NIST SPECIAL PUBLICATION 1800-15C

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

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

Volume C:
How-To Guides

Murugiah Souppaya
NIST

Yemi Fashina
Parisa Grayeli
Joshua Klosterman
Blaine Mulugeta
The MITRE Corporation

Eliot Lear
Cisco

William C. Barker
Dakota Consulting

April 2019

PRELIMINARY DRAFT

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

DISCLAIMER

Certain commercial entities, equipment, products, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST or NCCoE, nor is it intended to imply that the entities, equipment, products, or materials are necessarily the best available for the purpose.

National Institute of Standards and Technology Special Publication 1800-15C, Natl. Inst. Stand. Technol. Spec. Publ. 1800-15C, 78 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: August 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 providers.

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
Darshak Thakore	CableLabs
Mark Walker	CableLabs
Tao Wan	CableLabs
Russ Gyurek	Cisco
Peter Romness	Cisco
Brian Weis	Cisco
Rob Cantu	CTIA
Dean Coclin	DigiCert
Clint Wilson	DigiCert
Katherine Gronberg	ForeScout
Tim Jones	ForeScout
Adnan Baykal	Global Cyber Alliance
Drew Cohen	MasterPeace Solutions

Name	Organization
Nate Lesser	MasterPeace Solutions
Tom Martz	MasterPeace Solutions
Daniel Weller	MasterPeace Solutions
Kevin Yeich	MasterPeace Solutions
Mo Alhroub	Molex
Jaideep Singh	Molex
Bill Haag	NIST
Bryan Dubois	Patton Electronics
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

Name	Organization
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
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

Technology Partner/Collaborator	Build Involvement
Patton Electronics	Session Border Controller—SN5301/4B/EUI
Symantec	Subject Matter Expertise

Contents

158 # List of Figures

List of Tables

1 Introduction

This guide shows information technology (IT) professionals and security engineers how we implemented the example Manufacturer Usage Description (MUD)-based solution for mitigating Internet of Things (IoT)-based attacks by securing home and small-business IoT devices. We cover all of the products employed in this reference design. We do not re-create the product manufacturers' documentation, which is presumed to be widely available. Rather, these volumes show how we incorporated the products together in our environment.

Note: This is not a comprehensive tutorial. There are many possible service and security configurations for these products that are out of scope for this reference design.

1.1 Practice Guide Structure

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 the MUD-based solution for mitigating network-based attacks by securing home and small-business IoT devices. This reference design is modular and can be deployed in whole or in part.

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
- NIST SP 1800-15C: *How-To Guides*–instructions for building the example solution **(you are here)**

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

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

- challenges enterprises face in trying to mitigate network-based attacks by securing home and small-business IoT devices
- 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 NIST SP 1800-15B, which describes what we did and why. The following sections will be of particular interest:

- Section 3.4, Risk Assessment, 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 your leadership team members to help them understand the importance of adopting a standards-based solution for mitigating network-based attacks by securing home and small-business IoT devices.

IT professionals who want to implement an approach like this will find this whole practice guide useful. You can use this how-to portion of the guide, NIST SP 1800-15C, to replicate all or parts of the build created in our lab. This how-to portion of the 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 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-based solution. Your organization's security experts should identify the products that will best integrate with your existing tools and IT system infrastructure. We hope that you will seek products that are congruent with applicable standards and best practices. Section 4.3, Technologies, of NIST SP 1800-15B lists the products that we used and 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.

1.2 Build Overview

This NIST Cybersecurity Practice Guide addresses the challenge of using standards-based protocols and available technologies to mitigate network-based attacks by securing home and small-business IoT devices. It uses products that support the IETF MUD protocol to enable each MUD-capable IoT device that connects to the network to provide a uniform resource locator (URL) for a MUD file. The provided MUD file lists the domains of all external services with which the device is permitted to exchange traffic. The network router is then configured according to the information in the MUD file, thereby protecting the IoT device from being attacked by external entities and protecting external entities from being attacked by the IoT device. In addition, the MUD file specifies devices with which the IoT device is permitted to communicate based on, for example, the manufacturer or class of those other devices. If a local device does not have the specified manufacturer or is not a member of the specified class, the router will not permit it to communicate with the IoT device. So if a device on the local network becomes compromised and that device's manufacturer or class is not one that has been explicitly

permitted in a given IoT device's MUD file, the compromised device will not be permitted to send traffic to attack the given IoT device.

In addition to the protections provided by use of the MUD protocol, this build further secures IoT devices through update servers. Each IoT device on the build network periodically contacts its update server to download and apply security patches, ensuring that it is running the most up-to-date and secure code available. Last, the build uses IoT device discovery technology to discover, inventory, profile, and classify all attached devices. Such classification can be used to validate that the access that is being granted to each device is consistent with that device's type. Threat signaling has not been incorporated into the current build. However, in the future, the build network could be enhanced to periodically receive threat feeds from a threat signaling server to use as a basis for restricting certain types of network traffic.

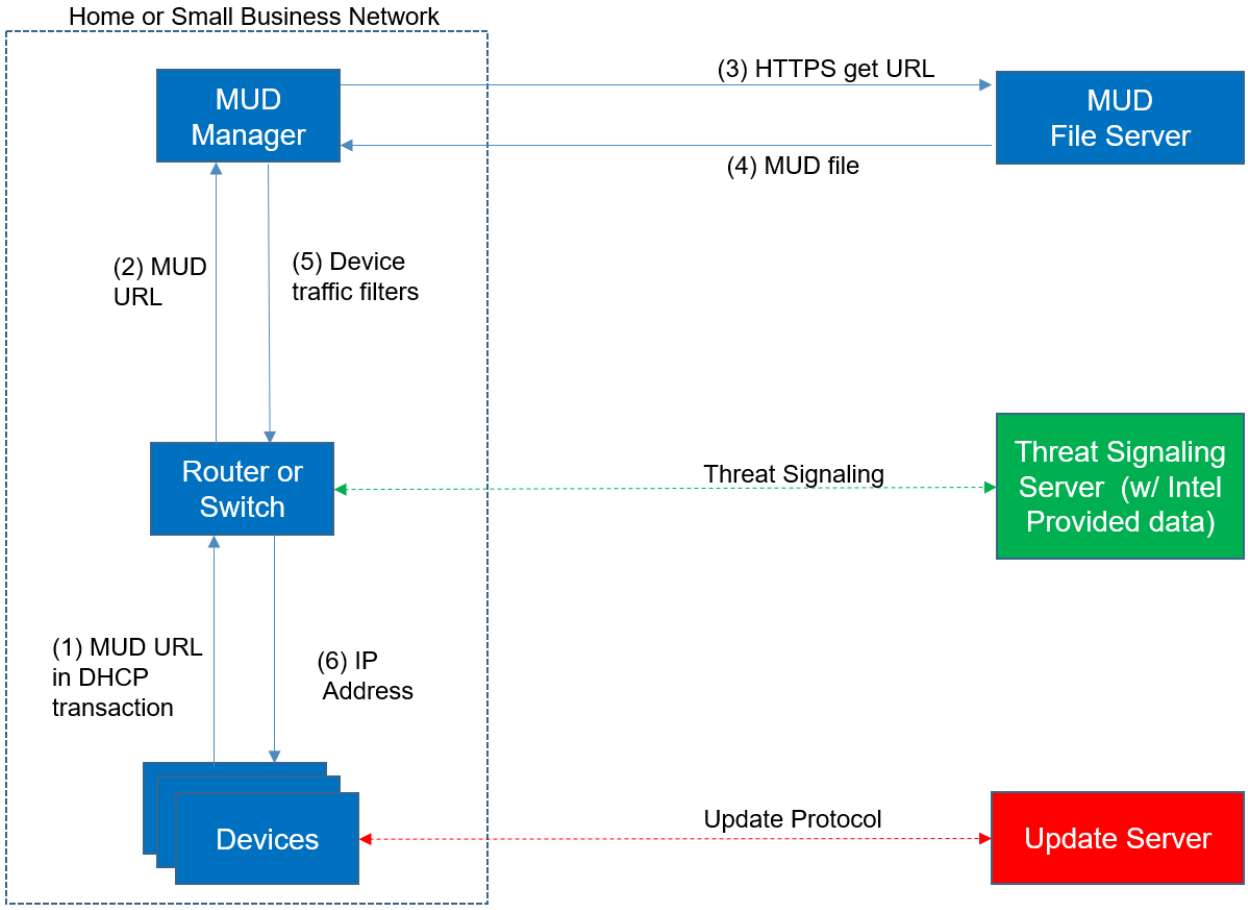
1.2.1 Usage Scenarios

The example implementation fulfills the use cases of a MUD-capable IoT device being onboarded and used on home and small-business networks, where plug-and-play deployment is required. The example implementation includes both MUD-capable and non-MUD-capable IoT devices. MUD-capable IoT devices include the Molex PoE Gateway and Light Engine as well as four development kits (devkits) that the NCCoE configured to perform actions such as power a LED bulb on and off, start network connections, and power a smart lighting device on and off. These MUD-capable IoT devices 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. Non-MUD-capable IoT devices deployed on the example implementation network include three cameras, two smartphones, two smart lighting devices, a smart assistant, a smart printer, a baby monitor with remote control and video and audio capabilities, a smart wireless access point, and a smart digital video recorder. The cameras, smart lighting devices, baby monitor, and digital video recorder are all controlled and managed by a smartphone. In combination, these devices are capable of generating a wide range of network traffic that could reasonably be expected on a home or small-business network.

1.2.2 Architectural Overview

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

262 **Figure 1-1: Generic MUD Logical Architecture**



263

264 A MUD-capable IoT device inserts its MUD URL into the dynamic host configuration protocol (DHCP)
265 address request that it generates when it attaches to the network (e.g., when it powers on).
266 Alternatively, it may include the MUD URL in a Link Layer Discovery Protocol (LLDP) message. The MUD
267 URL is passed to the MUD manager, which retrieves a MUD file from the designated website (denoted as
268 the MUD file server) by using hypertext transfer protocol secure (https). The MUD file describes the
269 communications requirements for the IoT device; the MUD manager converts the requirements into
270 traffic filters that are installed on the router or switch to enforce access controls on the network. This
271 enables the router or switch to deny traffic sent to or from the IoT device if that traffic is outside the
272 device's communications profile.

273 To provide further security, periodic updates are incorporated into the architecture. IoT devices
274 periodically contact the appropriate update server to download and apply security patches. To ensure

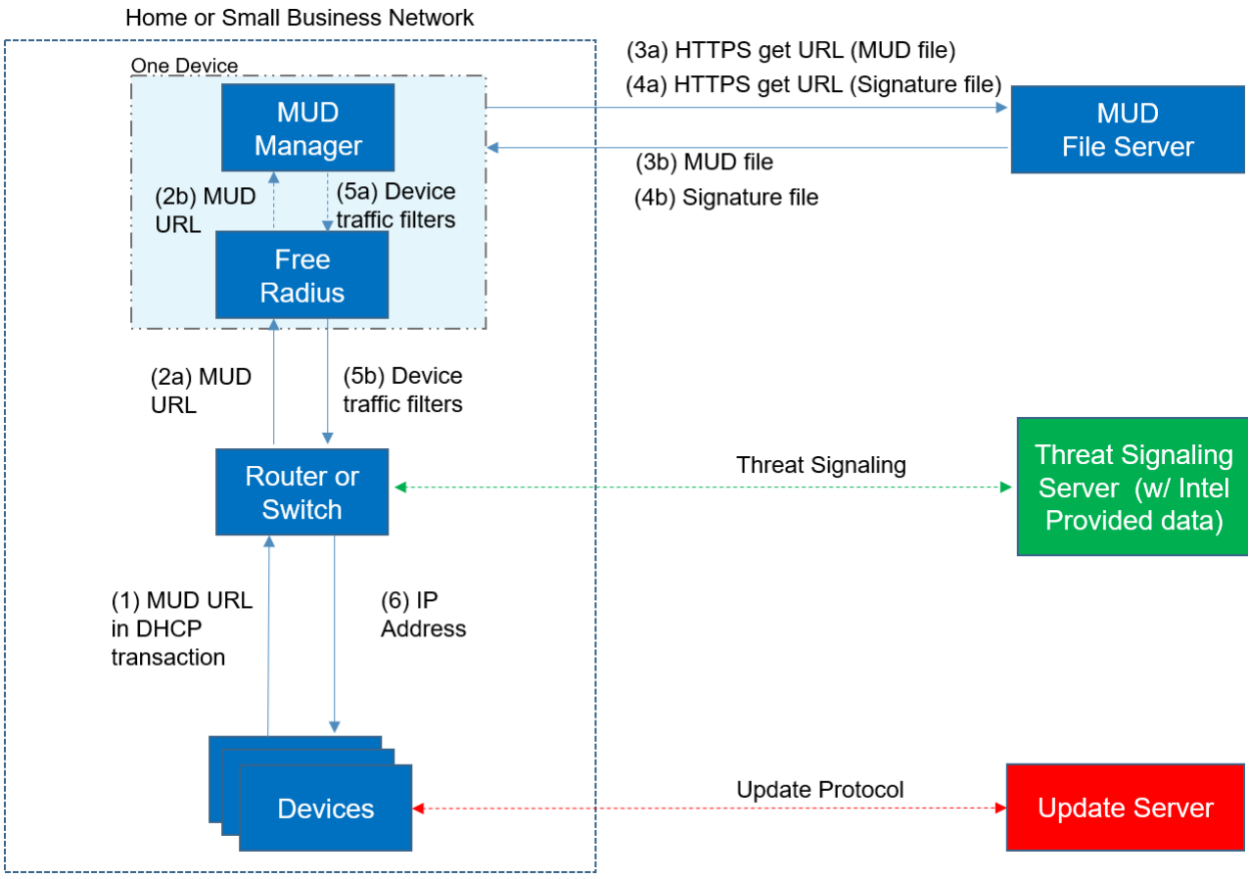
that such updates are possible, the IoT device's MUD file must explicitly permit the IoT device to receive traffic from the update server.

In the future, threat signaling could also be incorporated into the architecture, as shown in Figure 1-1. The router or switch would 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 could 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. The example implementation includes all of the architectural elements shown in Figure 1-1 except for threat signaling. Incorporation of threat signaling is planned for a future build of the example implementation.

1.3 Build 1 Architecture Summary

Figure 1-2 below describes the logical architecture of the first build of a MUD example implementation, referred to as Build 1. Build 1 is designed with a single device serving as the MUD manager and FreeRADIUS server that interfaces with the Catalyst 3850-S switch over transmission control protocol/internet protocol (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 device will emit its MUD URL in some approved method (LLDP, X.509, or DHCP). For this example implementation, only DHCP and LLDP were leveraged.

293 **Figure 1-2: Build 1 Logical Architecture**



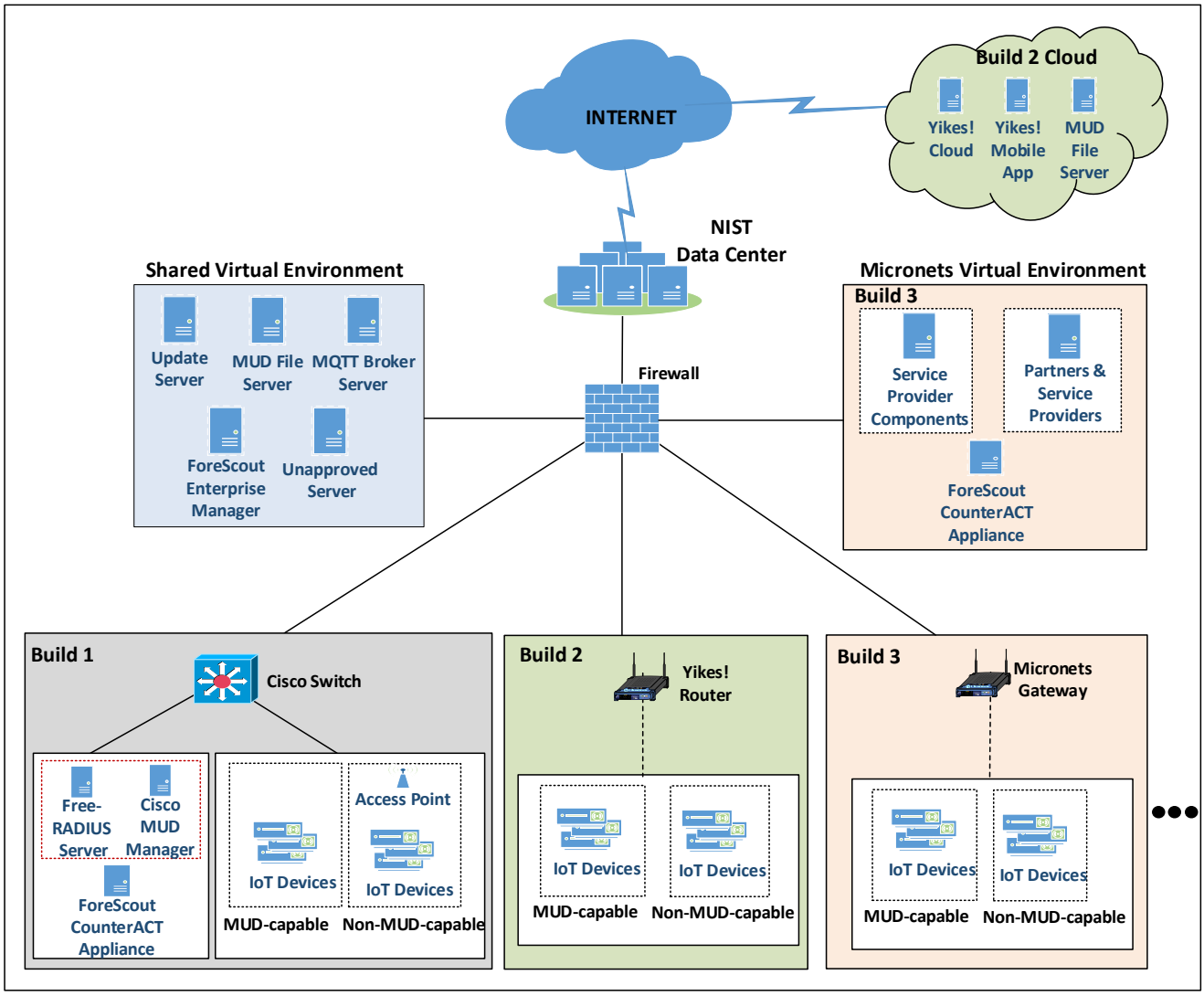
294

295 Figure 1-3 depicts the Build 1 architecture within the context of the NCCoE laboratory. Internet access is
296 available for connection to external hosts, while software components were installed as virtual servers
297 within the vSphere environment. This implementation provides flexibility to implement additional builds
298 in the future. As depicted, the NCCoE laboratory network is connected to the internet via the NIST data
299 center. Access to and from the NCCoE network is protected by a firewall. The NCCoE network includes a
300 virtual environment that houses an update server, a MUD file server, an unapproved server (i.e., a
301 server that is not listed as an approved communications end point in any MUD file), a Message Queuing
302 Telemetry Transport (MQTT) broker server, and a ForeScout Enterprise Manager. These components are
303 hosted at the NCCoE and will be used across builds. The TLS certificate and code signing certificates used
304 by the MUD file server are provided by DigiCert.

305 Only Build 1, as depicted in Figure 1-3, has been implemented during this phase of the project, so that is
306 what this document describes. Build 1 network components consist of a Cisco Catalyst 3850-S switch
307 and virtual instances of Cisco MUD Manager, FreeRADIUS server, and the ForeScout CounterACT
308 appliance. IoT devices used in this architecture comprise MUD-capable and non-MUD-capable devices.

The MUD-capable IoT devices for Build 1 include Raspberry Pi, ARTIK, u-blox, Intel UP Squared Devkits, and the Molex Light Engine controlled by PoE Gateway. Non-MUD-capable devices chosen for Build 1 include three cameras, two smartphones, two smart lighting devices, a smart assistant, a smart printer, a baby monitor with remote control and video and audio capabilities, a smart wireless access point, and a smart digital video recorder. The remainder of this document describes installation and configuration of the various components that are depicted in Figure 1-2.

Figure 1-3: Build 1 NCCoE Laboratory Architecture



1.4 Typographic Conventions

The following table presents typographic conventions used in this volume.

Typeface/Symbol	Meaning	Example
Italics	file names and path names; references to documents that are not hyperlinks; new terms; and placeholders	For 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, and status codes	`Mkdir`
`Monospace Bold`	command-line user input contrasted with computer output	**`service sshd start`**
blue text	link to other parts of the document, a web URL, or an email address	All publications from NIST's NCCoE are available at https://www.nccoe.nist.gov.

2 Product Installation Guides

This section of the practice guide contains detailed instructions for installing and configuring all of the products used to build an instance of the example solution.

2.1 Cisco MUD Manager

This section describes how to deploy Cisco's MUD Manager version 1.0, which uses a MUD-based authorization system in the network, by Cisco Catalyst switches, FreeRADIUS, and Cisco MUD Manager.

2.1.1 Cisco MUD Manager Overview

The Cisco MUD Manager is an open-source implementation that works with IoT devices that emit their MUD URLs. In this implementation we tested two MUD URL emission methods: DHCP and LLDP. The MUD manager is supported by a FreeRADIUS server that receives MUD URLs from the switch. The MUD URLs are extracted by the DHCP server and are sent to the MUD manager via RADIUS messages. The MUD manager is responsible for retrieving the MUD file and corresponding signature file associated with the MUD URL. The MUD manager verifies the legitimacy of the file and then translates the contents to an internet protocol (IP) access control list (ACL)-based policy that is specified in the MUD 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. At the time of implementation, the "model" construct was not yet implemented. In addition, if a DNS-based system changes its address, this will not be noticed. Also, IPv6 access has not been fully supported.

2.1.2 Cisco MUD Manager Configurations

The following subsections document the software, hardware, and network configurations for the Cisco MUD Manager.

2.1.2.1 Hardware Configuration

Cisco requires installing the MUD manager and FreeRADIUS on a single server with at least 2 gigabytes of random access memory. This server must integrate with at least one switch or router on the network. For this example implementation we used a Catalyst 3850-S switch.

2.1.2.2 Network Configuration

The MUD manager and FreeRADIUS server instances were installed and configured on a dedicated machine leveraged for hosting virtual machines in the Build 1 lab environment. This machine was then connected to virtual local area network (VLAN) 2 on the Catalyst 3850-S and assigned a static IP address.

2.1.2.3 Software Configuration

For this build, the Cisco MUD Manager was installed on an Ubuntu 18.04.01 64-bit server. However, there are many approaches for implementation. After completion of this implementation, the MUD manager can be built via Docker containers provided by Cisco.

The Cisco MUD Manager can operate on Linux operating systems, such as

- Ubuntu 18.04.01
- Amazon Linux

The Cisco MUD Manager requires the following installations and components:

- OpenSSL
- cJSON
- MongoDB
- Mongo C Driver
- Libcurl
- FreeRADIUS server

363 At a high level, the following software configurations and integrations are required:

364 ▪ The Cisco MUD Manager requires integration with a switch (such as a Catalyst 3850-S) that
365 connects to an authentication, authorization, and accounting (AAA) server that communicates
366 by using the RADIUS protocol (i.e., a RADIUS server).

367 ▪ The RADIUS server must be configured to identify a MUD URL received in an accounting request
368 message from a device it has authenticated.

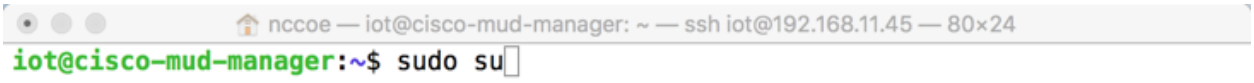
369 ▪ The MUD manager must be configured to process a MUD URL received from a RADIUS server
370 and return access control policy to the RADIUS server, which is then forwarded to the switch.

371 2.1.3 Preinstallation

372 Cisco's DevNet GitHub page provides documentation that we followed to complete this section:
373 https://github.com/CiscoDevNet/MUD-Manager/tree/1.0#dependancies

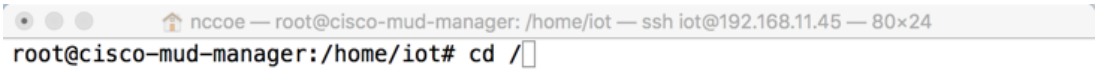
374 1. Open a terminal window, and enter the following command to log in as root:

375 `sudo su`



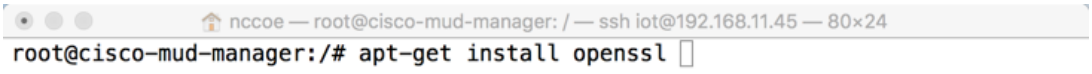
376 2. Change to the root directory:

377 `cd /`



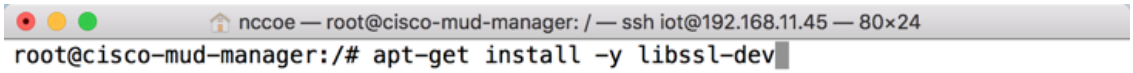
378 3. To install OpenSSL from the terminal, enter the following command:

379 `apt-get install openssl`



380 a. If unable to link to OpenSSL, install the following by entering this command:

381 `apt-get install -y libssl-dev`



4. To install cJSON, download it from GitHub by entering the following command:

```
git clone https://github.com/DaveGamble/cJSON
```

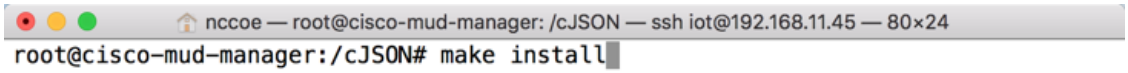
a. Change directories to the cJSON folder by entering the following command:

```
cd cJSON
```

b. Build cJSON by entering the following commands:

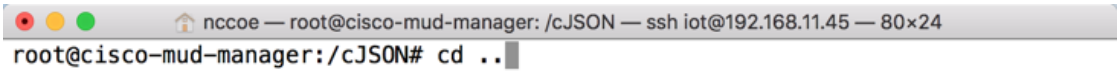
```
make
```


```
make install
```



5. Change directories back a folder by entering the following command:

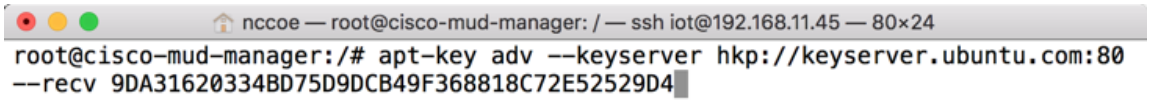
```
cd ..
```



6. To install MongoDB, enter the following commands:

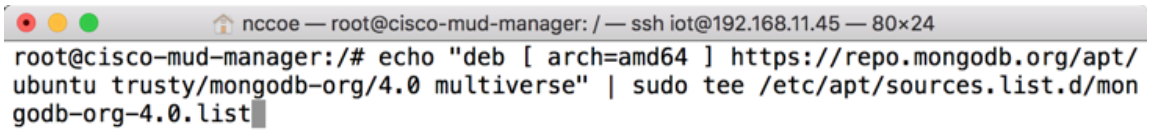
   a. Import the public key:

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
9DA31620334BD75D9DCB49F368818C72E52529D4
```



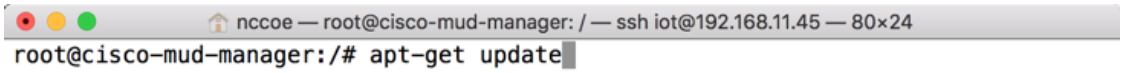
   b. Create a list file for MongoDB:

```
echo "deb [ arch=amd64 ] https://repo.mongodb.org/apt/ubuntu
trusty/mongodb-org/4.0 multiverse" | sudo tee
/etc/apt/sources.list.d/mongodb-org-4.0.list
```



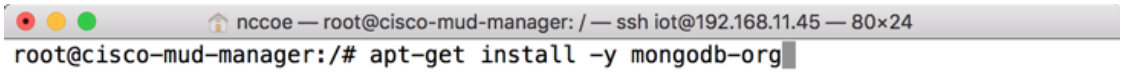
   c. Reload the local package database:

```
sudo apt-get update
```



   d. Install the MongoDB packages:

```
sudo apt-get install -y mongodb-org
```

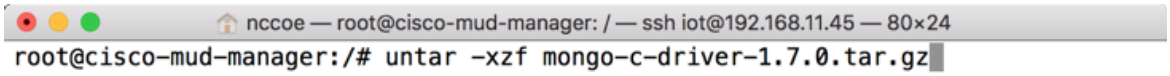


7. To install the Mongo C driver, enter the following command:

```
wget https://github.com/mongodb/mongo-c-driver/releases/download/1.7.0/mongo-c-driver-1.7.0.tar.gz
```

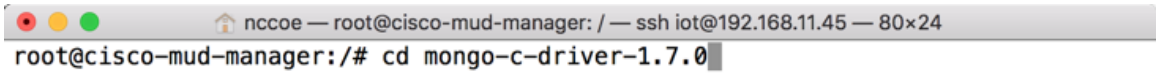
a. Untar the file by entering the following command:

```
untar -xzf mongo-c-driver-1.7.0.tar.gz
```



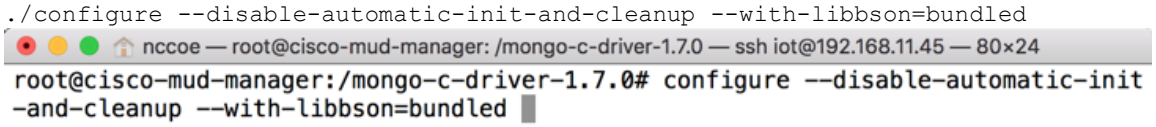
b. Change into the mongo-c-driver-1.7.0 directory by entering the following command:

```
cd mongo-c-driver-1.7.0/
```

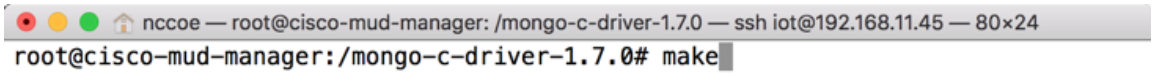


c. Build the Mongo C driver by entering the following commands:

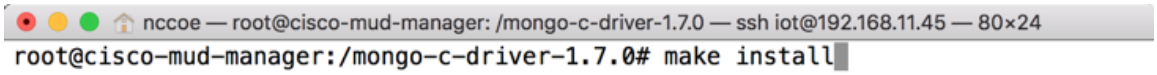
```
./configure --disable-automatic-init-and-cleanup --with-libbson=bundled
```




```
make
```

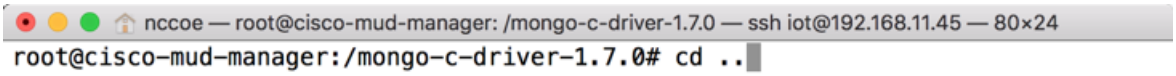


413 `make install`

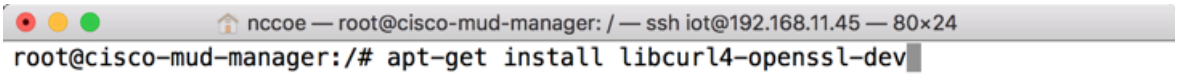


414 8. Change directories back a folder by entering the following command:

415 `cd ..`



416 9. To install libcurl, enter the following command:
417 `sudo apt-get install libcurl4-openssl-dev`



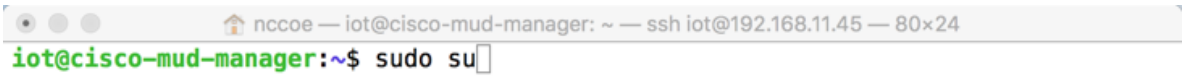
418 2.1.4 MUD Manager Installation

419 A portion of the steps in this section are documented on Cisco's DevNet GitHub page:

420 https://github.com/CiscoDevNet/MUD-Manager/tree/1.0#building-the-mud-manager

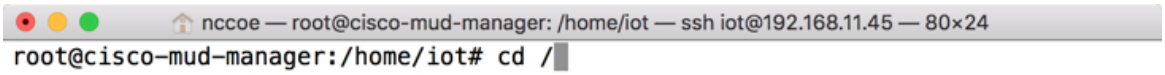
421 1. Open a terminal window, and enter the following command to log in as root:

```
sudo su
```



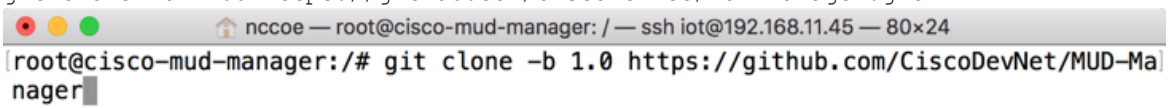
2. Change to the root directory by entering the following command:

```
cd /
```



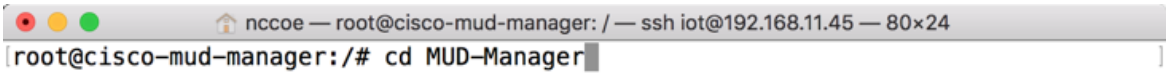
3. To install the MUD manager, download it from Cisco's GitHub by entering the following command:

```
git clone -br 1.0 https://github.com/CiscoDevNet/MUD-Manager.git
```



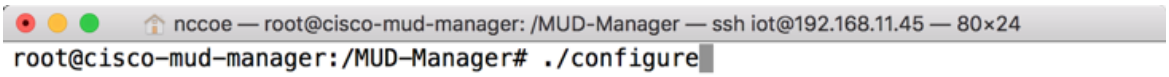
4. Change into the MUD manager directory:

```
cd MUD-Manager
```

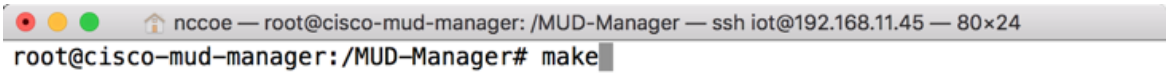


5. Build the MUD manager by entering the following commands:

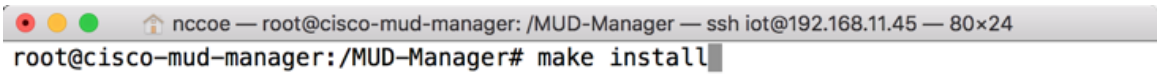
```
./configure
```




```
make
```




```
make install
```

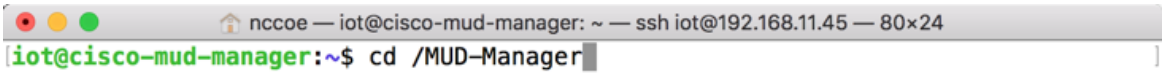


2.1.5 MUD Manager Configuration

This section describes configuring the MUD manager to communicate with the NCCoE MUD file server and defining the attributes used for translating the fetched MUD files. Details about the configuration file and additional fields that could be set within this file can be accessed here: https://github.com/CiscoDevNet/MUD-Manager#editing-the-configuration-file.

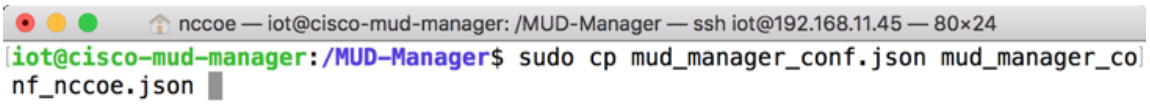
1. In the terminal, change to the MUD manager directory:

```
cd /MUD-Manager
```



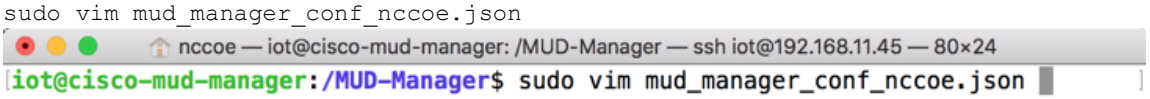
2. Copy the contents of the sample mud_manager_conf.json file to a different file:

```
sudo cp mud_manager_conf.json mud_manager_conf_nccoe.json
```



3. Modify the contents of the new MUD manager configuration file:

```
sudo vim mud_manager_conf_nccoe.json
```



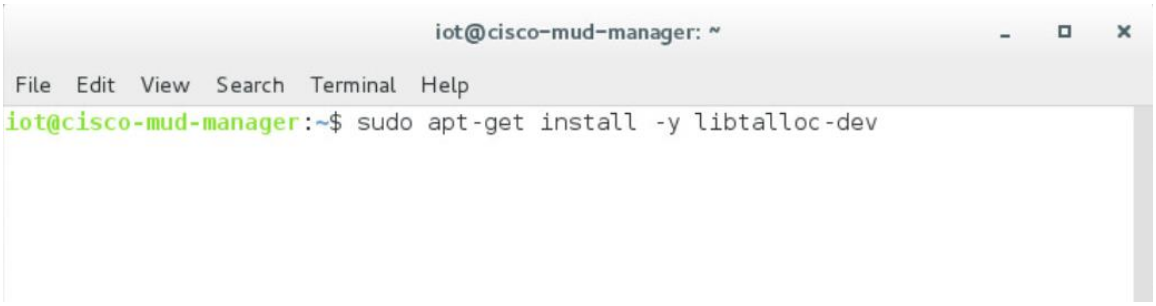
```
{
        "MUDManagerAPIProtocol" : "http",
        "ACL_Prefix" : "ACS:",
        "ACL_Type" : "dACL-ingress-only",
        "COA_Password" : "cisco",
        "Manufacturers" : [
                { "authority" : "mudfileserver",
                  "cert" : "/home/mudtester/digicertca-chain.crt",
                  "web_cert": "/home/mudtester/mud-intermediate.pem",
                  "my_controller_v4" : "192.168.10.104",
                  "my_controller_v6" : "2610:20:60CE:630:8000::7",
                  "local_networks_v4" : "192.168.10.0 0.0.0.255",
                  "local_networks_v6" : "2610:20:60CE:630:8000::",
                  "vlan_nw_v4" : "192.168.13.0 0.0.0.255",
                  "vlan" : 3
                },
                {
                "authority" : "www.devicetype.com",
                  "cert" : "/home/mudtester/digicertca-chain.crt",
                  "web_cert": "/home/mudtester/mud-intermediate.pem",
                  "vlan_nw_v4" : "192.168.14.0 0.0.0.255",
                  "vlan" : 4
                }
        ],
        "DNSMapping" : {
                "www.dominiontea.com" : "192.200.178.19",
                "www.updateserver.com": "192.168.4.7",
                "www.mqttbroker.com" : "192.168.4.6"

        },

        "DNSMapping_v6" : {
                "www.mqttbroker.com" : "2610:20:60CE:630:8000::6",
                "www.updateserver.com" : "2610:20:60CE:630:8000::7",
                "www.dominiontea.com": "2a03:2880:f10c:83:face:b00c:0:25de"
        },
        "ControllerMapping" : {
                "http://lightcontroller.example.com" : "192.168.10.104",
                "http://lightcontroller.example2.com" : "192.168.10.105"
        },
        "ControllerMapping_v6" : {
                "http://lightcontroller.example.com" : "ffff:2343:4444:::"
        },
        "DefaultACL" : ["permit tcp any eq 22 any","permit udp any eq 68 any eq 67",
        "DefaultACL_v6" : "permit udp any any eq 53", "deny ip any any"],
        "DefaultACL_v6" : ["permit udp any any eq 53", "deny ipv6 any any"]
}
~
"mud_manager_conf_nccoe.json" 46L, 1612C
```

Details about the contents of the configuration file can be found at the link provided at the start of this section.

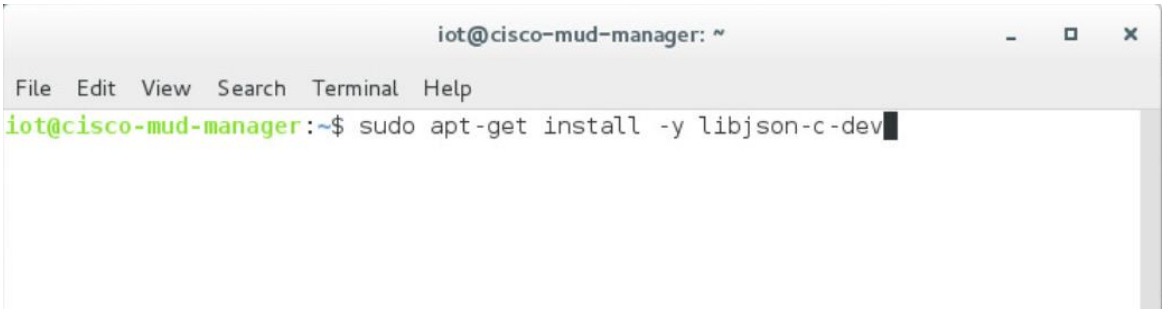
2.1.6 FreeRADIUS Installation

1. Install the dependencies for FreeRADIUS:

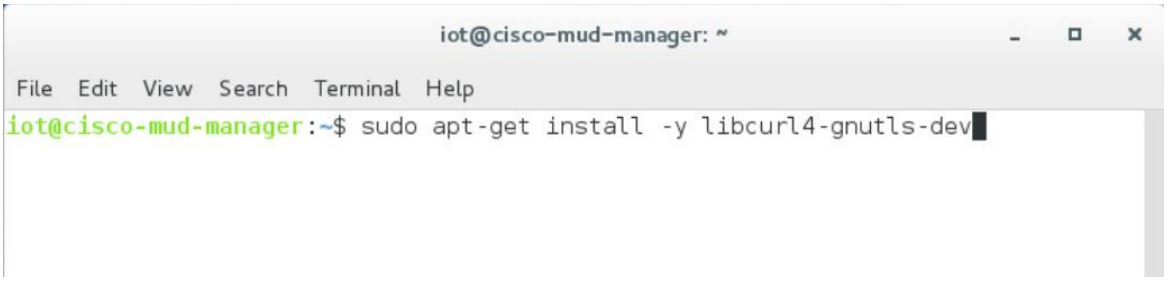
a. `sudo apt-get install -y libtalloc-dev`



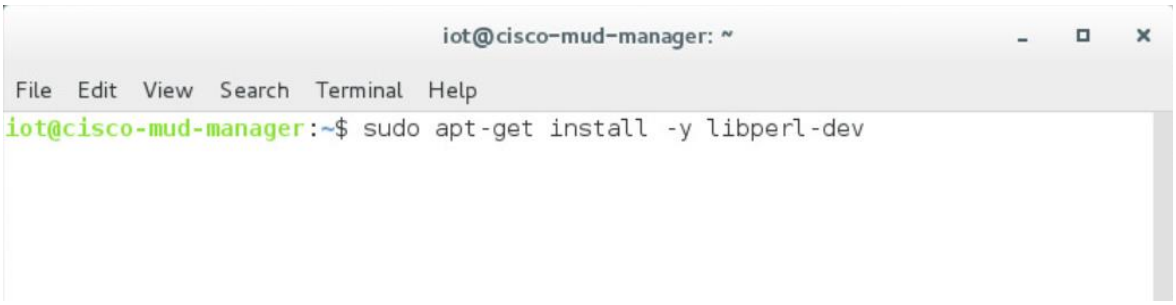
b. `sudo apt-get install -y libjson-c-dev`



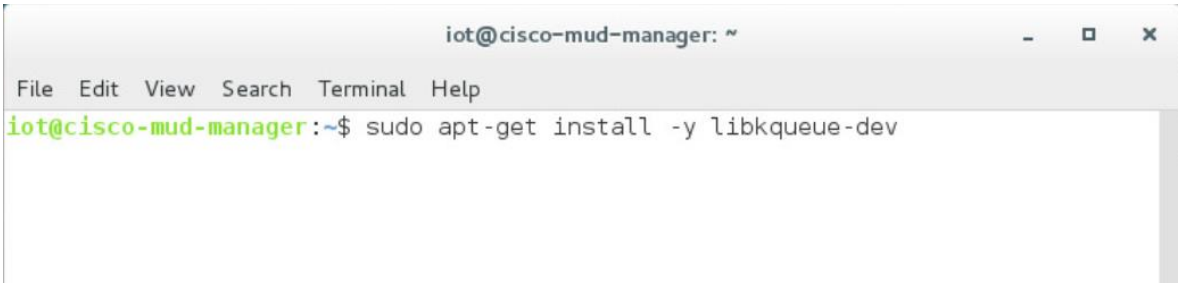
c. `sudo apt-get install -y libcurl4-gnutls-dev`



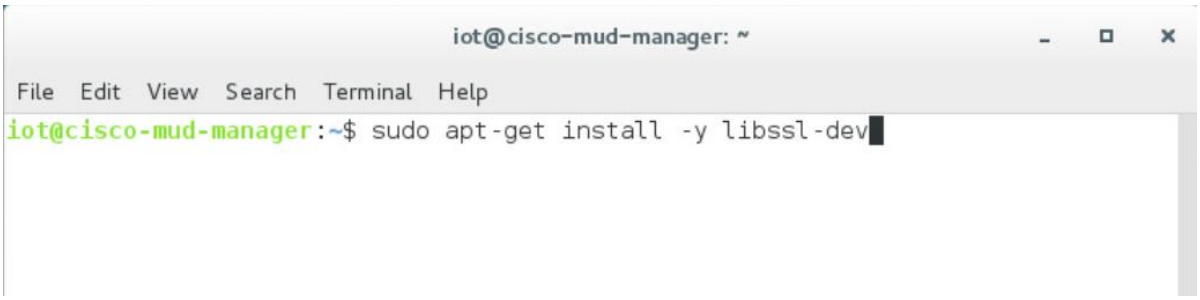
d. `sudo apt-get install -y libperl-dev`



e. `sudo apt-get install -y libkqueue-dev`

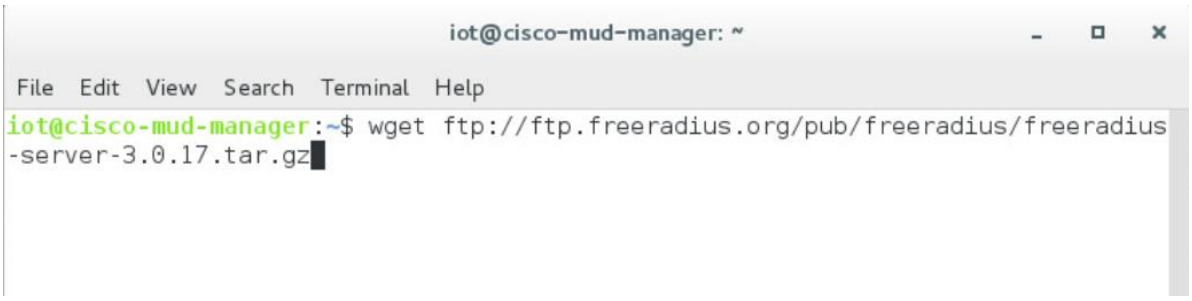


f. `sudo apt-get install -y libssl-dev`



2. Download the source by entering the following command (Note: Version 3.0.17 and later are recommended):

`wget ftp://ftp.freeradius.org/pub/freeradius/freeradius-server-3.0.17.tar.gz`

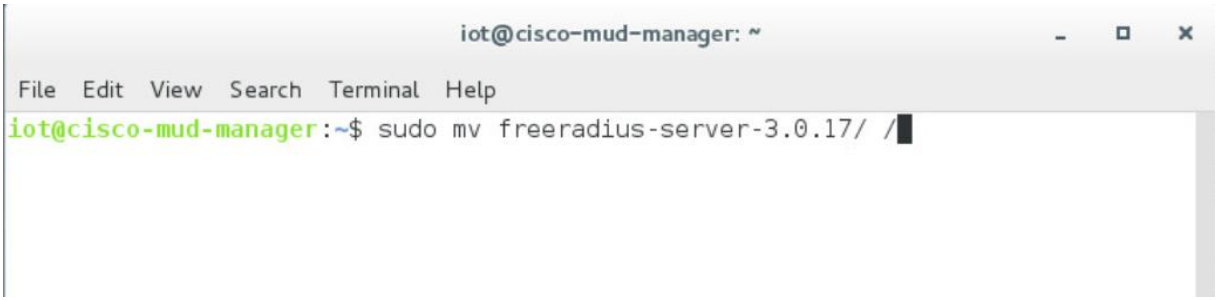


3. Untar the file pulled by entering the following command:

`tar -xf freeradius-server-3.0.17.tar.gz`

4. Move the FreeRADIUS directory to the root directory:

```
sudo mv freeradius-server-3.0.17/ /
```



5. Change to the FreeRADIUS directory:

```
cd /freeradius-server-3.0.17/
```

6. Make and install the source by entering the following:

a. `sudo ./configure --with-rest --with-json-c --with-perl`

b. `sudo make`

c. `sudo make install`

2.1.7 FreeRADIUS Configuration

1. Change to the FreeRADIUS subdirectory in the MUD manager directory:

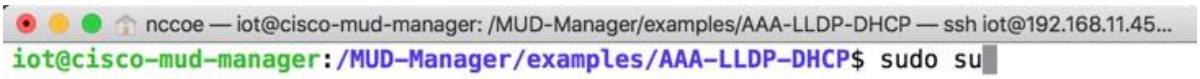
```
cd /MUD-Manager/examples/AAA-LLDP-DHCP/
```

2. Run the setup script:

```
sudo ./FR-setup.sh
```

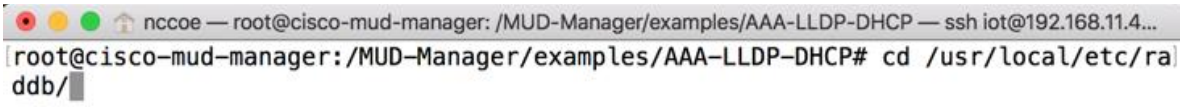
3. Enter the following command to log in as root:

```
sudo su
```



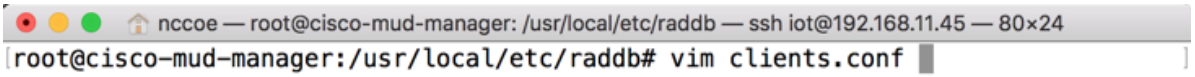
4. Change to the radius directory:

```
cd /usr/local/etc/raddb/
```



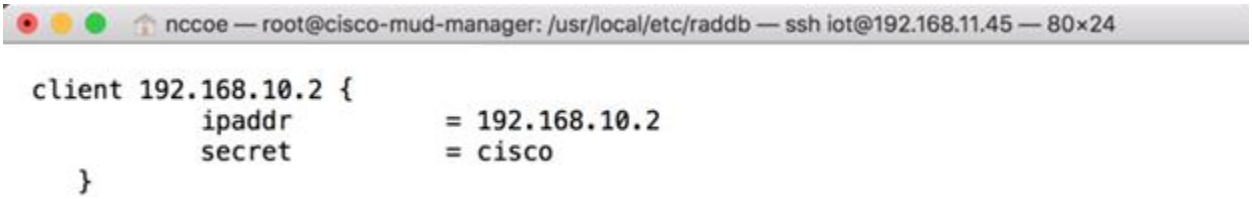
5. Open the *clients.conf* file:

```
vim clients.conf
```



6. Add the network address server (NAS) as an authorized client in the configuration file on the server by adding an entry for the NAS in the *client.conf* file opened (Note: Replace the IP address below with the IP address of the NAS and use the "secret" configured on the NAS to talk to RADIUS servers):

```
client 192.168.10.2 {
        ipaddr = 192.168.10.2
        secret = cisco
}
```

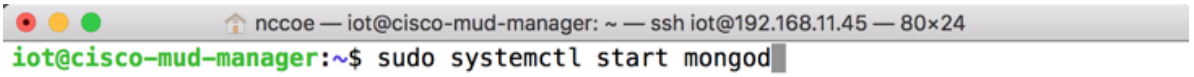


7. Save and close the file.

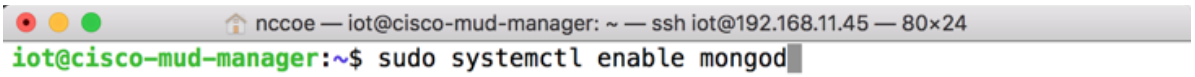
2.1.8 Start MUD Manager and FreeRADIUS Server

1. Start and enable the database by executing the following commands:

```
sudo systemctl start mongod
```

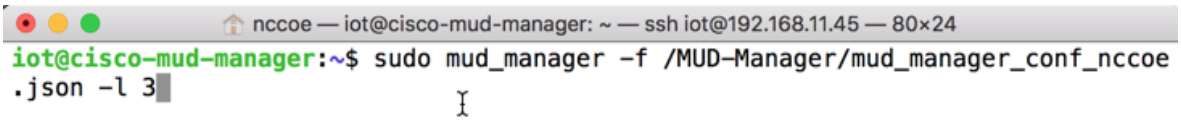



```
sudo systemctl enable mongod
```



2. Start the MUD manager in the foreground with logging enabled by entering the following command:

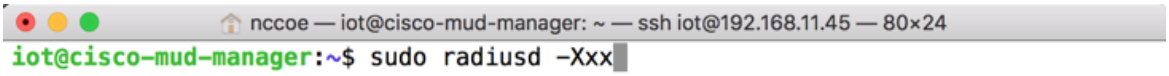
```
sudo mud_manager -f /MUD-Manager/mud_manager_conf_nccoe.json -l 3
```



The following output should appear if the service started successfully:

3. Start the FreeRADIUS service in the foreground with logging enabled by entering the following command:

```
sudo radiusd -Xxx
```



At this point all the necessary processes are up and running from the server side and it is time to move on to configuring the switch. Any DHCP activity on the network should appear here once the switch is configured accordingly.

2.2 MUD File Server

2.2.1 MUD File Server Overview

For this example implementation, the NCCoE built a MUD file server hosted within the lab infrastructure. This file server signs and stores the MUD files along with their corresponding signature files for the MUD-capable IoT devices used in the build. The MUD file server is also responsible for serving the MUD file and the corresponding signature file upon request from the MUD manager.

2.2.2 Configuration Overview

The following subsections document the software and network configurations for the MUD file server.

2.2.2.1 Network Configuration

This server was hosted in the NCCoE's virtual environment, functioning as a cloud service. The IP address was statically assigned.

2.2.2.2 Software Configuration

For this example implementation, the server ran on the CentOS 7 operating system. The MUD files and signatures were hosted by an Apache web server and configured to use secure sockets layer/ transport layer security (SSL/TLS) encryption.

2.2.2.3 Hardware Configuration

The MUD file server was hosted in the NCCoE's virtual environment, functioning as a cloud service.

2.2.3 Setup

The following subsections describe the setup process for configuring the MUD file server.

2.2.3.1 Apache Web Server

The Apache web server was set up by using the official Apache documentation at https://httpd.apache.org/docs/current/install.html. After that, SSL/TLS encryption was set up by using the digital certificate and key obtained from DigiCert. This was set up by using the official Apache documentation, found at https://httpd.apache.org/docs/current/ssl/ssl_howto.html.

2.2.3.2 MUD File Creation and Signing

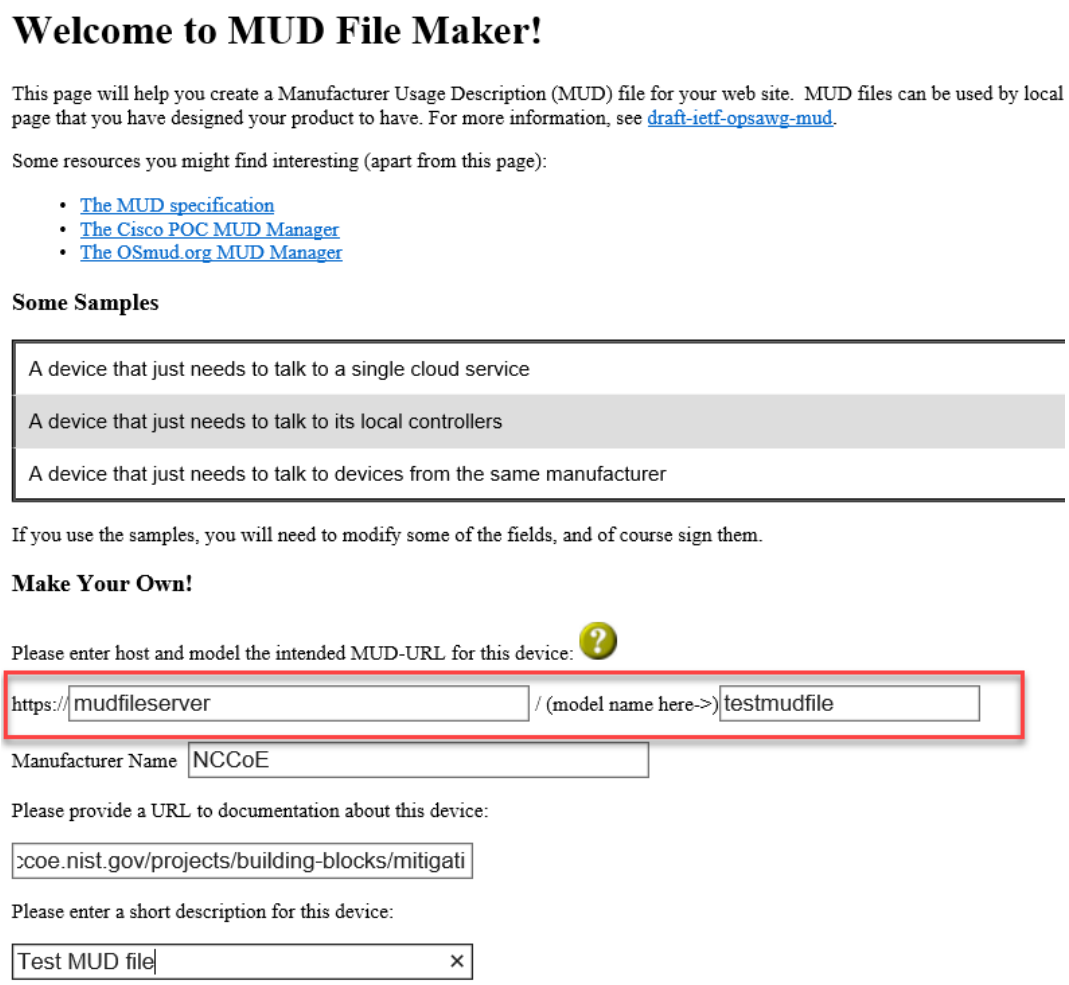
This section details creating and signing a MUD file on the MUD file server. It is not mandated by the MUD specification that this signing process be performed on the MUD file server itself.

2.2.3.2.1 MUD File Creation

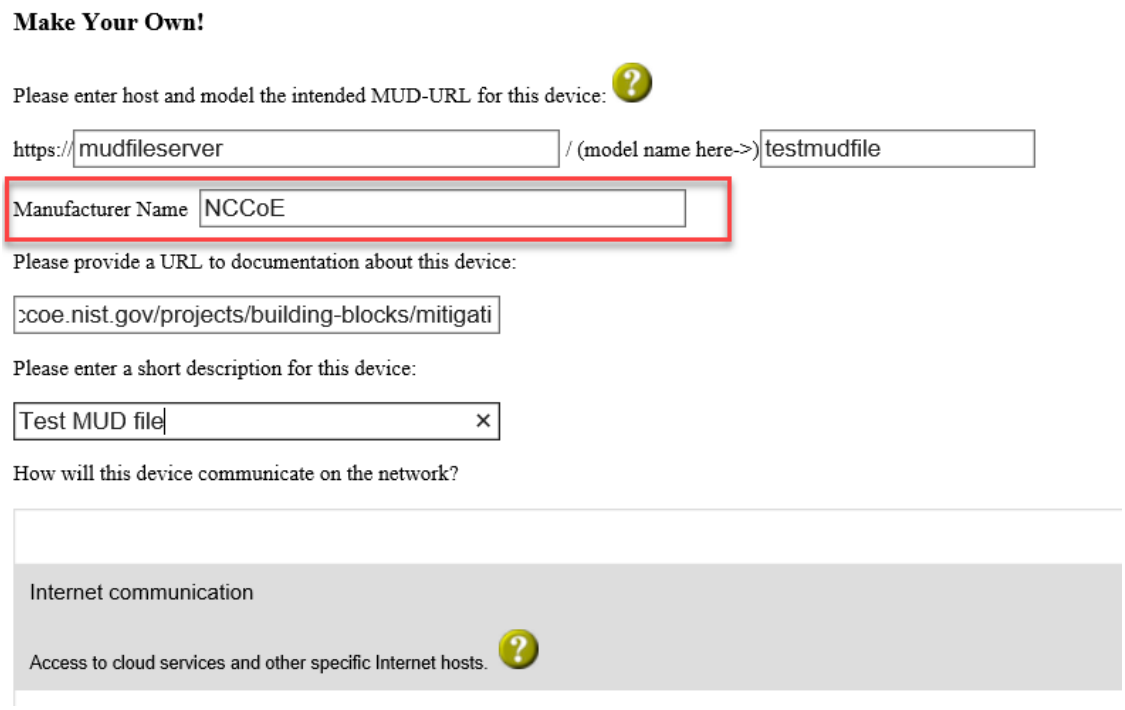
In this implementation, MUD Maker was used to build MUD files. Once the permitted communications have been defined for the IoT device, proceed to www.mudmaker.org to leverage the online tool. There is also a list of sample MUD files on the site, which can be used as a reference. Upon navigating to www.mudmaker.org, complete the following steps to create a MUD file:

1. Specify the host that will be serving the MUD file and the model name of the device in the following text fields (Note: This will result in the MUD URL for this device):

   Sample input: mudfileserver, testmudfile

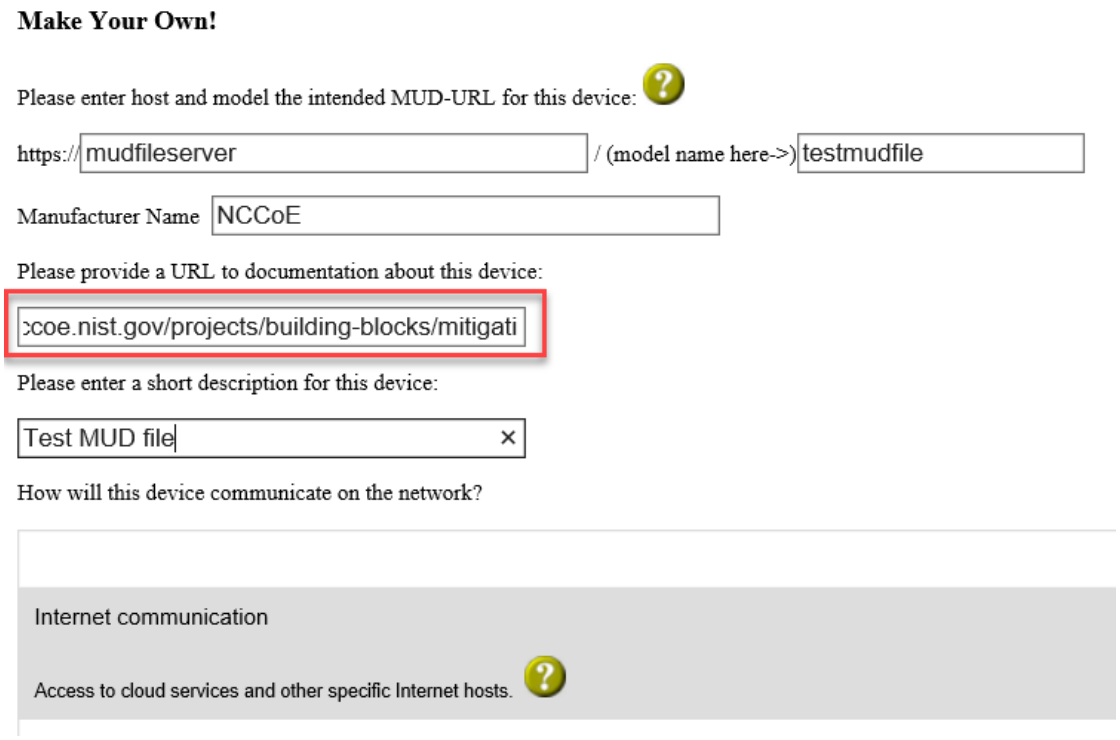


2. Specify the Manufacturer Name of the device in the following text field:



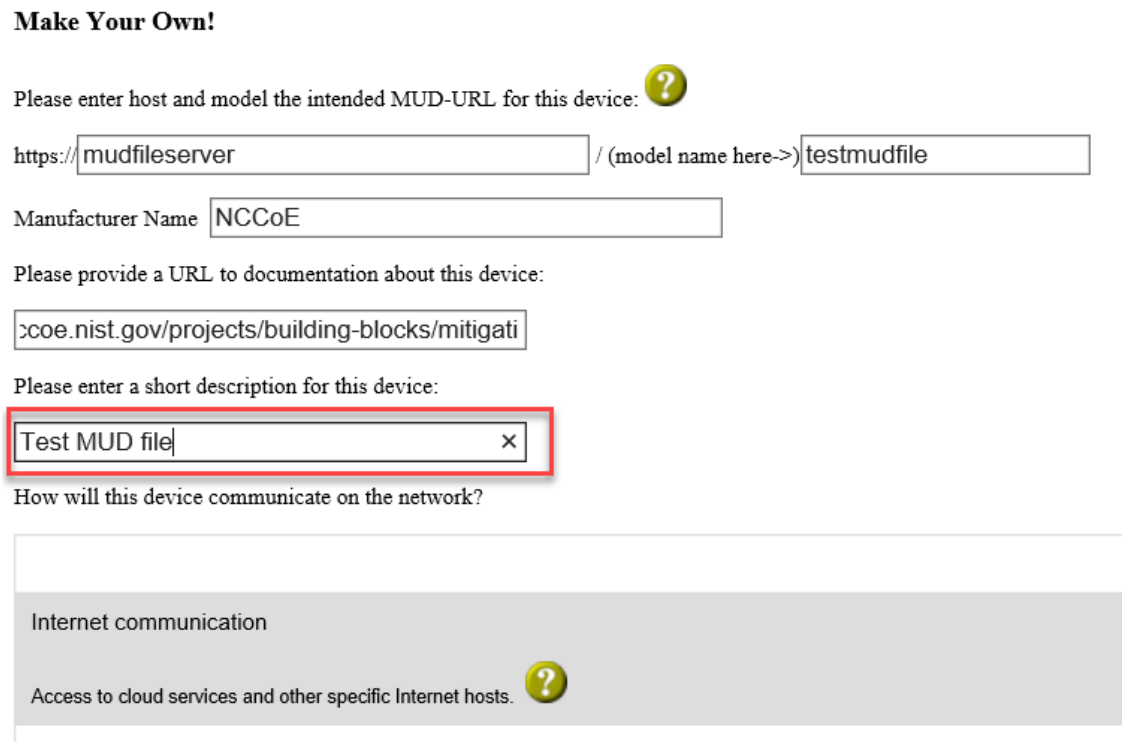
558

559 3. Include a URL to documentation about this device in the following text field:



560

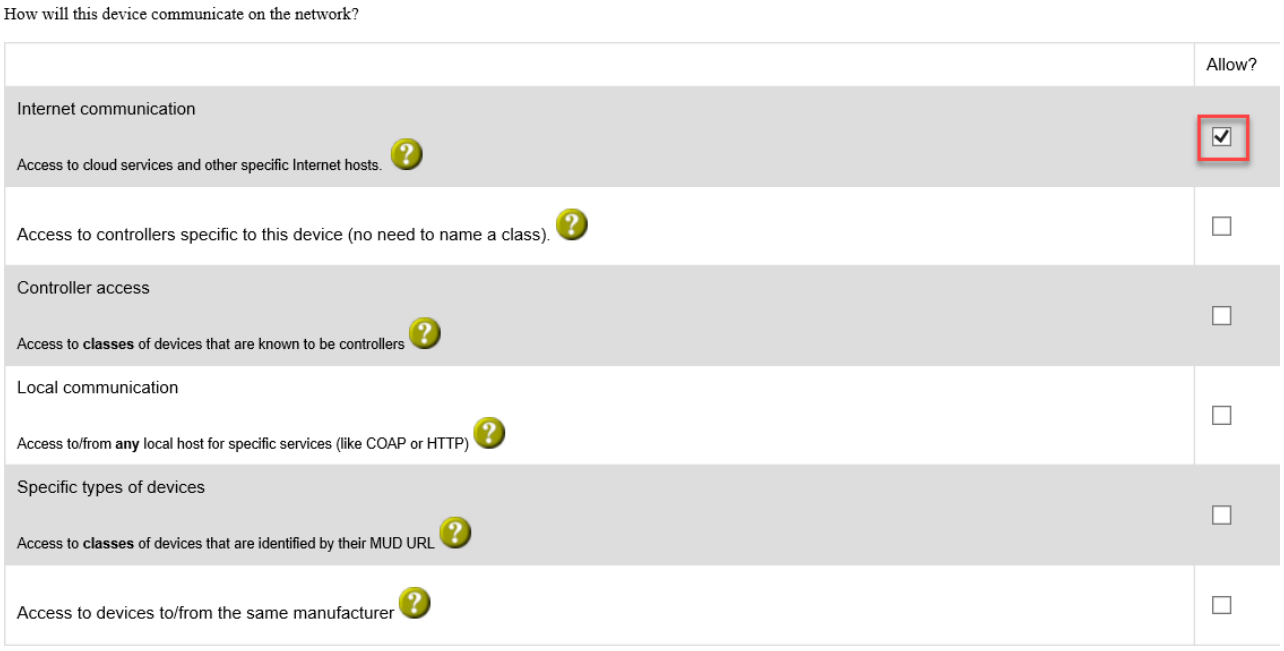
561 4. Include a short description of the device in the following text field:



562

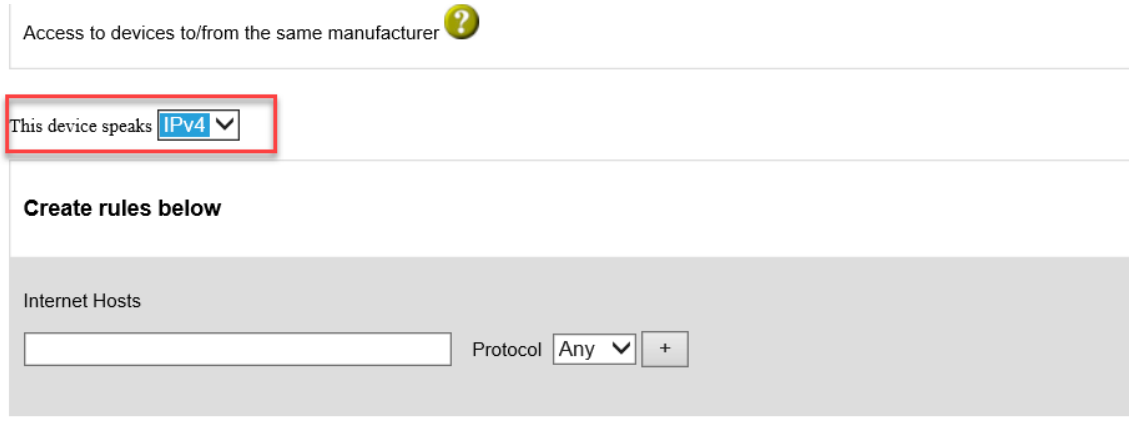
563 5. Check the boxes for the types of network communication that are allowed for the device:

564

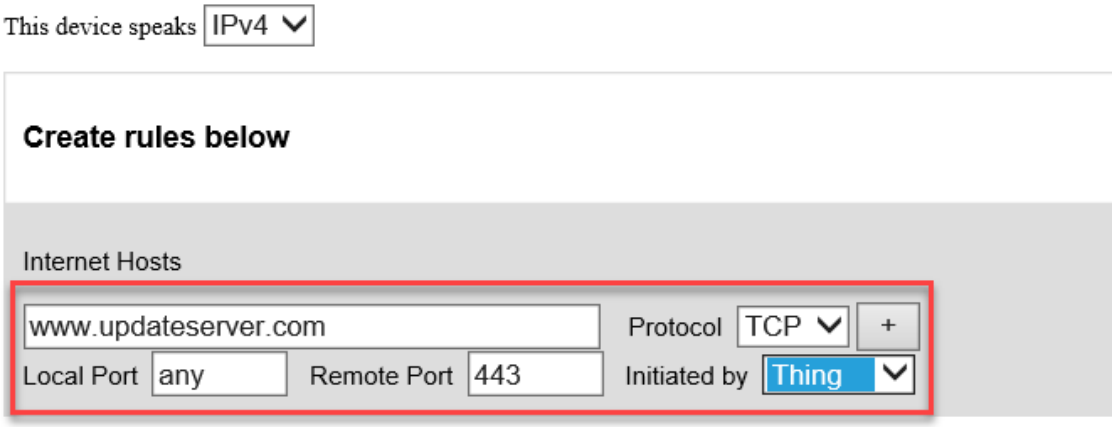


565

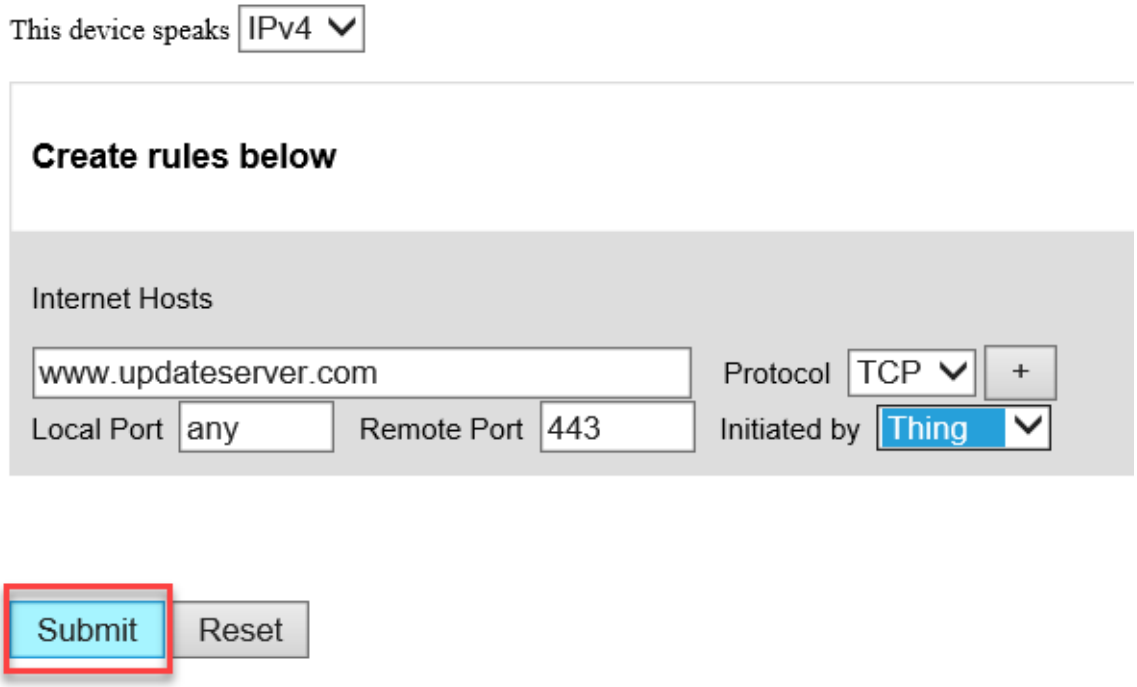
6. Specify the internet protocol version that the device leverages:



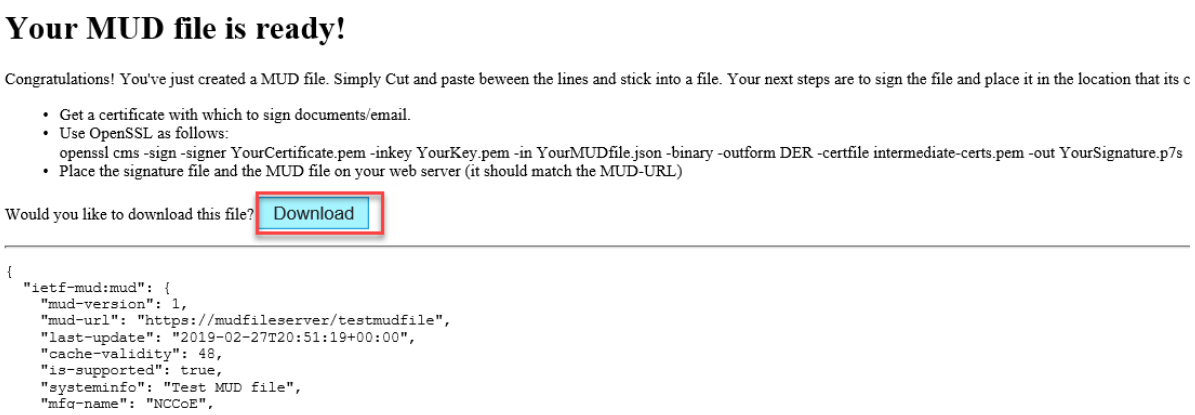
7. Specify the fields (Internet Hosts, Protocol, Local Port, Remote Port, and Initiated by) that this device will be communicating with:



8. Click **Submit** to generate the MUD file:



9. Once completed, the page will redirect to the following page that outputs the MUD file on the screen. Click **Download** to download the MUD file, which is a .JSON file:



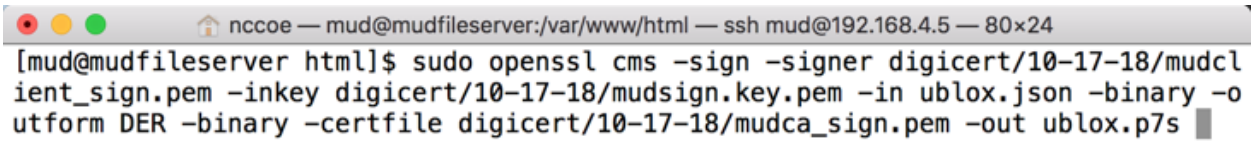
10. Click **Save** to store a copy of the MUD file:

2.2.3.2.2 MUD File Signature Creation and Verification

In this implementation, OpenSSL is used to sign and verify MUD files. This example uses the MUD file created in the previous section. To start this process, start with a MUD file (in our example, this file is named *ublox.json*), the Signing Certificate, the Private Key for the Signing Certificate, the Intermediate Certificate for the Signing Certificate, and the Certificate of the Trusted Root Certificate Authority for the Signing Certificate.

1. Sign the MUD file by using the following command:

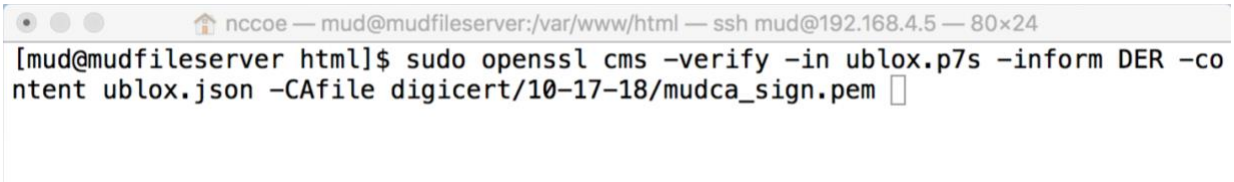
```
sudo openssl cms -sign -signer <Signing Certificate> -inkey <Private Key for Signing
Certificate> -in <Name of MUD File> -binary -outform DER -binary -certfile
<Intermediate Certificate for Signing Certificate> -out <Name of MUD File without the
.json file extension>.p7s
```



This will create a signature file for the MUD file that has the same name as the MUD file but ends with the.p7s file extension, i.e., in our case *ublox.p7s*.

2. Manually verify the MUD File Signature by using the following command:

```
sudo openssl cms -verify -in <Name of MUD File>.p7s -inform DER -content <Name of MUD
File>.json -CAfile <Certificate of Trusted Root Certificate Authority for Signing
Certificate>
```



If a valid file signature was created successfully, a corresponding message should appear. Both the MUD file and MUD File Signature should be placed on the MUD file server in the Apache server directory.

2.3 Cisco Switch–Catalyst 3850-S

2.3.1 Cisco 3850-S Catalyst Switch Overview

The switch used in this build is an enterprise class, Layer 3 switch, the Cisco Catalyst 3850-S that had been modified to support MUD functionality as a proof-of-concept implementation. In addition to providing DHCP services, the switch also acts as a broker for connected IoT devices for authentication,

authorization, and accounting through a 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.

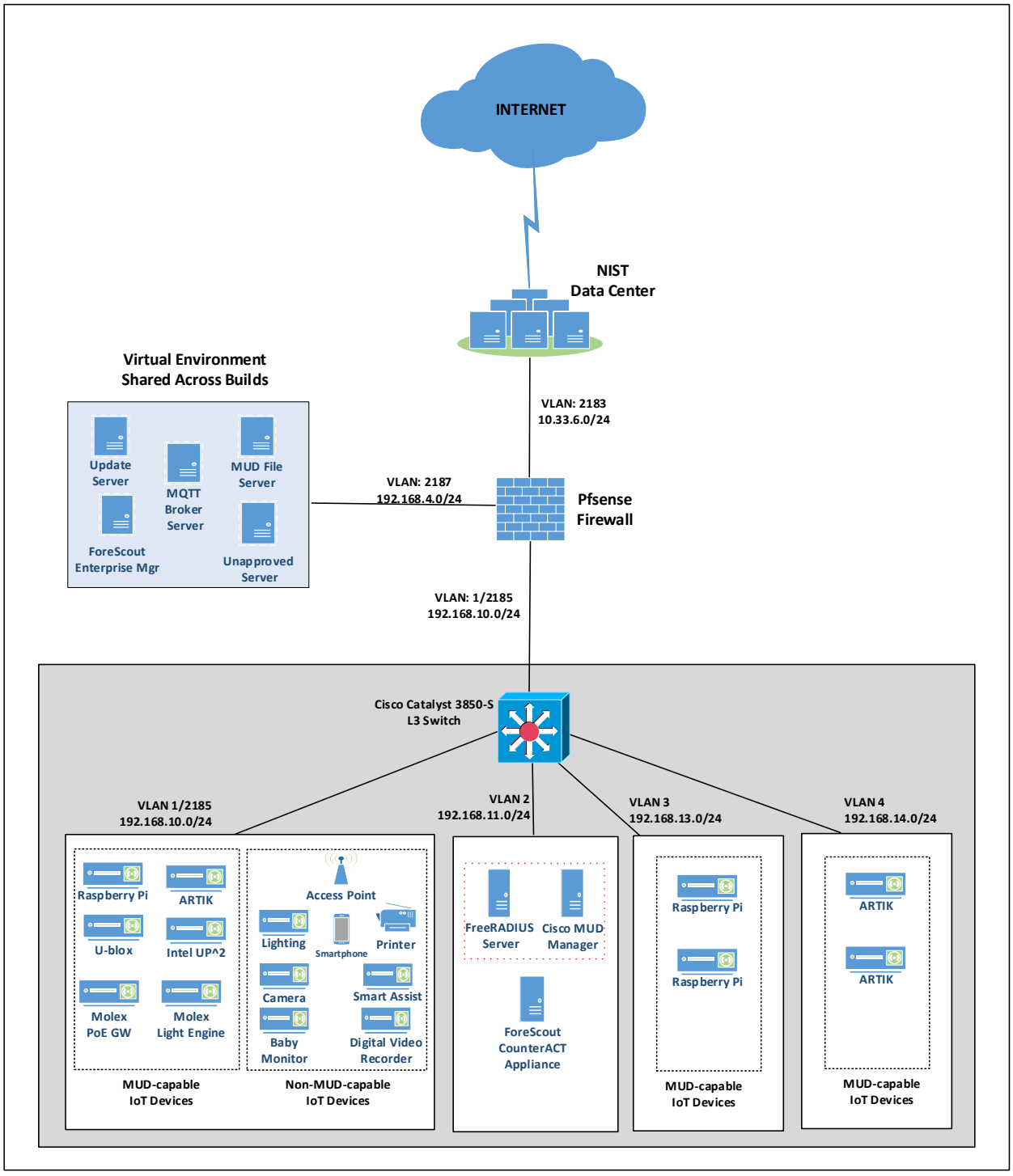
2.3.2 Configuration Overview

The following subsections document the network, software, and hardware configurations for the Cisco Catalyst 3850-S switch.

2.3.2.1 Network Configuration

This section describes how to configure the required Cisco Catalyst 3850-S switch to support the example implementation. A special image for the Catalyst 3850-S was provided by Cisco to support MUD-specific functionality. In our example implementation, the switch is integrated with a DHCP server and a FreeRADIUS server, which together support delivery of the MUD URL to the MUD manager via either DHCP or LLDP. The MUD manager is also able to generate and send a dynamic access list to the switch, via the RADIUS server, to permit or deny access to and from the IoT devices. In addition to hosting directly connected IoT devices on VLANs 1, 3, and 4, the switch also hosts both the MUD manager and the FreeRADIUS servers on VLAN 2. As illustrated in Figure 2-1, each locally configured VLAN is protected by a firewall that connects the lab environment to the NIST data center, which provides internet access for all connected devices.

620 **Figure 2-1: Physical Architecture–Build 1**



621

2.3.2.2 Software Configuration

The prototype, MUD-capable Cisco 3850-S used in this build is running internetwork operating system (IOS) version 16.09.02.

2.3.2.3 Hardware Configuration

The Catalyst 3850-S switch configured in the lab consists of 24 one-gigabit Ethernet ports with two optional 10-gigabit Ethernet uplink ports. A customized version of Cat-OS is installed on the switch. The versions of the operating system are as follows:

- Cat3k_caa-guestshell.16
- Cat3k_caa-rpbase.16.06.
- Cat3k_caa-rpcore.16.06.
- Cat3k_caa-srdriver.16.06.0
- Cat3k_caa-webui.16.06.0

2.3.3 Setup

The table below lists the Cisco 3850-S switch running configuration used for the lab environment. In addition to the IOS version and a few generic configuration items, configuration items specifically relating to integration with the MUD manager and IoT devices are highlighted in bold fonts; these include DHCP, LLDP, AAA, RADIUS, and policies regarding access session. The table also provides a description of each configuration item for ease of understanding.

Table 2-1 Cisco 3850-S Switch Running Configuration

Configuration Item	Description
version 16.9 no service pad service timestamps debug datetime msec service timestamps log datetime msec service call-home no platform punt-keepalive disable-kernel-core ! hostname Build1 !	General overview of configuration information needed to configure AAA to use RADIUS and configure the RADIUS server itself. Note that the FreeRADIUS and AAA passwords must match.
aaa new-model **!**	Enables AAA
aaa authentication dot1x default group radius	Creates an 802.1X AAA authentication method list

Configuration Item	Description
aaa authorization network default group radius	Configures network authorization via RADIUS, including network-related services such as VLAN assignment
aaa accounting identity default start-stop group radius	Enables accounting method list for Session Aware Networking subscriber services
aaa accounting network default start-stop group radius ! **aaa server radius dynamic-author** **client 192.168.11.45 server-key cisco** **server-key cisco** ! aaa session-id common	Enables accounting for all network-related service requests Enables dynamic authorization local server configuration mode and specifies a RADIUS client/key from which a device accepts Change of Authorization (CoA) and disconnect requests
radius server AAA **address ipv4 192.168.11.45 auth-port 1812** **acct-port 1813** **key cisco**	Enables AAA server from the list of multiple AAA servers configured Uses the IP address and ports on which the FreeRADIUS server is listening
ip routing ! **ip dhcp excluded-address 192.168.10.1 192.168.10.100** **!** **ip dhcp pool NCCOE-V3** **network 192.168.13.0 255.255.255.0** **default-router 192.168.13.1** **dns-server 8.8.8.8** **lease 0 12** ! **ip dhcp pool NCCOE-V4** **network 192.168.14.0 255.255.255.0** **default-router 192.168.14.1** **dns-server 8.8.8.8** **!**	**Define a DHCP pool for IoT devices** **Note** To reserve a static address, use the hardware-address command as opposed to client address. DHCP server configuration to exclude selected addresses from pool DHCP server configuration to assign IP address to devices on VLAN 3 DHCP server configuration to assign IP address to devices on VLAN 4

Configuration Item	Description
ip dhcp pool NCCOE **network 192.168.10.0 255.255.255.0** **default-router 192.168.10.2** **dns-server 8.8.8.8** **lease 0 12** ! ! **ip dhcp snooping** **ip dhcp snooping vlan 1,3**	DHCP server configuration to assign IP address to devices on VLAN 1 Enables DHCP snooping globally Specifically enables DHCP snooping on VLANs 1 and 3
! **access-session attributes filter-list list mudtest** **lldp** **dhcp** **access-session accounting attributes filter-spec include list mudtest** **access-session monitor**	Configures access-session attributes to cause LLDP TLVs (including the MUD URL) to be forwarded in an accounting message to the AAA server
! dot1x logging verbose	Global configuration command to filter 802.1x authentication verbose messages
ldp run **!**	Enables LLDP, a discovery protocol that runs over Layer 2 (the data link layer) to gather information on non-Cisco-manufactured devices
policy-map type control subscriber mud-mab-test **event session-started match-all** **10 class always do-until-failure** **10 authenticate using mab** **!** **template mud-mab-test** **switchport mode access** **mab** **access-session port-control auto** **service-policy type control subscriber mud-mab-test** !	Configures identity control policies that define the actions that Session Aware Networking takes in response to specified conditions and subscriber events Enables policy-map (mud-mab-test) and template to cause Media Access Control (MAC) Address Bypass (MAB) to happen Dynamically applies an interface template to a target Sets the authorization state of a port. The default value is force-authorized.

Configuration Item	Description
	Applies the above previously configured control policy called mud-mab-test

!
!
interface GigabitEthernet1/0/1
  no switchport
  no ip address
  power inline never
!
interface GigabitEthernet1/0/2
!
interface GigabitEthernet1/0/3
!
interface GigabitEthernet1/0/4
  switchport access vlan 5
!
interface GigabitEthernet1/0/5
!
interface GigabitEthernet1/0/6
!
interface GigabitEthernet1/0/7
  switchport access vlan 3
!
interface GigabitEthernet1/0/8
  switchport access vlan 3
!
interface GigabitEthernet1/0/9
!
interface GigabitEthernet1/0/10
!
interface GigabitEthernet1/0/11
!
interface GigabitEthernet1/0/12
!	

interface GigabitEthernet1/0/13 ** source template mud-mab-test** **!**	Statically applies an interface template to a target, i.e., an IoT device
interface GigabitEthernet1/0/14 ** source template mud-mab-test** **!**	Statically applies an interface template to a target, i.e., an IoT device

Configuration Item	Description
interface GigabitEthernet1/0/15 **source template mud-mab-test** **!**	Statically applies an interface template to a target, i.e., an IoT device
interface GigabitEthernet1/0/16 **source template mud-mab-test** **!**	Statically applies an interface template to a target, i.e., an IoT device
interface GigabitEthernet1/0/17 **source template mud-mab-test** **!**	Statically applies an interface template to a target, i.e., an IoT device
interface GigabitEthernet1/0/18 **source template mud-mab-test** **!**	Statically applies an interface template to a target, i.e., an IoT device
interface GigabitEthernet1/0/19 **source template mud-mab-test** **!**	Statically applies an interface template to a target, i.e., an IoT device
interface GigabitEthernet1/0/20 **source template mud-mab-test**	Statically applies an interface template to a target, i.e., an IoT device
! interface GigabitEthernet1/0/21 ! interface GigabitEthernet1/0/22 ! interface GigabitEthernet1/0/23 switchport access vlan 2 ! interface GigabitEthernet1/0/24 switchport access vlan 2 ! interface GigabitEthernet1/1/1 ! interface GigabitEthernet1/1/2 ! interface GigabitEthernet1/1/3 ! interface GigabitEthernet1/1/4 ! interface TenGigabitEthernet1/1/1 ! interface TenGigabitEthernet1/1/2 ! interface TenGigabitEthernet1/1/3	

Configuration Item	Description
! interface TenGigabitEthernet1/1/4 ! **interface Vlan1** **ip address 192.168.10.2 255.255.255.0** ! **interface Vlan2** **ip address 192.168.11.1 255.255.255.0** ! **interface Vlan3** **ip address 192.168.13.1 255.255.255.0** ! **interface Vlan4** **ip address 192.168.14.1 255.255.255.0** ! **interface Vlan5** **ip address 192.168.15.1 255.255.255.0** !	Configure and address VLAN1 interface for inter-VLAN routing Configure and address VLAN2 interface for inter-VLAN routing Configure and address VLAN3 interface for inter-VLAN routing Configure and address VLAN4 interface for inter-VLAN routing Configure and address VLAN5 interface for inter-VLAN routing
! ip default-gateway 192.168.10.1 ip forward-protocol nd ip http server ip http authentication local ip http secure-server ip route 0.0.0.0 0.0.0.0 192.168.10.1 ip route 192.168.12.0 255.255.255.0 192.168.5.1 !	

2.4 DigiCert Certificates

2.4.1 DigiCert CertCentral Overview

DigiCert's CertCentral web-based platform allows for provisioning and management of publicly trusted X.509 certificates for a variety of purposes. After establishing an account, clients can log in, request, renew, and revoke certificates by using only a browser. 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.

2.4.2 Configuration Overview

This section typically documents the network, software, and hardware configurations, but that is not necessary for this component.

2.4.3 Setup

DigiCert allows certificates to be requested through their web-based platform, CertCentral. A user account is needed to access CertCentral. For details on creating a user account and getting set up with an account, follow the steps described here:

https://www.digicert.com/certcentral-support/digicert-getting-started-guide.pdf

2.4.3.1 TLS Certificate

For this example implementation, we leveraged DigiCert's Private TLS certificate because the MUD file server is hosted internally. This certificate supports https connections to the MUD file server, which are required by the MUD manager. Additional information about the TLS certificates offered by DigiCert can be found at https://www.digicert.com/security-certificate-support/.

For instructions on how to request a certificate, proceed to the DigiCert documentation found here and follow the process for the specific certificate being requested: https://www.digicert.com/certcentral-support/basic-certcentral-getting-started-guide-v1.4_2018-12-10_opt.pdf

Once requested, integrate the certificate onto the MUD file server as described in Section 2.2.3.1.

2.4.3.2 Premium Certificate

To sign MUD files according to the MUD specification, a client certificate is required. For this implementation, we leveraged DigiCert's Premium certificate to sign MUD files. This certificate supports signing or encrypting secure/multipurpose internet mail extensions (S/MIME) messages, which is required by the specification.

For detailed instructions on how to request and implement a Premium certificate, proceed to the DigiCert documentation found here: https://www.digicert.com/certcentral-support/client-certificate-guide.pdf

Once requested, sign MUD files as described in Section 2.2.3.2.2.

2.5 IoT Devices

2.5.1 Molex PoE Gateway and Light Engine

This section provides configuration details of the MUD-capable Molex PoE Gateway and Light Engine used in the example implementation, which emits a MUD URL that uses LLDP.

2.5.1.1 Configuration Overview

The Molex PoE Gateway runs firmware created and provided by Molex. This firmware was modified by Molex to emit a MUD URL that uses an LLDP message.

2.5.1.1.1 Network Configuration

The Molex PoE Gateway is connected to the network over a wired Ethernet connection. The IP address is assigned dynamically by using DHCP.

2.5.1.1.2 Software Configuration

For this example implementation, the Molex PoE Gateway is configured with Molex's PoE Gateway firmware, version 1.6.1.8.4.

2.5.1.1.3 Hardware Configuration

The Molex PoE Gateway used in this build was Model Number 180993-0001, dated 03/2017.

2.5.1.2 Setup

The Molex PoE Gateway is controlled via the Constrained Application Protocol (CoAP), and CoAP commands were used to ensure that device functionality was maintained during the MUD process.

2.5.1.2.1 DHCP Client Configuration

The device uses the default DHCP client included in the Molex PoE Gateway firmware.

2.5.2 IoT Development Kits–Linux-Based

This section provides configuration details for the Linux-based IoT development kits used in the example implementation, which emit MUD URLs by using DHCP. It also provides information regarding a basic IoT application used to test the MUD process.

2.5.2.1 Configuration Overview

The devkits run various flavors of Linux-based operating systems and are configured to emit a MUD URL during a typical DHCP transaction. They also run a Python script that allows the devkits to receive and process commands by using the MQTT protocol, which can be sent to peripherals connected to the devkits.

2.5.2.1.1 Network Configuration

The devkits are connected to the network over a wired Ethernet connection. The IP address is assigned dynamically by using DHCP.

2.5.2.1.2 Software Configuration

For this example implementation, the Raspberry Pi is configured on Raspbian 9, the Samsung ARTIK 520 is configured on Fedora 24, and the Intel UP Squared Grove is configured on Ubuntu 16.04 LTS. The

devkits also utilized dhclient as the default DHCP client. This DHCP client is installed natively on many Linux distributions and can be installed using a preferred package manager if not currently present.

2.5.2.1.3 Hardware Configuration

The hardware used for these devkits included the Raspberry Pi 3 Model B, Samsung ARTIK 520, and Intel UP Squared Grove.

2.5.2.2 Setup

The following subsection describes setting up the devkits to send a MUD URL during the DHCP transaction and to act as a smart device by leveraging an MQTT broker server (we describe setting up the MQTT broker server in Section 2.8).

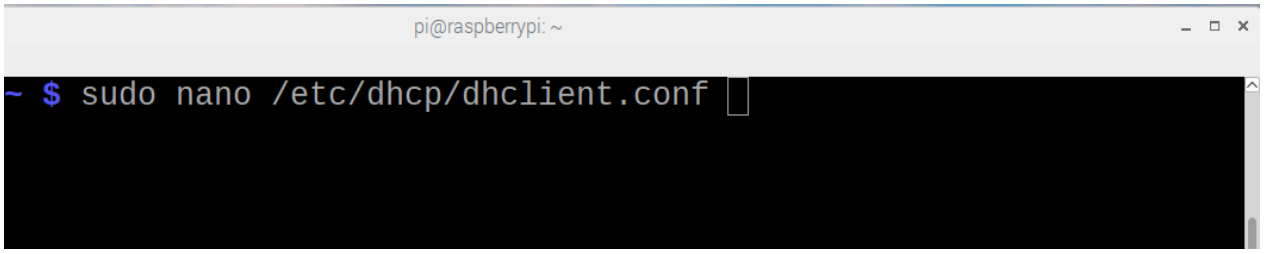
2.5.2.2.1 DHCP Client Configuration

We leveraged dhclient as the default DHCP client for these devices due to the availability of the DHCP client on different Linux platforms and the ease of emitting MUD URLs via DHCP.

To set up the dhclient configuration:

1. Open a terminal on the device.
2. Ensure that any other conflicting DHCP clients are disabled or removed.
3. Install the dhclient package (if needed).
4. Edit the *dhclient.conf* file by entering the following command:

```
sudo nano /etc/dhcp/dhclient.conf
```



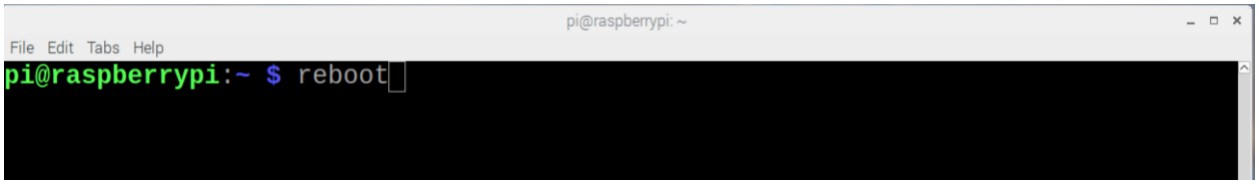
5. Add the following lines:

```
option mud-url code 161 = text;
send mud-url = "<insert URL for MUD File here>";
```

```
  GNU nano 2.7.4          File: /etc/dhcp/dhclient.conf          Modified

#lease {
#  interface "eth0";
#  fixed-address 192.33.137.200;
#  medium "link0 link1";
#  option host-name "andare.swiftmedia.com";
#  option subnet-mask 255.255.255.0;
#  option broadcast-address 192.33.137.255;
#  option routers 192.33.137.250;
#  option domain-name-servers 127.0.0.1;
#  renew 2 2000/1/12 00:00:01;
#  rebind 2 2000/1/12 00:00:01;
#  expire 2 2000/1/12 00:00:01;
#}

#DHCP MUD Option
option mud-url code 161 = text;
send mud-url = "https://mudfileserver/pi4";



^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

6. Save and close the file.
7. Reboot the device:

```
reboot
```



8. Open a terminal.
9. Execute the dhclient:

```
sudo dhclient -v
```

2.5.2.2.2 IoT Application for Testing

The following Python application was created by the NCCoE to enable the devkits to act as basic IoT devices:

```
#Program:                IoTapp.
#Version:                1.0
```

```
#Purpose:                  Provide IoT capabilities to devkit.
#Protocols:          MQTT.
#Functionality:      Allow remote control of LEDs on connected breadboard.

#Libraries
import paho.mqtt.client as mqttClient
import time
import RPi.GPIO as GPIO

#Global Variables
BrokerAddress = "192.168.1.87"    #IP address of Broker(Server), change as needed. Best
practice would be a registered domain name that can be queried for appropriate server
address.
BrokerPort = "1883"        #Default port used by most MQTT Brokers. Would be 1883 if
using Transport Encryption with TLS.
ConnectionStatus = "Disconnected"#Status of connection to Broker. Should be either
"Connected" or "Disconnected".
LED = 26

#Supporting Functions
def on_connect(client, userdata, flags, rc):   #Function for connection status to
Broker.
       if rc == 0:
              ConnectionStatus = "Connected to Broker!"
              print(ConnectionStatus)
       else:
              ConnectionStatus = "Connection Failed!"
              print(ConnectionStatus)

def on_message(client, userdata, msg):         #Function for parsing message data.
       if "ON" in msg.payload:
              print("ON!")
```

```
780                 GPIO.output(LED, 1)
781
782         if "OFF" in msg.payload:
783                 print("OFF!")
784                 GPIO.output(LED, 0)
785
786   def  MQTTapp():
787         client = mqttClient.Client()      #New instance.
788         client.on_connect = on_connect
789         client.on_message = on_message
790         client.connect(BrokerAddress, BrokerPort)
791         client.loop_start()
792         client.subscribe("test")
793         try:
794                 while True:
795                         time.sleep(1)
796         except KeyboardInterrupt:
797                 print("8")
798                 client.disconnect()
799                 client.loop_stop()
800
801   #Main Function
802   def main():
803
804         GPIO.setmode(GPIO.BCM)
805         GPIO.setup(LED, GPIO.OUT)
806
807         print("Main function has been executed!")
808         MQTTapp()
809
```

```
if __name__ == "__main__":
        main()
```

2.5.3 IoT Development Kit–u-blox C027-G35

This section details configuration of a u-blox C027-G35, which emits a MUD URL by using DHCP, and a basic IoT application used to test MUD rules.

2.5.3.1 Configuration Overview

This devkit runs the ARM Mbed-OS operating system and is configured to emit a MUD URL during a typical DHCP transaction. It also runs a basic IoT application to test MUD rules.

2.5.3.1.1 Network Configuration

The u-blox C027 is connected to the network over a wired Ethernet connection. The IP address is assigned dynamically by using DHCP.

2.5.3.1.2 Software Configuration

For this example implementation, the u-blox C027-G35 was configured on the Mbed-OS 5.10.4 operating system.

2.5.3.1.3 Hardware Configuration

The hardware used for this devkit is the u-blox C027-G35.

2.5.3.2 Setup

The following subsection describes setting up the u-blox C027-G35 to send a MUD URL in the DHCP transaction and act as a smart device by establishing network connections to the update server and other destinations.

2.5.3.2.1 DHCP Client Configuration

To add MUD functionality to the Mbed-OS DHCP client, the following two files inside Mbed-OS require modification:

- *mbed-os/features/lwipstack/lwip/src/include/lwip/prot/dhcp.h*
  - **NOT**: *mbed-os/features/lwipstack/lwip/src/include/lwip/dhcp.h*
- *mbed-os/features/lwipstack/lwip/src/core/ipv4/lwip_dhcp.c*

Changes to include/lwip/prot/dhcp.h:

1. Add the following line below the greatest DCHP option number (67) on Line 170:

```
#define DHCP_OPTION_MUD_URL_V4    161 /*MUD: RFC-ietf-opsawg-mud-25 draft-ietf-opsawg-mud-08, Manufacturer Usage Description*/
```

Changes to core/ipv4/lwip_dhcp.c:

1. Change within container around Line 141:
   To enum dhcp_option_idx (at Line 141) before the first #if, add

```
DHCP_OPTION_IDX_MUD_URL_V4, /*MUD: DHCP MUD URL Option*/
```

   It should now look like the screenshot below:

```
enum dhcp_option_idx {
  DHCP_OPTION_IDX_OVERLOAD = 0,
  DHCP_OPTION_IDX_MSG_TYPE,
  DHCP_OPTION_IDX_SERVER_ID,
  DHCP_OPTION_IDX_LEASE_TIME,
  DHCP_OPTION_IDX_T1,
  DHCP_OPTION_IDX_T2,
  DHCP_OPTION_IDX_SUBNET_MASK,
  DHCP_OPTION_IDX_ROUTER,
  DHCP_OPTION_IDX_MUD_URL_V4, /*MUD: DHCP MUD URL Option*/
#if LWIP_DHCP_PROVIDE_DNS_SERVERS
  DHCP_OPTION_IDX_DNS_SERVER,
  DHCP_OPTION_IDX_DNS_SERVER_LAST = DHCP_OPTION_IDX_DNS_SERVER +
LWIP_DHCP_PROVIDE_DNS_SERVERS - 1,
#endif /* LWIP_DHCP_PROVIDE_DNS_SERVERS */
#if LWIP_DHCP_GET_NTP_SRV
  DHCP_OPTION_IDX_NTP_SERVER,
  DHCP_OPTION_IDX_NTP_SERVER_LAST = DHCP_OPTION_IDX_NTP_SERVER +
LWIP_DHCP_MAX_NTP_SERVERS - 1,
#endif /* LWIP_DHCP_GET_NTP_SRV */
  DHCP_OPTION_IDX_MAX
};
```

2. Change within the function around Line 975:
   a. To list of local variables for static err_t dhcp_discover(struct netif *netif), add the desired MUD URL (www.example.com used here):

```
char* mud_url = "https://www.example.com";    /*MUD: MUD URL*/
```

NOTE: The MUD URL must be less than 255 octets/bytes/characters long.

   b. Within if (result == ERR_OK) after

```
dhcp_option(dhcp, DHCP_OPTION_PARAMETER_REQUEST_LIST,
LWIP_ARRAYSIZE(dhcp_discover_request_options));
  for (i = 0; i < LWIP_ARRAYSIZE(dhcp_discover_request_options); i++) {
   dhcp_option_byte(dhcp, dhcp_discover_request_options[i]);
  }
```

and before:

```
  dhcp_option_trailer(dhcp);
```

add:

```
  /*MUD: Begin - Add Option and URL to DISCOVER/REQUEST*/
#if (DHCP_DEBUG != LWIP_DBG_OFF)
  if (strlen(mud_url) > 255)
   LWIP_DEBUGF(DHCP_DEBUG | LWIP_DBG_TRACE, ("dhcp_discover: MUD URL is too large (>255)\n"));
#endif /* DHCP_DEBUG != LWIP_DBG_OFF */

  u8_t mud_url_len = (strlen(mud_url) < 255)? strlen(mud_url) : 255; //Ignores any URL greater than 255
bytes/octets
  dhcp_option(dhcp, DHCP_OPTION_MUD_URL_V4, mud_url_len);
  for (i = 0; i < mud_url_len; i++) {
   dhcp_option_byte(dhcp, mud_url[i]);
  }
  /*MUD: END - Add Option and URL to DISCOVER/REQUEST */
```

3. Change within the function around Line 1486:
   Within the following function:

```
static err_t
dhcp_parse_reply(struct dhcp *dhcp, struct pbuf *p)
```

Within switch(op) before default, add the following case (around line 1606):

```
    case(DHCP_OPTION_MUD_URL_V4): /* MUD Testing */
     LWIP_ERROR("len == 0", len == 0, return ERR_VAL;);
     decode_idx = DHCP_OPTION_IDX_MUD_URL_V4;
     break;
```

4. Compile by using the following command:

```
mbed compile -m ublox_c027 -t gcc_arm
```

2.5.3.2.2 IoT Application for Testing

The following application was created by the NCCoE to enable the devkit to test the example implementation as a MUD-capable device:

```
#include "mbed.h"
#include "EthernetInterface.h"

//DigitalOut led1(LED1);
PwmOut led2(LED2);
Serial pc(USBTX, USBRX);

float brightness = 0.0;

// Network interface
EthernetInterface net;

// Socket demo
int main() {
  int led1 = true;

  for (int i = 0; i < 4; i++) {

    led2 = (led1)? 0.5 : 0.0;

    led1 = !led1;
    wait(0.5);
  }

  for (int i = 0; i < 8; i++) {

    led2 = (led1)? 0.5 : 0.0;

```

```
    led1 = !led1;
    wait(0.25);
  }

  for (int i = 0; i < 8; i++) {

    led2 = (led1)? 0.5 : 0.0;

    led1 = !led1;
    wait(0.125);
  }
  TCPSocket socket;
  char sbuffer[] = "GET / HTTP/1.1\r\nHost: www.updateserver.com\r\n\r\n";
  char bbuffer[] = "GET / HTTP/1.1\r\nHost: www.unapprovedserver.com\r\n\r\n";
  int scount, bcount;
  char rbuffer[64];
  char brbuffer[64];
  int rcount, brcount;

  /* By default grab an IP address*/
  // Bring up the ethernet interface
  pc.printf("Ethernet socket example\r\n");
  net.connect();
  // Show the network address
  const char *ip = net.get_ip_address();
  pc.printf("IP address is: %s\r\n", ip ? ip : "No IP");
  socket.open(&net);
  /* End of default IP address */

  pc.printf("Press U to turn LED1 brightness up, D to turn it down, G to get IP, R to
release IP, H for HTTP request, B for blocked HTTP request\r\n");
```

```
while(1) {
  char c = pc.getc();
  if((c == 'u') && (brightness < 0.5)) {
    brightness += 0.01;
    led2 = brightness;
  }
  if((c == 'd') && (brightness > 0.0)) {
    brightness -= 0.01;
    led2 = brightness;
  }
  if(c == 'g'){
    // Bring up the ethernet interface
    pc.printf("Sending DHCP Request...\r\n");
    net.connect();
    // Show the network address
    const char *ip = net.get_ip_address();
    pc.printf("IP address is: %s\r\n", ip ? ip : "No IP");
  }
  if(c == 'r'){
    socket.close();
    net.disconnect();
    pc.printf("IP Address Released\r\n");
  }
  if(c == 'h'){

   pc.printf("Sending HTTP Request...\r\n");
   // Open a socket on the network interface, and create a TCP connection
   socket.open(&net);
   socket.connect("www.updateserver.com", 80);
```

```
      // Send a simple http request
      scount = socket.send(sbuffer, sizeof sbuffer);
      pc.printf("sent %d [%.*s]\r\n", scount, strstr(sbuffer, "\r\n")-sbuffer, sbuffer);
      // Receive a simple http response and print out the response line
      rcount = socket.recv(rbuffer, sizeof rbuffer);
      pc.printf("recv %d [%.*s]\r\n", rcount, strstr(rbuffer, "\r\n")-rbuffer, rbuffer);
      socket.close();
     }
     if(c == 'b'){
      pc.printf("Sending Blocked HTTP Request...\r\n");
      // Open a socket on the network interface, and create a TCP connection
      socket.open(&net);
      socket.connect("www.unapprovedserver.com", 80);
      // Send a simple http request
      bcount = socket.send(bbuffer, sizeof bbuffer);
      pc.printf("sent %d [%.*s]\r\n", bcount, strstr(bbuffer, "\r\n")-bbuffer, bbuffer);

      // Receive a simple http response and print out the response line
      brcount = socket.recv(brbuffer, sizeof brbuffer);
     pc.printf("recv %d [%.*s]\r\n", brcount, strstr(brbuffer, "\r\n")-brbuffer,
brbuffer);
      socket.close();
     }
   }
}
```

2.5.4 IoT Devices–Non-MUD Capable

This section details configuration of non-MUD-capable IoT devices attached to the implementation network. These include several types of devices, such as cameras, smartphones, lighting, a smart assistant, a printer, a baby monitor, a wireless access point, and a digital video recorder. These devices did not emit a MUD URL or have MUD capabilities of any kind.

2.5.4.1 Configuration Overview

These non-MUD-capable IoT devices are unmodified and still retain the default manufacturer configurations.

2.5.4.1.1 Network Configuration

These IoT devices are configured to obtain an IP address via DHCP.

2.5.4.1.2 Software Configuration

The software on these devices is configured according to standard manufacturer instructions.

2.5.4.1.3 Hardware Configuration

The hardware used in these devices is unmodified from manufacturer specifications.

2.5.4.2 Setup

These devices were set up according to the manufacturer instructions and connected to the Cisco switch via Ethernet cable or connected wirelessly through the wireless access point.

2.5.4.2.1 DHCP Client Configuration

These IoT devices used the default DHCP clients provided by the original manufacturer and were not modified in any way.

2.6 Update Server

This section describes how to implement a server that will act as an update server. It will attempt to access and be accessed by the IoT device, in this case one of the development kits we built in the lab.

2.6.1 Update Server Overview

The update server is an Apache web server that hosts mock software update files to be served as software updates to our IoT device devkits. When the server receives an http request, it sends the corresponding update file.

2.6.2 Configuration Overview

The following subsections document the software, hardware, and network requirements for the update server.

2.6.2.1 Network Configuration

The IP address was statically assigned.

2.6.2.2 Software Configuration

For this example implementation, the update server was configured on the Ubuntu 18.04 LTS operating system.

2.6.2.3 Hardware Configuration

The update server was hosted in the NCCoE's virtual environment, functioning as a cloud service.

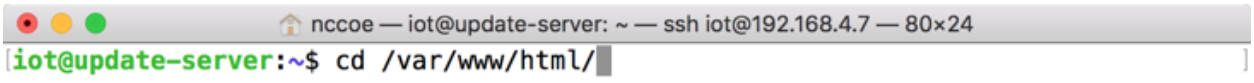
2.6.3 Setup

The Apache web server was set up by using the official Apache documentation at https://httpd.apache.org/docs/current/install.html. After this, SSL/TLS encryption was set up by using the digital certificate and key obtained from DigiCert. This was set up by using the official Apache documentation, found at https://httpd.apache.org/docs/current/ssl/ssl_howto.html.

The following configurations were made to the server to host the update file:

1. Open a terminal.
2. Change directories to the Hypertext Markup Language (HTML) folder:

```
cd /var/www/html
```



3. Create the update file (Note: this is a mock update file):

```
touch IoTsoftwareV2.tar.gz
```

2.7 Unapproved Server

This section describes how to implement a server that will act as an unapproved server. It will attempt to access and to be accessed by an IoT device, in this case one of the MUD-capable devices on the implementation network.

2.7.1 Unapproved Server Overview

The unapproved server is an internet host that is not explicitly authorized in the MUD file to communicate with the IoT device. When the IoT device attempts to connect to this server, the router or switch should not allow this traffic because it is not an approved internet service per the corresponding MUD file. Likewise, when the server attempts to connect to the IoT device, this traffic should be denied at the router or switch.

2.7.2 Configuration Overview

The following subsections document the software, hardware, and network configurations for the unapproved server.

2.7.2.1 Network Configuration

The unapproved server hosts a web server that is accessed via TCP port 80. Any applications that request access to this server need to be able to connect on this port. Use firewall-cmd, iptables, or any other system utility for manipulating the firewall to open this port.

2.7.2.2 Software Configuration

For this example implementation, the CentOS 7 operating system was leveraged with an Apache web server.

2.7.2.3 Hardware Configuration

The unapproved server was hosted in the NCCoE's virtual environment, functioning as a cloud service. The IP address was statically assigned.

2.7.3 Setup

The following subsection describes the setup process for configuring the unapproved server.

2.7.3.1 Apache Web Server

The Apache web server was set up by using the official Apache documentation at https://httpd.apache.org/docs/current/install.html. SSL/TLS encryption was not used for this server.

2.8 MQTT Broker Server

2.8.1 MQTT Broker Server Overview

For this example implementation, the open-source tool Mosquitto was used as the MQTT broker server. The server communicates publish and subscribe messages between multiple clients. For our implementation, this server 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 MQTT-enabled IoT device.

2.8.2 Configuration Overview

The following subsections document the software, hardware, and network requirements for the MQTT broker server.

2.8.2.1 Network Configuration

The MQTT broker server was hosted in the NCCoE's virtual environment, functioning as a cloud service. The IP address was statically assigned.

The server is accessed via TCP port 1883. Any clients that require access to this server need to be able to connect on this port. Use firewall-cmd, iptables, or any other system utility for manipulating the firewall to open this port.

2.8.2.2 Software Configuration

For this example implementation, the MQTT broker server was configured on an Ubuntu 18.04 LTS operating system.

2.8.2.3 Hardware Configuration

This server was hosted in the NCCoE's virtual environment, functioning as a cloud service. The IP address was statically assigned.

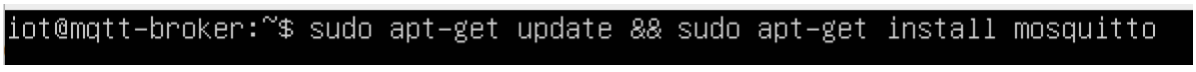
2.8.3 Setup

In this section we describe setting up the MQTT broker server to communicate messages to and from the controlling application and the IoT device.

2.8.3.1 Mosquitto Setup

1. Install the open-source MQTT broker server, Mosquitto, by entering the following command:

```
sudo apt-get update && sudo apt-get install mosquitto
```



Following the installation, this implementation leveraged the default configuration of the Mosquitto server. The MQTT broker server was set up by using the official Mosquitto documentation at https://mosquitto.org/man/.

2.9 ForeScout–IoT Device Discovery

This section describes how to implement ForeScout's CounterACT appliance and Enterprise Manager to provide device discovery on the network.

2.9.1 ForeScout Overview

The CounterACT appliance discovers, catalogs, profiles, and classifies the devices that are connected to the demonstration network. When a device is added to or removed from the network, the CounterACT appliance is updated and actively monitors these devices on the network. The administrator will be able to manage multiple CounterACT appliances from a central point by integrating the CounterACT appliance with the Enterprise Manager.

2.9.2 Configuration Overview

The following subsections document the software, hardware, and network requirements for the CounterACT appliance and Enterprise Manager.

2.9.2.1 Network Configuration

The virtual CounterACT appliance was hosted on VLAN 2 of the Cisco switch. It was set up with just the monitor interface. The network configuration for the CounterACT appliance was completed by using the official ForeScout documentation at https://www.forescout.com/wp-content/uploads/2018/10/CounterACT_Installation_Guide_8.0.1.pdf (see Chapters 2 and 8).

The virtual Enterprise Manager was hosted in the virtual environment that is shared across each build.

2.9.2.2 Software Configuration

The example implementation leveraged a virtual CounterACT appliance VCT-R version 8.0.1 along with a virtual Enterprise Manager VCEM-05 version 8.0.1. Both of the virtual appliances were built on a Linux operating system supported by ForeScout.

ForeScout provides software for managing the appliances on the network. The CounterACT Console is software that allows management of the CounterACT appliance/Enterprise Manager and visualization of the data gathered by the appliances.

2.9.2.3 Hardware Configuration

The example implementation leveraged a virtual CounterACT appliance, which was set up in the lab environment on a dedicated machine hosting the local virtual machines in Build 1.

The virtual Enterprise Manager was hosted in the NCCoE's virtual environment with a static IP assignment.

2.9.3 Setup

In this section we describe setting up the virtual CounterACT appliance and the virtual Enterprise Manager.

2.9.3.1 CounterACT Appliance Setup

The virtual CounterACT appliance was set up by using the official ForeScout documentation at https://www.forescout.com/wp-content/uploads/2018/10/CounterACT_Installation_Guide_8.0.1.pdf (see Chapters 3 and 8).

2.9.3.2 Enterprise Manager Setup

The Enterprise Manager was set up by using the official ForeScout documentation at https://www.forescout.com/wp-content/uploads/2018/10/CounterACT_Installation_Guide_8.0.1.pdf (see Chapters 4 and 8).

Using the Enterprise Manager, we configured the following modules:

- Endpoint
- Network
- Authentication
- Core Extension
- Device Profile Library—https://www.forescout.com/wp-content/uploads/2018/04/CounterACT_Device_Profile_Library.pdf
- IoT Posture Assessment Library—https://www.forescout.com/wp-content/uploads/2018/04/CounterACT_IoT_Posture_Assessment_Library-1.pdf
- network interface card (NIC) Vendor DB—https://www.forescout.com/wp-content/uploads/2018/04/CounterACT_NIC_Vendor_DB_17.0.12.pdf
- Windows Applications—https://www.forescout.com/wp-content/uploads/2018/04/CounterACT_Windows_Applications.pdf
- Windows Vulnerability database (DB)—https://www.forescout.com/wp-content/uploads/2018/04/CounterACT_Windows_Vulnerability_DB_18.0.2.pdf
- Open Integration Module—https://www.forescout.com/wp-content/uploads/2018/08/CounterACT_Open_Integration_Module_Overview_1.1.pdf

Appendix A List of Acronyms

2FA	Two-factor Authentication
AAA	Authentication, Authorization, and Accounting
CoA	Change of Authorization
CoAP	Constrained Application Protocol
CRADA	Cooperative Research and Development Agreement
Cybersecurity Framework	National Institute of Standards and Technology (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
MAB	MAC Address 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
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 (IETF Standard)
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 (Institute of Electrical and Electronics Engineers (IEEE) 802.11i-2004 standard)
WPA3	Wi-Fi Protected Access 3 Security Certificate Protocol
YANG	Yet Another Next Generation

1148

Appendix B Glossary

Audit	Independent review and examination of records and activities to assess the adequacy of system controls to ensure compliance with established policies and operational procedures (National Institute of Standards and Technology [NIST] Special Publication [SP] 800-12 Rev. 1)
Best Practice	A procedure that has been shown by research and experience to produce optimal results and that is established or proposed as a standard suitable for widespread adoption (Merriam-Webster)
Botnet	The word botnet is formed from the words robot and network. Cybercriminals 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 these devices. 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 IoT device that is capable of emitting a MUD uniform resource locator (URL) in compliance with the MUD specification
Network Address Translation (NAT)	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 systems 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)
Security Control	A safeguard or countermeasure prescribed for an information system or an organization, which is 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)
Server	A computer or device on a network that manages network resources. Examples are 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)
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 (Federal Information Processing Standard (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, traceback, 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 (hypertext transfer 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 Internet of Things devices.
Virtual Local Area Network (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)

1150

Appendix C References

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

[2] Cisco's developer MUD Manager GitHub page. Available: https://github.com/CiscoDevNet/MUD-Manager/tree/1.0#dependancies.

[3] Apache HTTP Server Project documentation, Version 2.4, Compiling and installing Apache [Website], https://httpd.apache.org/docs/current/install.html [accessed 3/5/19].

[4] Apache HTTP Server Project documentation, Version 2.4, Apache SSL/TLS Encryption [Website], https://httpd.apache.org/docs/current/ssl/ssl_howto.html [accessed 3/5/19].

[5] Welcome to MUD File maker! [Website], www.mudmaker.org [accessed 3/5/19].

[6] Advanced CertCentral™ Getting Started Guide, Version 9.2, DigiCert [Website], https://www.digicert.com/certcentral-support/digicert-getting-started-guide.pdf [accessed 3/5/19].

[7] SSL Certificate Support, DigiCert [Website], https://www.digicert.com/security-certificate-support/ [accessed 3/5/19].

[8] Basic CertCentral™ Getting Started Guide, Version 1.4, DigiCert [Website], https://www.digicert.com/certcentral-support/basic-certcentral-getting-started-guide-v1.4_2018-12-10_opt.pdf [accessed 3/19/19].

[9] CertCentral™ Client Certificate Guide, Version 1.9, DigiCert [Website], https://www.digicert.com/certcentral-support/client-certificate-guide.pdf [accessed 3/5/19].

[10] ForeScout CounterAct® Installation Guide, Version 8.0.1, ForeScout [Website], https://www.forescout.com/wp-content/uploads/2018/10/CounterACT_Installation_Guide_8.0.1.pdf [accessed 3/5/19].

[11] ForeScout CounterAct Device Profile Library Configuration Guide, Updated February 2018 [Website], https://www.forescout.com/wp-content/uploads/2018/04/CounterACT_Device_Profile_Library.pdf [accessed 3/5/19].

[12] ForeScout CounterAct IoT Posture Assessment Library Configuration Guide, Updated February 2018 [Website], https://www.forescout.com/wp-content/uploads/2018/04/CounterACT_IoT_Posture_Assessment_Library-1.pdf [accessed 3/5/19].

[13] ForeScout CounterAct Open Integration Module Overview Guide, Version 1.1 [Website], https://www.forescout.com/wp-content/uploads/2018/08/CounterACT_Open_Integration_Module_Overview_1.1.pdf [accessed 3/5/19].

[14] ForeScout CounterAct Windows Applications Configuration Guide, Updated February 2018 [Website], https://www.forescout.com/wp-content/uploads/2018/04/CounterACT_Windows_Applications.pdf [accessed 3/5/19].

[15] ForeScout CounterAct Windows Vulnerability DB Configuration Guide, Updated February 2018 [Website], https://www.forescout.com/wp-content/uploads/2018/04/CounterACT_Windows_Vulnerability_DB_18.0.2.pdf [accessed 3/5/19].

[16] ForeScout HPS NIC Vendor DB Configuration Guide, Version 1.2.4 [Website], https://www.forescout.com/wp-content/uploads/2018/04/HPS_NIC_Vendor_DB_1.2.4.pdf [accessed 3/5/19].