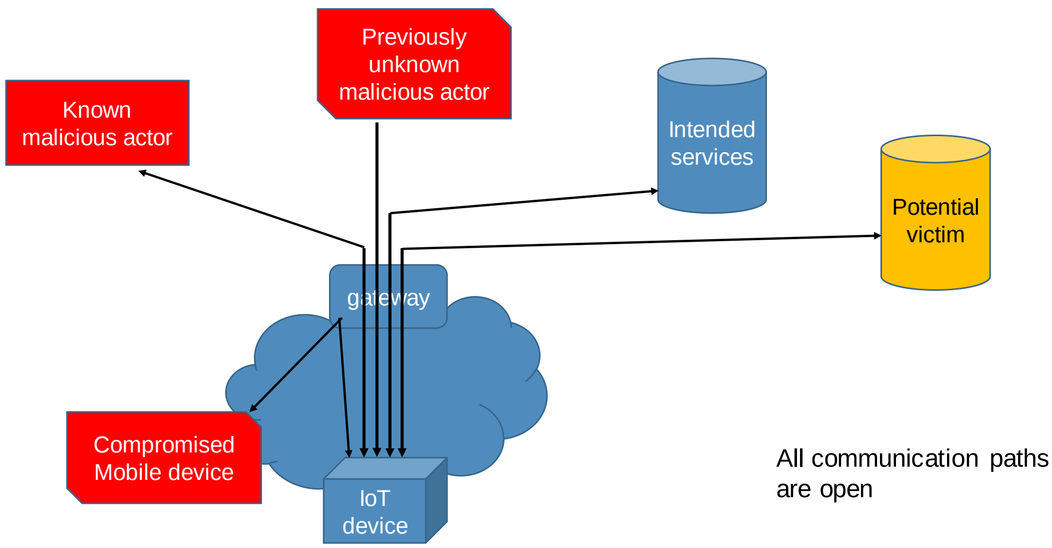
This section presents three threat scenarios for home and small-business networks involving increasing levels of security. In the first scenario, MUD is not deployed on the network, so IoT devices are vulnerable to being port scanned and are not restricted from exchanging traffic with either external sites or other devices on the local network. IoT devices in this first scenario are highly vulnerable to attack.

In the second scenario, MUD is deployed on the network, but the MUD files being used only restrict traffic from being sent between the IoT devices and some external internet domains (i.e., north-south traffic); IoT devices are still able to send and receive traffic from all other devices on the local network (i.e., east-west traffic). In the third scenario, the MUD file protections provided in scenario 2 are enhanced to also restrict traffic between IoT devices and some other devices on the local network, ensuring that the IoT devices are permitted to exchange traffic with only external domains and internal devices that are explicitly specified in their MUD file.

5.3.1 Scenario 1: No MUD Protection

In the *No MUD Protection* scenario, as shown in Figure 5-1, the home/small-business network (depicted by the blue cloud) does not have MUD deployed to provide security for its IoT devices.

Figure 5-1 No MUD Protection Threat Scenario



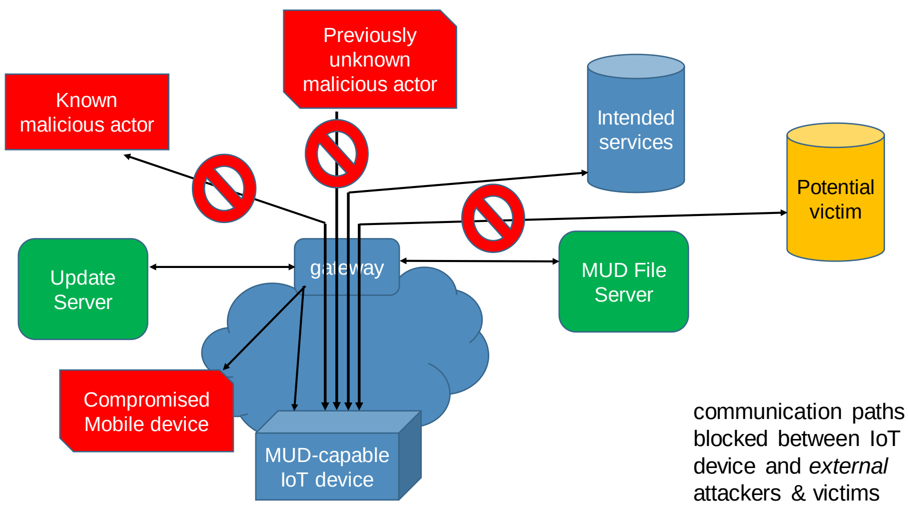
All IoT devices on the network can be port scanned (and perhaps hijacked) from anywhere. IoT devices are permitted access to and from intended services as desired. However, the IoT devices are also reachable by compromised mobile devices that are on their local network and by malicious external devices, making them vulnerable to attacks from these compromised and malicious devices. In addition, if an IoT device becomes compromised, there are no protections in place to stop it from launching an attack on outside devices, creating additional potential victims.

5.3.2 Scenario 2: MUD Protection from External Threats

In the *MUD Protection from External Threats* scenario, as shown in Figure 5-2, the home/small-business network (depicted by the blue cloud) has MUD deployed (the components of the MUD deployment are not depicted). The MUD file for the IoT device lists the domains of all external services with which the device is permitted to exchange traffic. All domains that are not explicitly permitted in the MUD file are denied. Therefore, the IoT device on the network can freely communicate with its intended external services, but all other attempted communications between the IoT device and external devices are blocked. The IoT device cannot be port scanned or receive traffic from external malicious actors, even if those actors are not known to be malicious. Furthermore, even if the IoT device is compromised in some way after being onboarded, it will not be permitted to send traffic to any external devices to attack those devices. One of the external devices with which the IoT device is permitted to communicate is an update server, from which the device receives regular software updates to ensure that it installs the most recent security patches as needed.

Unfortunately, the MUD file for the IoT device in this scenario also includes a construct that permits the IoT device to exchange traffic with all devices that are on the local network. If a device on the local network becomes compromised, that device will be able to attack the IoT device.

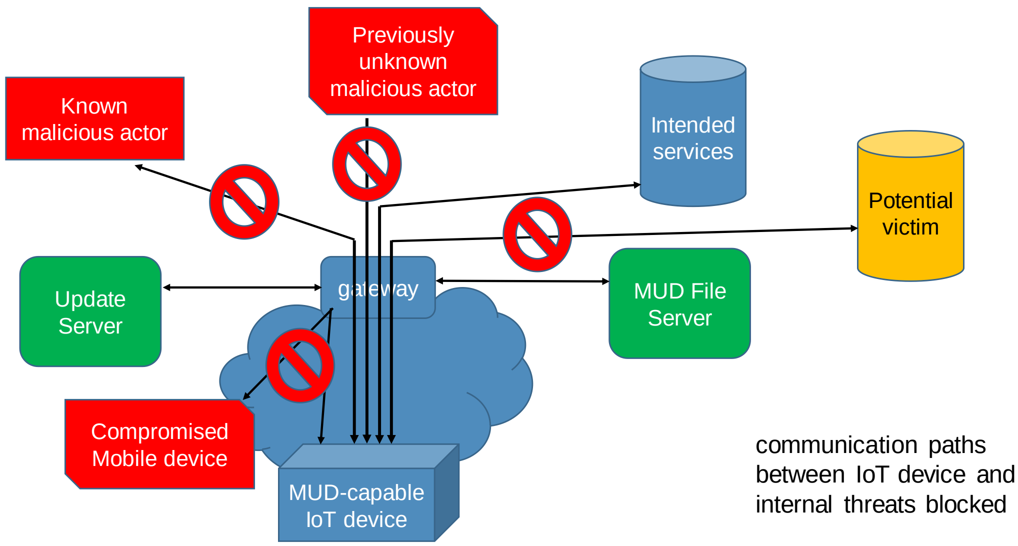
Figure 5-2 MUD Protection from External Threats Scenario



5.3.3 Scenario 3: MUD Protection from External and Internal Threats

In the *MUD Protection from External and Internal Threats* scenario, as shown in Figure 5-3, the home/small-business network (depicted by the blue cloud) has MUD deployed (the components of the MUD deployment are not depicted). As in the MUD Protection from External Threats scenario, the MUD file for the IoT device lists the domains of all external services with which the device is permitted to exchange traffic, thereby protecting the IoT devices from being attacked by external entities and protecting external entities from being attacked by the IoT device. In addition, unlike the previous scenario, the MUD file in this scenario does not include a construct that permits the IoT device to exchange traffic with all other devices that are on the local network. Instead, the MUD file specifies specific devices with which the IoT device is permitted to communicate based on, for example, the manufacturer of those other devices. If a local device is not from the specified manufacturer, it will not be permitted to communicate with the IoT device. So, if a device on the local network becomes compromised and that device's manufacturer or model is not one that has been explicitly permitted in the MUD file, that device will not be permitted to send traffic to attack the IoT device.

Figure 5-3 MUD Protection from External and Internal Threats Scenario



5.4 Build Evaluation

A functional evaluation of the IoT example implementation was conducted to verify that it meets the requirements. Table 5-3 summarizes the tests that were performed, their 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 described in a functional evaluation plan that is provided in 7.2Appendix D. Not all tests defined in the functional evaluation plan are listed in Table 5-3 because not all those tests were applicable to this build of the IoT example implementation. Boldface text is used in the Test Summary and Expected Outcome columns to highlight the gist of the information that is being conveyed.

Table 5-3 Summary of Functional Tests

Test	Applicable Cybersecurity Framework Subcategories & 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.	A **MUD-enabled IoT device is configured to emit a MUD URL.** The	Upon connection to the network, the MUD-enabled IoT	Pass

	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 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	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 into local network configurations that it installs on the router or switch that is serving as the MUD PEP for the IoT device.**	device has its MUD **policy enforcement point (PEP) router/switch automatically configured according to the MUD file's route filtering policies.**	

	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-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 the time of signing. Local policy has been configured to ensure that if the MUD file for a device has a signature that was signed by a**	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

		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 the time of 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 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 security principles (e.g., concept of least functionality).	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 respect to traffic being permitted to/from some internet locations; and traffic being implicitly blocked to/from	Pass (for testable procedure –ingress cannot be tested)

	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		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 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 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 (for testable procedure –ingress cannot be tested)
IoT-7	**PR.IP-3:** Configuration change control processes are in place. **NIST SP 800-53 Rev. 4** CM-3, CM-4, SA-10	Test IoT-1 has run successfully, meaning that the MUD PEP **router/switch has**	When the MUD-enabled **IoT device explicitly releases its IP address lease,**	Pass

	PR.DS-3: Assets are formally managed throughout removal, transfers, and disposition.	**been configured based on the MUD file** for a specific MUD-enabled 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.**	the MUD-related configuration for that IoT device will be removed from its 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-enabled 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-enabled **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)
IoT-12	**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.	A MUD-enabled 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**	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-	Pass

	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. **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.	**the cache-validity period for that MUD file), a second IoT device that has been configured to emit the same MUD URL is connected to the network.**	filtering rules and installs these rules onto the PEP for the second IoT device.	

	NIST SP 800-53 Rev. 4 AC-3, CM-7 **PR.DS-2:** Data in transit is protected.			
IoT-13	**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.**	Pass
IoT-14	**ID.AM-1:** Physical devices and systems within the organization are inventoried.	A **MUD-enabled IoT device is capable of emitting 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 Upon initialization, the MUD-enabled IoT device **emits a**	Pass

			MUD URL as an LLDP extension.	

6 Findings and Recommendations

This section introduces findings based on the build implementation and demonstration, as well as recommendations.

6.1 Findings

- It is possible to provide significantly higher security than is typically achieved in today's (non-MUD-capable) home and small-business networks by deploying and using MUD on those networks.
- The NCCoE example solution demonstrates 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 components on the internal network that are not from authorized manufacturers or authorized device classes
    - preventing the MUD-capable IoT device from being used to access unauthorized external domains
    - preventing the MUD-capable IoT device from being used to access other components on the internal network that are not from authorized manufacturers or that are not authorized device types
- MUD can help prevent MUD-capable IoT devices from being used to launch DDoS and other attacks that are typically possible by commandeering non-MUD-capable IoT devices on today's home and small-business networks. To enable 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 are able to 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 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. It is

not clear that the current suite of available products is sufficiently user-friendly to enable the typical, nontechnical user to easily and seamlessly deploy MUD on their home and small-business networks. However, improvements in the usability of MUD components are expected as more manufacturers implement support for MUD and as current manufacturers that support MUD refine the design of their MUD components to target the home and small-business user.

- 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, yet that will continue to be used. The ability to leverage MUD to limit the communications profiles of such unsupported devices will be important for protecting these highly vulnerable devices from attack by unauthorized end points 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's certificate. In either of these cases, if the default behavior that the device is configured to perform is not acceptable, the user would need to configure the device to perform the desired behavior. Ideally the device would provide a user-friendly interface through which to do so.
- There is 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.
- 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.

- Specifically, for Build 1 we observed the following limitations that are informing improvements to the current proof-of-concept implementation:
    - MUD Manager (version 1.0):
        - DNS resolution of internet host names in the MUD file is performed manually and remains static. The MUD manager does not invoke a DNS resolution service to automatically resolve the fully qualified domain names (FQDNs) that are referenced in MUD files dynamically at the time that it reads those MUD files. Instead, DNS resolution of FQDNs referenced in MUD files is performed manually by a human operator before beginning execution of the MUD manager service. The operator inserts this address resolution information into the MUD manager's .JSON configuration file, where it remains static for the duration of the operation of the MUD manager service. The MUD manager consults this configuration file, which is passed to it at the time the MUD manager service is started, to obtain this static DNS resolution information.

          Use of the .JSON configuration file as the mechanism to provide the MUD manager with DNS resolution information means that every FQDN that will be referenced in any MUD file must be resolved and inserted into the MUD manager's configuration file manually before beginning execution of the MUD manager service. So, for example, if a MUD file that will be used on the network permits traffic to be sent to domains *www.example.com* and *www.company.com*, the MUD manager's .JSON configuration file must be provided with lines that indicate that the FQDN *www.example.com* resolves to IP address 128.56.54.3 and the FQDN *www.company.com* resolves to IP address 128.54.35.2.

          Use of the .JSON configuration file as the mechanism to provide the MUD manager with DNS resolution information also means that the operator who is configuring the .JSON file must have prior knowledge of all FQDNs that will be referenced in all MUD files that will used by devices on the network. If a MUD file references an FQDN that has not been listed and associated with a corresponding IP address in the MUD manager's .JSON configuration file, the MUD manager will not be able to configure an ACL to enforce controls related to that FQDN.

          In addition, because the DNS resolution information is passed to the MUD manager at execution time, this address resolution information remains static for as long as the MUD manager service continues to operate. To add new FQDN address resolutions or change existing ones, the MUD manager's .JSON file would have to be edited and the MUD manager process killed and restarted by using the new .JSON configuration file.

          Dynamic resolution of FQDNs is expected to be supported in the future.

    - Translation and implementation of the "model" construct from the MUD file was not supported at the time of testing. 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 capable of supporting 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-enabled IoT devices on the network. According to the MUD specification, the DHCP server should notify the MUD manager if the MUD-enabled IoT device's IP address lease expires or has been released. However, this version of the DHCP server does not do so at the time of testing. 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 a particular 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.

- 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

The NCCoE recommends additional work with IoT security that builds on the broader set of security controls.

6.2 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 communications vectors that can be used to attack a vulnerable IoT device and the communications 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 implementing 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, and so by its very nature can be disruptive if used by unauthorized parties." Protecting the MUD infrastructure includes ensuring the security of the IoT device MUD URL emission, the MUD manager, the DHCP server, 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 bad actor creates a compromised MUD manager; or if a router is compromised so that it does not enforce its ACL rules—then MUD can be used to enable rather than prevent potentially damaging communications between affected IoT devices and other domains.
- If a bad actor is able to 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 file for their IoT devices to ensure it specifies communications that are appropriate for the device. Unfortunately, on home and small-business networks, where users are not likely to have the technical expertise to enable themselves to understand how to read MUD files, users will be required to trust that the MUD files specify communications appropriate for the device or rely on a third party to perform this review for them.

- 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 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 DoS 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 communications 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 enable 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 be used to provide strong authentication of 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 does the MUD build 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 nonauthenticated 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), 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
    - To protect against rogue Certificate Authorities, the MUD implementation should be configured to raise an alert and require administrator approval if the MUD manager detects that the signer of a MUD file has changed.
    - To protect compromised IoT devices that seek to be associated with malevolent MUD files, the MUD implementation should be configured to raise an alert and require administrator approval if the MUD manager detects that a device's MUD file has changed.

  - To protect against domain name ownership changes that would permit a bad 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 above bullets 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.

6.3 Recommendations

The following are recommendations for using MUD:

- Home and small-business network owners should enable MUD on their networks by deploying a MUD-capable infrastructure.
- Home and small-business network owners should select and use MUD-capable IoT devices on their networks.
- ISPs should consider providing and supporting MUD-capable home routers for their customers.
- IoT device manufacturers should configure their devices to emit a MUD URL by default.
- 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, in the event that the communications 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 communications 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 end point with which the device may communicate.

- Manufacturers of MUD managers, MUD-capable DHCP servers, and MUD-capable routers that are 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 have visibility into every device on their network. Any device is a potential attack or reconnaissance point that must be discovered and secured. In particular, non-MUD-capable devices are inviting targets.
- Home and small-business network owners should segment their networks where possible. In small-business and home environments it may not be possible to apply good segmentation policies. But at a minimum, where there are IoT devices that are known to have security risks, e.g., non-MUD-capable devices, keep these on a separate network segment from the everyday computing devices that are afforded with 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 use the information presented in the Security Considerations section of the MUD Specification (RFC 8520) to enhance protection of MUD deployments.
- 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 as a whole.
- 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.
- 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 suggestions for continuing activity with the collaboration team:

- Continue work with collaborators to enhance MUD capabilities in their commercial products (see Section 6.1).
- 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.
- 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 communications 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.

7 Future Build Considerations

As the MUD build proceeded, some emerging components were not included in our initial demonstration platform. The physical architecture described in Section 4.2 and the technologies identified in Section 4.3 describe those components that were demonstrated in the course of developing this preliminary practice guide. The team is working on additional builds that include additional components that have become available. Findings from the demonstration of components, including those described in Appendix A will appear in a subsequent version of this draft practice guide.

The number of components that can employ the MUD protocol continues to grow rapidly. It is recommended that this project be further extended to demonstrate the capabilities of the maturing offerings of technology providers and to integrate additional related capabilities such as threat signaling. In addition, IPv6, for which MUD-capable products were unavailable for the initial demonstration sequences, adds a new dimension to using MUD to help mitigate IoT-based DDoS threats. As discussed in Section 7.2 below, inclusion of IPv6-capability should be considered for future builds.

7.1 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 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. Appendix A describes components scheduled for demonstration in the second phase of this project.

7.2 Recommended Demonstration of IPv6 Implementation

Due to product limitations, the initial phase 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 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 Information Provided by Collaborators Related to Phase 2

This appendix provides information provided by collaborators regarding potential components to be used in future builds.

A.1 MasterPeace

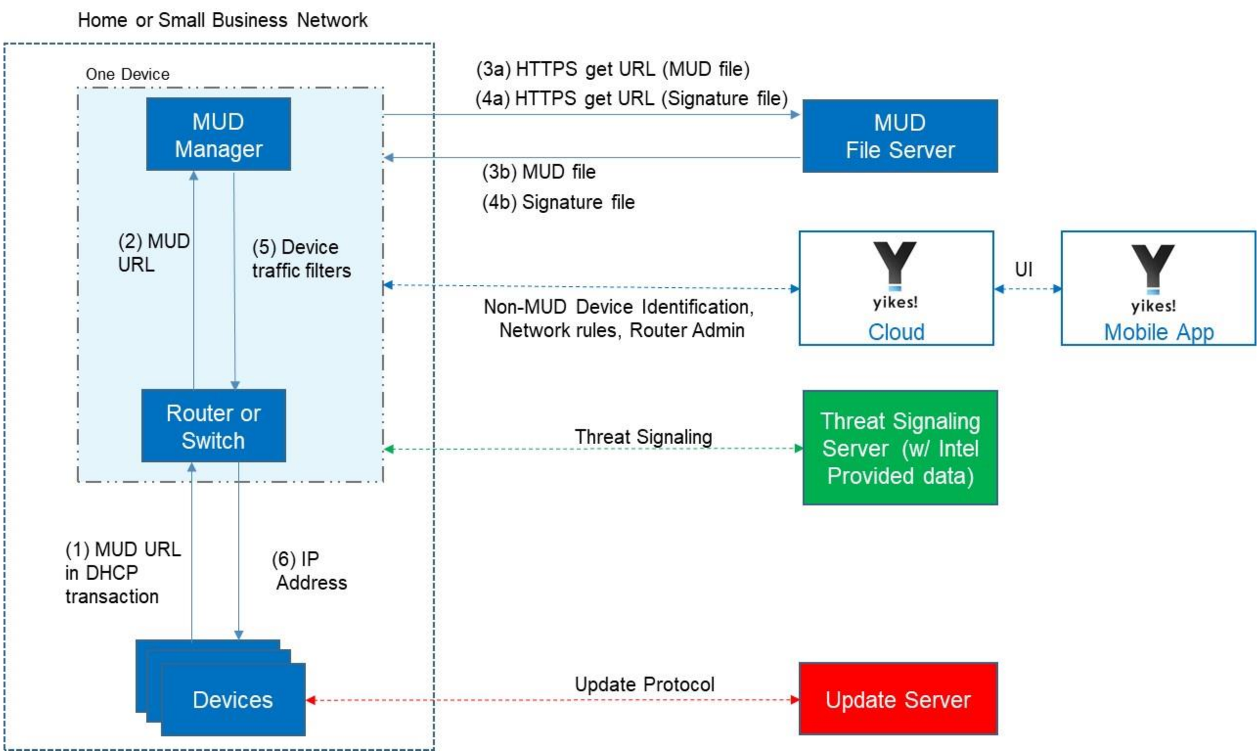
MasterPeace Solutions Ltd is a cybersecurity company in Columbia, Maryland, that focuses on serving federal intelligence community agencies. MasterPeace also operates the MasterPeace LaunchPad start-up studio, chartered with launching cyber-oriented technology product companies. A current LaunchPad start-up 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 to by Wi-Fi-enabled devices, and managed by individuals who possess relatively little IT or cyber background compared with enterprise IT and cyber teams.

A.1.1 Yikes!

Yikes! is a commercial router/cloud service solution focused on consumer and small-business markets. Yikes! starts by thinking about networking differently and isolating every device on the network. As devices are added to a Yikes! home/small-business network, Yikes! leverages a cloud-based process to automatically identify and categorize devices and, based on categories, enables specific internet access (north/south [N/S]) and internal network access to specific devices (east/west [E/W]). Yikes! also provides a mobile application to allow users further fine-grained device filtering control. Yikes! includes a MUD manager that is compliant with the IETF MUD Specification. MUD rules for MUD-enabled devices are enforced automatically.

Figure A-1 depicts a deployment of Yikes! and required related components. Yikes! is designed to run as a router with a connection to the Yikes! cloud and managed via the Yikes! mobile application.

Figure A-1 Yikes! Architecture



A.1.2 Yikes! Components

A.1.2.1 Yikes! Router

The Yikes! router initially isolates all devices connected to the router from all other devices on the network. 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 services. The Yikes! router receives from the Yikes! cloud service rules for N/S and E/W filtering based on the Yikes! cloud processing (see Yikes! Cloud) and custom user settings (see Yikes! App). The Yikes! router also handles IoT devices that emit MUD URLs via DHCP. Yikes! reads these MUD URLs from the device, retrieves MUD files associated with those URLs, and configures the traffic filters (access control lists) in the router to enforce the communication limitations specified in the MUD file for each device.

A.1.2.2 Yikes! Cloud

The Yikes! cloud uses proprietary techniques and machine learning to analyze the DHCP header, MAC address, operating system, and connection characteristics provided by the Yikes! router to automatically classify the device, including Make, Model, and Yikes! Device category. Yikes! has a comprehensive list of categories that includes these examples:

- Mobile: Phone, Tablet, eBook, Smart Watch, Wearable, Car
- Home & Office: Computer, Laptop, Printer, IP Phone, Scanner
- Smart Home: IP Camera, Smart Device, Smart Plug, Light, Voice Assistant, Thermostat, Doorbell, Baby Monitor
- Network: Router, Wi-Fi Extender
- Server: NAS, Server
- Engineering: Raspberry Pi, Arduino

The Yikes! cloud then uses the Yikes! Category to define specific E/W 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 N/S filtering rules for that device. The E/W and N/S rules are then returned to the Yikes! router for local enforcement.

The Yikes! cloud also provides information about the device, its categorization, and filtering rules to the Yikes! application (see Yikes! App). This information is presented to the user, and the user can make specific changes. These changes are also provided to the Yikes! router for enforcement.

A.1.2.3 Yikes! Application

The Yikes! application is a mobile application (available from the iPhone App Store). It communicates with the Yikes! cloud, receiving information about the Yikes! router and all the devices connected to the router, including Yikes! device identification and categorization information and associated N/S and E/W network rules. The Yikes! application allows users to override the automated device information and change the filtering rules. User changes are communicated to the Yikes! cloud and then provided to the Yikes! router for enforcement.

A.1.3 Requirements

The Yikes! router is a self-contained router, Wi-Fi access point, and firewall that communicates locally with Wi-Fi devices and wired devices. Yikes! requires wide area network (WAN) access to the internet that allows MQTT access to the Yikes! cloud service.

A.1.3.1 Hardware

Yikes! provides a customized original equipment manufacturer (OEM) router.

A.1.3.2 Supported Network Architecture

Yikes! currently supports networks that use DHCP. Future work may be done to allow for protocols supporting RADIUS or LLDP compatibility.

A.1.3.3 Required Components Status

Yikes! WAN access to the internet must allow MQTT access to the Yikes! cloud service.

A.1.4 Known Limitations

Here is a list of some important things to be aware of with the current state of Yikes!:

- Does not implement MUD via LLDP
- Does not implement MUD via protocols supporting X.509 certificates
- The Yikes IoT Device Library currently supports automated N/S filtering for only a limited set of IoT devices.

A.2 CableLabs

As the leading Innovation and R&D lab for the cable industry, CableLabs creates global impact through its more than 60 cable-network-operator members around the world, representing approximately 180 million subscribers and roughly 500 million individuals.

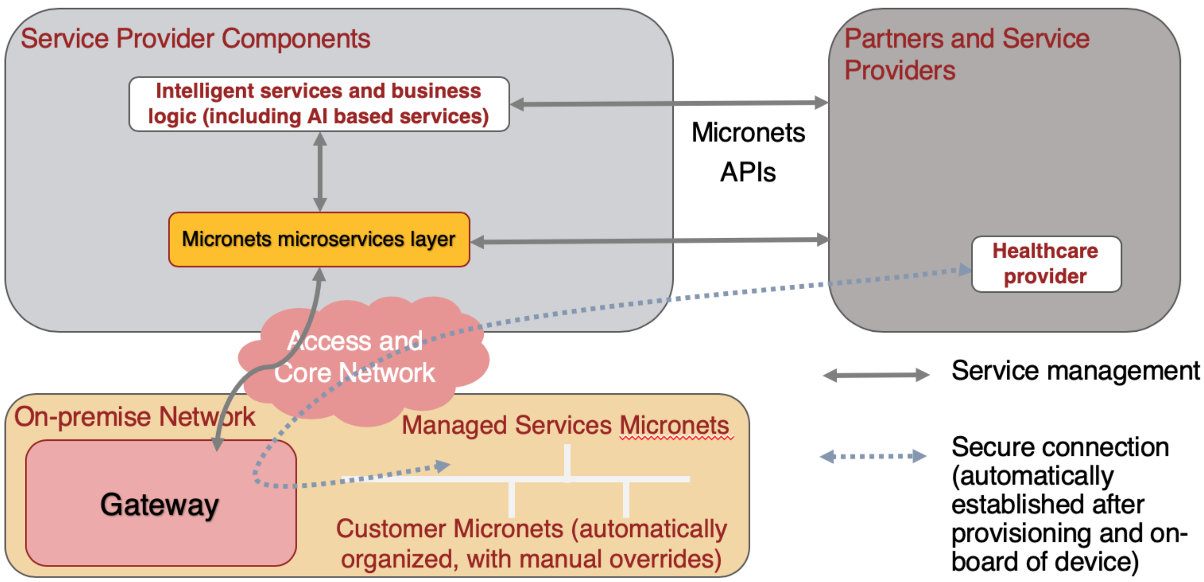
A.2.1 CableLabs Micronets

In November 2018, CableLabs publicly announced Micronets, a next-generation on-premise network platform focused on providing adaptive security for all devices connecting to a residential or small-business network 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 on-premise network platform. 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. More detailed description can be found in CableLabs' Micronets white paper.

In addition to MUD, Micronets incorporates a number of device identity and fingerprinting techniques to enable real-time detection and quarantining of compromised IoT devices, minimizing the risk to other devices on the local network and to the broader internet. CableLabs deployed a Micronets instance in the NCCoE lab based on the open-source reference implementation available on GitHub. CableLabs plans to continue to develop and add new features and functionality to the open-source reference implementation.

As illustrated in Figure A-2 and described in more detail below, Micronets consists of the following architectural components: an intelligent services and business logic layer (e.g., machine learning-based services), a Micronets microservices layer, the on-premise network that includes the Micronets gateway, and the Micronets APIs that are exposed by each of these components.

Figure A-2 Micronets Reference Architecture



A.2.1.1 Intelligent Services and Business Logic

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 combines them into operational decisions that are handed over to the Micronets microservices for execution. It may receive information from various Micronets' microservices (such as the software defined networking [SDN] controller) and in turn use that information to dynamically update the access rules for connected IoT devices. For example, to support devices that do not emit a MUD URL, a "synthetic" MUD file generator and MUD server are provided that can host crowdsourced MUD files that are provided to the Micronets microservices. Another example is an IoT fingerprinting service that allows detection of devices in the network or an artificial intelligence/machine learning-based malware detection service that can provide updated MUD files or access policies based on actively detected threats in the network.

A.2.1.2 Micronets Microservices

This layer hosts a number of network management-related microservices that interact with the on-premise gateway to manage the devices and network connectivity. One of the core microservices, the Micronets Manager, coordinates the entire state of the Micronets-enabled on-premise network. It orchestrates the overall services delivery to the devices and ultimately to the user. Several microservices are engaged and managed by the Micronets Manager like the SDN controller, DHCP/DNS manager, AAA server, and MUD manager.

A.2.1.2.1 MUD Manager

The MUD manager is responsible for retrieving MUD files, processing the MUD files, and generating the rules/policies that are consumed by the Micronets Manager to enforce SDN-based flow rules on the on-premise gateway. The MUD manager can get the files from either the MUD server or any other external location.

A.2.1.3 Micronets Gateway

The core networking component of Micronets is the gateway. The gateway implements an SDN-capable switch that is also integrated with the Wi-Fi access point. The gateway supports connectivity for both wired and wireless components.

A.2.1.3.1 Supported Hardware

The gateway components are implemented on an Ubuntu 16.04-based next unit of computing (NUC) with the Atheros AR9462 Wi-Fi chipset. A port of the current components on a Linksys WRT1900ACS based on OpenWRT is in progress.

A.2.1.4 On-Premise Micronets

The Micronets gateway is responsible for creation and enforcement of the Micronets. Each Micronet represents a distinct trust domain and at the minimum represents a distinct IP subnet.

A.2.1.4.1 MUD-Driven Policies

The Micronets definition and the device allocation within a given Micronets are governed by the Micronets manager and are driven by specific policies. A MUD-based policy drives the allocation of devices into specific Micronets.

A.2.1.4.2 Customer Micronets

Customers will acquire and connect their own devices. They may even integrate entire service-oriented networks, such as a smart home lighting system. 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.

A.2.1.5 Micronets API Framework

Each component (the microservices 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 provision and deliver specific services that the customer has requested.

Appendix B List of Acronyms

2FA	Two-factor Authentication
AAA	Authentication, Authorization, and Accounting
ACL	Access Control List
COA	Change of Authorization
CoAP	Constrained Application Protocol
CRADA	Cooperative Research and Development Agreement
Cybersecurity Framework	NIST Framework for Improving Critical Infrastructure Cybersecurity
DACL	Dynamic Access Control List
DB	Database
DDoS	Distributed Denial of Service
Devkit	Development Kit
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
FIPS	Federal Information Processing Standard
FQDN	Fully Qualified Domain Name
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
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
IT	Information Technology
ITL	NIST's Information Technology Laboratory
LAN	Local Area Network
LED	Light-Emitting Diode
LLDP	Link Layer Discovery Protocol (IEEE 802.1AB)
MAB	MAC Authentication Bypass
MAC	Media Access Control
MQTT	Message Queuing Telemetry Transport
MUD	Manufacturer Usage Description
NAS	Network Address Server
NAT	Network Address Translation
NCCoE	National Cybersecurity Center of Excellence
NIST	National Institute of Standards and Technology
NISTIR	NIST Interagency/Internal Report
OS	Operating System
PC	Personal Computer

PEP	Policy Enforcement Point
PoE	Power over Ethernet
RADIUS	Remote Authentication Dial-In User Service
RFC	Request for Comments
RMF	Risk Management Framework
SP	Special Publication
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
URL	Uniform Resource Locator
VLAN	Virtual Local Area Network
WPA2	Wi-Fi Protected Access 2 Security Certificate Protocol (IEEE 802.11i-2004 standard)
WPA3	Wi-Fi Protected Access 3 Security Certificate protocol
YANG	Yet Another Next Generation

1557 Appendix C Definitions

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/Internal Report 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 (NIST Interagency/Internal Report 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 is capable of emitting a MUD uniform resource locator (URL) in compliance with the MUD specification
Network Address Translation	A function by which internet protocol (IP) 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)
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 NIST Interagency/Internal Report 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 (OSI) 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 (NIST Interagency/Internal Report 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 (FIPS 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. Updates are often provided by the software publisher 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 D Functional Evaluation Plan

The functional evaluation plan describes the test cases conducted.

D.1 Build 1

Functional evaluations of the Internet of Things (IoT) example implementations, as constructed in the Mitigating IoT-Based DDoS lab, were conducted to verify they meet their requirements. Figure 4-6 details the lab environment. Additionally, the architecture shown in Figure 4-1 is used as a guiding principle for this test plan.

It is assumed that prior to testing the example implementation, all communication paths to the IoT devices on the network are open and could potentially be used to attack systems on the internet. It is also assumed that for traffic to be sent to/from one IoT device to another that traffic must pass through the router/switch.

The MUD Specification defines three methods for an IoT device to emit a MUD URL: DHCP, LLDP, and X.509 extensions. This functional evaluation plan shows the methods that were leveraged in the example implementation–emission via DHCP and LLDP.

D.1.1 Build 1 Requirements

Each functional test case is designed to verify that the example implementation meets a specific set of requirements. These requirements are closely aligned to the order of operations in the Manufacturer Usage Description (MUD) Specification (RFC 8520). These requirements are listed in Table D-1. Each of these requirements contains two separate tests, one using IPv4, and one using IPv6 for devices that support IPv6. At the time of testing, the IPv6 functionality was not fully supported by the implementation and was not evaluated. The names of the tests in which each requirement is tested are listed in the right-most column of Table D-1. Tests that end with the suffix "v4" are those in which IPv4 addressing is used; tests that end with the suffix "v6" are those in which IPv6 addressing is used.

Table D-1: Functional IoT Use Case Requirements

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR 1	The IoT DDoS example implementation shall include a MUD-enabled **IoT device that can emit a MUD URL.**			IoT-1-v4, IoT-1-v6, IoT-14-v4, IoT-14-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR 1.a		Upon initialization, the MUD-enabled IoT device shall broadcast a DHCP message on the network, including at most one **MUD URL, in https scheme, within the DHCP transaction.**		IoT-1-v4, IoT-1-v6, IoT-14-v4, IoT-14-v6
CR-1.a.1			The DHCP server shall be able to receive **DHCPv4 DISCOVER and/or REQUEST with IANA code 161** (OPTION_MUD_URL_V4) from the MUD-enabled IoT device.	IoT-1-v4, IoT-14-v4
CR-1.a.2			The DHCP server shall be able to receive **DHCPv6 Solicit and/or Request with IANA code 112** (OPTION_MUD_URL_V6) from the MUD-enabled IoT device.	IoT-1-v6, IoT-14-v6
CR-1.b		Upon initialization, the MUD-enabled IoT device shall **emit the MUD URL as an LLDP extension.**		IoT-1-v4, IoT-1-v6, IoT-14-v4, IoT-14-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-1.b.1			The network service shall be able to **process** the MUD URL that is received as an LLDP e**xtension.**	IoT-1-v4, IoT-1-v6, IoT-14-v4, IoT-14-v6
CR 2	The IoT DDoS example implementation shall include the capability for the extracted MUD URL **to be forwarded to a MUD manager.**			IoT-1-v4, IoT-1-v6
CR-2.a		The DHCP server shall **assign an IP address lease** to the MUD-enabled IoT device.		IoT-1-v4, IoT-1-v6
CR-2.a.1			The MUD-enabled IoT device shall **receive the IP address.**	IoT-1-v4, IoT-1-v6
CR-2.b		**The DHCP server shall** receive the DHCP message and **extract the MUD URL, which is then passed to the MUD manager.**		IoT-1-v4, IoT-1-v6
CR-2.b.1			**The MUD manager shall receive the MUD URL.**	IoT-1-v4, IoT-1-v6
CR 3	The IoT DDoS example implementation shall include a			IoT-1-v4, IoT-1-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
	MUD manager that is capable of requesting a MUD file and signature from a MUD file server.			
CR-3.a		The MUD manager shall use the "GET" method (RFC 7231) to **request MUD and signature files** (per RFC 7230) from the MUD file server and is able to **validate the MUD file server's TLS certificate** by using the rules in RFC 2818.		IoT-1-v4, IoT-1-v6
CR-3.a.1			**The MUD file server shall receive the https request from the MUD manager.**	IoT-1-v4, IoT-1-v6
CR-3.b		**The MUD manager** shall use the GET method (RFC 7231) to request MUD and signature files (per RFC 7230) from the MUD file server but it **is not able to validate the MUD file server's TLS certificate** by using the rules in RFC 2818.		IoT-2-v4, IoT-2-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-3.b.1			**The MUD manager shall drop the connection** to the MUD file server.	IoT-2-v4, IoT-2-v6
CR-3.b.2			**The MUD manager shall send locally defined policy to the router or switch** that handles whether to allow or block traffic to/from the MUD-enabled IoT device.	IoT-2-v4, IoT-2-v6
CR 4	The IoT DDoS example implementation shall include a **MUD file server that is capable of serving a MUD file and signature to the MUD manager**.			IoT-1-v4, IoT-1-v6
CR-4.a		**The MUD file server shall serve the file and signature to the MUD manager, and the MUD manager shall check to determine whether the certificate used to sign the MUD file** (signed using distinguished encoding rules (DER)-encoded Cryptographic Message Syntax [CMS]		IoT-1-v4, IoT-1-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
		[RFC 5652) was valid at the time of signing, i.e., the **certificate had not expired.**		
CR-4.b		**The MUD file server shall serve the file and signature to the MUD manager, and the MUD manager shall check to determine whether the certificate used to sign the MUD file** was valid at the time of signing, i.e., the **certificate had already expired when it was used to sign the MUD file.**		IoT-3-v4, IoT-3-v6
CR-4.b.1			The MUD manager shall cease to process the MUD file.	IoT-3-v4, IoT-3-v6
CR-4.b.2			The MUD manager shall send locally defined policy to the router or switch that handles whether to allow or block traffic to/from the MUD-enabled IoT device.	IoT-3-v4, IoT-3-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR 5	The IoT DDoS example implementation shall include a **MUD manager** that **is capable of translating local network configurations based on the MUD file.**			IoT-1-v4, IoT-1-v6
CR-5.a		**The MUD manager shall successfully validate the signature of the MUD file.**		IoT-1-v4, IoT-1-v6
CR-5.a.1			The MUD manager, after validation of the MUD file signature, shall **check for an existing MUD file, compare both files, and translate abstractions in the MUD file to router or switch configurations** if the new MUD file is an update.	IoT-1-v4, IoT-1-v6
CR-5.a.2			The MUD manager shall **cache** this newly received MUD file.	IoT-12-v4, IoT-12-v6
CR-5.b		The MUD manager shall attempt to validate the signature of the **MUD file,** but the **signature validation fails** (even though the		IoT-4-v4, IoT-4-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
		certificate that had been used to create the signature had not been expired at the time of signing, i.e., the signature is invalid for a different reason).		
CR-5.b.1			**The MUD manager shall cease processing the MUD file.**	IoT-4-v4, IoT-4-v6
CR 5.b.2			**The MUD manager shall send locally defined policy to the router or switch** that handles whether to allow or block traffic to/from the MUD-enabled IoT device.	IoT-4-v4, IoT-4-v6
CR 6	The IoT DDoS example implementation shall include a **MUD manager that is capable of configuring the MUD PEP,** i.e., the router or switch nearest the MUD-enabled IoT device that emitted the URL.			IoT-1-v4, IoT-1-v6
CR-6.a		**The MUD manager shall install a router configuration** on the router or switch nearest the MUD-enabled		IoT-1-v4, IoT-1-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
		IoT device that emitted the URL.		
CR-6.a.1			**The router or switch shall have been configured to enforce the route filter sent by the MUD manager.**	IoT-1-v4, IoT-1-v6
CR 7	The IoT DDoS example implementation shall **allow the MUD-enabled IoT device to communicate with approved internet services in the MUD file.**			IoT-5-v4, IoT-5-v6
CR-7.a		The MUD-enabled IoT device shall attempt to **initiate outbound traffic to approved internet services.**		IoT-5-v4, IoT-5-v6
CR-7.a.1			The router or switch shall receive the attempt and shall **allow it to pass** based on the filters from the MUD file.	IoT-5-v4, IoT-5-v6
CR-7.b		An approved **internet service shall attempt**		IoT-5-v4, IoT-5-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
		to initiate a connection to the MUD-enabled IoT device.		
CR-7.b.1			The router or switch shall receive the attempt and shall **allow it to pass** based on the filters from the MUD file.	IoT-5-v4, IoT-5-v6
CR 8	The IoT DDoS example implementation shall **deny communications from a MUD-enabled IoT device to unapproved internet services** (i.e., services that are denied by virtue of not being explicitly approved).			IoT-5-v4, IoT-5-v6
CR-8.a		The MUD-enabled IoT device shall **attempt to initiate outbound traffic to unapproved** (implicitly denied) **internet services.**		IoT-5-v4, IoT-5-v6
CR-8.a.1			**The router or switch shall receive the attempt and** shall **deny it** based on the filters from the MUD file.	IoT-5-v4, IoT-5-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-8.b		**An unapproved** (implicitly denied) **internet service shall attempt to initiate a connection to the MUD-enabled IoT device.**		IoT-5-v4, IoT-5-v6
CR-8.b.1			**The router or switch shall receive the attempt and shall deny it** based on the filters from the MUD file.	IoT-5-v4, IoT-5-v6
CR-8.c		The MUD-enabled IoT device shall initiate communications to an internet service that is **approved to initiate communication with the MUD-enabled device but not approved to receive communications initiated by the MUD-enabled device.**		IoT-5-v4, IoT-5-v6
CR-8.c.1			**The router or switch shall receive the attempt and shall deny it** based on the filters from the MUD file.	IoT-5-v4, IoT-5-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-8.d		An internet service shall initiate communication to a MUD-enabled device that is **approved to initiate communications with the internet service but that is not approved to receive communications initiated by the internet service.**		IoT-5-v4, IoT-5-v6
CR-8.d.1			**The router or switch shall receive the attempt and shall deny it** based on the filters from the MUD file.	IoT-5-v4, IoT-5-v6
CR 9	The IoT DDoS example implementation shall **allow the MUD-enabled IoT device to communicate laterally with devices that are approved** in the MUD file.			IoT-6-v4, IoT-6-v6
CR-9.a		The MUD-enabled IoT device shall **attempt to initiate lateral traffic to approved devices.**		IoT-6-v4, IoT-6-v6
CR-9.a.1			**The router or switch shall receive the at-**	IoT-6-v4, IoT-6-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
			tempt and shall allow it to pass based on the filters from the MUD file.	
CR-9.b		An approved device **shall attempt to initiate a lateral connection to the MUD-enabled IoT device.**		IoT-6-v4, IoT-6-v6
CR-9.b.1			**The router or switch shall receive the attempt and shall allow it to pass** based on the filters from the MUD file.	IoT-6-v4, IoT-6-v6
CR 10	The IoT DDoS example implementation shall **deny lateral communications from MUD-enabled IoT device to devices that are not approved** in the MUD file (i.e., devices that are implicitly denied by virtue of not being explicitly approved).			IoT-6-v4, IoT-6-v6
CR-10.a		The MUD-enabled IoT device shall **attempt to initiate lateral traffic to unapproved** (implicitly denied) **devices.**		IoT-6-v4, IoT-6-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-10.a.1			**The router or switch shall receive the attempt and shall deny it** based on the filters from the MUD file.	IoT-6-v4, IoT-6-v6
CR-10.b		**An unapproved** (implicitly denied) **device shall attempt to initiate a lateral connection** to the MUD-enabled IoT device.		IoT-6-v4, IoT-6-v6
CR-10.b.1			**The router or switch shall receive the attempt and shall deny it** based on the filters from the MUD file.	IoT-6-v4, IoT-6-v6
CR 11	The IoT DDoS example implementation shall **remove the implemented policy when the MUD-enabled IoT device changes DHCP state.**			IoT-7-v4, IoT-7-v6
CR-11.a		The MUD-enabled IoT **device shall explicitly release the IP address lease** (i.e., it sends a DHCP release message to the DHCP server).		IoT-7-v4, IoT-7-v6
CR-11.a.1			**The DHCP server shall notify the MUD**	IoT-7-v4, IoT-7-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
			manager that the device's IP address lease has been released.	
CR-11.a.2			**The MUD manager shall remove all policies** associated with the disconnected IoT device that had been configured on the MUD PEP router/switch.	IoT-7-v4, IoT-7-v6
CR-11.b		The MUD-enabled IoT **device's IP address lease shall expire.**		IoT-8-v4, IoT-8-v6
CR-11.b.1			**The DHCP server shall notify the MUD manager that the device's IP address lease has expired.**	IoT-8-v4, IoT-8-v6
CR-11.b.2			**The MUD manager shall remove all policies** associated with the affected IoT device that had been configured on the MUD PEP router/switch.	IoT-8-v4, IoT-8-v6
CR 12	The IoT DDoS example implementation shall include a			IoT-9-v4, IoT-9-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
	router or switch that is capable of receiving and implementing threat signaling information from a threat signaling server that includes "blacklisted" URLs and hosts that are not considered to be trusted. The router/switch shall be capable of being configured so it **applies threat signaling rules to both MUD-enabled and non-MUD-capable devices, depending on local policy.**			
CR-12.a		The router or switch shall receive **threat signals** from the threat signaling server and translate them into appropriate permit/deny configuration rules that **will pertain to all devices (both MUD-enabled and non-MUD-capable).**		IoT-9-v4, IoT-9-v6
CR-12.a.1			**A non-MUD-capable IoT device shall attempt to initiate outbound traffic to a blacklisted internet URL. The router or switch shall receive**	IoT-9-v4, IoT-9-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
			the attempt and shall deny it based on the filters from the threat signaling server.	
CR-12.a.2			**There shall be an attempt to initiate a connection from a blacklisted internet URL to a non-MUD-capable IoT device. The router or switch shall receive the attempt and shall deny it** based on the filters from the threat signaling server.	IoT-9-v4, IoT-9-v6
CR-12.a.3			**The MUD-enabled IoT device shall attempt to initiate outbound traffic to an internet URL that is explicitly permitted in its MUD file but that has been blacklisted by the threat signaling service. The router or switch shall receive the attempts and shall deny it** based on the filters from the threat signaling server.	IoT-9-v4, IoT-9-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-12.a.4			**A blacklisted internet URL that is explicitly permitted in a MUD-enabled device's MUD file shall attempt to send traffic to the MUD-enabled device. The router or switch shall receive the attempt and shall deny it** based on the filters from the threat signaling server.	IoT-9-v4, IoT-9-v6
CR 13	The IoT DDoS example implementation shall include a **router or switch that is capable of receiving and implementing threat signaling information from a threat signaling server that includes "blacklisted" URLs and hosts that are not considered to be trusted. The router/switch shall be capable of being configured so that it applies threat signaling rules only to non-MUD-capable devices, based on local policy.**			IoT-10-v4, IoT-10-v6
CR-13.a		The router or switch shall receive **threat signals** from the threat signaling server		IoT-10-v4, IoT-10-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
		and translate them into appropriate permit/deny configuration rules that **will pertain only to non-MUD-capable devices.**		
CR-13.a.1			**The non-MUD-capable IoT device shall attempt to initiate outbound traffic to blacklisted internet URL. The router or switch shall receive the attempt and shall deny it** based on the filters from the threat signaling server.	IoT-10-v4, IoT-10-v6
CR-13.a.2			**There shall be an attempt to initiate a connection from a blacklisted internet URL to the non-MUD-capable IoT device. The router or switch shall receive the attempt and shall deny it** based on the filters from the threat signaling server.	IoT-10-v4, IoT-10-v6
CR-13.a.3			**The MUD-enabled IoT device shall at-**	IoT-10-v4, IoT-10-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
			tempt to initiate outbound traffic to an internet URL that is explicitly permitted in its MUD file but that has been blacklisted by the threat signaling service. The router or switch shall receive the attempt and shall permit it because the filters from the threat signaling server do not apply to MUD-enabled devices.	
CR-13.a.4			**A device from a blacklisted internet URL that is explicitly permitted in a MUD-enabled device's MUD file shall attempt to send traffic to the MUD-enabled IoT device. The router or switch shall receive the attempt and shall permit it** because the filters from the threat signaling server do not apply to MUD-enabled devices.	IoT-10-v4, IoT-10-v6
CR-14	The IoT DDoS example implementation shall include the			IoT-11

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
	capability to handle life-cycle changes, such as decommissioning IoT devices.			
CR-14.a		MUD-enabled IoT device **manufacturers should provide a final MUD file for devices no longer being supported.**		IoT-11
CR-14.a.1			The MUD manager shall use the final MUD file to configure traffic filters for the IoT device per CR 1-6.	IoT-11
CR-15	The IoT DDoS example implementation shall include a **MUD manager that uses a cached MUD file rather than retrieve a new one if the cache-validity time period has not yet elapsed** for the MUD file indicated by the MUD URL**.**			IoT-12-v4, IoT-12-v6
CR-15.a		The MUD manager shall check if the file associated with the **MUD URL is present in its cache** and shall determine that it is.		IoT-12-v4, IoT-12-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-15.a.1			The MUD manager shall **check whether the amount of time that has elapsed since the cached file was retrieved is less than or equal to the number of hours in the cache-validity value for this MUD file.** If so, the MUD manager shall apply the contents of the cached MUD file.	IoT-12-v4, IoT-12-v6
CR-15.a.2			The MUD manager **shall check whether the amount of time that has elapsed since the cached file was retrieved is greater than the number of hours in the cache-validity value for this MUD file.** If so, the MUD manager may (but does not have to) fetch a new file by using the MUD URL received.	IoT-12-v4, IoT-12-v6
CR-16	The IoT DDoS example implementation shall include a visibility component that is able to **detect, identify, categorize, and monitor the status**			IoT-13-v4, IoT-13-v6

Capability Requirement (CR) ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
	of IoT devices that are on the network.			
CR-16.a		The visibility component shall **detect and identify** the attributes and category of a newly connected IoT device.		IoT-13-v4, IoT-13-v6
CR-16.a.1			The visibility component shall **monitor the status** of the IoT device (e.g., notice if the device goes offline).	IoT-13-v4, IoT-13-v6

D.1.2 Build 1 Test Cases

Each test case consists of multiple fields that collectively identify the goal of the test, the specifics required to implement the test, and how to assess the results of the test. Table D-2 describes each field in any given test case.

Table D-2: Test Case Fields

Test Case Field	Description
Parent Requirement	Identifies the top-level requirement or the series of top-level requirements leading to the testable requirement.
Testable Requirement	Guides the definition of the remainder of the test case fields. Specifies the capability to be evaluated.

Test Case Field	Description
Description	Describes the objective of the test case.
Associated Test Cases	In some instances, a test case may be based on the outcome of (an)other test case(s). For example, analysis-based test cases produce a result that is verifiable through various means (e.g., log entries, reports, and alerts).
Associated Cybersecurity Framework Subcategories	Lists the Cybersecurity Framework Subcategories addressed by the test case.
IoT Device(s) Under Test	Text identifying which IoT device is being connected to the network in this test
MUD File(s) Used	Name of MUD file(s) used
Preconditions	The starting state of the test case. Preconditions indicate various starting state items, such as a specific capability configuration required or specific protocol and content.
Procedure	The step-by-step actions required to implement the test case. A procedure may consist of a single sequence of steps or multiple sequences of steps (with delineation) to indicate variations in the test procedure.
Expected Results	The expected results for each variation in the test procedure
Actual Results	The observed results
Overall Results	The overall result of the test as pass/fail

The remainder of this section contains the test cases that were used to verify that the example implementations met the requirements listed in Table D-1. Each test case is presented in the format described in Table D-2. It is assumed that all tests are run on the lab architecture shown in the figure above. It is further assumed that for all tests, the MUD PEP is the router/switch that is nearest the IoT device in question. Only the IPv4 versions of each test are listed explicitly below. For each test that has both an IPv4 and an IPv6 version, the IPv4 version of the test, IoT-n-v4, is identical to the IPv6 version of the test, IoT-n-v6, except

- IoT-n-v6 devices are configured to use IPv6, whereas IoT-n-v4 devices are configured to use IPv4.
- IoT-n-v6 devices are configured to use DHCPv6, whereas IoT-n-v4 devices are configured to use DHCPv4.
- The IoT-n-v6 DHCPv6 message that is emitted includes the MUD URL option that uses IANA code 112, whereas the IoT-n-v4 DHCPv4 message that is emitted includes the MUD URL option that uses IANA code 161.

In addition to the lab setup that is depicted in Figure 4-6, the following hosts and web servers must also be set up and available to support the tests defined below. On the local network where the MUD manager and IoT devices are located, hosts with the following names must exist and be reachable from an IoT device that is plugged into the local network:

- *unnamed-host* (i.e., a local host that is not from the same manufacturer as the IoT device in question and whose MUD URL is not explicitly mentioned in the MUD file of the IoT device in question as denoting a class of devices with which the IoT device in question is permitted to communicate. For example, if device A's MUD file says that it may communicate locally with devices that have MUD URLs *www.zzz.com* and *www.xxx.com*, then a local host that has a MUD file of *www.qqq.com* could be *unnamed-host*.)
- *anyhost-to* (i.e., a local host to which the IoT device in question is permitted to initiate communications but not vice versa)
- *anyhost-from* (i.e., a local host that is permitted to initiate communications to the IoT device but not vice versa)
- *same-manufacturer-host* (i.e., a local host that is from the same manufacturer as the IoT device in question. For example, if device A's MUD file is found at URL *www.aaa.com* and device B's MUD file is also found at URL *www.aaa.com*, then device B could be *same-manufacturer-host*.)

On the internet (i.e., outside the local network), the following web servers must be set up and reachable from an IoT device that is plugged into the local network:

- *https://yes-permit-to.com* (i.e., an internet location to which the IoT device in question is permitted to initiate communications, but not vice versa)
- *https://yes-permit-from.com* (i.e., an internet location that is permitted to initiate communications to the IoT device but not vice versa)
- *https://unnamed.com* (i.e., an internet location with which the IoT device is not permitted to communicate)

Please note all the listed capabilities will be included in a single MUD file in which we will highlight the specific capability being evaluated in the respective test.

D.1.2.1 Test Case IoT-1-v4

Table D-3: Test Case IoT-1-v4

Test Case Field	Description
Parent Requirements	(CR 1) The IoT DDoS example implementation shall include a MUD-enabled IoT device that is capable of emitting a MUD URL. (CR 2) The IoT DDoS example implementation shall include the capability for the extracted MUD URL to be forwarded to a MUD manager. (CR 3) The IoT DDoS example implementation shall include a MUD manager that is capable of requesting a MUD file and signature from the MUD file server. (CR 4) The IoT DDoS example implementation shall include a MUD file server that is capable of serving a MUD file and signature to the MUD manager. (CR 5) The IoT DDoS example implementation shall include a MUD manager that is capable of translating local network configurations based on the MUD file. (CR 6) The IoT DDoS example implementation shall include a MUD manager that is capable of configuring the router or switch nearest the MUD-enabled IoT device that emitted the URL.
Testable Requirements	(CR 1.a) Upon initialization, the MUD-enabled IoT device shall broadcast a DHCP message on the network, including at most one MUD URL, in https scheme, within the DHCP transaction. (CR 1.a.1) DHCP server shall be able to receive DHCPv4 DISCOVER and /or REQUEST with IANA code 161 (OPTION_MUD_URL_V4) from the MUD-enabled IoT device (NOTE: Test IoT-1-v6 does not test this requirement; instead, it tests CR 1.a.2, which pertains to DHCPv6 rather than DHCPv4.) OR (CR 1.b) Upon initialization, the MUD-enabled IoT device shall emit MUD URL as an LLDP extension.

Test Case Field	Description
	(CR 1.b.1) The network service shall be able to process the MUD URL that is received as an LLDP extension. (CR 2.a) The DHCP server shall assign an IP address lease to the MUD-enabled IoT device. (CR 2.a.1) MUD-enabled IoT device shall receive the IP address. (CR 2.b) The DHCP server shall receive the DHCP message and extract the MUD URL, which is then passed to the MUD manager. (CR 2.b.1) The MUD manager shall receive the MUD URL. (CR 3.a) The MUD manager shall use the GET method (RFC 7231) to request MUD and signature files (per RFC 7230) from the MUD file server and is able to validate the MUD file server's TLS certificate by using the rules in RFC 2818. (CR 3.a.1) MUD file server shall receive the https request from the MUD manager. (CR 4.a) The MUD file server shall serve the file and signature to the MUD manager, and the MUD manager shall check to determine whether the certificate used to sign the MUD file (signed using DER-encoded CMS [RFC 5652]) was valid at the time of signing, i.e., the certificate had not expired. (CR 5.a) The MUD manager shall successfully validate the signature of the MUD file. (CR 5.a.1) The MUD manager, after validation of the MUD file signature, shall check for an existing MUD file, compare both files, and translate abstractions in the MUD file to router or switch configurations if the new MUD file is an update. (CR 6.a) The MUD manager shall install a router configuration on the router or switch nearest the MUD-enabled IoT device that emitted the URL. (CR 6.a.1) The router or switch shall have been configured to enforce the route filter sent by the MUD manager.
Description	Shows that, upon connection to the network, a MUD-enabled IoT device used in the IoT DDoS example implementation has its MUD PEP router/switch automatically configured to enforce the route filtering that is described in the device's MUD file, assuming the MUD file has a

Test Case Field	Description
	valid signature and is served from a MUD file server that has a valid TLS certificate.
Associated Test Cases	N/A
Associated Cybersecurity Framework Subcategories	ID.AM-1, ID.AM-2, ID.AM-3, PR.DS-5, DE.AE-1, PR.AC-4, PR.AC-5, PR.IP-1, PR.IP-3, PR.PT-3, PR.DS-2
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	ciscopi2.json
Preconditions	1. All devices have been configured to use IPv4, and the IoT device under test has been configured to emit a DHCPv4 message that includes the URL of its MUD file by using the DHCPv4 MUD URL option (IANA code 161). (Note that in the v6 version of this test, IPv6, DHCPv6, and IANA code 112 will be used.) 2. This MUD file is not currently cached at the MUD manager. 3. The device's MUD file has a valid signature that was signed by a certificate that had not yet expired, and it is being hosted on a MUD file server that has a valid TLS certificate. 4. The MUD PEP router/switch does not yet have any configuration settings pertaining to the IoT device being used in the test. 5. The MUD file for the IoT device being used in the test is identical to the MUD file provided in section D.1.3.
Procedure	Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. Also verify that the MUD file of the IoT device to be used is not currently cached at the MUD manager.

Test Case Field	Description
	Power-on the IoT device and connect it to the test network. This should set in motion the following series of steps, which should occur automatically: 1. IoT device automatically emits a MUD URL in one of the following methods: a. DHCPv4 message containing the device's MUD URL (IANA code 161) (Note that in the v6 version of this test, IPv6, DHCPv6, and IANA code 112 will be used.) b. LLDP message containing the device's MUD URL in its extension 2. Corresponding service is responsible for the following actions: a. DHCP server receives the DHCP message containing the IoT device's MUD URL. b. LLDP server receives LLDP advertisement containing the IoT device's MUD URL. 3. Respective service (LLDP or DHCP) extracts the MUD URL. 4. The MUD URL is then provided to the MUD manager. 5. The MUD manager automatically contacts the MUD file server that is located using the MUD URL, verifies that it has a valid TLS certificate, requests and receives the MUD file and signature from the MUD file server, validates the MUD file's signature, and translates the MUD file's contents into appropriate route filtering rules and installs these rules onto the MUD PEP for the IoT device in question so that this router/switch is now configured to enforce the policies specified in the MUD file. 6. DHCP server offers an IP address lease to the newly connected IoT device. 7. The IoT device requests this IP address lease, which the DHCP server acknowledges.
Expected Results	The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to enforce the policies specified in the IoT device's MUD file. The expected configuration should resemble the following details: Extended IP access list mud-81726-v4fr.in

Test Case Field	Description
	10 permit tcp any host 192.168.4.7 eq www ack syn 20 permit tcp any host 192.168.10.104 eq www 30 permit tcp any host 192.168.10.105 eq www 50 permit tcp any 192.168.10.0 0.0.0.255 eq www 60 permit tcp any 192.168.13.0 0.0.0.255 eq www 70 permit tcp any 192.168.14.0 0.0.0.255 eq www 80 permit tcp any eq 22 any 81 permit udp any eq bootpc any eq bootps 82 permit udp any any eq domain 83 deny ip any any All protocol exchanges described in steps 1–4 above are expected to occur and can be viewed via Wireshark if desired. If the router/switch does not get configured in accordance with the MUD file, each exchange of DHCP and MUD-related protocol traffic should be viewed on the network via Wireshark to determine which transactions did not proceed as expected, and the observed and absent protocol exchanges should be described here.
Actual Results	**Dynamic access-session on switch:** Build1#sh access-session int g1/0/15 det Interface: GigabitEthernet1/0/15 IIF-ID: 0x1B6BCEA5 MAC Address: b827.ebeb.6c8b IPv6 Address: Unknown IPv4 Address: 192.168.13.9 User-Name: b827ebeb6c8b Status: Authorized Domain: DATA Oper host mode: multi-auth Oper control dir: both Session timeout: N/A Common Session ID: C0A80A02000000A6A9828F06 Acct Session ID: 0x0000003b

Test Case Field	Description
	Handle: 0x2200009c Current Policy: mud-mab-test Server Policies: ACS ACL: mud-81726-v4fr.in Vlan Group: Vlan: 3 Method status list: Method State mab Authc Success **access-list on switch:** Build1#sh access-list mud-81726-v4fr.in Extended IP access list mud-81726-v4fr.in 10 permit tcp any host 192.168.4.7 eq www ack syn 20 permit tcp any host 192.168.10.104 eq www 30 permit tcp any host 192.168.10.105 eq www 50 permit tcp any 192.168.10.0 0.0.0.255 eq www 60 permit tcp any 192.168.13.0 0.0.0.255 eq www 70 permit tcp any 192.168.14.0 0.0.0.255 eq www 80 permit tcp any eq 22 any 81 permit udp any eq bootpc any eq bootps 82 permit udp any any eq domain 83 deny ip any any
Overall Results	Pass

Test Case IoT-1-v6 is identical to test case IoT-1-v4 except that IoT-1-v6 tests requirement CR-1.a.2, whereas IoT-1-v4 tests requirement CR-1.a.1. Hence, as explained above, test IoT-1-v6 it uses IPv6, DHCPv6, and IANA code 112 instead of using IPv4, DHCPv4, and IANA code 161.

1635 ### D.1.2.2 Test Case IoT-2-v4

1636 **Table D-4: Test Case IoT-2-v4**

Test Case Field	Description
Parent Requirement	(CR 3) The IoT DDoS example implementation shall include a MUD manager that is capable of requesting a MUD file and signature from the MUD file server.
Testable Requirement	(CR 3.b) The MUD manager shall use the GET method (RFC 7231) to request MUD and signature files (per RFC 7230) from the MUD file server, but it is not able to validate the MUD file server's TLS certificate using the rules in RFC 2818. (CR 3.b.1) The MUD manager shall drop the connection to the MUD file server. (CR 3.b.2) MUD manager shall send locally defined policy to the router or switch that handles whether to allow or block traffic to/from the MUD-enabled IoT device.
Description	Shows that if a MUD manager is not able to validate the TLS certificate of a MUD file server when trying to retrieve the MUD file for a specific IoT device, the MUD manager will drop the connection to the MUD file server and configure the router/switch according to locally defined policy regarding whether to allow or block traffic to the IoT device in question.
Associated Test Cases	IoT-14-v4 (for the v6 version of this test, IoT-14-v6)
Associated Cybersecurity Framework Subcategories	PR.AC-7
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	ciscopi2.json

Test Case Field	Description
Preconditions	1. All devices have been configured to use IPv4, and the IoT device under test has been configured to emit a DHCPv4 message that includes the URL of its MUD file by using the DHCPv4 MUD URL option (IANA code 161). (Note that in the v6 version of this test, IPv6, DHCPv6, and IANA code 112 will be used.) 2. This MUD file is not currently cached at the MUD Manager. 3. The MUD file server that is hosting the MUD file of the device under test does not have a valid TLS certificate. 4. Local policy has been defined to ensure that if the MUD file for a device is located on a server with an invalid certificate, the router/switch will be configured to deny all communication to/from the device. 5. The MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings with respect to the IoT device being used in the test.
Procedure	Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. Power-on the IoT device and connect it to the test network. This should set in motion the following series of steps, which should occur automatically: 1. IoT device automatically emits a DHCPv4 message containing the device's MUD URL (IANA code 161). (Note that in the v6 version of this test, IPv6, DHCPv6, and IANA code 112 will be used.) 2. DHCP server receives the DHCP message containing the IoT device's MUD URL. 3. DHCP server offers an IP address lease to the newly connected IoT device. 4. The IoT device requests this IP address lease, which the DHCP server acknowledges. 5. DHCP server sends the MUD URL to the MUD manager.

Test Case Field	Description
	6. The MUD manager automatically contacts the MUD file server that is located by using the MUD URL, determines that it does not have a valid TLS certificate, and drops the connection to the MUD file server. 7. The MUD manager configures the router/switch that is closest to the IoT device so that it denies all communications to and from the IoT device.
Expected Results	The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to local policy for communication to/from the IoT device.
Actual Results	***MUDC [STATUS][send_mudfs_request:2005]--> Request URI <*https://mudfileserver/ciscopi2*> </home/mudtester/ca.cert.pem> * Trying 192.168.4.5... * TCP_NODELAY set * Connected to mudfileserver (192.168.4.5) port 443 (#0) * found 1 certificate in /home/mudtester/ca.cert.pem * found 400 certificates in /etc/ssl/certs * ALPN, offering http/1.1 * SSL connection using TLS1.2 / ECDHE_RSA_AES_256_GCM_SHA384 * server certificate verification failed. CAfile: /home/mudtester/ca.cert.pem CRLfile: none * stopped the pause stream! * Closing connection 0 ***MUDC [ERROR][fetch_file:182]--> curl_easy_perform() failed: Peer certificate cannot be authenticated with given CA certificates ***MUDC [INFO][send_mudfs_request:2019]--> Unable to reach MUD fileserver to fetch MUD file. Will try to append .json * Trying 192.168.4.5...

Test Case Field	Description
	* TCP_NODELAY set * Connected to mudfileserver (192.168.4.5) port 443 (#0) * found 1 certificate in /home/mudtester/ca.cert.pem * found 400 certificates in /etc/ssl/certs * ALPN, offering http/1.1 * SSL connection using TLS1.2 / EC-DHE_RSA_AES_256_GCM_SHA384 * server certificate verification failed. CAfile: /home/mudtester/ca.cert.pem CRLfile: none * stopped the pause stream! * Closing connection 0 ***MUDC [ERROR][fetch_file:182]--> curl_easy_perform() failed: Peer certificate cannot be authenticated with given CA certificates ***MUDC [ERROR][send_mudfs_request:2027]--> Unable to reach MUD fileserver to fetch .json file ***MUDC [INFO][mudc_construct_head:135]--> status_code: 204, content_len: 14, extra_headers: (null) ***MUDC [INFO][mudc_construct_head:152]--> HTTP header: HTTP/1.1 204 No Content Content-Length: 14 ***MUDC [INFO][send_error_result:176]--> error from FS ***MUDC [ERROR][send_mudfs_request:2170]--> mudfs_conn failed --- Build1#sho access-session int g1018 det Interface GigabitEthernet1018 IIF-ID 0x181835C2 MAC Address b827.eba7.0533 IPv6 Address Unknown IPv4 Address 192.168.10.106 User-Name b827eba70533

Test Case Field	Description
	Status Authorized Domain DATA Oper host mode multi-auth Oper control dir both Session timeout NA Common Session ID C0A80A02000000CCBDB267F8 Acct Session ID 0x00000046 Handle 0x100000c2 Current Policy mud-mab-test Server Policies Method status list Method State mab Authc Success
Overall Results	Pass

1637 As explained above, test IoT-2-v6 is identical to test IoT-2-v4 except that it uses IPv6, DHCPv6, and IANA
1638 code 112 instead of using IPv4, DHCPv4, and IANA code 161.

1639 D.1.2.3 Test Case IoT-3-v4

1640 **Table D-5: Test Case IoT-3-v4**

Test Case Field	Description
Parent Requirement	(CR 4) The IoT DDoS example implementation shall include a MUD file server that is capable of serving a MUD file and signature to the MUD manager.
Testable Requirement	(CR 4.b) The MUD file server shall serve the file and signature to MUD manager, and the MUD manager shall check to determine whether the certificate used to sign the MUD file was valid at the time of signing, i.e., the certificate had already expired when it was used to sign the MUD file.

Test Case Field	Description
	(CR 4.b.1) The MUD manager shall cease to process the MUD file. (CR 4.b.2) The MUD manager shall send locally defined policy to router or switch that handles whether to allow or block traffic to/from the MUD-enabled IoT device.
Description	Shows that if a MUD file server serves a MUD file with a signature that was created with an expired certificate, the MUD manager will cease processing the MUD file.
Associated Test Cases	IoT-14-v4 (for the v6 version of this test, IoT-14-v6)
Associated Cybersecurity Framework Subcategories	PR.DS-6
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	expiredcerttest.json
Preconditions	1. All devices have been configured to use IPv4, and the IoT device under test has been configured to emit a DHCPv4 message that includes the URL of its MUD file by using the DHCPv4 MUD URL option (IANA code 161). (Note that in the v6 version of this test, IPv6, DHCPv6, and IANA code 112 will be used.) 2. This MUD file is not currently cached at the MUD manager. 3. The IoT device’s MUD file is being hosted on a MUD file server that has a valid TLS certificate, but the MUD file signature was signed by a certificate that had already expired at the time of signature. 4. Local policy has been defined 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 communication to/from the device.

Test Case Field	Description
	5. The MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings with respect to the IoT device being used in the test.
Procedure	Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. Power-on the IoT device and connect it to the test network. This should set in motion the following series of steps, which should occur automatically: 1. IoT device automatically emits a DHCPv4 message containing the device's MUD URL (IANA code 161). (Note that in the v6 version of this test, IPv6, DHCPv6, and IANA code 112 will be used.) 2. DHCP server receives the DHCP message containing the IoT device's MUD URL. 3. DHCP server offers an IP address lease to the newly connected IoT device. 4. The IoT device requests this IP address lease, which the DHCP server acknowledges. 5. DHCP server sends the MUD URL to the MUD manager. 6. The MUD manager automatically contacts the MUD file server that is located by using the MUD URL, verifies that it has a valid TLS certificate, and requests the MUD file and signature from the MUD file server. 7. The MUD file server serves the MUD file and signature to MUD manager, and the MUD manager detects that the MUD file's signature was created by using a certificate that had already expired at the time of signing. 8. The MUD manager configures the router/switch that is closest to the IoT device so that it denies all communications to and from the IoT device.

Test Case Field	Description
Expected Results	The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to deny all communication to/from the IoT device. The expected configuration should resemble the following: Expecting a show access session without a MUD file as seen below: show access-session int g1018 det Interface GigabitEthernet1018 IIF-ID 0x181835C2 MAC Address b827.eba7.0533 IPv6 Address Unknown IPv4 Address 192.168.10.106 User-Name b827eba70533 Status Authorized Domain DATA Oper host mode multi-auth Oper control dir both Session timeout NA Common Session ID C0A80A02000000CCBDB267F8 Acct Session ID 0x00000046 Handle 0x100000c2 Current Policy mud-mab-test Server Policies Method status list Method State mab Authc Success
Actual Results	***MUDC [INFO][verify_mud_content:1594]--> BIO_reset <1> ***MUDC [ERROR][verify_mud_content:1604]--> Verification Failure

Test Case Field	Description
	139713269933824:error:2E099064:CMS routines:cms_signerinfo_verify_cert:certificate verify error:../crypto/cms/cms_smime.c:253:Verify error:**certificate has expired** ***MUDC [INFO][send_mudfs_request:2092]--> Verification failed. Manufacturer Index <0> ***MUDC [INFO][mudc_construct_head:135]--> status_code: 401, content_len: 19, extra_headers: (null) ***MUDC [INFO][mudc_construct_head:152]--> HTTP header: HTTP/1.1 401 Unauthorized Content-Length: 19 ***MUDC [INFO][send_error_result:176]--> **Verification failed** ***MUDC [ERROR][send_mudfs_request:2170]--> mudfs_conn failed --- Build1#sho access-session int g1018 det Interface GigabitEthernet1018 IIF-ID 0x181835C2 MAC Address b827.eba7.0533 IPv6 Address Unknown IPv4 Address 192.168.10.106 User-Name b827eba70533 Status Authorized Domain DATA Oper host mode multi-auth Oper control dir both Session timeout NA Common Session ID C0A80A02000000CCBDB267F8 Acct Session ID 0x00000046 Handle 0x100000c2 Current Policy mud-mab-test

Test Case Field	Description
	Server Policies Method status list Method State mab Authc Success
Overall Results	Pass

1641 As explained above, test IoT-3-v6 is identical to test IoT-3-v4 except that it uses IPv6, DHCPv6, and IANA
1642 code 112 instead of using IPv4, DHCPv4, and IANA code 161.

1643 D.1.2.4 Test Case IoT-4-v4

1644 **Table D-6: Test Case IoT-4-v4**

Test Case Field	Description
Parent Requirement	(CR 5) The IoT DDoS example implementation shall include a MUD manager that is capable of translating local network configurations based on the MUD file.
Testable Requirement	(CR-5.b) The MUD manager shall attempt to validate the signature of the MUD file, but the signature validation fails (even though the certificate that had been used to create the signature had not been expired at the time of signing, i.e., the signature is invalid for a different reason). (CR-5.b.1) The MUD manager shall cease processing the MUD file. (CR 5.b.2) MUD manager shall send locally defined policy to router or switch that handles whether to allow or block traffic to/from the MUD-enabled IoT device.
Description	Shows that if the MUD manager determines that the signature on the MUD file it receives from the MUD file server is invalid, it will cease processing the MUD file and configure the

Test Case Field	Description
	router/switch according to locally defined policy regarding whether to allow or block traffic to the IoT device in question.
Associated Test Cases	IoT-14-v4 (for the v6 version of this test, IoT-14-v6)
Associated Cybersecurity Framework Subcategories	PR.DS-6
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	ciscop2.json
Preconditions	1. All devices have been configured to use IPv4, and the IoT device under test has been configured to emit a DHCPv4 message that includes the URL of its MUD file by using the DHCPv4 MUD URL option (IANA code 161). (Note that in the v6 version of this test, IPv6, DHCPv6, and IANA code 112 will be used.) 2. This MUD file is not currently cached at the MUD manager. 3. The MUD file that is served from the MUD file server to the MUD manager has a signature that is invalid, even though it was signed by a certificate that had not expired at the time of signing. 4. Local policy has been defined to ensure that if the MUD file for a device has an invalid signature, the device's MUD PEP router/switch will be configured to deny all communication to/from the device. 5. The MUD PEP router/switch does not yet have any configuration settings with respect to the IoT device being used in the test.

Test Case Field	Description
Procedure	Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. Power-on the IoT device and connect it to the test network. This should set in motion the following series of steps, which should occur automatically: 1. IoT device automatically emits a DHCPv4 message containing the device's MUD URL (IANA code 161). (Note that in the v6 version of this test, IPv6, DHCPv6, and IANA code 112 will be used.) 2. DHCP server receives the DHCP message containing the IoT device's MUD URL. 3. DHCP server offers an IP address lease to the newly connected IoT device. 4. The IoT device requests this IP address lease, which the DHCP server acknowledges. 5. DHCP server sends the MUD URL to the MUD manager. 6. The MUD manager automatically contacts the MUD file server that is located by using the MUD URL, verifies that it has a valid TLS certificate, and requests the MUD file and signature from the MUD file server. 7. The MUD file server sends the MUD file, and the MUD manager detects that the MUD file's signature is invalid. 8. The MUD manager configures the router/switch that is closest to the IoT device so that it denies all communications to and from the IoT device.
Expected Results	The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to deny all communication to/from the IoT device. The expected configuration should resemble the following: Expecting a show access session without a MUD file as seen below:

Test Case Field	Description
	sho access-session int g1018 det Interface GigabitEthernet1018 IIF-ID 0x181835C2 MAC Address b827.eba7.0533 IPv6 Address Unknown IPv4 Address 192.168.10.106 User-Name b827eba70533 Status Authorized Domain DATA Oper host mode multi-auth Oper control dir both Session timeout NA Common Session ID C0A80A02000000CCBDB267F8 Acct Session ID 0x00000046 Handle 0x100000c2 Current Policy mud-mab-test Server Policies Method status list Method State mab Authc Success
Actual Results	> GET /ciscopi2.json HTTP/1.1 Host: mudfileserver Accept: */* [omitted for sake of length] ***MUDC [STATUS][send_mudfs_request:2060]--> Request signature URI <*https://mudfileserver/ciscopi2.p7s*> </home/mudtester/mud-intermediate.pem> * Trying 192.168.4.5... * TCP_NODELAY set

Test Case Field	Description
	* Connected to mudfileserver (192.168.4.5) port 443 (#0) * found 1 certificate in /home/mudtester/mud-intermediate.pem * found 400 certificates in /etc/ssl/certs * ALPN, offering http/1.1 * SSL connection using TLS1.2 / ECDHE_RSA_AES_256_GCM_SHA384 * server certificate verification OK * server certificate status verification SKIPPED * common name: mudfileserver (matched) * server certificate expiration date OK * server certificate activation date OK * certificate public key: RSA * certificate version: #3 * subject: C=US,ST=Maryland,L=Rockville,O=National Cybersecurity Center of Excellence - NIST,CN=mudfileserver * start date: Fri, 05 Oct 2018 00:00:00 GMT * expire date: Wed, 13 Oct 2021 12:00:00 GMT * issuer: C=US,O=DigiCert Inc,CN=DigiCert Test SHA2 Intermediate CA-1 * compression: NULL * ALPN, server did not agree to a protocol > GET /ciscopi2.p7s HTTP/1.1 Host: mudfileserver Accept: */* [omitted for sake of length] ***MUDC [INFO][send_mudfs_request:2080]--> MUD signature file successfully retrieved ***MUDC [DEBUG][verify_mud_content:1543]--> MUD signature file (length 4680) [shortened logs] ***MUDC [INFO][verify_mud_content:1594]--> BIO_reset <1>

Test Case Field	Description
	***MUDC [ERROR][verify_mud_content:1604]--> **Verification Failure** 140561528563456:error:2E09A09E:CMS routines:CMS_SignerInfo_verify_content:verification failure:../crypto/cms/cms_sd.c:819: 140561528563456:error:2E09D06D:CMS routines:CMS_verify:content verify error:../crypto/cms/cms_smime.c:393: ***MUDC [INFO][send_mudfs_request:2092]--> **Verification failed. Manufacturer Index <0>** ***MUDC [INFO][mudc_construct_head:135]--> status_code: 401, content_len: 19, extra_headers: (null) ***MUDC [INFO][mudc_construct_head:152]--> HTTP header: HTTP/1.1 401 Unauthorized Content-Length: 19 ***MUDC [INFO][send_error_result:176]--> **Verification failed** ***MUDC [ERROR][send_mudfs_request:2170]--> **mudfs_conn failed**
	Switch access-session: Build1#sho access-session int g1/0/18 det Interface: GigabitEthernet1/0/18 IIF-ID: 0x11C404C6 MAC Address: b827.eba7.0533 IPv6 Address: Unknown IPv4 Address: 192.168.10.106 User-Name: b827eba70533 Status: Authorized Domain: DATA Oper host mode: multi-auth Oper control dir: both Session timeout: N/A Common Session ID: C0A80A02000000CDBDB68A30

Test Case Field	Description
	Acct Session ID: 0x00000047 Handle: 0x690000c3 Current Policy: mud-mab-test Server Policies: Method status list: Method State mab Authc Success
Overall Results	Pass

As explained above, test IoT-4-v6 is identical to test IoT-4-v4 except that it uses IPv6, DHCPv6, and IANA code 112 instead of using IPv4, DHCPv4, and IANA code 161.

D.1.2.5 Test Case IoT-5-v4

Table D-7: Test Case IoT-5-v4

Test Case Field	Description
Parent Requirement	(CR 7) The IoT DDoS example implementation shall allow a MUD-enabled IoT device to communicate with approved internet services in MUD file. (CR 8) The IoT DDoS example implementation shall deny a MUD-enabled IoT device to communicate with unapproved internet services (i.e., services that are implicitly denied by virtue of not being explicitly approved).
Testable Requirement	(CR-7.a) The MUD-enabled IoT device shall attempt to initiate outbound traffic to approved internet services. (CR-7.a.1) The router or switch shall receive the attempt and shall allow it to pass based on the filters from MUD file.

Test Case Field	Description
	(CR-7.b) An approved internet service shall attempt to initiate connection to MUD-enabled IoT device. (CR-7.b.1) The router or switch shall receive the attempt and shall allow it to pass based on the filters from MUD file. (CR-8.a) The MUD-enabled IoT device shall attempt to initiate outbound traffic to unapproved (implicitly denied) internet services. (CR-8.a.1) The router or switch shall receive the attempt and shall deny it based on the filters from MUD file. (CR-8.b) An unapproved (implicitly denied) internet service shall attempt to initiate a connection to the MUD-enabled IoT device. (CR-8.b.1) The router or switch shall receive the attempt and shall deny it based on the filters from the MUD file. (CR-8.c) The MUD-enabled IoT device shall initiate communications to an internet service that is approved to initiate communication with the MUD-enabled device but not approved to receive communications initiated by the MUD-enabled device. (CR-8.c.1) The router or switch shall receive the attempt and shall deny it based on the filters from the MUD file. (CR-8.d) An internet service shall initiate communications to a MUD-enabled device that is approved to initiate communications with the internet service but that is not approved to receive communications initiated by the internet service. (CR-8.d.1) The router or switch shall receive the attempt and shall deny it based on the filters from the MUD file.
Description	Shows that, upon connection to the network, a MUD-enabled IoT device used in the IoT DDoS example implementation has its MUD PEP router/switch automatically configured to enforce the

Test Case Field	Description
	route filtering that is described in the device's MUD file with respect to communication with internet services. Further shows that the policies that are configured on the MUD PEP router/switch with respect to communication with internet services will be enforced as expected, with communications that are configured as denied being blocked, and communications that are configured as permitted being allowed.
Associated Test Cases	IoT-1-v4 (for the v6 version of this test, IoT-1-v6)
Associated Cybersecurity Framework Subcategories	ID.AM-3, PR.DS-5, PR.IP-1, PR.PT-3
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	ciscopi2.json
Preconditions	Test IoT-1-v4 (or IoT-1-v6) has run successfully, meaning that the MUD PEP router/switch has been configured to enforce the following policies for the IoT device in question (as defined in the MUD file in section D.1.3): a) Explicitly permit *https://yes-permit-from.com* to initiate communication with the IoT device. b) Explicitly permit to the IoT device to initiate communication with *https://yes-permit-to.com*. c) Implicitly deny all other communications with the internet, including denying i) the IoT device to initiate communication with *https://yes-permit-from.com* ii) *https://yes-permit-to.com* to initiate communication with the IoT device iii) communication between the IoT device and all other internet locations, such as *https://unnamed-*

Test Case Field	Description
	*to.com (*by not mentioning this or any other URLs in the MUD file)
Procedure	Note: Procedure steps with strike-through are not tested in this phase as ingress DACLs are not supported in this implementation. 1. As stipulated in the preconditions, right before this test, test IoT-1-v4 (or IoT-1-v6) must have been run successfully. 2. Initiate communications from the IoT device to *https://yes-permit-to.com* and verify that this traffic is received at *https://yes-permit-to.com*. (egress) 3. Initiate communications to the IoT device from *https://yes-permit-to.com* and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at the IoT device. (ingress) ~~4. Initiate communications to the IoT device from *https://yes-permit-from.com* and verify that this traffic is received at the IoT device. (ingress)~~ ~~5. Initiate communications from the IoT device to *https://yes-permit-from.com* and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at *https://yes-permit-from.com*. (ingress)~~ 6. Initiate communications from the IoT device to *https://unnamed.com* and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at *https://unnamed.com*. (egress) 7. Initiate communications to the IoT device from *https://unnamed.com* and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at the IoT device. (ingress)
Expected Results	Each of the results that is listed as needing to be verified in procedure steps above occurs as expected.

Test Case Field	Description
Actual Results	Procedure 2– Connection to update server successfully initiated by IoT device: pi@raspberrypi:~ $ wget *http://www.updateserver.com/* --2018-12-13 21:28:00-- *http://www.updateserver.com/* Resolving *www.updateserver.com* (*www.updateserver.com*)... 192.168.4.7 Connecting to *www.updateserver.com* (*www.updateserver.com*)|192.168.4.7|:80... connected. HTTP request sent, awaiting response... 200 OK Length: 10918 (11K) [text/html] Saving to: 'index.html.2' index.html.2 100%[==================>] 10.66K --.-KB/s in 0s 2018-12-13 21:28:00 (30.6 MB/s) - 'index.html.2' saved [10918/10918]
	Procedure 3– Update server failed to connect to IoT device: iot@update-server:~$ wget *http://192.168.13.9* --2018-12-13 21:49:36-- *http://192.168.13.9/* Connecting to 192.168.13.9:80... failed: Connection timed out. Retrying.
	Procedure 6– IoT device failed to connect to unapproved server: pi@raspberrypi:~ $ wget *http://192.168.4.105* --2018-12-14 16:42:36-- *http://192.168.4.105/* Connecting to 192.168.4.105:80... failed: Connection timed out. Retrying.

Test Case Field	Description
	Procedure 7– Unapproved server attempts to connect to IoT device: [mud@unapprovedserver ~]$ wget *http://192.168.13.14* --2018-12-14 13:03:32-- *http://192.168.13.14/* Connecting to 192.168.13.14:80... failed: Connection timed out. Retrying.
Overall Results	Pass (for testable procedures–as stated, ingress cannot be tested)

As explained above, test IoT-5-v6 is identical to test IoT-5-v4 except that it uses IPv6, DHCPv6, and IANA code 112 instead of using IPv4, DHCPv4, and IANA code 161.

D.1.2.6 Test Case IoT-6-v4

Table D-8: Test Case IoT-6-v4

Test Case Field	Description
Parent Requirement	(CR 9) The IoT DDoS example implementation shall allow MUD-enabled IoT device to communicate laterally with devices that are approved in MUD file. (CR 10) The IoT DDoS example implementation shall deny MUD-enabled IoT device to communicate laterally with devices that are not approved in MUD file (i.e., devices that are implicitly denied by virtue of not being explicitly approved).
Testable Requirement	(CR-9.a) The MUD-enabled IoT device shall attempt to initiate lateral traffic to approved devices. (CR-9.a.1) The router or switch shall receive the attempt and shall allow it to pass based on the filters from MUD file. (CR-9.b) An approved device shall attempt to initiate a lateral connection to the MUD-enabled IoT device.

Test Case Field	Description
	(CR-9.b.1) The router or switch shall receive the attempt and shall allow it to pass based on the filters from MUD file. (CR-10.a) The MUD-enabled IoT device shall attempt to initiate lateral traffic to unapproved (implicitly denied) devices. (CR-10-a.1) The router or switch shall receive the attempt and shall deny it based on the filters from MUD file. (CR-10-b) An unapproved (implicitly denied) device shall attempt to initiate a lateral connection to MUD-enabled IoT device. (CR-10-b.1) The router or switch shall receive the attempt and shall deny it based on the filters from MUD file.
Description	Shows that, upon connection to the network, a MUD-enabled IoT device used in the IoT DDoS example implementation has its MUD PEP router/switch automatically configured to enforce the route filtering that is described in the device's MUD file with respect to communication with lateral devices. Further shows that the policies that are configured on the MUD PEP router/switch with respect to communication with lateral devices will be enforced as expected, with communications that are configured as denied being blocked and communications that are configured as being allowed.
Associated Test Cases	IoT-1-v4 (for the v6 version of this test, IoT-1-v6)
Associated Cybersecurity Framework Subcategories	ID.AM-3, PR.DS-5, PR.AC-5, PR.IP-1, PR.PT-3, PR.IP-3, PR.DS-3
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	ciscopi2.json

Test Case Field	Description
Preconditions	Test IoT-1-v4 (or IoT-1-v6) has run successfully, meaning that the MUD PEP router/switch has been configured to enforce the following policies for the IoT device in question with respect to local communications (as defined in the MUD files in section D.1.3): a) Local-network class—Explicitly permit **local communication to/from IoT device and any local hosts** (including the specific local hosts *anyhost-to* and *anyhost-from*) **for specific services,** as specified in MUD file by source port any, destination port: 80; and protocol: TCP, and which party initiates the connection. b) Manufacturer class—Explicitly permit **local communication to/from IoT device and other classes of IoT devices, as identified by their MUD URL (*www.device-type.com*), and further constrained** by source port: any; destination port: 80; and protocol: TCP. c) Same-manufacturer class—Explicitly permit **local communication to/from IoT devices of the same manufacturer as the IoT device in question (the domain in the MUD URLs (mudfileserver) of the other IoT devices is the same as the domain in the MUD URL (mudfileserver) of the IoT device in question),** and further constrained by source port: any; destination port: 80; and protocol: TCP. d) Implicitly deny all other local communication that is not explicitly permitted in the MUD file, including denying i) **anyhost-to to initiate communications** with the IoT device **ii)** the **IoT device to initiate communications** with anyhost-to by **using a source port, destination port, or protocol (TCP or UDP) that is not explicitly permitted** iii) the **IoT device to initiate communications with anyhost-from** iv) **anyhost-from to initiate communications** with the IoT device by **using a source port, destination port,**

Test Case Field	Description
	or protocol (TCP or UDP) that is not explicitly permitted v) communications between the IoT device and all lateral hosts (including *unnamed-host*) whose **MUD URLs are not explicitly mentioned** as being permissible in the MUD file vi) communications between the IoT device and all lateral hosts whose **MUD URLS are explicitly mentioned** as being permissible, **but using a source port, destination port, or protocol (TCP or UDP) that is not explicitly permitted** vii) communications between the IoT device and all lateral hosts that are **not from the same manufacturer** as the IoT device in question viii) communications between the IoT device and a lateral host that **is from the same manufacturer, but using a source port, destination port, or protocol (TCP or UDP) that is not explicitly permitted**
Procedure	Note: Procedure steps with strike-through are not tested in this phase as ingress DACLs are not supported in this implementation. 1. As stipulated in the preconditions, right before this test, test IoT-1-v4 (or IoT-1-v6) must have been run successfully. ~~2. Local-network (ingress): Initiate communications to the IoT device from *anyhost-from* **for specific permitted service,** and verify that this traffic is received at the IoT device.~~ 3. Local-network (egress): **Initiate communications from the IoT device to *anyhost-from*** for specific permitted service, and verify that this traffic is received at the MUD PEP, but it **is not forwarded** by the MUD PEP, nor is it received at *anyhost-from*.

Test Case Field	Description
	4. Local-network, controller, my-controller, manufacturer class (egress): Initiate communications from the IoT device to *anyhost-to* **for specific permitted service,** and verify that this traffic **is received** at *anyhost-to.* 5. ~~Local-network , controller, my-controller, manufacturer class (ingress): **Initiate communications to the IoT device from *anyhost-to*** for specific permitted service, and verify that this traffic is received at the MUD PEP, but it **is not forwarded** by the MUD PEP, nor is it received at the IoT device.~~ 6. No associated class (egress): Initiate communications from the IoT device to *unnamed-host* (where *unnamed-host* is a host that is not from the same manufacturer as the IoT device in question and whose **MUD URL is not explicitly mentioned in the MUD file as being permitted),** and verify that this traffic is received at the MUD PEP, but it **is not forwarded** by the MUD PEP, nor is it received at *unnamed-host*. 7. ~~No associated class (ingress): Initiate communications to the IoT device from *unnamed-host* (where *unnamed-host* is a host that is not from the same manufacturer as the IoT device in question and whose **MUD URL is not explicitly mentioned in the MUD file as being permitted)**, and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at the IoT device.~~ 8. Same-manufacturer class (egress): Initiate communications from the IoT device to *same-manufacturer-host* (where *same-manufacturer-host* is **a host that is from the same manufacturer as the IoT device** in question), and verify that this traffic **is received** at *same-manufacturer-host*. 9. Same-manufacturer class (egress): Initiate communications from the IoT device to *same-manufacturer-host* (where *same-manufacturer-host* is **a host that is from the same manufacturer as the IoT device** in question) **but using a port or protocol that is not specified,** and verify that this traffic is received at the MUD PEP, but it **is not forwarded** by the MUD PEP, nor is it received at *same-manufacturer-host*.

Test Case Field	Description
Expected Results	Each of the results that is listed as needing to be verified in the procedure steps above occurs as expected.
Actual Results	3. Local_network (egress)–blocked: pi@raspberrypi:~ $ wget *https://192.168.10.106/* --2019-01-31 19:59:23-- *https://192.168.10.106/* Connecting to 192.168.10.106:443... failed: Connection timed out. Retrying. 4. Local-network, controller, my-controller, manufacturer class (egress)–allowed: Local_Network: pi@raspberrypi:~ $ wget *http://192.168.10.175* --2018-12-14 15:11:50-- *http://192.168.10.175/* Connecting to 192.168.10.175:80... connected. HTTP request sent, awaiting response... 200 OK Length: 10701 (10K) [text/html] Saving to: 'index.html.4' index.html.4 100%[==================>] 10.45K --.-KB/s in 0s 2018-12-14 15:11:50 (41.4 MB/s) - 'index.html.4' saved [10701/10701] Controller: pi@raspberrypi:~ $ wget *http://192.168.10.105/* --2019-01-31 21:03:45-- *http://192.168.10.105/* Connecting to 192.168.10.105:80... connected. HTTP request sent, awaiting response... 200 OK Length: 277 Saving to: 'index.html.10'

Test Case Field	Description
	in- dex.html.10 100%[==================>] 277 --.-KB/s in 0s 2019-01-31 21:03:45 (18.8 MB/s) - ‘index.html.10’ saved [277/277] My-controller: pi@raspberrypi:~ $ wget *http://192.168.10.104/* --2019-01-31 21:06:39-- *http://192.168.10.104/* Connecting to 192.168.10.104:80... connected. HTTP request sent, awaiting response... 200 OK Length: 10701 (10K) [text/html] Saving to: ‘index.html.11’ in- dex.html.11 100%[==================>] 10.45K --.-KB/s in 0s 2019-01-31 21:06:39 (32.5 MB/s) - ‘index.html.11’ saved [10701/10701] Manufacturer: pi@raspberrypi:~ $ wget *http://192.168.14.2/* --2019-01-31 21:13:47-- *http://192.168.14.2/* Connecting to 192.168.14.2:80... connected. HTTP request sent, awaiting response... 200 OK Length: 10701 (10K) [text/html] Saving to: ‘index.html.12’ in- dex.html.12 100%[==================>] 10.45K --.-KB/s in 0s

Test Case Field	Description
	2019-01-31 21:13:47 (39.6 MB/s) - ‘index.html.12’ saved [10701/10701]
	6. No associated class (egress)–blocked: pi@raspberrypi:~ $ wget *http://192.168.15.105* --2018-12-14 17:15:36-- *http://192.168.15.105/* Connecting to 192.168.15.105:80... failed: Connection timed out. Retrying.
	8. Same-manufacturer class (egress)–allowed: pi@raspberrypi:~ $ wget *http://192.168.13.8/* --2019-01-31 21:16:41-- *http://192.168.13.8/* Connecting to 192.168.13.8:80... connected. HTTP request sent, awaiting response... 200 OK Length: 10701 (10K) [text/html] Saving to: ‘index.html.13’ in- dex.html.13 100%[===================>] 10.45K --.-KB/s in 0s 2019-01-31 21:16:41 (37.9 MB/s) - ‘index.html.13’ saved [10701/10701]
	9. Same-manufacturer class (egress)–blocked: pi@raspberrypi:~ $ wget *https://192.168.13.8/* --2019-01-31 21:17:15-- *https://192.168.13.8/* Connecting to 192.168.13.8:443... failed: Connection timed out. Retrying.

Test Case Field	Description
Overall Results	Pass (for testable procedures–as stated, ingress cannot be tested)

1653 As explained above, test IoT-6-v6 is identical to test IoT-6-v4 except that it uses IPv6, DHCPv6, and IANA
1654 code 112 instead of using IPv4, DHCPv4, and IANA code 161.

1655 ### D.1.2.7 Test Case IoT-7-v4

1656 **Table D-9: Test Case IoT-7-v4**

Test Case Field	Description
Parent Requirement	(CR 11) The IoT DDoS example implementation shall remove the implemented policy when the MUD-enabled IoT device changes DHCP state.
Testable Requirement	(CR-11.a) The MUD-enabled IoT device shall explicitly release the IP address lease (i.e., it sends a DHCP release message to the DHCP server). (CR-11.a.1) The DHCP server shall notify MUD manager that the device's IP address lease has been released. (CR-11.a.2) The MUD manager shall remove all policies associated with the disconnected IoT device that had been configured on the MUD PEP router/switch.
Description	Shows that when a MUD-enabled 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.
Associated Test Cases	IoT-1-v4 (or IoT-1-v6 when IPv6 addressing is used)
Associated Cybersecurity Frame-work Subcategories	PR.IP-3, PR.DS-3

Test Case Field	Description
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	ciscopi2.json
Preconditions	Test IoT-1-v4 (or IoT-1-v6) has run successfully, meaning that the MUD PEP router/switch has been configured to enforce the policies defined in the MUD file in section D.1.3 for the IoT device in question.
Procedure	1. As stipulated in the preconditions, right before this test, test IoT-1-v4 (or IoT-1-v6) must have been run successfully. Verify that the MUD PEP router/switch for the IoT device has been configured to enforce the policies listed in the preconditions section above for the IoT device in question. 2. Cause a DHCP release of the IoT device in question. 3. Verify that all the configuration rules listed above have been removed from the MUD PEP router/switch for the IoT device in question.
Expected Results	All of the configuration rules listed above have been removed from the MUD PEP router/switch for the IoT device in question.
Actual Results	Procedure 1– Build1#sh access-session int g1/0/15 det Interface: GigabitEthernet1/0/15 IIF-ID: 0x1B6BCEA5 MAC Address: b827.ebeb.6c8b IPv6 Address: Unknown IPv4 Address: 192.168.13.17 User-Name: b827ebeb6c8b Status: Authorized Domain: DATA Oper host mode: multi-auth Oper control dir: both

Test Case Field	Description
	Session timeout: N/A Common Session ID: C0A80A0200000A6A9828F06 Acct Session ID: 0x0000003b Handle: 0x2200009c Current Policy: mud-mab-test Server Policies: ACS ACL: mud-81726-v4fr.in Vlan Group: Vlan: 3 Method status list: Method State mab Authc Success
	Procedure 2– pi@raspberrypi:~ $ sudo dhclient -v -r
	Build1#sh access-session int g1/0/15 det Interface: GigabitEthernet1/0/15 IIF-ID: 0x1B6BCEA5 MAC Address: b827.ebeb.6c8b IPv6 Address: Unknown IPv4 Address: Unknown User-Name: b827ebeb6c8b Status: Authorized Domain: DATA Oper host mode: multi-auth Oper control dir: both Session timeout: N/A

Test Case Field	Description
	Common Session ID: C0A80A0200000A6A9828F06 Acct Session ID: 0x0000003b Handle: 0x2200009c Current Policy: mud-mab-test Server Policies: ACS ACL: mud-81726-v4fr.in Vlan Group: Vlan: 3 Method status list: Method State mab Authc Success
Overall Results	Failed

1657 As explained above, test IoT-7-v6 is identical to test IoT-7-v4 except that it uses IPv6, DHCPv6, and IANA
1658 code 112 instead of using IPv4, DHCPv4, and IANA code 161.

1659 D.1.2.8 Test Case IoT-8-v4

1660 **Table D-10: Test Case IoT-8-v4**

Test Case Field	Description
Parent Requirement	(CR 11) The IoT DDoS example implementation shall remove the implemented policy when the MUD-enabled IoT device changes DHCP state.
Testable Requirement	(CR-11.b) The MUD-enabled IoT device’s IP address lease shall expire.

Test Case Field	Description
	(CR-11.b.1) The DHCP server shall notify the MUD manager that the device's IP address lease has expired. (CR-11.b.2) The MUD manager shall remove all policies associated with the affected IoT device that had been configured on the MUD PEP router/switch.
Description	Shows that when a MUD-enabled IoT device's IP address lease expires, the MUD-related configuration for that IoT device will be removed from its MUD PEP router/switch.
Associated Test Cases	IoT-1-v4 (or IoT-1-v6 when IPv6 addressing is used)
Associated Cybersecurity Framework Subcategories	PR.IP-3, PR.DS-3
IoT Device(s) Under Test	TBD (Not testable in Build 1)
MUD File(s) Used	TBD (Not testable in Build 1)
Preconditions	Test IoT-1-v4 (or IoT-1-v6) has run successfully, meaning that the MUD PEP router/switch has been configured to enforce the policies defined in the MUD file in section D.1.3 for the IoT device in question.
Procedure	1. Configure the DHCP server to have a DHCP lease time of 10 minutes. 2. Run test IoT-1-v4 (or IoT-1-v6). 3. Verify that the MUD PEP router/switch for the IoT device has been configured to enforce the policies listed above for the IoT device in question. 4. Disconnect the IoT device in question from the network. 5. After 10 minutes have elapsed, verify that all of the configuration rules listed above have been removed from the MUD PEP router/switch for the IoT device in question.

Test Case Field	Description
Expected Results	Once 10 minutes have elapsed after disconnecting the IoT device from the network, all of the configuration rules listed above have been removed from the MUD PEP router/switch for the IoT device in question.
Actual Results	TBD (Not testable in Build 1)
Overall Results	TBD (Not testable in Build 1)

As explained above, test IoT-8-v6 is identical to test IoT-8-v4 except that it uses IPv6, DHCPv6, and IANA code 112 instead of using IPv4, DHCPv4, and IANA code 161.

D.1.2.9 Test Case IoT-9-v4

Table D-11: Test Case IoT-9-v4

Test Case Field	Description
Parent Requirement	(CR 12) The IoT DDoS example implementation shall include a router or switch that is capable of receiving and implementing threat signaling information from a threat signaling server that includes "blacklisted" URLs and hosts that are not considered to be trusted. The router/switch can be configured so that it applies threat signaling rules to both MUD-enabled and non-MUD-capable devices, based on local policy.
Testable Requirement	(CR-12.a) The router or switch shall receive threat signals from the threat signaling server and translate them into appropriate permit/deny configuration rules that will pertain to all devices (both MUD-enabled and non-MUD-capable). (CR-12.a.1) A non-MUD-capable IoT device shall attempt to initiate outbound traffic to blacklisted internet URL. The router or switch shall receive the attempt and shall deny it based on the filters from the threat signaling server.

Test Case Field	Description
	(CR-12.a.2) There shall be an attempt to initiate a connection from a blacklisted internet URL to a non-MUD-capable IoT device. The router or switch shall receive the attempt and shall deny it based on the filters from the threat signaling server. (CR-12.a.3) The MUD-enabled IoT device shall attempt to initiate outbound traffic to an internet URL that is explicitly permitted in its MUD file but that has been blacklisted by the threat signaling service. The router or switch shall receive the attempt and shall deny it based on the filters from the threat signaling server. (CR-12.a.4) A blacklisted internet URL that is explicitly permitted in a MUD-enabled device's MUD file shall attempt to send traffic to the MUD-enabled device. The router or switch shall receive the attempt and shall deny it based on the filters from the threat signaling server.
Description	Shows that the devices in the IoT DDoS example implementation can be configured to apply threat signaling to both non-MUD-capable and MUD-enabled devices so that threat-signaling information will override MUD file policy for a MUD-enabled device, denying communication with a blacklisted URL or host even though communication with that URL or host may be explicitly permitted in the device's MUD file.
Associated Test Cases	IoT-1-v4 (or IoT-1-v6)
Associated Cybersecurity Framework Subcategories	TBD (Not testable in Build 1)
IoT Device(s) Under Test	TBD (Not testable in Build 1)
MUD File(s) Used	TBD (Not testable in Build 1)

Test Case Field	Description
Preconditions	1. A threat signaling device is available for use in the IoT DDoS example implementation. 2. The threat signaling device has been configured to generate threat signaling information that blacklists the following host types and internet services, but the threat signaling device has not yet been turned on: *a. https://yes-permit-to.com* *b. https://yes-permit-from.com* 3. The MUD manager and the device that receives the threat signaling information (if different from the MUD manager) are configured so that threat signaling will be applied to both MUD-enabled and non-MUD-capable devices. 4. Two different types of IoT devices are available for use in this test: one that is MUD-enabled and one that is non-MUD-capable.
Procedure	1. Run test IoT-1-v4 (or IoT-1-v6). Verify that the MUD PEP router/switch for the MUD-enabled IoT device has been configured to enforce the following policies: a. Explicitly permit the IoT device to initiate communication with *https://yes-permit-to.com.* b. Explicitly permit *https://yes-permit-from.com* to initiate communication with the IoT device. 2. Initiate communication from the MUD-enabled IoT device to *https://yes-permit-to.com,* and verify that this traffic is received at *https://yes-permit-to.com.* 3. Initiate communication to the MUD-enabled IoT device from *https://yes-permit-from.com,* and verify that this traffic is received at the IoT device. 4. Initiate communication from the non-MUD-capable IoT device to *https://yes-permit-to.com,* and verify that this traffic is received at *https://yes-permit-to.com.* 5. Initiate communication to the MUD-incapable IoT device from *https://yes-permit-from.com,* and verify that this traffic is received at the IoT device.

Test Case Field	Description
	6. The above steps verify that both the MUD-enabled and non-MUD-capable devices are operating as expected in the absence of threat signaling information. 7. Turn on the threat signaler and wait sufficient time for it to send threat signaling information. 8. Initiate communication from the MUD-enabled IoT device to *https://yes-permit-to.com,* and verify that this traffic is neither forwarded to the internet nor received at *https://yes-permit-to.com.* 9. Initiate communication to the MUD-enabled IoT device from *https://yes-permit-from.com,* and verify that this traffic is not received at the IoT device. 10. Initiate communication from the non-MUD-capable IoT device to *https://yes-permit-to.com,* and verify that this traffic is neither forwarded to the internet nor received at *https://yes-permit-to.com.* 11. Initiate communication to the MUD-incapable IoT device from *https://yes-permit-from.com,* and verify that this traffic is not received at the IoT device.
Expected Results	All expectations stated in the procedural steps listed above will be verified as described.
Actual Results	TBD (Not testable in Build 1)
Overall Results	TBD (Not testable in Build 1)

As explained above, test IoT-9-v6 is identical to test IoT-9-v4 except that it uses IPv6, DHCPv6, and IANA code 112 instead of using IPv4, DHCPv4, and IANA code 161.

D.1.2.10 Test Case IoT-10-v4

Table D-12: Test Case IoT-10-v4

Test Case Field	Description
Parent Requirement	(CR 13) The IoT DDoS example implementation shall include a router or switch that is capable of receiving and implementing threat signaling information from a threat signaling server that includes "blacklisted" URLs and hosts that are not considered to be trusted. The router/switch can be configured so that it applies threat signaling rules only to non-MUD-capable devices, based on local policy.
Testable Requirement	(CR-13.a) The router or switch shall receive threat signals from the threat signaling server and translate them into appropriate permit/deny configuration rules that will pertain only to non-MUD-capable devices. (CR-13.a.1) The non-MUD-capable IoT device shall attempt to initiate outbound traffic to blacklisted internet URL. The router or switch shall receive the attempt and shall deny it based on the filters from the threat signaling server. (CR-13.a.2) There shall be an attempt to initiate a connection from a blacklisted internet URL to the non-MUD-capable IoT device. The router or switch shall receive the attempt and shall deny it based on the filters from the threat signaling server. CR-13.a.3) The MUD-enabled IoT device shall attempt to initiate outbound traffic to an internet URL that is explicitly permitted in its MUD file but that has been blacklisted by the threat signaling service. The router or switch shall receive the attempt and shall permit it because the filters from the threat signaling server do not apply to MUD-enabled devices. (CR-13.a.4) A device from a blacklisted internet URL that is explicitly permitted in a MUD-enabled device's MUD file shall attempt to send traffic to the MUD-enabled IoT device. The router or switch shall receive the attempt and shall permit it because the filters from the threat signaling server do not apply to MUD-enabled devices.

Test Case Field	Description
Description	Shows that the devices in the IoT DDoS example implementation can be configured to apply threat signaling only to non-MUD-capable devices and not to MUD-enabled devices, even if the threat-signaling information contradicts the policy in the MUD-enabled device's MUD file.
Associated Test Cases	IoT-1-v4 (or IoT-1-v6)
Associated Cybersecurity Framework Subcategories	TBD (Not testable in Build 1)
IoT Device(s) Under Test	TBD (Not testable in Build 1)
MUD File(s) Used	TBD (Not testable in Build 1)
Preconditions	1. A threat signaling device is available for use in the IoT DDoS example implementation. 2. The threat signaling device has been configured to generate threat signaling information that blacklists the following host types and internet services, but the threat signaling device has not yet been turned on: a. *https://yes-permit-to.com* b. *https://yes-permit-from.com* 3. The MUD manager and the device that receives the threat signaling information (if different from the MUD manager) are configured so that threat signaling will be applied only to non-MUD-capable devices. 4. Two different types of IoT devices are available for use in this test: one that is MUD-enabled and one that is non-MUD-capable.
Procedure	1. Run test IoT-1-v4 (or IoT-1-v6). Verify that the MUD PEP router/switch for the MUD-enabled IoT device has been configured to enforce the following policies:

Test Case Field	Description
	a. Explicitly permit the IoT device to initiate communication with *https://yes-permit-to.com*. b. Explicitly permit *https://yes-permit-from.com* to initiate communication with the IoT device. 2. Initiate communication from the MUD-enabled IoT device to *https://yes-permit-to.com,* and verify that this traffic is received at *https://yes-permit-to.com.* 3. Initiate communication to the MUD-enabled IoT device from *https://yes-permit-from.com,* and verify that this traffic is received at the IoT device. 4. Initiate communication from the non-MUD-capable IoT device to *https://yes-permit-to.com,* and verify that this traffic is received at *https://yes-permit-to.com.* 5. Initiate communication to the MUD-incapable IoT device from *https://yes-permit-from.com,* and verify that this traffic is received at the IoT device. 6. The above steps verify that both the MUD-enabled and non-MUD-capable devices are operating as expected in the absence of threat signaling information. 7. Turn on the threat signaler and wait sufficient time for it to send threat signaling information. 8. Initiate communication from the MUD-enabled IoT device to *https://yes-permit-to.com,* and verify that this traffic is received at *https://yes-permit-to.com*. 9. Initiate communication to the MUD-enabled IoT device from *https://yes-permit-from.com,* and verify that this traffic is received at the IoT device. 10. Initiate communication from the non-MUD-capable IoT device to *https://yes-permit-to.com,* and verify that this traffic is neither forwarded to the internet nor received at *https://yes-permit-to.com.* 11. Initiate communication to the MUD-incapable IoT device from *https://yes-permit-from.com,* and verify that this traffic is not received at the IoT device.

Test Case Field	Description
Expected Results	TBD (Not testable in Build 1)
Actual Results	TBD (Not testable in Build 1)
Overall Results	TBD (Not testable in Build 1)

As explained above, test IoT-10-v6 is identical to test IoT-10-v4 except that it uses IPv6, DHCPv6, and IANA code 112 instead of using IPv4, DHCPv4, and IANA code 161.

D.1.2.11 Test Case IoT-11-v4

Table D-13 contains test case requirements, associated test cases, and descriptions of the test scenarios for the IoT capabilities of the example implementations. At the time of testing, this test case was not evaluated due to not being fully supported by the implementation.

Table D-13: Test Case IoT-11-v4

Test Case Field	Description
Parent Requirement	(CR-14) The IoT DDoS example implementation shall include capability to handle life-cycle changes, such as decommissioning, of IoT devices.
Testable Requirement	(CR-14.a) MUD-enabled IoT device manufacturers should provide a final MUD file for devices no longer being supported. (CR-14.a.1) The MUD manager shall use the final MUD file to configure traffic filters for the IoT device per CR 1-6.
Description	Shows that the MUD-enabled IoT device manufacturer and the MUD manager in the IoT DDoS example implementation are able to successfully maintain usage of IoT devices throughout life-cycle changes of the IoT device.
Associated Test Cases	N/A

Test Case Field	Description
Associated Cybersecurity Framework Subcategories	TBD (Not testable in Build 1)
IoT Device(s) Under Test	TBD (Not testable in Build 1)
MUD File(s) Used	TBD (Not testable in Build 1)
Preconditions	TBD (Not testable in Build 1)
Procedure	TBD (Not testable in Build 1)
Expected Results	TBD (Not testable in Build 1)
Actual Results	TBD (Not testable in Build 1)
Overall Results	TBD (Not testable in Build 1)

1676 ### D.1.2.12 Test Case IoT-12-v4

1677 **Table D-14: Test Case IoT-12-v4**

Test Case Field	Description
Parent Requirements	(CR 15) The IoT DDoS example implementation shall include a MUD manager that uses a cached MUD file rather than retrieving a new one if the cache-validity time period has not yet elapsed for the MUD file indicated by the MUD URL.
Testable Requirements	(CR 15.a) The MUD manager shall check if the file associated with the MUD URL is present in its cache and shall determine that it is.

Test Case Field	Description
	(CR 15.a.1) The MUD manager shall check whether the amount of time that has elapsed since the cached file was retrieved is less than or equal to the number of hours in the cache-validity value for this MUD file. If so, the MUD manager shall apply the contents of the cached MUD file. (CR 15.a.2) The MUD manager shall check whether the amount of time that has elapsed since the cached file was retrieved is greater than the number of hours in the cache-validity value for this MUD file. If so, the MUD manager may (but does not have to) fetch a new file by using the MUD URL received.
Description	Shows that, upon connection to the network, a MUD-enabled IoT device used in the IoT DDoS example implementation has its MUD PEP router/switch automatically configured to enforce the route filtering that is described in the cached MUD file for that device's MUD URL, assuming that the amount of time that has elapsed since the cached MUD file was retrieved is less than or equal to the number of hours in the file's cache-validity value. If the cache validity has expired in for the respective file, the MUD manager should fetch a new MUD file from the MUD file server.
Associated Test Cases	N/A
Associated Cybersecurity Frame-work Subcategories	ID.AM-1, ID.AM-2, ID.AM-3, PR.DS-5, DE.AE-1, PR.AC-4, PR.AC-5, PR.IP-1, PR.IP-3, PR.DS-2
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	ciscopi2.json
Preconditions	1. All devices have been configured to use IPv4, and the IoT de-vice under test has been configured to emit a DHCPv4 mes-sage that includes the URL of its MUD file by using the DHCPv4 MUD URL option (IANA code 161). (Note that in the

Test Case Field	Description
	v6 version of this test, IPv6, DHCPv6, and IANA code 112 will be used.) 2. The MUD PEP router/switch does not yet have any configuration settings pertaining to the IoT device being used in the test. 3. The MUD file for the IoT device being used in the test is identical to the MUD file provided in section D.1.3.
Procedure	Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. 1. Run test IoT-1-v4 (or IoT-1-v6). 2. Within 24 hours (i.e., within the cache-validity period for the MUD file) of running test IoT-1-v4 (or IoT-1-v6), verify that the IoT device that was connected during test IoT-1-v4 (or IoT-1-v6) is still up and running on the network. Power-on a second IoT device that has been configured to emit the same MUD URL as the device that was connected during test IoT-1-v4 (or IoT-1-v6), and connect it to the test network. This should set in motion the following series of steps, which should occur automatically: 3. IoT device automatically emits a DHCPv4 message containing the device's MUD URL (IANA code 161). (Note that in the v6 version of this test, IPv6, DHCPv6, and IANA code 112 will be used.) 4. DHCP server receives the DHCPv4 message containing the IoT device's MUD URL. 5. DHCP server offers an IP address lease to the newly connected IoT device. 6. The IoT device requests this IP address lease, which the DHCP server acknowledges. 7. DHCP server sends the MUD URL to the MUD manager. 8. The MUD manager determines that it has this MUD file cached and checks that the amount of time that has elapsed

Test Case Field	Description
	since the cached file was retrieved is less than or equal to the number of hours in the cache-validity value for this MUD file. If the cache validity has been exceeded, the MUD manager will fetch a new MUD file. 9. The MUD manager translates the MUD file's contents into appropriate route filtering rules and installs these rules onto the MUD PEP for the IoT device in question so that this router/switch is now configured to enforce the policies specified in the MUD file.
Expected Results	The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to enforce the policies specified in the IoT device's MUD file. The expected configuration should resemble the following: **Cache is valid** (the MUD manager does NOT retrieve the MUD file from the MUD file server)**:** Extended IP access list mud-81726-v4fr.in 10 permit tcp any host 192.168.4.7 eq www ack syn 20 permit tcp any host 192.168.10.104 eq www 30 permit tcp any host 192.168.10.105 eq www 50 permit tcp any 192.168.10.0 0.0.0.255 eq www 60 permit tcp any 192.168.13.0 0.0.0.255 eq www 70 permit tcp any 192.168.14.0 0.0.0.255 eq www 80 permit tcp any eq 22 any 81 permit udp any eq bootpc any eq bootps 82 permit udp any any eq domain 83 deny ip any any **Cache is valid** (the MUD manager does NOT retrieve the MUD file from the MUD file server)**:** Extended IP access list mud-81726-v4fr.in 10 permit tcp any host 192.168.4.7 eq www ack syn 20 permit tcp any host 192.168.10.104 eq www 30 permit tcp any host 192.168.10.105 eq www

Test Case Field	Description
	50 permit tcp any 192.168.10.0 0.0.0.255 eq www 60 permit tcp any 192.168.13.0 0.0.0.255 eq www 70 permit tcp any 192.168.14.0 0.0.0.255 eq www 80 permit tcp any eq 22 any 81 permit udp any eq bootpc any eq bootps 82 permit udp any any eq domain 83 deny ip any any **Cache is not valid** (the MUD manager does retrieve the MUD file from the MUD file server)**:** Extended IP access list mud-81726-v4fr.in 10 permit tcp any host 192.168.4.7 eq www ack syn 20 permit tcp any host 192.168.10.104 eq www 30 permit tcp any host 192.168.10.105 eq www 50 permit tcp any 192.168.10.0 0.0.0.255 eq www 60 permit tcp any 192.168.13.0 0.0.0.255 eq www 70 permit tcp any 192.168.14.0 0.0.0.255 eq www 80 permit tcp any eq 22 any 81 permit udp any eq bootpc any eq bootps 82 permit udp any any eq domain 83 deny ip any any All protocol exchanges described in steps 1–9 above are expected to occur and can be viewed via Wireshark if desired. If the router/switch does not get configured in accordance with the MUD file, each exchange of DHCP and MUD-related protocol traffic should be viewed on the network via Wireshark to determine which transactions did not proceed as expected, and the observed and absent protocol exchanges should be described here.

Actual Results	**MUD manager logs for valid cache:**
	
	`**MUDC [INFO][mudc_print_request_info:2185]--> print parsed HTTP request header info`
	`***MUDC [INFO][mudc_print_request_info:2186]--> request method: POST`

Test Case Field	Description
	***MUDC [INFO][mudc_print_request_info:2187]--> request uri: /getaclname ***MUDC [INFO][mudc_print_request_info:2188]--> local uri: /getaclname ***MUDC [INFO][mudc_print_request_info:2189]--> http version: 1.1 ***MUDC [INFO][mudc_print_request_info:2190]--> query string: (null) ***MUDC [INFO][mudc_print_request_info:2191]--> content_length: 27 ***MUDC [INFO][mudc_print_request_info:2192]--> remote ip addr: 0xe7719c38 ***MUDC [INFO][mudc_print_request_info:2193]--> remote port: 49344 ***MUDC [INFO][mudc_print_request_info:2194]--> remote_user: (null) ***MUDC [INFO][mudc_print_request_info:2195]--> is ssl: 0 ***MUDC [INFO][mudc_print_request_info:2199]--> header(0): name: <Host>, value: <127.0.0.1:8000> ***MUDC [INFO][mudc_print_request_info:2199]--> header(1): name: <User-Agent>, value: <FreeRADIUS 3.0.17> ***MUDC [INFO][mudc_print_request_info:2199]--> header(2): name: <Accept>, value: <*/*> ***MUDC [INFO][mudc_print_request_info:2199]--> header(3): name: <Content-Type>, value: <application/json> ***MUDC [INFO][mudc_print_request_info:2199]--> header(4): name: <X-FreeRADIUS-Section>, value: <authorize> ***MUDC [INFO][mudc_print_request_info:2199]--> header(5): name: <X-FreeRADIUS-Server>, value: <default> ***MUDC [INFO][mudc_print_request_info:2199]--> header(6): name: <Content-Length>, value: <27> ***MUDC [INFO][handle_get_aclname:2506]--> Mac address <b827ebeb6c8b> ***MUDC [INFO][fetch_uri_from_macaddr:1702]--> found the fields <{ "_id" : { "$oid" : "5c182c7edb40218cde918776" }, "URI" : "https://mudfileserver/ciscopi2" }> ***MUDC [INFO][fetch_uri_from_macaddr:1711]--> ============= Returning URI:https://mudfileserver/ciscopi2 ***MUDC [INFO][handle_get_aclname:2513]--> Found URI https://mudfileserver/ciscopi2 for MAC address b827ebeb6c8b

Test Case Field	Description

	***MUDC [INFO][validate_muduri:2373]--> uri: https://mudfileserver/ciscopi2
	***MUDC [INFO][validate_muduri:2399]--> ip: mud-fileserver, filename: ciscopi2
	***MUDC [INFO][handle_get_aclname:2558]--> Got URL from message <https://mudfileserver/ciscopi2>
	
	***MUDC [INFO][query_policies_by_uri:1419]--> found the record <{ "_id" : { "$oid" : "5c182d9cdb40218cde91884a" }, "DACL_Name" : "ACS:Cis-coSecure-Defined-ACL=mud-81726-v4fr.in", "DACL" : "[\"ip:inacl#10=permit tcp any host 192.168.4.7 range 80 80 syn ack\", \"ip:inacl#20=permit tcp any host 192.168.10.104 range 80 80\", \"ip:inacl#30=permit tcp any host 192.168.10.105 range 80 80\", \"ip:in-acl#40=permit tcp any host 192.168.10.104 range 80 80\", \"ip:inacl#50=permit tcp any 192.168.10.0 0.0.0.255 range 80 80\", \"ip:inacl#60=permit tcp any 192.168.13.0 0.0.0.255 range 80 80\", \"ip:in-acl#70=permit tcp any 192.168.14.0 0.0.0.255 range 80 80\", \"ip:inacl#80=permit tcp any eq 22 any\", \"ip:inacl#81=permit udp any eq 68 any eq 67\", \"ip:inacl#82=permit udp any any eq 53\", \"ip:in-acl#83=deny ip any any\"]", "URI" : "https://mud-fileserver/ciscopi2", "VLAN" : 3 }>
	
	***MUDC [INFO][query_policies_by_uri:1461]--> Re-sponse <{
	"Cisco-AVPair": ["ACS:CiscoSecure-Defined-ACL=mud-81726-v4fr.in"],
	"Tunnel-Type": "VLAN",
	"Tunnel-Medium-Type": "IEEE-802",
	"Tunnel-Private-Group-Id": 3
	}>
	
	***MUDC [INFO][mudc_construct_head:135]--> sta-tus_code: 200, content_len: 160, extra_headers: Con-tent-Type: application/aclname
	***MUDC [INFO][mudc_construct_head:152]--> HTTP header: HTTP/1.1 200 OK
	Content-Type: application/aclname
	Content-Length: 160
	
	***MUDC [INFO][query_policies_by_uri:1464]--> {
	"Cisco-AVPair": ["ACS:CiscoSecure-Defined-ACL=mud-81726-v4fr.in"],
	"Tunnel-Type": "VLAN",
	"Tunnel-Medium-Type": "IEEE-802",
	"Tunnel-Private-Group-Id": 3
	}

Test Case Field	Description
	***MUDC [INFO][handle_get_aclname:2568]--> Got ACLs from the MUD URL MUD Manager logs for expired cache: ***MUDC [INFO][mudc_print_request_info:2185]--> print parsed HTTP request header info ***MUDC [INFO][mudc_print_request_info:2186]--> request method: POST ***MUDC [INFO][mudc_print_request_info:2187]--> request uri: /getaclname ***MUDC [INFO][mudc_print_request_info:2188]--> local uri: /getaclname ***MUDC [INFO][mudc_print_request_info:2189]--> http version: 1.1 ***MUDC [INFO][mudc_print_request_info:2190]--> query string: (null) ***MUDC [INFO][handle_get_aclname:2506]--> Mac address <**b827ebeb6c8b**> ***MUDC [INFO][fetch_uri_from_macaddr:1702]--> found the fields <{ "_id" : { "$oid" : "5c182c7edb40218cde918776" }, "URI" : "https://mud-fileserver/ciscopi2" }> ***MUDC [INFO][fetch_uri_from_macaddr:1711]--> ============= Returning URI:https://mud-fileserver/ciscopi2 ***MUDC [INFO][handle_get_aclname:2513]--> **Found URI** https://mudfileserver/ciscopi2 **for MAC address b827ebeb6c8b** ***MUDC [INFO][validate_muduri:2373]--> uri: https://mudfileserver/ciscopi2 ***MUDC [INFO][validate_muduri:2399]--> ip: mud-fileserver, filename: ciscopi2 ***MUDC [INFO][handle_get_aclname:2558]--> **Got URL from message <https://mudfileserver/ciscopi2>** ***MUDC [INFO][query_policies_by_uri:1399]--> **Cache has expired** [omitted for sake of length] ***MUDC [STATUS][send_mudfs_request:2005]--> Request URI <https://mudfileserver/ciscopi2> </home/mudtester/mud-intermediate.pem>

Test Case Field	Description

	`*   Trying 192.168.4.5...`
	`* TCP_NODELAY set`
	`* Connected to mudfileserver (192.168.4.5) port 443 (#0)`
	`* found 1 certificate in /home/mudtester/mud-interme-diate.pem`
	`* found 400 certificates in /etc/ssl/certs`
	`* ALPN, offering http/1.1`
	`* SSL connection using TLS1.2 / EC-DHE_RSA_AES_256_GCM_SHA384`
	`*       server certificate verification OK`
	`*       server certificate status verification SKIPPED`
	`*       common name: mudfileserver (matched)`
	`*       server certificate expiration date OK`
	`*       server certificate activation date OK`
	`*       certificate public key: RSA`
	`*       certificate version: #3`
	`*       subject: C=US,ST=Maryland,L=Rockville,O=Na-tional Cybersecurity Center of Excellence - NIST,CN=mudfileserver`
	`*       start date: Fri, 05 Oct 2018 00:00:00 GMT`
	`*       expire date: Wed, 13 Oct 2021 12:00:00 GMT`
	`*       issuer: C=US,O=DigiCert Inc,CN=DigiCert Test SHA2 Intermediate CA-1`
	`*       compression: NULL`
	`* ALPN, server did not agree to a protocol`
	`> GET /ciscopi2 HTTP/1.1`
	`Host: mudfileserver`
	`Accept: */*`
	
	[omitted for sake of length]

Overall Results	Pass

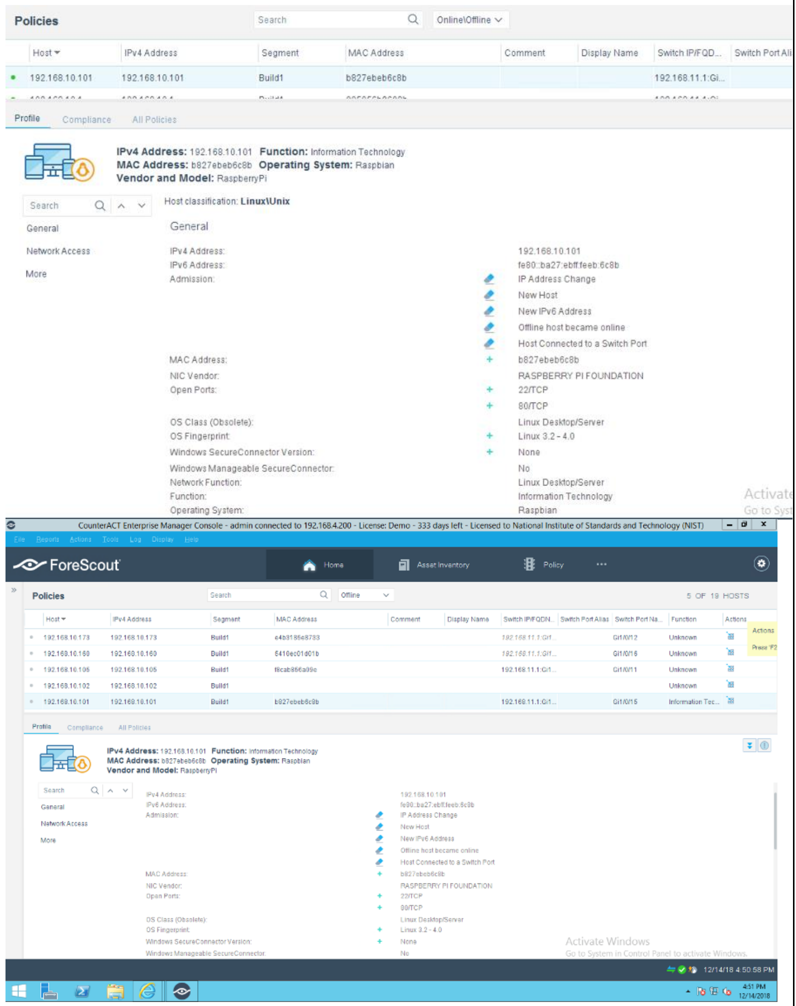
1678 Test Case IoT-12-v6 is identical to test case IoT-12-v4 except that IoT-1-v6 tests requirement CR-1.a.2,
1679 whereas IoT-1-v4 tests requirement CR-1.a.1. Hence, as explained above, test IoT-1-v6 it uses IPv6,
1680 DHCPv6, and IANA code 112 instead of using IPv4, DHCPv4, and IANA code 161.

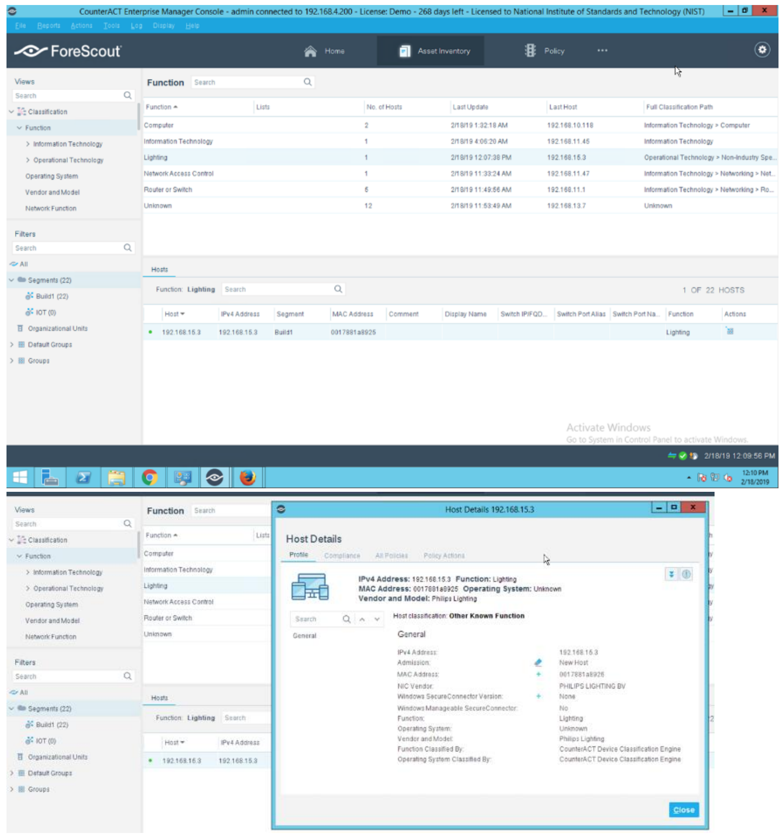
D.1.2.13 Test Case IoT-13-v4

Table D-15: Test Case IoT-13-v4

Test Case Field	Description
Parent Requirements	(CR 16) The IoT DDoS example implementation shall include visibility component that is able to detect, identify, categorize, and monitor the status of IoT devices that are on the network.
Testable Requirements	(CR 16.a) The visibility component shall detect and identify the attributes and category of a newly connected IoT device. (CR 16.a.1) The visibility component shall monitor the status of the IoT device (e.g., notice if the device goes offline).
Description	Shows that the IoT DDoS example implementation includes a visibility component that is capable of performing the following: upon connection of a live IoT device to the network, the device will be detected; identified in terms of attributes such as its IP address, operating system, device type, etc.; and continuously monitored as long as it remains live on the network. If the device becomes disconnected or turns off, this change of status will also be detected.
Associated Test Cases	N/A
Associated Cybersecurity Framework Subcategories	ID.AM-1, ID.AM-2, ID.AM-3, DE.AE-1, DE.CM-1
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	Not applicable for this test
Preconditions	1. The visibility component is up and running and attached to the network.

Test Case Field	Description
Procedure	1. Power on a device and connect it to the network. 2. Verify that the device was detected by the visibility component and that its type, address, operating system (OS), and other features were identified, and device was categorized correctly. 3. Turn off the device. 4. Verify that its absence from the network is detected. 5. Power the device back on. 6. Verify that its presence is detected and its features are identified correctly. 7. Disconnect the device from the network. 8. Verify that its absence from the network is detected.
Expected Results	All expectations as enumerated in items 2, 4, 6, and 8 above are observed.
Actual Results	**At Power-On:** pi@raspberrypi:~ $ ifconfig eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500 inet 192.168.10.101 netmask 255.255.255.0 broadcast 192.168.10.255 ether b8:27:eb:eb:6c:8b txqueuelen 1000 (Ethernet) RX packets 9193 bytes 8208593 (7.8 MiB) RX errors 0 dropped 5 overruns 0 frame 0 TX packets 7210 bytes 822414 (803.1 KiB) TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0 lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536 inet 127.0.0.1 netmask 255.0.0.0 inet6 ::1 prefixlen 128 scopeid 0x10<host> loop txqueuelen 1000 (Local Loopback) RX packets 16 bytes 1467 (1.4 KiB) RX errors 0 dropped 0 overruns 0 frame 0 TX packets 16 bytes 1467 (1.4 KiB) TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

Test Case Field	Description
	Screenshot from ForeScout: IoT device status is indicated by green or gray light shown in the screen capture. 
	Categorizing IoT device: We tested this function with a smart light bulb. See the example screenshots below.

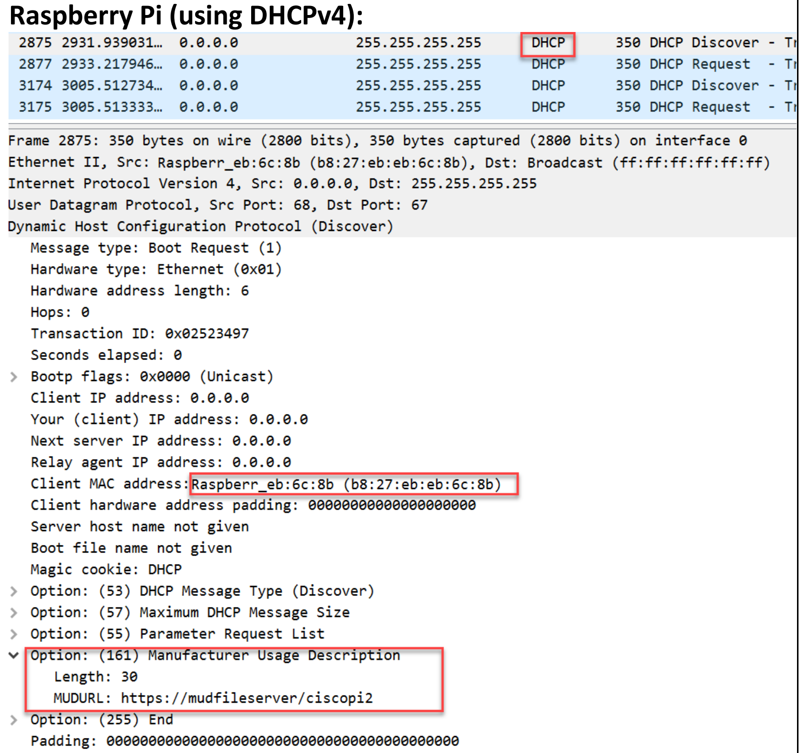
Test Case Field	Description
	
Overall Results	Pass

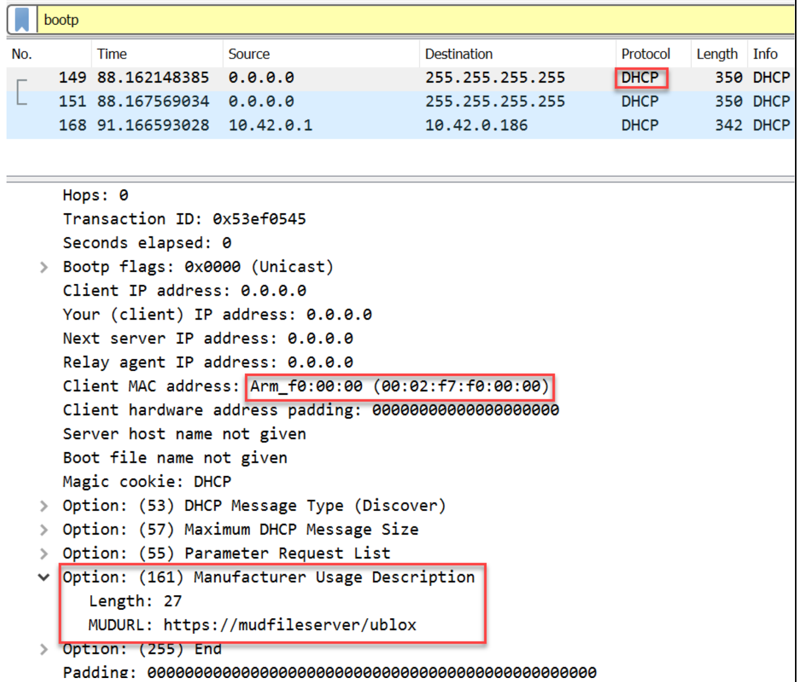
D.1.2.14 Test Case IoT-14-v4

Table D-16: Test Case IoT-14-v4

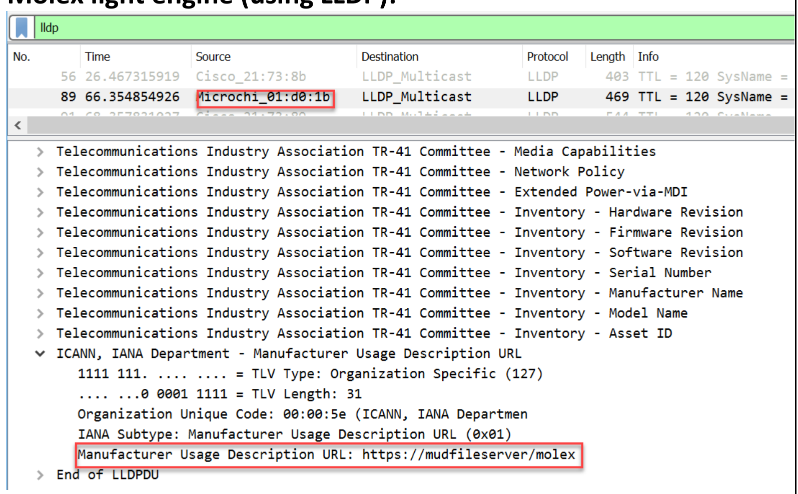
Test Case Field	Description
Parent Requirements	(CR 1) The IoT DDoS example implementation shall include a MUD-enabled IoT device that can emit a MUD URL.

Test Case Field	Description
Testable Requirements	(CR 1.a) Upon initialization, the MUD-enabled IoT device shall broadcast a DHCP message on the network, including at most one MUD URL, in https scheme, within the DHCP transaction. (CR 1.a.1) The DHCP server shall be able to receive **DHCPv4 DISCOVER and/or REQUEST with IANA code 161** (OPTION_MUD_URL_V4) from the MUD-enabled IoT device OR (CR 1.b) Upon initialization, the MUD-enabled IoT device shall emit the MUD URL as an LLDP extension. (CR 1.b.1) The network service shall be able to **process** the MUD URL that is received as an LLDP e**xtension.**
Description	Shows that the IoT DDoS example implementation includes IoT devices that are capable of emitting a MUD URL via DHCP or LLDP.
Associated Test Cases	N/A
Associated Cybersecurity Framework Subcategories	ID.AM-1
IoT Device(s) Under Test	Raspberry Pi, Molex light engine, u-blox C027-G35
MUD File(s) Used	Ciscopi2.json, molex.json, ublox.json
Preconditions	1. Device has been developed to emit MUD URL in DHCP transaction.

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td>Procedure</td><td>1. Power on a device and connect it to the network.
2. Verify that the device emits MUD URL in DHCP transaction.
a. Use Wireshark to capture DHCP transaction with options present.</td></tr>
<tr><td>Expected Results</td><td>DHCP transaction with MUD option 161 or LLDP TLV MUD extension enabled and MUD URL included.</td></tr>
<tr><td>Actual Results</td><td>Raspberry Pi (using DHCPv4):

2875 2931.939031… 0.0.0.0 255.255.255.255 DHCP 350 DHCP Discover - Tr
2877 2933.217946… 0.0.0.0 255.255.255.255 DHCP 350 DHCP Request - Tr
3174 3005.512734… 0.0.0.0 255.255.255.255 DHCP 350 DHCP Discover - Tr
3175 3005.513333… 0.0.0.0 255.255.255.255 DHCP 350 DHCP Request - Tr
Frame 2875: 350 bytes on wire (2800 bits), 350 bytes captured (2800 bits) on interface 0
Ethernet II, Src: Raspberr_eb:6c:8b (b8:27:eb:eb:6c:8b), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 68, Dst Port: 67
Dynamic Host Configuration Protocol (Discover)
Message type: Boot Request (1)
Hardware type: Ethernet (0x01)
Hardware address length: 6
Hops: 0
Transaction ID: 0x02523497
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0
Your (client) IP address: 0.0.0.0
Next server IP address: 0.0.0.0
Relay agent IP address: 0.0.0.0
Client MAC address: Raspberr_eb:6c:8b (b8:27:eb:eb:6c:8b)
Client hardware address padding: 00000000000000000000
Server host name not given
Boot file name not given
Magic cookie: DHCP
Option: (53) DHCP Message Type (Discover)
Option: (57) Maximum DHCP Message Size
Option: (55) Parameter Request List
Option: (161) Manufacturer Usage Description
Length: 30
MUDURL: https://mudfileserver/ciscopi2
Option: (255) End
Padding: 000000000000000000000000000000000000000000
u-blox C027-G35 (using DHCPv4):</td></tr>
</table>

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td></td><td>


Molex light engine (using LLDP):



</td></tr>
<tr><td>Overall Results</td><td>Pass</td></tr>
</table>

D.1.3 Build 1 MUD Files

D.1.3.1 Ciscopi2.json


```
{
    "ietf-mud:mud": {
        "mud-version": 1,
        "mud-url":"https://mudfileserver/ciscopi2",
        "last-update": "2018-12-05T19:42:01+00:00",
        "cache-validity": 24,
        "is-supported": true,
        "systeminfo": "ingress/egress ",
        "from-device-policy": {
            "access-lists": {
                "access-list": [{
                    "name": "mud-81726-v4fr"
                }]
            }
        },
        "to-device-policy": {
            "access-lists": {
                "access-list": [{
                    "name": "mud-81726-v4to"
                }]
            }
        }
    },
    "ietf-access-control-list:acls": {
        "acl": [{
                "name": "mud-81726-v4to",
                "type": "ipv4-acl-type",
                "aces": {
                    "ace": [{
                            "name": "cl0-todev",
                            "matches": {
                                "ipv4": {
                                    "ietf-acldns:src-dnsname": "www.updateserver.com",
                                    "protocol": 6
                                },
                                "tcp": {
                                    "ietf-mud:direction-initiated": "from-device",
                                    "source-port": {
                                        "operator": "eq",
                                        "port": 80
                                    }
                                }
                            },
                            "actions": {
                                "forwarding": "accept"
                            }
                        },
                        {
```

```
    "name": "ent0-todev",
    "matches": {
        "ietf-mud:mud": {
            "controller": "http://lightcontroller.example.com"
        },
        "ipv4": {
            "protocol": 6
        },
        "tcp": {
            "source-port": {
                "operator": "eq",
                "port": 80
            }
        }
    },
    "actions": {
        "forwarding": "accept"
    }
},
{
    "name": "ent1-todev",
    "matches": {
        "ietf-mud:mud": {
            "controller": "http://lightcontroller.example2.com"
        },
        "ipv4": {
            "protocol": 6
        },
        "tcp": {
            "source-port": {
                "operator": "eq",
                "port": 80
            }
        }
    },
    "actions": {
        "forwarding": "accept"
    }
},
{
    "name": "myctl0-todev",
    "matches": {
        "ietf-mud:mud": {
            "my-controller": [
                null
            ]
        },
        "ipv4": {
            "protocol": 6
        },
        "tcp": {
            "source-port": {
```

```
                "operator": "eq",
                "port": 80
            }
        }
    },
    "actions": {
        "forwarding": "accept"
    }
},
{
    "name": "loc0-todev",
    "matches": {
        "ietf-mud:mud": {
            "local-networks": [
                null
            ]
        },
        "ipv4": {
            "protocol": 6
        },
        "tcp": {
            "source-port": {
                "operator": "eq",
                "port": 80
            }
        }
    },
    "actions": {
        "forwarding": "accept"
    }
},
{
    "name": "myman0-todev",
    "matches": {
        "ietf-mud:mud": {
            "same-manufacturer": [
                null
            ]
        },
        "ipv4": {
            "protocol": 6
        },
        "tcp": {
            "source-port": {
                "operator": "eq",
                "port": 80
            }
        }
    },
    "actions": {
        "forwarding": "accept"
    }
```

```
            },
            {
                "name": "man0-todev",
                "matches": {
                    "ietf-mud:mud": {
                        "manufacturer": "www.devicetype.com"
                    },
                    "ipv4": {
                        "protocol": 6
                    },
                    "tcp": {
                        "source-port": {
                            "operator": "eq",
                            "port": 80
                        }
                    }
                },
                "actions": {
                    "forwarding": "accept"
                }
            }

        ]
    }
},
{
    "name": "mud-81726-v4fr",
    "type": "ipv4-acl-type",
    "aces": {
        "ace": [{
                "name": "cl0-frdev",
                "matches": {
                    "ipv4": {
                        "ietf-acldns:dst-dnsname": "www.updateserver.com",
                        "protocol": 6
                    },
                    "tcp": {
                        "ietf-mud:direction-initiated": "from-device",
                        "destination-port": {
                            "operator": "eq",
                            "port": 80
                        }
                    }
                },
                "actions": {
                    "forwarding": "accept"
                }
            },
            {
                "name": "ent0-frdev",
                "matches": {
                    "ietf-mud:mud": {
```

```
            "controller": "http://lightcontroller.example.com"
        },
        "ipv4": {
            "protocol": 6
        },
        "tcp": {
            "destination-port": {
                "operator": "eq",
                "port": 80
            }
        }
    },
    "actions": {
        "forwarding": "accept"
    }
},
{
    "name": "ent1-frdev",
    "matches": {
        "ietf-mud:mud": {
            "controller": "http://lightcontroller.example2.com"
        },
        "ipv4": {
            "protocol": 6
        },
        "tcp": {
            "destination-port": {
                "operator": "eq",
                "port": 80
            }
        }
    },
    "actions": {
        "forwarding": "accept"
    }
},
{
    "name": "myctl0-frdev",
    "matches": {
        "ietf-mud:mud": {
            "my-controller": [
                null
            ]
        },
        "ipv4": {
            "protocol": 6
        },
        "tcp": {
            "destination-port": {
                "operator": "eq",
                "port": 80
            }
```

```
        }
    },
    "actions": {
        "forwarding": "accept"
    }
},
{
    "name": "loc0-frdev",
    "matches": {
        "ietf-mud:mud": {
            "local-networks": [
                null
            ]
        },
        "ipv4": {
            "protocol": 6
        },
        "tcp": {
            "destination-port": {
                "operator": "eq",
                "port": 80
            }
        }
    },
    "actions": {
        "forwarding": "accept"
    }
},
{
    "name": "myman0-frdev",
    "matches": {
        "ietf-mud:mud": {
            "same-manufacturer": [
                null
            ]
        },
        "ipv4": {
            "protocol": 6
        },
        "tcp": {
            "destination-port": {
                "operator": "eq",
                "port": 80
            }
        }
    },
    "actions": {
        "forwarding": "accept"
    }
},
{
    "name": "man0-frdev",
```

```
                            "matches": {
                                "ietf-mud:mud": {
                                    "manufacturer": "www.devicetype.com"
                                },
                                "ipv4": {
                                    "protocol": 6
                                },
                                "tcp": {
                                    "destination-port": {
                                        "operator": "eq",
                                        "port": 80
                                    }
                                }
                            },
                            "actions": {
                                "forwarding": "accept"
                            }
                        }
                    ]
                }
            }
        ]
    }
}
```


D.1.3.2 expiredcerttest.json


```
{
    "ietf-mud:mud": {
        "mud-version": 1,
        "mud-url":"https://mudfileserver/expiredcerttest.json",
        "last-update": "2018-12-05T19:42:01+00:00",
        "cache-validity": 24,
        "is-supported": true,
        "systeminfo": "ingress/egress ",
        "from-device-policy": {
            "access-lists": {
                "access-list": [{
                    "name": "mud-81726-v4fr"
                }]
            }
        },
        "to-device-policy": {
            "access-lists": {
                "access-list": [{
                    "name": "mud-81726-v4to"
                }]
            }
        }
    },
    "ietf-access-control-list:acls": {
        "acl": [{
```

```
"name": "mud-81726-v4to",
"type": "ipv4-acl-type",
"aces": {
    "ace": [{
            "name": "cl0-todev",
            "matches": {
                "ipv4": {
                    "ietf-acldns:src-dnsname": "www.updateserver.com",
                    "protocol": 6
                },
                "tcp": {
                    "ietf-mud:direction-initiated": "from-device",
                    "source-port": {
                        "operator": "eq",
                        "port": 80
                    }
                }
            },
            "actions": {
                "forwarding": "accept"
            }
        },
        {
            "name": "ent0-todev",
            "matches": {
                "ietf-mud:mud": {
                    "controller":"http://lightcontroller.example.com"
                },
                "ipv4": {
                    "protocol": 6
                },
                "tcp": {
                    "source-port": {
                        "operator": "eq",
                        "port": 80
                    }
                }
            },
            "actions": {
                "forwarding": "accept"
            }
        },
        {
            "name": "ent1-todev",
            "matches": {
                "ietf-mud:mud": {
                    "controller": "http://lightcontroller.example2.com"
                },
                "ipv4": {
                    "protocol": 6
                },
                "tcp": {
```

```
                        "source-port": {
                            "operator": "eq",
                            "port": 80
                        }
                    }
                },
                "actions": {
                    "forwarding": "accept"
                }
            },
            {
                "name": "myctl0-todev",
                "matches": {
                    "ietf-mud:mud": {
                        "my-controller": [
                            null
                        ]
                    },
                    "ipv4": {
                        "protocol": 6
                    },
                    "tcp": {
                        "source-port": {
                            "operator": "eq",
                            "port": 80
                        }
                    }
                },
                "actions": {
                    "forwarding": "accept"
                }
            },
            {
                "name": "loc0-todev",
                "matches": {
                    "ietf-mud:mud": {
                        "local-networks": [
                            null
                        ]
                    },
                    "ipv4": {
                        "protocol": 6
                    },
                    "tcp": {
                        "source-port": {
                            "operator": "eq",
                            "port": 80
                        }
                    }
                },
                "actions": {
                    "forwarding": "accept"
```

```
                }
            },
            {
                "name": "myman0-todev",
                "matches": {
                    "ietf-mud:mud": {
                        "same-manufacturer": [
                            null
                        ]
                    },
                    "ipv4": {
                        "protocol": 6
                    },
                    "tcp": {
                        "source-port": {
                            "operator": "eq",
                            "port": 80
                        }
                    }
                },
                "actions": {
                    "forwarding": "accept"
                }
            },
            {
                "name": "man0-todev",
                "matches": {
                    "ietf-mud:mud": {
                        "manufacturer": "www.devicetype.com"
                    },
                    "ipv4": {
                        "protocol": 6
                    },
                    "tcp": {
                        "source-port": {
                            "operator": "eq",
                            "port": 80
                        }
                    }
                },
                "actions": {
                    "forwarding": "accept"
                }
            }

        ]
    }
},
{
    "name": "mud-81726-v4fr",
    "type": "ipv4-acl-type",
    "aces": {
```

```
"ace": [{
        "name": "cl0-frdev",
        "matches": {
            "ipv4": {
                "ietf-acldns:dst-dnsname": "www.updateserver.com",
                "protocol": 6
            },
            "tcp": {
                "ietf-mud:direction-initiated": "from-device",
                "destination-port": {
                    "operator": "eq",
                    "port": 80
                }
            }
        },
        "actions": {
            "forwarding": "accept"
        }
    },
    {
        "name": "ent0-frdev",
        "matches": {
            "ietf-mud:mud": {
                "controller": "http://lightcontroller.example.com"
            },
            "ipv4": {
                "protocol": 6
            },
            "tcp": {
                "destination-port": {
                    "operator": "eq",
                    "port": 80
                }
            }
        },
        "actions": {
            "forwarding": "accept"
        }
    },
    {
        "name": "ent1-frdev",
        "matches": {
            "ietf-mud:mud": {
                "controller": "http://lightcontroller.example2.com"
            },
            "ipv4": {
                "protocol": 6
            },
            "tcp": {
                "destination-port": {
                    "operator": "eq",
                    "port": 80
```

```
                        }
                    }
                },
                "actions": {
                    "forwarding": "accept"
                }
            },
            {
                "name": "myctl0-frdev",
                "matches": {
                    "ietf-mud:mud": {
                        "my-controller": [
                            null
                        ]
                    },
                    "ipv4": {
                        "protocol": 6
                    },
                    "tcp": {
                        "destination-port": {
                            "operator": "eq",
                            "port": 80
                        }
                    }
                },
                "actions": {
                    "forwarding": "accept"
                }
            },
            {
                "name": "loc0-frdev",
                "matches": {
                    "ietf-mud:mud": {
                        "local-networks": [
                            null
                        ]
                    },
                    "ipv4": {
                        "protocol": 6
                    },
                    "tcp": {
                        "destination-port": {
                            "operator": "eq",
                            "port": 80
                        }
                    }
                },
                "actions": {
                    "forwarding": "accept"
                }
            },
            {
```

```
                            "name": "myman0-frdev",
                            "matches": {
                                "ietf-mud:mud": {
                                    "same-manufacturer": [
                                        null
                                    ]
                                },
                                "ipv4": {
                                    "protocol": 6
                                },
                                "tcp": {
                                    "destination-port": {
                                        "operator": "eq",
                                        "port": 80
                                    }
                                }
                            },
                            "actions": {
                                "forwarding": "accept"
                            }
                        },
                        {
                            "name": "man0-frdev",
                            "matches": {
                                "ietf-mud:mud": {
                                    "manufacturer": "www.devicetype.com"
                                },
                                "ipv4": {
                                    "protocol": 6
                                },
                                "tcp": {
                                    "destination-port": {
                                        "operator": "eq",
                                        "port": 80
                                    }
                                }
                            },
                            "actions": {
                                "forwarding": "accept"
                            }
                        }
                    ]
                }
            }
        ]
    }
}
```


D.1.3.3 Molex.json


```
{
```

```
        "ietf-mud:mud": {
                "mud-version": 1,
                "mud-url": "https://mudfileserver/molex",
                "last-update": "2019-02-05T13:18:04+00:00",
                "cache-validity": 48,
                "is-supported": true,
                "systeminfo": "Molex Coresync POE Gateway",
                "mfg-name": "Molex",
                "documentation": "https://www.molex.com/coresync/documentation/gateway",
                "model-name": "coresync",
                "from-device-policy": {
                        "access-lists": {
                                "access-list": [{
                                        "name": "mud-10436-v4fr"
                                }]
                        }
                },
                "to-device-policy": {
                        "access-lists": {
                                "access-list": [{
                                        "name": "mud-10436-v4to"
                                }]
                        }
                }
        },
        "ietf-access-control-list:acls": {
                "acl": [{
                                "name": "mud-10436-v4to",
                                "type": "ipv4-acl-type",
                                "aces": {
```

```
        "ace": [{
                "name": "myctl0-todev",
                "matches": {
                    "ietf-mud:mud": {
                        "my-controller": [
                            null
                        ]
                    }
                },
                "actions": {
                    "forwarding": "accept"
                }
            },
            {
                "name": "loc0-todev",
                "matches": {
                    "ietf-mud:mud": {
                        "local-networks": [
                            null
                        ]
                    }
                },
                "actions": {
                    "forwarding": "accept"
                }
            }
        ]
    }
},
{
```

```
            "name": "mud-10436-v4fr",
            "type": "ipv4-acl-type",
            "aces": {
                "ace": [{
                        "name": "myctl0-frdev",
                        "matches": {
                            "ietf-mud:mud": {
                                "my-controller": [
                                    null
                                ]
                            }
                        },
                        "actions": {
                            "forwarding": "accept"
                        }
                    },
                    {
                        "name": "loc0-frdev",
                        "matches": {
                            "ietf-mud:mud": {
                                "local-networks": [
                                    null
                                ]
                            }
                        },
                        "actions": {
                            "forwarding": "accept"
                        }
                    }
                ]
```

```
                    }
                }
            ]
        }
}
```

D.1.3.4 u-blox.json


```
{
        "ietf-mud:mud": {
                "mud-version": 1,
                "mud-url": "https://mudfileserver/ublox",
                "last-update": "2018-10-31T16:23:26+00:00",
                "cache-validity": 48,
                "is-supported": true,
                "systeminfo": "configurations for ipv4 pi connected to cisco switch",
                "from-device-policy": {
                        "access-lists": {
                                "access-list": [{
                                        "name": "mud-65989-v4fr"
                                }]
                        }
                },
                "to-device-policy": {
                        "access-lists": {
                                "access-list": [{
                                        "name": "mud-65989-v4to"
                                }]
                        }
                }
        },
        "ietf-access-control-list:acls": {
```

```
            "acl": [{
                    "name": "mud-65989-v4to",
                    "type": "ipv4-acl-type",
                    "aces": {
                        "ace": [{
                                "name": "cl0-todev",
                                "matches": {
                                    "ipv4": {
                                        "ietf-acldns:src-dnsname":
"www.updateserver.com",
                                        "protocol": 6
                                    },
                                    "tcp": {
                                        "ietf-mud:direction-
initiated": "from-device"
                                    }
                                },
                                "actions": {
                                    "forwarding": "accept"
                                }
                            },
                            {
                                "name": "cl1-todev",
                                "matches": {
                                    "ipv4": {
                                        "ietf-acldns:src-dnsname":
"www.nossl.net",
                                        "protocol": 6
                                    }
                                },
                                "actions": {
                                    "forwarding": "accept"
```

```
                                        }
                                }
                        ]
                }
        },
        {
                "name": "mud-65989-v4fr",
                "type": "ipv4-acl-type",
                "aces": {
                        "ace": [{
                                        "name": "cl0-frdev",
                                        "matches": {
                                                "ipv4": {
                                                        "ietf-acldns:dst-dnsname": "www.updateserver.com",
                                                        "protocol": 6
                                                },
                                                "tcp": {
                                                        "ietf-mud:direction-initiated": "from-device"
                                                }
                                        },
                                        "actions": {
                                                "forwarding": "accept"
                                        }
                                },
                                {
                                        "name": "cl1-frdev",
                                        "matches": {
                                                "ipv4": {
                                                        "ietf-acldns:dst-dnsname": "www.nossl.net",
```

```
                                    "protocol": 6
                                }
                            },
                            "actions": {
                                "forwarding": "accept"
                            }
                        }
                    ]
                }
            }
        ]
    }
}
```

Appendix E References

E.1 Core Standards

"Dynamic Host Configuration Protocol," Request for Comments (RFC) 2131, Mar. 1997. See http://www.rfc-editor.org/info/rfc2131

"HTTP Over TLS," RFC 2818, May 2000. See http://www.rfc-editor.org/info/rfc2818

"Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280, May 2008. See http://www.rfc-editor.org/info/rfc5280

"Cryptographic Message Syntax (CMS)," RFC 5652, Sept. 2009. See http://www.rfc-editor.org/info/rfc5652

"YANG—A Data Modeling Language for the Network Configuration Protocol (NETCONF)," RFC 6020, Oct. 2010. See http://www.rfc-editor.org/info/rfc6020

"Manufacturer Usage Description Specification," RFC 8520, ISSN: 2070-1721, Mar. 2019. See https://datatracker.ietf.org/doc/rfc8520/

IEEE802.1AB Link Layer Discovery Protocol (LLDP)

E.2 Ongoing MUD Standards Activities

K. Boeckl et al., "Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks," Draft, National Institute of Standards and Technology (NIST) Interagency/Internal Report 8228, Sept. 2018. See https://csrc.nist.gov/publications/detail/nistir/8228/draft

S. Rich and T. Dahm, "MUD Lifecyle: A Network Operator's Perspective," Mar. 12, 2017. See https://tools.ietf.org/html/draft-srich-opsawg-mud-net-lifecycle-01

S. Rich and T. Dahm, "MUD Lifecyle: A Manufacturer's Perspective," Mar. 27, 2017. See https://tools.ietf.org/html/draft-srich-opsawg-mud-manu-lifecycle-01

E.3 Secure Update Standards

NIST Special Publication (SP) 800-40, *Guide to Enterprise Patch Management Technologies*. See https://csrc.nist.gov/publications/detail/sp/800-40/rev-3/final

NIST SP 800-147, *BIOS Protection Guidelines,* and NIST SP 800-147B, *BIOS Protection Guidelines for Servers.* See https://csrc.nist.gov/publications/detail/sp/800-147/final and https://nvl-pubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-147B.pdf

NIST Interagency/Internal Report 7823, *Advanced Metering Infrastructure Smart Meter Upgradeability Test Framework.* See http://csrc.nist.gov/publications/drafts/nistir-7823/draft_nistir-7823.pdf

NIST SP 800-193, *Platform Firmware Resiliency Guidelines*. See https://csrc.nist.gov/publications/detail/sp/800-193/draft

Multi-stakeholder Working Group for Secure Update of IoT devices (ongoing and established by the National Telecommunications Information Administration as part of its Internet Policy Task Force). See https://www.ntia.doc.gov/category/internet-things

E.4 Industry Best Practices for Software Quality

SANS TOP 25 Most Dangerous Software Errors, SANS Institute. See https://www.sans.org/top25-software-errors/

E.5 Best Practices for Identification and Authentication

NIST SP 800-63-3, *Digital Identity Guidelines*. See https://csrc.nist.gov/publications/detail/sp/800-63/3/final

NIST SP 800-63-B, *Digital Identity Guidelines: Authentication and Lifecycle Management*. See https://csrc.nist.gov/publications/detail/sp/800-63b/final

FIDO Alliance specifications. See https://fidoalliance.org/specifications/overview/

E.6 Cryptographic Standards and Best Practices

OMB Circular A-130, Executive Office of the President, Office of Management and Budget, *Managing Federal Information as a Strategic Resource,* July 28, 2016. See https://obamawhitehouse.archives.gov/omb/circulars_a130_a130trans4/

NIST SP 800-57 Part 1 Revision 4, *Recommendation for Key Management*. See https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdfh

NIST SP 800-52 Revision 1, *Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations*. See http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r1.pdf

E.7 Risk, Risk Assessment, and Risk Management Guidance

NIST SP 800-30, *Risk Management Guide for Information Technology Systems.* See https://www.hhs.gov/sites/default/files/ocr/privacy/hipaa/administrative/securityrule/nist800-30.pdf

NIST SP 800-30 Revision 1, *Guide for Conducting Risk Assessments*. See https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-30r1.pdf

NIST SP 800-37 Revision 2, *Risk Management Framework for Information Systems and Organizations*. See https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-37r2.pdf

NIST Computer Security Resource Center Risk Management Framework guidance. See https://csrc.nist.gov/projects/risk-management/risk-management-framework-quick-start-guides

NIST Interagency/Internal Report 8228, *Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks* (Draft). See https://csrc.nist.gov/publications/detail/nistir/8228/draft

NIST SP 800-53 Rev. 5, *Security and Privacy Controls for Information Systems and Organizations* (Draft). See https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/draft

NIST Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1, April 16, 2018. See https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf