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

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

Functional Demonstration Results
Supplement to NIST Special Publication 1800-15B

Mudumbai Ranganathan
NIST

William C. Barker
Dakota Consulting

Drew Cohen
Kevin Yeich
MasterPeace Solutions

Eliot Lear
Cisco

Adnan Baykal
Global Cyber Alliance

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

November 2019

PRELIMINARY DRAFT

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

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

NATIONAL CYBERSECURITY CENTER OF EXCELLENCE

Contents

List of Tables

1 Introduction

The National Institute of Standards and Technology (NIST) Cybersecurity Practice Guide explains how the [Manufacturer Usage Description (MUD) Specification (Internet Engineering Task Force [IETF] Request for Comments [RFC] 8520)](https://tools.ietf.org/html/rfc8520) can be used to reduce the vulnerability of Internet of Things (IoT) devices to botnets and other network-based threats as well as reduce the potential for harm from exploited IoT devices. It describes the logical architecture of a standards-based reference design for using MUD, threat signaling, and employing software updates to significantly increase the effort required by malicious actors to compromise and exploit IoT devices on a home or small-business network. It provides users with the information they need to replicate deployment of the MUD protocol to mitigate IoT-based distributed denial of service (DDoS) threats. The 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 solutions

This document, *Functional Demonstration Results,* is a supplement to NIST SP 1800-15B, *Approach, Architecture, and Security Characteristics*. This proof-of-concept document describes the functional demonstration results for three implementations of the reference design that were demonstrated as part of this National Cybersecurity Center of Excellence (NCCoE) project. These implementations are referred to as *builds.* Four builds are implemented, one of which is still under development. The functional demonstration results of three of these builds are reported in this document:

- Build 1 uses equipment from Cisco Systems and Forescout. The Cisco MUD Manager is used to provide support for MUD, and the Forescout Virtual Appliances and Enterprise Manager are used to perform non-MUD-related device discovery on the network.
- Build 2 uses equipment from MasterPeace Solutions Ltd., Global Cyber Alliance (GCA), and ThreatSTOP. The MasterPeace Solutions Yikes! router, cloud service, and mobile application are used to support MUD, as well as to perform device discovery on the network and to apply additional traffic rules to both MUD-capable and non-MUD-capable devices based on device manufacturer and model. The GCA Quad9 DNS Service and the ThreatSTOP Threat MUD File Server are used to support threat signaling.
- Build 3 uses equipment from CableLabs to onboard devices and support MUD. Although limited functionality of a preliminary version of this build has been demonstrated as part of this project, elements of Build 3 are still under development. Therefore, it has not yet been subjected to functional evaluation or demonstration of the full range of its capabilities.
- Build 4 uses software developed at the NIST Advanced Networking Technologies laboratory. This software serves as a working prototype for demonstrating the feasibility and scalability characteristics of the MUD RFC.

For a more comprehensive description of each build and a detailed explanation of each build's architecture and technologies, refer to NIST SP 1800-15B.

1.1 Objective

This document, *Functional Demonstration Results*, reports the results of the functional evaluation and demonstration of Builds 1, 2, and 4. For each of these builds, we defined a list of requirements unique to that build and then developed a set of test cases to verify that the build meets those requirements. The requirements, test cases, and test results for each of these three builds are documented below.

1.2 Functional Demonstration Activities

Builds 1, 2, and 4 were tested to determine the extent to which they correctly implement basic functionality defined within the MUD RFC. Builds 1 and 2 were also subjected to additional exercises that were designed to demonstrate non-MUD-related capabilities. These additional exercises were demonstrative rather than evaluative. They did not verify the build's behavior for conformance to a standard or specification; they were designed to demonstrate advertised capabilities of the builds related to their ability to increase device and network security in ways that are independent of the MUD RFC. These additional capabilities may provide security for both non-MUD-capable and MUD-capable devices. Examples of this type of capability include device discovery, identification and classification, and support for threat signaling.

1.3 Assumptions

The physical architecture of each build as deployed in the NCCoE laboratory environment is depicted and described in NIST SP 1800-15B. Tests for each build were run on the lab architecture documented in NIST SP 1800-15B. Prior to testing each build, all communication paths to the IoT devices on the network were open and could potentially be used to attack systems on the internet. For traffic to be sent between IoT devices, it was required to pass through the router/switch that served as the policy enforcement point (PEP) for the MUD rules.

In the lab setup for each build, the following hosts and web servers were required to be set up and available to support the tests defined below. On the local network where the IoT devices are located, hosts with the following names must exist and be reachable from an IoT device that is plugged into the local network:

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

- *anyhost-to* (i.e., a local host to which the IoT device in question is permitted to initiate communications but not vice versa)
- *anyhost-from* (i.e., a local host that is permitted to initiate communication to the IoT device but not vice versa)
- *same-manufacturer-host* (i.e., a local host that is from the same manufacturer as the IoT device in question. For example, if device A's MUD file is found at URL www.aaa.com and device B's MUD file is also found at URL www.aaa.com, then device B could be *same-manufacturer-host*.)

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

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

We also defined several MUD files for each build (provided in each build section below) that were used to evaluate specific capabilities.

1.4 Document Conventions

For each build, a set of requirements and a corresponding set of functional test cases were defined to verify that the build meets a specific set of requirements that are unique to that build. For evaluating MUD-related capabilities, these requirements are closely aligned to the order of operations in the Manufacturer Usage Description Specification (RFC 8520). However, even for MUD-specific tests, there are tests that are applicable to some builds but not to others, depending on how any given build is implemented.

For each build, the MUD-related requirements for that build are listed in a table. Each of these requirements is associated with two separate tests, one using Internet Protocol version 4 (IPv4) and one using IPv6. At the time of testing, however, IPv6 functionality was not fully supported by any of the builds and so was not evaluated. The names of the tests in which each requirement is tested are listed in the rightmost column of the requirements table for each build. Tests that end with the suffix "v4" are those in which IPv4 addressing is used; tests that end with the suffix "v6" are those in which IPv6 addressing is used. Only the IPv4 versions of each test are listed explicitly in this document. For each test that has both an IPv4 and an IPv6 version, the IPv4 version of the test, IoT-n-v4, is identical to the IPv6 version of the test, IoT-n-v6, except:

- IoT-n-v6 devices are configured to use IPv6, whereas IoT-n-v4 devices are configured to use IPv4.
- IoT-n-v6 devices are configured to use Dynamic Host Configuration Protocol version 6 (DHCPv6), whereas IoT-n-v4 devices are configured to use DHCPv4.
- The IoT-n-v6 DHCPv6 message that is emitted includes the MUD URL option that uses Internet Assigned Numbers Authority (IANA) code 112, whereas the IoT-n-v4 DHCPv4 message that is emitted includes the MUD URL option that uses IANA code 161.

Each test consists of multiple fields that collectively identify the goal of the test, the specifics required to implement the test, and how to assess the results of the test. Table 1-1 describes all test fields.

Table 1-1: Test Case Fields

Test Case Field	Description
Parent Requirement	Identifies the top-level requirement or the series of top-level requirements leading to the testable requirement
Testable Requirement	Guides the definition of the remainder of the test case fields, and specifies the capability to be evaluated
Description	Describes the objective of the test case
Associated Test Case(s)	In some instances, a test case may be based on the outcome of (an)other test case(s). For example, analysis-based test cases produce a result that is verifiable through various means (e.g., log entries, reports, and alerts).
Associated Cybersecurity Framework Subcategory(ies)	Lists the Cybersecurity Framework Subcategories addressed by the test case
IoT Device(s) Under Test	Text identifying which IoT device is being connected to the network in this test
MUD File(s) Used	Name of MUD file(s) used
Preconditions	Starting state of the test case. Preconditions indicate various starting-state items, such as a specific capability configuration required or specific protocol and content.

Test Case Field	Description
Procedure	Step-by-step actions required to implement the test case. A procedure may consist of a single sequence of steps or multiple sequences of steps (with delineation) to indicate variations in the test procedure.
Expected Results	Expected results for each variation in the test procedure
Actual Results	Observed results
Overall Results	Overall result of the test as pass/fail

Each test case is presented in the format described in Table 1-1.

1.5 Document Organization

The remainder of this document describes the evaluation and demonstration activities that were performed for Builds 1, 2, and 4. Each build has a section devoted to it, with that section being divided into subsections that describe the evaluation of MUD-related capabilities and the demonstration of non-MUD-related capabilities (if applicable). The MUD files used for each build are also provided.

Acronyms used in this document can be found in the Acronyms Appendix in NIST SP 1800-15B.

1.6 Typographic Conventions

The following table presents typographic conventions used in this document.

Typeface/ Symbol	Meaning	Example
Italics	file names and pathnames; references to documents that are not hyperlinks; new terms; and placeholders	For detailed definitions of terms, see the *NCCoE Glossary.*
Bold	names of menus, options, command buttons, and fields	Choose **File > Edit.**

Typeface/ Symbol	Meaning	Example
`Monospace`	command-line input, onscreen computer output, sample code examples, status codes	`Mkdir`
`Monospace Bold`	command-line user input contrasted with computer output	**`service sshd start`**
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 Build 1

Build 1 uses equipment from Cisco Systems and Forescout. The Cisco MUD Manager is used to support MUD and the Forescout Virtual Appliances, and Enterprise Manager is used to perform non-MUD-related device discovery on the network.

2.1 Evaluation of MUD-Related Capabilities

The functional evaluation that was conducted to verify that Build 1 conforms to the MUD specification was based on the Build 1-specific requirements defined in Table 2-1.

2.1.1 Requirements

Table 2-1: MUD Use Case Functional Requirements

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-1	The IoT DDoS example implementation shall include a mechanism for associating a device with a MUD file URL (e.g., by having the MUD-enabled **IoT device emit a MUD file URL via DHCP, Link Layer Discovery Protocol [LLDP], or X.509 or by using some other mechanism to enable the network to associate a device with a MUD file URL).**			IoT-1-v4, IoT-1-v6, IoT-11-v4, IoT-11-v6
CR-1.a		Upon initialization, the MUD-enabled IoT device shall broadcast a DHCP message on the network, including at most one **MUD URL, in hypertext transfer protocol secure**		IoT-1-v4, IoT-1-v6, IoT-11-v4, IoT-11-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
		(https) scheme, within the DHCP transaction.		
CR-1.a.1			The DHCP server shall be able to receive **DHCPv4 DISCOVER and REQUEST with IANA code 161** (OPTION_MUD_URL_V4) from the MUD-enabled IoT device.	IoT-1-v4, IoT-11-v4
CR-1.a.2			The DHCP server shall be able to receive **DHCPv6 Solicit and Request with IANA code 112** (OPTION_MUD_URL_V6) from the MUD-enabled IoT device.	IoT-1-v6, IoT-11-v6
CR-1.b		Upon initialization, the MUD-enabled IoT device shall **emit the MUD URL as an LLDP extension.**		IoT-1-v4, IoT-1-v6, IoT-11-v4, IoT-11-v6
CR-1.b.1			The network service shall be able to **process** the MUD URL that is received as an **LLDP extension.**	IoT-1-v4, IoT-1-v6, IoT-11-v4, IoT-11-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-2	The IoT DDoS example implementation shall include the capability for the MUD URL **to be provided to a MUD manager.**			IoT-1-v4, IoT-1-v6
CR-2.a		The DHCP server shall **assign an IP address lease** to the MUD-enabled IoT device.		IoT-1-v4, IoT-1-v6
CR-2.a.1			The MUD-enabled IoT device shall **receive the IP address.**	IoT-1-v4, IoT-1-v6
CR-2.b		**The DHCP server shall** receive the DHCP message and **extract the MUD URL, which is then passed to the MUD manager.**		IoT-1-v4, IoT-1-v6
CR-2.b.1			**The MUD manager shall receive the MUD URL.**	IoT-1-v4, IoT-1-v6
CR-3	The IoT DDoS example implementation shall include a **MUD manager that can request a MUD file and signature from a MUD file server.**			IoT-1-v4, IoT-1-v6
CR-3.a		The MUD manager shall use the GET method (RFC 7231) to		IoT-1-v4, IoT-1-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
		request MUD and signature files (per RFC 7230) from the MUD file server and can **validate the MUD file server's Transport Layer Security (TLS) certificate** by using the rules in RFC 2818.		
CR-3.a.1			**The MUD file server shall receive the https request from the MUD manager.**	IoT-1-v4, IoT-1-v6
CR-3.b		**The MUD manager** shall use the GET method (RFC 7231) to request MUD and signature files (per RFC 7230) from the MUD file server, but it **cannot validate the MUD file server's TLS certificate** by using the rules in RFC 2818.		IoT-2-v4, IoT-2-v6
CR-3.b.1			**The MUD manager shall drop the connection** to the MUD file server.	IoT-2-v4, IoT-2-v6
CR-3.b.2			**The MUD manager shall send locally defined policy to the**	IoT-2-v4, IoT-2-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
			router or switch that handles whether to allow or block traffic to and from the MUD-enabled IoT device.	
CR-4	The IoT DDoS example implementation shall include a **MUD file server that can serve a MUD file and signature to the MUD manager**.			IoT-1-v4, IoT-1-v6
CR-4.a		**The MUD file server shall serve the file and signature to the MUD manager, and the MUD manager shall check to determine whether the certificate used to sign the MUD file** (signed using distinguished encoding rules [DER]-encoded Cryptographic Message Syntax [CMS] [RFC 5652]) was valid at the time of signing, i.e., the **certificate had not expired.**		IoT-1-v4, IoT-1-v6
CR-4.b		**The MUD file server shall serve the file and signature to the**		IoT-3-v4, IoT-3-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
		MUD manager, and the MUD manager shall check to determine whether the certificate used to sign the MUD file was valid at the time of signing, i.e., the **certificate had already expired when it was used to sign the MUD file.**		
CR-4.b.1			The MUD manager shall cease to process the MUD file.	IoT-3-v4, IoT-3-v6
CR-4.b.2			The MUD manager shall send locally defined policy to the router or switch that handles whether to allow or block traffic to and from the MUD-enabled IoT device.	IoT-3-v4, IoT-3-v6
CR-5	The IoT DDoS example implementation shall include a **MUD manager** that **can translate local network configurations based on the MUD file.**			IoT-1-v4, IoT-1-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-5.a		**The MUD manager shall successfully validate the signature of the MUD file.**		IoT-1-v4, IoT-1-v6
CR-5.a.1			The MUD manager, after validation of the MUD file signature, shall **check for an existing MUD file and translate abstractions in the MUD file to router or switch configurations.**	IoT-1-v4, IoT-1-v6
CR-5.a.2			The MUD manager shall **cache** this newly received MUD file.	IoT-10-v4, IoT-10-v6
CR-5.b		The MUD manager shall attempt to validate the signature of the **MUD file,** but the **signature validation fails** (even though the certificate that had been used to create the signature had not been expired at the time of signing, i.e., the signature is invalid for a different reason).		IoT-4-v4, IoT-4-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-5.b.1			**The MUD manager shall cease processing the MUD file.**	IoT-4-v4, IoT-4-v6
CR-5.b.2			**The MUD manager shall send locally defined policy to the router or switch** that handles whether to allow or block traffic to and from the MUD-enabled IoT device.	IoT-4-v4, IoT-4-v6
CR-6	The IoT DDoS example implementation shall include a **MUD manager that can configure the MUD PEP,** i.e., the router or switch nearest the MUD-enabled IoT device that emitted the URL.			IoT-1-v4, IoT-1-v6
CR-6.a		**The MUD manager shall install a router configuration** on the router or switch nearest the MUD-enabled IoT device that emitted the URL.		IoT-1-v4, IoT-1-v6
CR-6.a.1			**The router or switch shall have been configured to enforce the route filter sent**	IoT-1-v4, IoT-1-v6

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

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

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

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
		communications initiated by the internet service.		
CR-8.d.1			**The router or switch shall receive the attempt and shall deny it** based on the filters from the MUD file.	IoT-5-v4, IoT-5-v6
CR-9	The IoT DDoS example implementation shall **allow the MUD-enabled IoT device to communicate laterally with devices that are approved** in the MUD file.			IoT-6-v4, IoT-6-v6
CR-9.a		The MUD-enabled IoT device shall **attempt to initiate lateral traffic to approved devices.**		IoT-6-v4, IoT-6-v6
CR-9.a.1			**The router or switch shall receive the attempt and shall allow it to pass** based on the filters from the MUD file.	IoT-6-v4, IoT-6-v6
CR-9.b		An approved device **shall attempt to initiate a lateral connection to the MUD-enabled IoT device.**		IoT-6-v4, IoT-6-v6

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

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
		ate a lateral connection to the MUD-enabled IoT device.		
CR-10.b.1			**The router or switch shall receive the attempt and shall deny it** based on the filters from the MUD file.	IoT-6-v4, IoT-6-v6
CR-11	If the IoT DDoS example implementation is such that its DHCP server does not act as a MUD manager and it forwards a MUD URL to a MUD manager, **the DHCP server must notify the MUD manager of any corresponding change to the DHCP state** of the MUD-enabled IoT device, and the MUD manager should **remove the implemented policy configuration in the router/switch pertaining to that MUD-enabled IoT device.**			IoT-7-v4, IoT-7-v6
CR-11.a		The MUD-enabled IoT **device shall explicitly release the IP address lease** (i.e., it sends a DHCP release message to the DHCP server).		IoT-7-v4, IoT-7-v6

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

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-12	The IoT DDoS example implementation shall include a **MUD manager that uses a cached MUD file rather than retrieve a new one if the cache-validity time period has not yet elapsed** for the MUD file indicated by the MUD URL**. The MUD manager should fetch a new MUD file if the cache-validity time period has already elapsed.**			IoT-10-v4, IoT-10-v6
CR-12.a		The MUD manager shall check if the file associated with the **MUD URL is present in its cache** and shall determine that it is.		IoT-10-v4, IoT-10-v6
CR-12.a.1			The MUD manager shall **check whether the amount of time that has elapsed since the cached file was retrieved is less than or equal to the number of hours in the cache-validity value for this MUD file.** If so, the MUD manager shall apply the contents of the cached MUD file.	IoT-10-v4, IoT-10-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-12.a.2			The MUD manager **shall check whether the amount of time that has elapsed since the cached file was retrieved is greater than the number of hours in the cache-validity value for this MUD file.** If so, the MUD manager may (but does not have to) fetch a new file by using the MUD URL received.	IoT-10-v4, IoT-10-v6
CR-13	The IoT DDoS example implementation shall ensure that for each rule in a MUD file that pertains to an external domain, the MUD PEP router/switch will get configured with **all possible instantiations of that rule,** insofar as **each instantiation contains one of the IP addresses to which the domain in that MUD file rule may be resolved when queried by the MUD PEP router/switch.**			IoT-9-v4, IoT-9-v6
CR-13.a		The MUD file for a device shall contain a rule involving a **domain that can resolve**		IoT-9-v4, IoT-9-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
		to multiple IP addresses when queried by the MUD PEP router/switch. **An Access Control List (ACL) for permitting access to each of those IP addresses will be inserted into the MUD PEP router/switch** for the device in question, and the device will be permitted to communicate with all of those IP addresses.		
CR-13.a.1			IPv4 addressing is used on the network.	IoT-9-v4
CR-13.a.2			IPv6 addressing is used on the network.	IoT-9-v6

205 ### 2.1.2 Test Cases

206 This section contains the test cases that were used to verify that Build 1 met the requirements listed in
207 Table 2-1.

208 #### *2.1.2.1 Test Case IoT-1-v4*

209 **Table 2-2: Test Case IoT-1-v4**

Test Case Field	Description
Parent Requirements	(CR-1) The IoT DDoS example implementation shall include a mechanism for associating a device with a MUD file URL (e.g., by having the MUD-enabled IoT device emit a MUD file URL via DHCP, Link Layer Discovery

Test Case Field	Description
	Protocol [LLDP], or X.509 or by using some other mechanism to enable the network to associate a device with a MUD file URL). (CR-2) The IoT DDoS example implementation shall include the capability for the MUD URL to be provided to a MUD manager. (CR-3) The IoT DDoS example implementation shall include a MUD manager that can request a MUD file and signature from a MUD file server. (CR-4) The IoT DDoS example implementation shall include a MUD file server that can serve a MUD file and signature to the MUD manager. (CR-5) The IoT DDoS example implementation shall include a MUD manager that can translate local network configurations based on the MUD file. (CR-6) The IoT DDoS example implementation shall include a MUD manager that can configure the router or switch nearest the MUD-enabled IoT device that emitted the URL.
Testable Requirements	(CR-1.a) Upon initialization, the MUD-enabled IoT device shall broadcast a DHCP message on the network, including at most one MUD URL, in https scheme, within the DHCP transaction. (CR-1.a.1) The DHCP server shall be able to receive DHCPv4 DISCOVER and REQUEST with IANA code 161 (OPTION_MUD_URL_V4) from the MUD-enabled IoT device. (NOTE: Test IoT-1-v6 does not test this requirement; instead, it tests CR-1.a.2, which pertains to DHCPv6 rather than DHCPv4.) OR (CR-1.b) Upon initialization, the MUD-enabled IoT device shall emit the MUD URL as an LLDP extension. (CR-1.b.1) The network service shall be able to process the MUD URL that is received as an LLDP extension. (CR-2.a) The DHCP server shall assign an IP address lease to the MUD-enabled IoT device. (CR-2.a.1) The MUD-enabled IoT device shall receive the IP address. (CR-2.b) The DHCP server shall receive the DHCP message and extract the MUD URL, which is then passed to the MUD manager. (CR-2.b.1) The MUD manager shall receive the MUD URL.

Test Case Field	Description
	(CR-3.a) The MUD manager shall use the "GET" method (RFC 7231) to request MUD and signature files (per RFC 7230) from the MUD file server and can validate the MUD file server's TLS certificate by using the rules in RFC 2818. (CR-3.a.1) The MUD file server shall receive the https request from the MUD manager. (CR-4.a) The MUD file server shall serve the file and signature to the MUD manager, and the MUD manager shall check to determine whether the certificate used to sign the MUD file (signed using DER-encoded CMS [RFC 5652]) was valid at the time of signing, i.e., the certificate had not expired. (CR-5.a) The MUD manager shall successfully validate the signature of the MUD file. (CR-5.a.1) The MUD manager, after validation of the MUD file signature, shall check for an existing MUD file and translate abstractions in the MUD file to router or switch configurations. (CR-6.a) The MUD manager shall install a router configuration on the router or switch nearest the MUD-enabled IoT device that emitted the URL. (CR-6.a.1) The router or switch shall have been configured to enforce the route filter sent by the MUD manager.
Description	Shows that, upon connection to the network, a MUD-enabled IoT device used in the IoT DDoS example implementation has its MUD PEP router/switch automatically configured to enforce the route filtering that is described in the device's MUD file, assuming the MUD file has a valid signature and is served from a MUD file server that has a valid TLS certificate
Associated Test Case(s)	N/A
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-1, ID.AM-2, ID.AM-3, PR.DS-5, DE.AE-1, PR.AC-4, PR.AC-5, PR.IP-1, PR.IP-3, PR.PT-3, PR.DS-2
IoT Device(s) Under Test	Raspberry Pi

Test Case Field	Description
MUD File(s) Used	*ciscopi2.json*
Preconditions	1. All devices have been configured to use IPv4. 2. This MUD file is not currently cached at the MUD manager. 3. The device's MUD file has a valid signature that was signed by a certificate that had not yet expired, and it is being hosted on a MUD file server that has a valid TLS certificate. 4. The MUD PEP router/switch does not yet have any configuration settings pertaining to the IoT device being used in the test. 5. The MUD file for the IoT device being used in the test is identical to the MUD file provided in Section 2.1.3.
Procedure	Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. Also verify that the MUD file of the IoT device to be used is not currently cached at the MUD manager. Power on the IoT device and connect it to the test network. This should set in motion the following series of steps, which should occur automatically: 1. IoT device automatically emits a MUD URL in one of the following methods: a. DHCPv4 message containing the device's MUD URL (IANA code 161) (Note that in the v6 version of this test, IPv6, DHCPv6, and IANA code 112 will be used.) b. LLDP message containing the device's MUD URL in its extension 2. Corresponding service is responsible for the following actions: a. The DHCP server receives a DHCP message containing the IoT device's MUD URL. b. The LLDP server receives an LLDP advertisement containing the IoT device's MUD URL. 3. The respective service (LLDP or DHCP) extracts the MUD URL. 4. The MUD URL is then provided to the MUD manager.

Test Case Field	Description
	5. The MUD manager automatically contacts the MUD file server that is located using the MUD URL, verifies that it has a valid TLS certificate, requests and receives the MUD file and signature from the MUD file server, validates the MUD file's signature, and translates the MUD file's contents into appropriate route filtering rules. It then installs these rules onto the MUD PEP for the IoT device in question so that this router/switch is now configured to enforce the policies specified in the MUD file. 6. The DHCP server offers an IP address lease to the newly connected IoT device. 7. The IoT device requests this IP address lease, which the DHCP server acknowledges.
Expected Results	The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to enforce the policies specified in the IoT device's MUD file. The expected configuration should resemble the following details: `Extended IP access list mud-81726-v4fr.in` `   10 permit tcp any host 192.168.4.7 eq www ack syn` `   20 permit tcp any host 192.168.10.104 eq www` `   30 permit tcp any host 192.168.10.105 eq www` `   50 permit tcp any 192.168.10.0 0.0.0.255 eq www` `   60 permit tcp any 192.168.13.0 0.0.0.255 eq www` `   70 permit tcp any 192.168.14.0 0.0.0.255 eq www` `   80 permit tcp any eq 22 any` `   81 permit udp any eq bootpc any eq bootps` `   82 permit udp any any eq domain` `   83 deny ip any any` All protocol exchanges described in steps 1–7 above are expected to occur and can be viewed via Wireshark if desired. If the router/switch does not get configured in accordance with the MUD file, each exchange of DHCP and MUD-related protocol traffic should be viewed on the network via Wireshark to determine which transactions did not proceed as expected, and the observed and absent protocol exchanges should be described here.
Actual Results	**Dynamic access-session on switch:**

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td></td><td><pre>Build1#sh access-session int g1/0/15 det
          Interface: GigabitEthernet1/0/15
          IIF-ID: 0x1B6BCEA5
          MAC Address: b827.ebeb.6c8b
          IPv6 Address: Unknown
          IPv4 Address: 192.168.13.9
          User-Name: b827ebeb6c8b
            Status: Authorized
            Domain: DATA
      Oper host mode: multi-auth
    Oper control dir: both
     Session timeout: N/A
   Common Session ID: C0A80A02000000A6A9828F06
     Acct Session ID: 0x0000003b
            Handle: 0x2200009c
      Current Policy: mud-mab-test

Server Policies:
           ACS ACL: mud-81726-v4fr.in
         Vlan Group: Vlan: 3

Method status list:
      Method   State
        mab   Authc Success</pre>
<u>access-list on switch</u>:
<pre>Build1#sh access-list mud-81726-v4fr.in
Extended IP access list mud-81726-v4fr.in
   10 permit tcp any host 192.168.4.7 eq www ack syn
   20 permit tcp any host 192.168.10.104 eq www
   30 permit tcp any host 192.168.10.105 eq www
   50 permit tcp any 192.168.10.0 0.0.0.255 eq www
   60 permit tcp any 192.168.13.0 0.0.0.255 eq www
   70 permit tcp any 192.168.14.0 0.0.0.255 eq www
   80 permit tcp any eq 22 any
   81 permit udp any eq bootpc any eq bootps
   82 permit udp any any eq domain
   83 deny ip any any</pre></td></tr>
<tr><td>Overall Results</td><td>Pass</td></tr>
</table>

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

2.1.2.2 Test Case IoT-2-v4

Table 2-3: Test Case IoT-2-v4

Test Case Field	Description
Parent Requirement	(CR-3) The IoT DDoS example implementation shall include a MUD manager that can request a MUD file and signature from a MUD file server.
Testable requirement	(CR-3.b) The MUD manager shall use the GET method (RFC 7231) to request MUD and signature files (per RFC 7230) from the MUD file server, but it cannot validate the MUD file server's TLS certificate by using the rules in RFC 2818. (CR-3.b.1) The MUD manager shall drop the connection to the MUD file server. (CR-3.b.2) The MUD manager shall send locally defined policy to the router or switch that handles whether to allow or block traffic to and from the MUD-enabled IoT device.
Description	Shows that if a MUD manager is not able to validate the TLS certificate of a MUD file server when trying to retrieve the MUD file for a specific IoT device, the MUD manager will drop the connection to the MUD file server and configure the router/switch according to locally defined policy regarding whether to allow or block traffic to the IoT device in question
Associated Test Case(s)	IoT-11-v4 (for the v6 version of this test, IoT-11-v6)
Associated Cybersecurity Framework Subcategory(ies)	PR.AC-7
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*ciscopi2.json*

Test Case Field	Description
Preconditions	1. All devices have been configured to use IPv4. 2. This MUD file is not currently cached at the MUD manager. 3. The MUD file server that is hosting the MUD file of the device under test does not have a valid TLS certificate. 4. Local policy has been defined to ensure that if the MUD file for a device is located on a server with an invalid certificate, the router/switch will be configured to deny all communication to and from the device. 5. The MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings with respect to the IoT device being used in the test.
Procedure	Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. Power on the IoT device and connect it to the test network. This should set in motion the following series of steps, which should occur automatically: 1. The IoT device automatically emits a DHCPv4 message containing the device's MUD URL (IANA code 161). (Note that in the v6 version of this test, IPv6, DHCPv6, and IANA code 112 will be used.) 2. The DHCP server receives the DHCP message containing the IoT device's MUD URL. 3. The DHCP server offers an IP address lease to the newly connected IoT device. 4. The IoT device requests this IP address lease, which the DHCP server acknowledges. 5. The DHCP server sends the MUD URL to the MUD manager. 6. The MUD manager automatically contacts the MUD file server that is located by using the MUD URL, determines that it does not have a valid TLS certificate, and drops the connection to the MUD file server.

Test Case Field	Description
	7. The MUD manager configures the router/switch that is closest to the IoT device so that it denies all communication to and from the IoT device.
Expected Results	The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to local policy for communication to/from the IoT device.
Actual Results	`***MUDC [STATUS][send_mudfs_request:2005]--> Request URI <https://mudfileserver/ciscopi2> </home/mudtester/ca.cert.pem>` `*   Trying 192.168.4.5...` `* TCP_NODELAY set` `* Connected to mudfileserver (192.168.4.5) port 443 (#0)` `* found 1 certificate in /home/mudtester/ca.cert.pem` `* found 400 certificates in /etc/ssl/certs` `* ALPN, offering http/1.1` `* SSL connection using TLS1.2 / ECDHE_RSA_AES_256_GCM_SHA384` `* server certificate verification failed. CAfile: /home/mudtester/ca.cert.pem CRLfile: none` `* stopped the pause stream!` `* Closing connection 0` `***MUDC [ERROR][fetch_file:182]--> curl_easy_perform() failed: Peer certificate cannot be authenticated with given CA certificates` `***MUDC [INFO][send_mudfs_request:2019]--> Unable to reach MUD fileserver to fetch MUD file.  Will try to append .json` `*   Trying 192.168.4.5...` `* TCP_NODELAY set` `* Connected to mudfileserver (192.168.4.5) port 443 (#0)` `* found 1 certificate in /home/mudtester/ca.cert.pem` `* found 400 certificates in /etc/ssl/certs` `* ALPN, offering http/1.1` `* SSL connection using TLS1.2 / ECDHE_RSA_AES_256_GCM_SHA384` `* server certificate verification failed. CAfile: /home/mudtester/ca.cert.pem CRLfile: none` `* stopped the pause stream!` `* Closing connection 0` `***MUDC [ERROR][fetch_file:182]--> curl_easy_perform() failed: Peer certificate cannot be authenticated with given CA certificates` `***MUDC [ERROR][send_mudfs_request:2027]--> Unable to reach MUD fileserver to fetch .json file` `***MUDC [INFO][mudc_construct_head:135]--> status_code: 204, content_len: 14, extra_headers: (null)`

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td></td><td><pre>***MUDC [INFO][mudc_construct_head:152]--> HTTP header:
HTTP/1.1 204 No Content
Content-Length: 14

***MUDC [INFO][send_error_result:176]--> error from FS

***MUDC [ERROR][send_mudfs_request:2170]--> mudfs_conn
failed</pre>
<hr>
<pre>Build1#sho access-session int g1018 det
            Interface GigabitEthernet1018
               IIF-ID 0x181835C2
          MAC Address b827.eba7.0533
         IPv6 Address Unknown
         IPv4 Address 192.168.10.106
            User-Name b827eba70533
               Status Authorized
               Domain DATA
       Oper host mode multi-auth
     Oper control dir both
      Session timeout NA
    Common Session ID C0A80A02000000CCBDB267F8
      Acct Session ID 0x00000046
               Handle 0x100000c2
       Current Policy mud-mab-test

Server Policies

Method status list
       Method   State
          mab   Authc Success</pre></td></tr>
<tr><td>Overall Results</td><td>Pass</td></tr>
</table>

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

2.1.2.3 Test Case IoT-3-v4

Table 2-4: Test Case IoT-3-v4

Test Case Field	Description
Parent Requirement	(CR-4) The IoT DDoS example implementation shall include a MUD file server that can serve a MUD file and signature to the MUD manager.
Testable Requirement	(CR-4.b) The MUD file server shall serve the file and signature to the MUD manager, and the MUD manager shall check to determine whether the certificate used to sign the MUD file was valid at the time of signing, i.e., the certificate had already expired when it was used to sign the MUD file. (CR-4.b.1) The MUD manager shall cease to process the MUD file. (CR-4.b.2) The MUD manager shall send locally defined policy to the router or switch that handles whether to allow or block traffic to and from the MUD-enabled IoT device.
Description	Shows that if a MUD file server serves a MUD file with a signature that was created with an expired certificate, the MUD manager will cease processing the MUD file
Associated Test Case(s)	IoT-11-v4 (for the v6 version of this test, IoT-11-v6)
Associated Cybersecurity Framework Subcategory(ies)	PR.DS-6
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*expiredcerttest.json*
Preconditions	1. All devices have been configured to use IPv4. 2. This MUD file is not currently cached at the MUD manager. 3. The IoT device's MUD file is being hosted on a MUD file server that has a valid TLS certificate, but the MUD file signature was signed by a certificate that had already expired at the time of signature.

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

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td></td><td>8. The MUD manager configures the router/switch that is closest to the IoT device so that it denies all communication to and from the IoT device.</td></tr>
<tr><td>Expected Results</td><td>The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to deny all communication to and from the IoT device. The expected configuration should resemble the details below.
Expecting a show access session without a MUD file as seen below:<pre>
Build1#show access-session int g1018 det
            Interface GigabitEthernet1018
               IIF-ID 0x181835C2
          MAC Address b827.eba7.0533
         IPv6 Address Unknown
         IPv4 Address 192.168.10.106
            User-Name b827eba70533
               Status Authorized
               Domain DATA
       Oper host mode multi-auth
     Oper control dir both
      Session timeout NA
    Common Session ID C0A80A02000000CCBDB267F8
      Acct Session ID 0x00000046
               Handle 0x100000c2
       Current Policy mud-mab-test

Server Policies

Method status list
       Method   State
          mab   Authc Success
</pre></td></tr>
</table>

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td>Actual Results</td><td><pre>
***MUDC [INFO][verify_mud_content:1594]--> BIO_reset <1>

***MUDC [ERROR][verify_mud_content:1604]--> Verification
Failure

139713269933824:error:2E099064:CMS routines:cms_sign-
erinfo_verify_cert:certificate verify er-
ror:../crypto/cms/cms_smime.c:253:Verify error:certificate
has expired
***MUDC [INFO][send_mudfs_request:2092]--> Verification
failed. Manufacturer Index <0>

***MUDC [INFO][mudc_construct_head:135]--> status_code: 401,
content_len: 19, extra_headers: (null)
***MUDC [INFO][mudc_construct_head:152]--> HTTP header:
HTTP/1.1 401 Unauthorized
Content-Length: 19

***MUDC [INFO][send_error_result:176]--> Verification failed
***MUDC [ERROR][send_mudfs_request:2170]--> mudfs_conn
failed
</pre>
<hr>
<pre>
Build1#sho access-session int g1018 det
            Interface GigabitEthernet1018
               IIF-ID 0x181835C2
          MAC Address b827.eba7.0533
         IPv6 Address Unknown
         IPv4 Address 192.168.10.106
            User-Name b827eba70533
               Status Authorized
               Domain DATA
       Oper host mode multi-auth
     Oper control dir both
      Session timeout NA
    Common Session ID C0A80A02000000CCBDB267F8
      Acct Session ID 0x00000046
               Handle 0x100000c2
       Current Policy mud-mab-test

Server Policies

Method status list
       Method   State
          mab   Authc Success
</pre></td></tr>
<tr><td>Overall Results</td><td>Pass</td></tr>
</table>

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

221 2.1.2.4 Test Case IoT-4-v4

222 **Table 2-5: Test Case IoT-4-v4**

Test Case Field	Description
Parent Requirement	(CR-5) The IoT DDoS example implementation shall include a MUD manager that can translate local network configurations based on the MUD file.
Testable Requirement	(CR-5.b) The MUD manager shall attempt to validate the signature of the MUD file, but the signature validation fails (even though the certificate that had been used to create the signature had not been expired at the time of signing, i.e., the signature is invalid for a different reason). (CR-5.b.1) The MUD manager shall cease processing the MUD file. (CR-5.b.2) The MUD manager shall send locally defined policy to the router or switch that handles whether to allow or block traffic to and from the MUD-enabled IoT device.
Description	Shows that if the MUD manager determines that the signature on the MUD file it receives from the MUD file server is invalid, it will cease processing the MUD file and configure the router/switch according to locally defined policy regarding whether to allow or block traffic to the IoT device in question
Associated Test Case(s)	IoT-11-v4 (for the v6 version of this test, IoT-11-v6)
Associated Cybersecurity Framework Subcategory(ies)	PR.DS-6
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*ciscop2.json*

Test Case Field	Description
Preconditions	1. All devices have been configured to use IPv4. 2. This MUD file is not currently cached at the MUD manager. 3. The MUD file that is served from the MUD file server to the MUD manager has a signature that is invalid, even though it was signed by a certificate that had not expired at the time of signing. 4. Local policy has been defined to ensure that if the MUD file for a device has an invalid signature, the device's MUD PEP router/switch will be configured to deny all communication to and from the device. 5. The MUD PEP router/switch does not yet have any configuration settings with respect to the IoT device being used in the test.
Procedure	Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. Power on the IoT device and connect it to the test network. This should set in motion the following series of steps, which should occur automatically: 1. The IoT device automatically emits a DHCPv4 message containing the device's MUD URL (IANA code 161). (Note that in the v6 version of this test, IPv6, DHCPv6, and IANA code 112 will be used.) 2. The DHCP server receives the DHCP message containing the IoT device's MUD URL. 3. The DHCP server offers an IP address lease to the newly connected IoT device. 4. The IoT device requests this IP address lease, which the DHCP server acknowledges. 5. The DHCP server sends the MUD URL to the MUD manager. 6. The MUD manager automatically contacts the MUD file server that is located by using the MUD URL, verifies that it has a valid TLS certificate, and requests the MUD file and signature from the MUD file server. 7. The MUD file server sends the MUD file, and the MUD manager detects that the MUD file's signature is invalid.

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

8. The MUD manager configures the router/switch that is closest to the IoT device so that it denies all communication to and from the IoT device.

</td></tr>
<tr><td>Expected Results</td><td>

The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to deny all communication to/from the IoT device. The expected configuration should resemble the following details.

Expecting a show access session without a MUD file as seen below:

```
Build1#sho access-session int g1018 det
            Interface  GigabitEthernet1018
               IIF-ID  0x181835C2
          MAC Address  b827.eba7.0533
         IPv6 Address  Unknown
         IPv4 Address  192.168.10.106
            User-Name  b827eba70533
               Status  Authorized
               Domain  DATA
       Oper host mode  multi-auth
     Oper control dir  both
      Session timeout  NA
    Common Session ID  C0A80A02000000CCBDB267F8
      Acct Session ID  0x00000046
               Handle  0x100000c2
       Current Policy  mud-mab-test


Server Policies


Method status list
       Method           State
          mab          Authc Success
```

</td></tr>
<tr><td>Actual Results</td><td>

```
> GET /ciscopi2.json HTTP/1.1
Host: mudfileserver
Accept: */*
```

[Omitted for brevity]

```
***MUDC [STATUS][send_mudfs_request:2060]-->
Request signature URI <https://mudfileserver/ciscopi2.p7s>
</home/mudtester/mud-intermediate.pem>
```

</td></tr>
</table>

Test Case Field	Description
	`*   Trying 192.168.4.5...` `* TCP_NODELAY set` `* Connected to mudfileserver (192.168.4.5) port 443 (#0)` `* found 1 certificate in /home/mudtester/mud-intermedi-ate.pem` `* found 400 certificates in /etc/ssl/certs` `* ALPN, offering http/1.1` `* SSL connection using TLS1.2 / ECDHE_RSA_AES_256_GCM_SHA384` `*       server certificate verification OK` `*       server certificate status verification SKIPPED` `*       common name: mudfileserver (matched)` `*       server certificate expiration date OK` `*       server certificate activation date OK` `*       certificate public key: RSA` `*       certificate version: #3` `*       subject: C=US,ST=Maryland,L=Rockville,O=National Cy-bersecurity Center of Excellence - NIST,CN=mudfileserver` `*       start date: Fri, 05 Oct 2018 00:00:00 GMT` `*       expire date: Wed, 13 Oct 2021 12:00:00 GMT` `*       issuer: C=US,O=DigiCert Inc,CN=DigiCert Test SHA2 Intermediate CA-1` `*       compression: NULL` `* ALPN, server did not agree to a protocol` `> GET /ciscopi2.p7s HTTP/1.1` `Host: mudfileserver` `Accept: */*` **[Omitted for brevity]** `***MUDC [INFO][send_mudfs_request:2080]--> MUD signature file successfully retrieved` `***MUDC [DEBUG][verify_mud_content:1543]--> MUD signature file (length 4680)` `[shortened logs]` `***MUDC [INFO][verify_mud_content:1594]--> BIO_reset <1>` `***MUDC [ERROR][verify_mud_content:1604]-->` **`Verification Failure`** `140561528563456:error:2E09A09E:CMS routines:CMS_Sign-erInfo_verify_content:verification fail-ure:../crypto/cms/cms_sd.c:819:` `140561528563456:error:2E09D06D:CMS routines:CMS_verify:con-tent verify error:../crypto/cms/cms_smime.c:393:`

Test Case Field	Description
	`***MUDC [INFO][send_mudfs_request:2092]-->` **Verification failed. Manufacturer Index <0>** `***MUDC [INFO][mudc_construct_head:135]--> status_code: 401, content_len: 19, extra_headers: (null)` `***MUDC [INFO][mudc_construct_head:152]--> HTTP header: HTTP/1.1 401 Unauthorized` `Content-Length: 19` `***MUDC [INFO][send_error_result:176]-->` **Verification failed** `***MUDC [ERROR][send_mudfs_request:2170]-->` **mudfs_conn failed**
	Switch access-session: `Build1#sho access-session int g1/0/18 det` `Interface:  GigabitEthernet1/0/18` `IIF-ID:  0x11C404C6` `MAC Address:  b827.eba7.0533` `IPv6 Address:  Unknown` `IPv4 Address:  192.168.10.106` `User-Name:  b827eba70533` `Status:  Authorized` `Domain:  DATA` `Oper host mode:  multi-auth` `Oper control dir:  both` `Session timeout:  N/A` `Common Session ID:  C0A80A02000000CDBDB68A30` `Acct Session ID:  0x00000047` `Handle:  0x690000c3` `Current Policy:  mud-mab-test` `Server Policies:` `Method status list:` `Method          State` `mab           Authc Success`
Overall Results	Pass

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

2.1.2.5 Test Case IoT-5-v4

Table 2-6: Test Case IoT-5-v4

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

Test Case Field	Description
	internet service but that is not approved to receive communications initiated by the internet service. (CR-8.d.1) The router or switch shall receive the attempt and shall deny it based on the filters from the MUD file.
Description	Shows that, upon connection to the network, a MUD-enabled IoT device used in the IoT DDoS example implementation has its MUD PEP router/switch automatically configured to enforce the route filtering that is described in the device's MUD file with respect to communication with internet services. Further shows that the policies that are configured on the MUD PEP router/switch with respect to communication with internet services will be enforced as expected, with communications that are configured as denied being blocked, and communications that are configured as permitted being allowed.
Associated Test Case(s)	IoT-1-v4 (for the v6 version of this test, IoT-1-v6)
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-3, PR.DS-5, PR.IP-1, PR.PT-3
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*ciscopi2.json*
Preconditions	Test IoT-1-v4 (or IoT-1-v6) has run successfully, meaning that the MUD PEP router/switch has been configured to enforce the following policies for the IoT device in question (as defined in the MUD file in Section 2.1.3): a) Explicitly permit *https://yes-permit-from.com* to initiate communication with the IoT device. b) Explicitly permit the IoT device to initiate communication with *https://yes-permit-to.com*. c) Implicitly deny all other communications with the internet, including denying

Test Case Field	Description
	i) the IoT device to initiate communication with *https://yes-permit-from.com* ii) *https://yes-permit-to.com* to initiate communication with the IoT device iii) communication between the IoT device and all other internet locations, such as *https://unnamed-to.com* (by not mentioning this or any other URLs in the MUD file)
Procedure	Note: Procedure steps with strikethrough are not tested in this phase because ingress Dynamic Access Control Lists (DACLs) are not supported in this implementation. 1. As stipulated in the preconditions, right before this test, test IoT-1-v4 (or IoT-1-v6) must have been run successfully. 2. Initiate communications from the IoT device to *https://yes-permit-to.com* and verify that this traffic is received at *https://yes-permit-to.com*. (egress) 3. Initiate communications to the IoT device from *https://yes-permit-to.com* and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at the IoT device. (ingress) ~~4. Initiate communications to the IoT device from *https://yes-permit-from.com* and verify that this traffic is received at the IoT device. (ingress)~~ ~~5. Initiate communications from the IoT device to *https://yes-permit-from.com* and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at *https://yes-permit-from.com*. (ingress)~~ 6. Initiate communications from the IoT device to *https://unnamed.com* and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at *https://unnamed.com*. (egress) 7. Initiate communications to the IoT device from *https://unnamed.com* and verify that this traffic is received at the MUD PEP,

Test Case Field	Description
	but it is not forwarded by the MUD PEP, nor is it received at the IoT device. (ingress)
Expected Results	Each of the results that is listed as needing to be verified in procedure steps above occurs as expected.
Actual Results	Procedure 2: Connection to update server successfully initiated by IoT device: `pi@raspberrypi:~ $ wget http://www.updateserver.com/` `--2018-12-13 21:28:00--  http://www.updateserver.com/` `Resolving www.updateserver.com (www.updateserver.com)... 192.168.4.7` `Connecting to www.updateserver.com (www.up-dateserver.com)|192.168.4.7|:80... connected.` `HTTP request sent, awaiting response... 200 OK` `Length: 10918 (11K) [text/html]` `Saving to: ‘index.html.2’` `index.html.2      100%[===================>]  10.66K  --.-KB/s    in 0s` `2018-12-13 21:28:00 (30.6 MB/s) - ‘index.html.2’ saved [10918/10918]`
	Procedure 3: Update server failed to connect to IoT device: `iot@update-server:~$ wget http://192.168.13.9` `--2018-12-13 21:49:36--  http://192.168.13.9/` `Connecting to 192.168.13.9:80... failed: Connection timed out.` `Retrying.`
	Procedure 6: IoT device failed to connect to unapproved server: `pi@raspberrypi:~ $ wget http://192.168.4.105` `--2018-12-14 16:42:36--  http://192.168.4.105/` `Connecting to 192.168.4.105:80... failed: Connection timed out.` `Retrying.`

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

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

2.1.2.6 Test Case IoT-6-v4

Table 2-7: Test Case IoT-6-v4

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

Test Case Field	Description
	(CR-10.a) The MUD-enabled IoT device shall attempt to initiate lateral traffic to unapproved (implicitly denied) devices. (CR-10.a.1) The router or switch shall receive the attempt and shall deny it based on the filters from the MUD file. (CR-10.b) An unapproved (implicitly denied) device shall attempt to initiate a lateral connection to the MUD-enabled IoT device. (CR-10.b.1) The router or switch shall receive the attempt and shall deny it based on the filters from the MUD file.
Description	Shows that, upon connection to the network, a MUD-enabled IoT device used in the IoT DDoS example implementation has its MUD PEP router/switch automatically configured to enforce the route filtering that is described in the device's MUD file with respect to communication with lateral devices. Further shows that the policies that are configured on the MUD PEP router/switch with respect to communication with lateral devices will be enforced as expected, with communications that are configured as denied being blocked, and communications that are configured as permitted being allowed.
Associated Test Case(s)	IoT-1-v4 (for the v6 version of this test, IoT-1-v6)
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-3, PR.DS-5, PR.AC-5, PR.IP-1, PR.PT-3, PR.IP-3, PR.DS-3
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*ciscopi2.json*
Preconditions	Test IoT-1-v4 (or IoT-1-v6) has run successfully, meaning that the MUD PEP router/switch has been configured to enforce the following policies for the IoT device in question with respect to local communications (as defined in the MUD files in Section 2.1.3): a) Local-network class—Explicitly permit **local communication to and from the IoT device and any local hosts** (including the spe-

Test Case Field	Description
	cific local hosts *anyhost-to* and *anyhost-from*) **for specific services,** as specified in the MUD file by source port: any; destination port: 80; and protocol: TCP, and which party initiates the connection. b) Manufacturer class—Explicitly permit **local communication to and from the IoT device and other classes of IoT devices, as identified by their MUD URL (*www.devicetype.com*), and further constrained** by source port: any; destination port: 80; and protocol: TCP. c) Same-manufacturer class—Explicitly permit **local communication to and from IoT devices of the same manufacturer as the IoT device in question (the domain in the MUD URLs [mudfileserver] of the other IoT devices is the same as the domain in the MUD URL [mudfileserver] of the IoT device in question),** and further constrained by source port: any; destination port: 80; and protocol: TCP. d) Implicitly deny all other local communication that is not explicitly permitted in the MUD file, including denying i) ***anyhost-to* to initiate communications** with the IoT device ii) the **IoT device to initiate communications** with *anyhost-to* by **using a source port, destination port, or protocol (TCP or UDP) that is not explicitly permitted** iii) the **IoT device to initiate communications with *anyhost-from*** iv) ***anyhost-from* to initiate communications** with the IoT device by **using a source port, destination port, or protocol (TCP or UDP) that is not explicitly permitted** v) communications between the IoT device and all lateral hosts (including *unnamed-host*) whose **MUD URLs are not explicitly mentioned** as being permissible in the MUD file vi) communications between the IoT device and all lateral hosts whose **MUD URLS are explicitly mentioned** as being permissible, **but using a source port, destination port, or protocol (TCP or UDP) that is not explicitly permitted**

Test Case Field	Description
	vii) communications between the IoT device and all lateral hosts that are **not from the same manufacturer** as the IoT device in question viii) communications between the IoT device and a lateral host that **is from the same manufacturer, but using a source port, destination port, or protocol (TCP or UDP) that is not explicitly permitted**
Procedure	Note: Procedure steps with strikethrough are not tested in this phase because ingress DACLs are not supported in this implementation. 1. As stipulated in the preconditions, right before this test, test IoT-1-v4 (or IoT-1-v6) must have been run successfully. ~~2. Local-network (ingress): Initiate communications to the IoT device from *anyhost-from* **for specific permitted service,** and verify that this traffic is received at the IoT device.~~ 3. Local-network (egress): **Initiate communications from the IoT device to *anyhost-from*** for specific permitted service, and verify that this traffic is received at the MUD PEP, but it **is not forwarded** by the MUD PEP, nor is it received at *anyhost-from*. 4. Local-network, controller, my-controller, manufacturer class (egress): Initiate communications from the IoT device to *anyhost-to* **for specific permitted service,** and verify that this traffic **is received** at *anyhost-to.* ~~5. Local-network, controller, my-controller, manufacturer class (ingress): **Initiate communications to the IoT device from *anyhost-to*** for specific permitted service, and verify that this traffic is received at the MUD PEP, but it **is not forwarded** by the MUD PEP, nor is it received at the IoT device.~~ 6. No associated class (egress): Initiate communications from the IoT device to *unnamed-host* (where *unnamed-host* is a host that is not from the same manufacturer as the IoT device in question and whose **MUD URL is not explicitly mentioned in the MUD file as being permitted),** and verify that this traffic is received at the MUD PEP, but it **is not forwarded** by the MUD PEP, nor is it received at *unnamed-host*.

Test Case Field	Description
	~~7. No associated class (ingress): Initiate communications to the IoT device from *unnamed-host* (where *unnamed-host* is a host that is not from the same manufacturer as the IoT device in question and whose **MUD URL is not explicitly mentioned in the MUD file as being permitted),** and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at the IoT device.~~ 8. Same-manufacturer class (egress): Initiate communications from the IoT device to *same-manufacturer-host* (where *same-manufacturer-host* is **a host that is from the same manufacturer as the IoT device** in question) and verify that this traffic **is received** at *same-manufacturer-host*. 9. Same-manufacturer class (egress): Initiate communications from the IoT device to *same-manufacturer-host* (where *same-manufacturer-host* is **a host that is from the same manufacturer as the IoT device** in question) **but using a port or protocol that is not specified,** and verify that this traffic is received at the MUD PEP, but it **is not forwarded** by the MUD PEP, nor is it received at *same-manufacturer-host*.
Expected Results	Each of the results that is listed as needing to be verified in the procedure steps above occurs as expected.
Actual Results	3. Local_network (egress)—blocked: `pi@raspberrypi:~ $ wget https://192.168.10.106/` `--2019-01-31 19:59:23--  https://192.168.10.106/` `Connecting to 192.168.10.106:443... failed: Connection timed out.` `Retrying.` 4. Local-network, controller, my-controller, manufacturer class (egress)—allowed: Local_Network: `pi@raspberrypi:~ $ wget http://192.168.10.175` `--2018-12-14 15:11:50--  http://192.168.10.175/` `Connecting to 192.168.10.175:80... connected.` `HTTP request sent, awaiting response... 200 OK` `Length: 10701 (10K) [text/html]`

Test Case Field	Description
	`Saving to: 'index.html.4'` `index.html.4 100%[===================>] 10.45K --.-KB/s in 0s` `2018-12-14 15:11:50 (41.4 MB/s) - 'index.html.4' saved [10701/10701]` Controller: `pi@raspberrypi:~ $` **`wget`** ***`http://192.168.10.105/`*** `--2019-01-31 21:03:45-- ` *`http://192.168.10.105/`* `Connecting to 192.168.10.105:80... connected.` `HTTP request sent, awaiting response... 200 OK` `Length: 277` `Saving to: 'index.html.10'` `in-dex.html.10 100%[===================>] 277 --.-KB/s in 0s` `2019-01-31 21:03:45 (18.8 MB/s) - 'index.html.10' saved [277/277]` My-controller: `pi@raspberrypi:~ $` **`wget`** ***`http://192.168.10.104/`*** `--2019-01-31 21:06:39-- ` *`http://192.168.10.104/`* `Connecting to 192.168.10.104:80... connected.` `HTTP request sent, awaiting response... 200 OK` `Length: 10701 (10K) [text/html]` `Saving to: 'index.html.11'` `in-dex.html.11 100%[===================>] 10.45K --.-KB/s in 0s` `2019-01-31 21:06:39 (32.5 MB/s) - 'index.html.11' saved [10701/10701]` Manufacturer: `pi@raspberrypi:~ $` **`wget`** ***`http://192.168.14.2/`*** `--2019-01-31 21:13:47-- ` *`http://192.168.14.2/`* `Connecting to 192.168.14.2:80... connected.`

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

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

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

2.1.2.7 Test Case IoT-7-v4

Table 2-8: Test Case IoT-7-v4

Test Case Field	Description
Parent Requirement	(CR-11) If the IoT DDoS example implementation is such that its DHCP server does not act as a MUD manager and it forwards a MUD URL to a MUD manager, the DHCP server must notify the MUD manager of any corresponding change to the DHCP state of the MUD-enabled IoT device, and the MUD manager should remove the implemented policy configuration in the router/switch pertaining to that MUD-enabled IoT device.
Testable Requirement	(CR-11.a) The MUD-enabled IoT device shall explicitly release the IP address lease (i.e., it sends a DHCP release message to the DHCP server). (CR-11.a.1) The DHCP server shall notify the MUD manager that the device's IP address lease has been released. (CR-11.a.2) The MUD manager should remove all policies associated with the disconnected IoT device that had been configured on the MUD PEP router/switch.
Description	Shows that when a MUD-enabled IoT device explicitly releases its IP address lease, the MUD-related configuration for that IoT device will be removed from its MUD PEP router/switch
Associated Test Case(s)	IoT-1-v4 (or IoT-1-v6 when IPv6 addressing is used)
Associated Cybersecurity Framework Subcategory(ies)	PR.IP-3, PR.DS-3

Test Case Field	Description
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*ciscopi2.json*
Preconditions	Test IoT-1-v4 (or IoT-1-v6) has run successfully, meaning that the MUD PEP router/switch has been configured to enforce the policies defined in the MUD file in section 2.1.3 for the IoT device in question.
Procedure	1. As stipulated in the preconditions, right before this test, test IoT-1-v4 (or IoT-1-v6) must have been run successfully. Verify that the MUD PEP router/switch for the IoT device has been configured to enforce the policies listed in the preconditions section above for the IoT device in question. 2. Cause a DHCP release of the IoT device in question. 3. Verify that all the configuration rules listed above have been re-moved from the MUD PEP router/switch for the IoT device in ques-tion.
Expected Results	All of the configuration rules listed above have been removed from the MUD PEP router/switch for the IoT device in question.
Actual Results	Procedure 1: `Build1#`**`sh access-session int g1/0/15 det`** `Interface:  GigabitEthernet1/0/15` `IIF-ID:  0x1B6BCEA5` `MAC Address:  b827.ebeb.6c8b` `IPv6 Address:  Unknown` `IPv4 Address:  192.168.13.17` `User-Name:  b827ebeb6c8b` `Status:  Authorized` `Domain:  DATA` `Oper host mode:  multi-auth` `Oper control dir:  both` `Session timeout:  N/A` `Common Session ID:  C0A80A0200000A6A9828F06` `Acct Session ID:  0x0000003b` `Handle:  0x2200009c` `Current Policy:  mud-mab-test`

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

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

2.1.2.8 Test Case IoT-8-v4

Table 2-9: Test Case IoT-8-v4

Test Case Field	Description
Parent Requirement	(CR-11) If the IoT DDoS example implementation is such that its DHCP server does not act as a MUD manager and it forwards a MUD URL to a MUD manager, the DHCP server must notify the MUD manager of any corresponding change to the DHCP state of the MUD-enabled IoT device, and the MUD manager should remove the implemented policy configuration in the router/switch pertaining to that MUD-enabled IoT device.
Testable Requirement	(CR-11.b) The MUD-enabled IoT device's IP address lease shall expire. (CR-11.b.1) The DHCP server shall notify the MUD manager that the device's IP address lease has expired. (CR-11.b.2) The MUD manager should remove all policies associated with the affected IoT device that had been configured on the MUD PEP router/switch.
Description	Shows that when a MUD-enabled IoT device's IP address lease expires, the MUD-related configuration for that IoT device will be removed from its MUD PEP router/switch
Associated Test Case(s)	IoT-1-v4 (or IoT-1-v6 when IPv6 addressing is used)
Associated Cybersecurity Framework Subcategory(ies)	PR.IP-3, PR.DS-3
IoT Device(s) Under Test	TBD (Not testable in Build 1)
MUD File(s) Used	TBD (Not testable in Build 1)

Test Case Field	Description
Preconditions	Test IoT-1-v4 (or IoT-1-v6) has run successfully, meaning that the MUD PEP router/switch has been configured to enforce the policies defined in the MUD file in Section 2.1.3 for the IoT device in question.
Procedure	1. Configure the DHCP server to have a DHCP lease time of 10 minutes. 2. Run test IoT-1-v4 (or IoT-1-v6). 3. Verify that the MUD PEP router/switch for the IoT device has been configured to enforce the policies listed above for the IoT device in question. 4. Disconnect the IoT device in question from the network. 5. After 10 minutes have elapsed, verify that all of the configuration rules listed above have been removed from the MUD PEP router/switch for the IoT device in question.
Expected Results	Once 10 minutes have elapsed after disconnecting the IoT device from the network, all of the configuration rules listed above have been removed from the MUD PEP router/switch for the IoT device in question.
Actual Results	TBD (Not testable in Build 1)
Overall Results	TBD (Not testable in Build 1)

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

241 *2.1.2.9 Test Case IoT-9-v4*

242 **Table 2-10: Test Case IoT-9-v4**

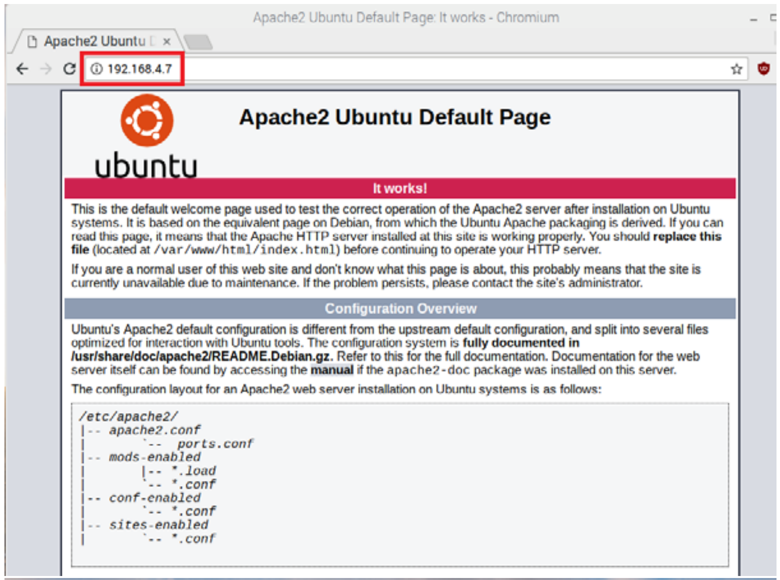
Test Case Field	Description
Parent Requirements	(CR-13) The IoT DDoS example implementation shall ensure that for each rule in a MUD file that pertains to an external domain, the MUD PEP router/switch will get configured with all possible instantiations of that rule, insofar as each instantiation contains one of the IP addresses

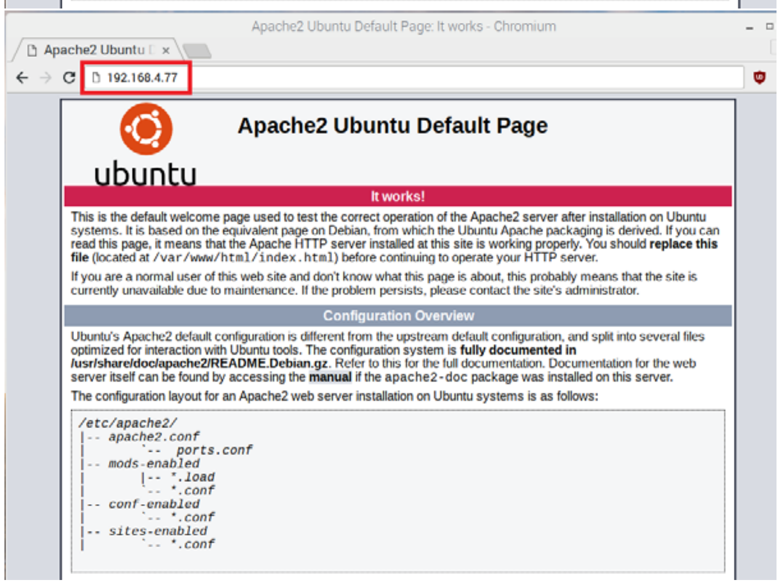
Test Case Field	Description
	to which the domain in that MUD file rule may be resolved when queried by the MUD PEP router/switch.
Testable Requirements	(CR-13.a) The MUD file for a device shall contain a rule involving an external domain that can resolve to multiple IP addresses when queried by the MUD PEP router/switch. An ACL for permitting access to each of those IP addresses will be inserted into the MUD PEP router/switch for the device in question, and the device will be permitted to communicate with all of those IP addresses.
Description	Shows that if a domain in a MUD file rule resolves to multiple IP addresses when the address resolution is queried by the network gateway, then 1. ACLs instantiating that MUD file rule corresponding to each of these IP addresses will be configured in the gateway for the IoT device associated with the MUD file, and 2. the IoT device associated with the MUD file will be permitted to communicate with all of the IP addresses to which that domain resolves
Associated Test Case(s)	N/A
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-1, ID.AM-2, ID.AM-3, PR.DS-5, DE.AE-1, PR.AC-4, PR.AC-5, PR.IP-1, PR.IP-3, PR.DS-2
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*dnstest.json*
Preconditions	1. The MUD PEP router/switch does not yet have any configuration settings pertaining to the IoT device being used in the test. 2. The MUD file for the IoT device being used in the test is identical to the MUD file provided in Section 2.1.3. (Therefore, the MUD file used in the test permits the device to send data to *www.updateserver.com*.)

Test Case Field	Description
	3. The tester has access to a domain name system (DNS) server that will be used by the MUD PEP router/switch and can configure it such that it will resolve the domain *www.updateserver.com* to any of these addresses when queried by the MUD PEP router/switch: x1.x1.x1.x1, y1.y1.y1.y1, and z1.z1.z1.z1. 4. There is an update server running at each of these three IP addresses.
Procedure	1. Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. 2. Run test IoT-1-v4 (or IoT-1-v6). The result should be that the MUD PEP router/switch has been configured to explicitly permit the IoT device to initiate communication with *www.updateserver.com*. 3. Verify that the MUD PEP router/switch has been configured with ACLs that permit the IoT device to send data to IP addresses x1.x1.x1.x1, y1.y1.y1.y1, and z1.z1.z1.z1. 4. Have the device in question attempt to connect to x1.x1.x1.x1, y1.y1.y1.y1, and z1.z1.z1.z1.
Expected Results	The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to permit the IoT device to send data to IP addresses x1.x1.x1.x1, y1.y1.y1.y1, and z1.z1.z1.z1. The IoT device is permitted to send data to each of the update servers at these addresses.
Actual Results	**Procedures 1–2:** **Completed; excluded for brevity** **Procedure 3:** **MUD MANAGER:** `***MUDC [INFO][fetch_uri_from_macaddr:2166]--> ============= Returning URI:`*`https://mudfileserver/dnstest.json`* `***MUDC [INFO][handle_get_aclname:3149]--> Found URI `*`https://mudfileserver/dnstest.json`*` for MAC address b827ebcf7b81`

Test Case Field	Description
	`***MUDC [INFO][validate_muduri:3009]--> uri:` *https://mudfileserver/dnstest.jsonhttps://mudfileserver/dnstest.json* `***MUDC [INFO][validate_muduri:3035]--> ip: mudfileserver, filename: dnstest.json` `***MUDC [INFO][handle_get_aclname:3194]--> Got URL from message` <*https://mudfileserver/dnstest.json*> `***MUDC [INFO][query_policies_by_uri:1873]--> found the record <{ "_id" : { "$oid" : "5d51d0eb0ff2eb76576ee38b" }, "DACL_Name" : "ACS:CiscoSecure-Defined-ACL=mud-77797-v4fr.in", "DACL" : "[\"ip:inacl#10=permit tcp any host` **192.168.4.7** `range 80 80 syn ack\", \"ip:inacl#20=permit tcp any host` **192.168.4.78** `range 80 80 syn ack\", \"ip:inacl#30=permit tcp any host` **192.168.4.77** `range 80 80 syn ack\", \"ip:inacl#40=permit tcp any eq 22 any\", \"ip:inacl#41=permit udp any eq 68 any eq 67\", \"ip:inacl#42=permit udp any any eq 53\", \"ip:inacl#43=deny ip any any\"]", "URI" : "`*https://mudfileserver/dnstest.json*`" }>` `***MUDC [INFO][query_policies_by_uri:1915]--> Response <{` `"Cisco-AVPair": ["ACS:CiscoSecure-Defined-ACL=mud-77797-v4fr.in"]` `}>` `***MUDC [INFO][mudc_construct_head:63]--> status_code: 200, content_len: 70, extra_headers: Content-Type: application/aclname` `***MUDC [INFO][mudc_construct_head:80]--> HTTP header: HTTP/1.1 200 OK` `Content-Type: application/aclname` `Content-Length: 70` `***MUDC [INFO][query_policies_by_uri:1918]--> {` `"Cisco-AVPair": ["ACS:CiscoSecure-Defined-ACL=mud-77797-v4fr.in"]` `}` `***MUDC [INFO][handle_get_aclname:3204]--> Got ACLs from the MUD URL` **Switch/PEP:**

Test Case Field	Description
	`Build1#`**`show access-lists`** `Extended IP access list mud-77797-v4fr.in` `   10 permit tcp any host `**`192.168.4.7`**` eq www ack syn` `   20 permit tcp any host `**`192.168.4.78`**` eq www ack syn` `   30 permit tcp any host `**`192.168.4.77`**` eq www ack syn` `   40 permit tcp any eq 22 any` `   41 permit udp any eq bootpc any eq bootps` `   42 permit udp any any eq domain` `   43 deny ip any any` **Procedure 4:**

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




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

243 Test Case IoT-9-v6 is identical to test case IoT-9-v4 except that IoT-9-v6 uses IPv6 addresses rather than
244 IPv4 addresses.

2.1.2.10 Test Case IoT-10-v4

Table 2-11: Test Case IoT-10-v4

Test Case Field	Description
Parent Requirements	(CR-12) The IoT DDoS example implementation shall include a MUD manager that uses a cached MUD file rather than retrieve a new one if the cache-validity time period has not yet elapsed for the MUD file indicated by the MUD URL. The MUD manager should fetch a new MUD file if the cache-validity time period has already elapsed.
Testable Requirements	(CR-12.a) The MUD manager shall check if the file associated with the MUD URL is present in its cache and shall determine that it is. (CR-12.a.1) The MUD manager shall check whether the amount of time that has elapsed since the cached file was retrieved is less than or equal to the number of hours in the cache-validity value for this MUD file. If so, the MUD manager shall apply the contents of the cached MUD file. (CR-12.a.2) The MUD manager shall check whether the amount of time that has elapsed since the cached file was retrieved is greater than the number of hours in the cache-validity value for this MUD file. If so, the MUD manager may (but does not have to) fetch a new file by using the MUD URL received.
Description	Shows that, upon connection to the network, a MUD-enabled IoT device used in the IoT DDoS example implementation has its MUD PEP router/switch automatically configured to enforce the route filtering that is described in the cached MUD file for that device's MUD URL, assuming that the amount of time that has elapsed since the cached MUD file was retrieved is less than or equal to the number of hours in the file's cache-validity value. If the cache validity has expired for the respective file, the MUD manager should fetch a new MUD file from the MUD file server.
Associated Test Case(s)	N/A
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-1, ID.AM-2, ID.AM-3, PR.DS-5, DE.AE-1, PR.AC-4, PR.AC-5, PR.IP-1, PR.IP-3, PR.DS-2, PR.PT-3

Test Case Field	Description
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*Ciscopi2.json*
Preconditions	1. All devices have been configured to use IPv4. 2. The MUD PEP router/switch does not yet have any configuration settings pertaining to the IoT device being used in the test. 3. The MUD file for the IoT device being used in the test is identical to the MUD file provided in Section 2.1.3.
Procedure	Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. 1. Run test IoT-1-v4 (or IoT-1-v6). 2. Within 24 hours (i.e., within the cache-validity period for the MUD file) of running test IoT-1-v4 (or IoT-1-v6), remove the IoT device that was connected during test IoT-1-v4 (or IoT-1-v6) from the network. Ensure all traffic filters associated to IoT device have been removed, and reconnect it to the test network. This should set in motion the following series of steps, which should occur automatically. 3. The IoT device automatically emits a DHCPv4 message containing the device's MUD URL (IANA code 161). (Note that in the v6 version of this test, IPv6, DHCPv6, and IANA code 112 will be used.) 4. The DHCP server receives the DHCPv4 message containing the IoT device's MUD URL. 5. The DHCP server offers an IP address lease to the newly connected IoT device. 6. The IoT device requests this IP address lease, which the DHCP server acknowledges. 7. The DHCP server sends the MUD URL to the MUD manager. 8. The MUD manager determines that it has this MUD file cached and checks that the amount of time that has elapsed since the cached file was retrieved is less than or equal to the number of hours in the cache-validity value for this MUD file. If the cache validity has been

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td></td><td>exceeded, the MUD manager will fetch a new MUD file. (Run the test both ways—with a cache-validity period that has expired and with one that has not.)
9. The MUD manager translates the MUD file’s contents into appropriate route filtering rules and installs these rules onto the MUD PEP for the IoT device in question so that this router/switch is now configured to enforce the policies specified in the MUD file.</td></tr>
<tr><td>Expected Results</td><td>The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to enforce the policies specified in the IoT device’s MUD file. The expected configuration should resemble the following.

Cache is valid (the MUD manager does NOT retrieve the MUD file from the MUD file server):

<pre>Extended IP access list mud-81726-v4fr.in
    10 permit tcp any host 192.168.4.7 eq www ack syn
    20 permit tcp any host 192.168.10.104 eq www
    30 permit tcp any host 192.168.10.105 eq www
    50 permit tcp any 192.168.10.0 0.0.0.255 eq www
    60 permit tcp any 192.168.13.0 0.0.0.255 eq www
    70 permit tcp any 192.168.14.0 0.0.0.255 eq www
    80 permit tcp any eq 22 any
    81 permit udp any eq bootpc any eq bootps
    82 permit udp any any eq domain
    83 deny ip any any</pre>
Cache is valid (the MUD manager does NOT retrieve the MUD file from the MUD file server):

<pre>Extended IP access list mud-81726-v4fr.in
    10 permit tcp any host 192.168.4.7 eq www ack syn
    20 permit tcp any host 192.168.10.104 eq www
    30 permit tcp any host 192.168.10.105 eq www
    50 permit tcp any 192.168.10.0 0.0.0.255 eq www
    60 permit tcp any 192.168.13.0 0.0.0.255 eq www
    70 permit tcp any 192.168.14.0 0.0.0.255 eq www
    80 permit tcp any eq 22 any
    81 permit udp any eq bootpc any eq bootps
    82 permit udp any any eq domain</pre></td></tr>
</table>

Test Case Field	Description
	`83 deny ip any any` **Cache is not valid** (the MUD manager does retrieve the MUD file from the MUD file server)**:** `Extended IP access list mud-81726-v4fr.in` `10 permit tcp any host 192.168.4.7 eq www ack syn` `20 permit tcp any host 192.168.10.104 eq www` `30 permit tcp any host 192.168.10.105 eq www` `50 permit tcp any 192.168.10.0 0.0.0.255 eq www` `60 permit tcp any 192.168.13.0 0.0.0.255 eq www` `70 permit tcp any 192.168.14.0 0.0.0.255 eq www` `80 permit tcp any eq 22 any` `81 permit udp any eq bootpc any eq bootps` `82 permit udp any any eq domain` `83 deny ip any any` All protocol exchanges described in steps 1–9 above are expected to occur and can be viewed via Wireshark if desired. If the router/switch does not get configured in accordance with the MUD file, each exchange of DHCP and MUD-related protocol traffic should be viewed on the network via Wireshark to determine which transactions did not proceed as expected, and the observed and absent protocol exchanges should be described here.
Actual Results	**MUD manager logs for valid cache:** `**MUDC [INFO][mudc_print_request_info:2185]--> print parsed HTTP request header info` `***MUDC [INFO][mudc_print_request_info:2186]--> request method: POST` `***MUDC [INFO][mudc_print_request_info:2187]--> request uri: /getaclname` `***MUDC [INFO][mudc_print_request_info:2188]--> local uri: /getaclname` `***MUDC [INFO][mudc_print_request_info:2189]--> http ver-sion: 1.1` `***MUDC [INFO][mudc_print_request_info:2190]--> query string: (null)` `***MUDC [INFO][mudc_print_request_info:2191]--> con-tent_length: 27` `***MUDC [INFO][mudc_print_request_info:2192]--> remote ip addr: 0xe7719c38` `***MUDC [INFO][mudc_print_request_info:2193]--> remote port: 49344`

Test Case Field	Description
	***MUDC [INFO][mudc_print_request_info:2194]--> remote_user: (null) ***MUDC [INFO][mudc_print_request_info:2195]--> is ssl: 0 ***MUDC [INFO][mudc_print_request_info:2199]--> header(0): name: <Host>, value: <127.0.0.1:8000> ***MUDC [INFO][mudc_print_request_info:2199]--> header(1): name: <User-Agent>, value: <FreeRADIUS 3.0.17> ***MUDC [INFO][mudc_print_request_info:2199]--> header(2): name: <Accept>, value: <*/*> ***MUDC [INFO][mudc_print_request_info:2199]--> header(3): name: <Content-Type>, value: <application/json> ***MUDC [INFO][mudc_print_request_info:2199]--> header(4): name: <X-FreeRADIUS-Section>, value: <authorize> ***MUDC [INFO][mudc_print_request_info:2199]--> header(5): name: <X-FreeRADIUS-Server>, value: <default> ***MUDC [INFO][mudc_print_request_info:2199]--> header(6): name: <Content-Length>, value: <27> ***MUDC [INFO][handle_get_aclname:2506]--> Mac address <b827ebeb6c8b> ***MUDC [INFO][fetch_uri_from_macaddr:1702]--> found the fields <{ "_id" : { "$oid" : "5c182c7edb40218cde918776" }, "URI" : "*https://mudfileserver/ciscopi2*" }> ***MUDC [INFO][fetch_uri_from_macaddr:1711]--> ============= Returning URI:*https://mudfileserver/ciscopi2* ***MUDC [INFO][handle_get_aclname:2513]--> Found URI *https://mudfileserver/ciscopi2* for MAC address b827ebeb6c8b ***MUDC [INFO][validate_muduri:2373]--> uri: *https://mud-fileserver/ciscopi2* ***MUDC [INFO][validate_muduri:2399]--> ip: mudfileserver, filename: ciscopi2 ***MUDC [INFO][handle_get_aclname:2558]--> Got URL from message <*https://mudfileserver/ciscopi2*> ***MUDC [INFO][query_policies_by_uri:1419]--> found the record <{ "_id" : { "$oid" : "5c182d9cdb40218cde91884a" }, "DACL_Name" : "ACS:CiscoSecure-Defined-ACL=mud-81726-v4fr.in", "DACL" : "[\"ip:inacl#10=permit tcp any host 192.168.4.7 range 80 80 syn ack\", \"ip:inacl#20=permit tcp any host 192.168.10.104 range 80 80\", \"ip:inacl#30=permit tcp any host 192.168.10.105 range 80 80\", \"ip:inacl#40=permit tcp any host 192.168.10.104 range 80 80\", \"ip:inacl#50=permit tcp any 192.168.10.0 0.0.0.255 range 80 80\", \"ip:inacl#60=permit tcp any 192.168.13.0 0.0.0.255 range 80 80\", \"ip:inacl#70=permit tcp any 192.168.14.0 0.0.0.255 range 80 80\", \"ip:inacl#80=permit tcp any eq 22 any\", \"ip:inacl#81=permit udp any eq 68 any eq 67\", \"ip:inacl#82=permit udp any any eq 53\", \"ip:inacl#83=deny

Test Case Field	Description
	ip any any\"]", "URI" : "*https://mudfileserver/ciscopi2*", "VLAN" : 3 }> ***MUDC [INFO][query_policies_by_uri:1461]--> Response <{ "Cisco-AVPair": ["ACS:CiscoSecure-Defined-ACL=mud-81726-v4fr.in"], "Tunnel-Type": "VLAN", "Tunnel-Medium-Type": "IEEE-802", "Tunnel-Private-Group-Id": 3 }> ***MUDC [INFO][mudc_construct_head:135]--> status_code: 200, content_len: 160, extra_headers: Content-Type: application/aclname ***MUDC [INFO][mudc_construct_head:152]--> HTTP header: HTTP/1.1 200 OK Content-Type: application/aclname Content-Length: 160 ***MUDC [INFO][query_policies_by_uri:1464]--> { "Cisco-AVPair": ["ACS:CiscoSecure-Defined-ACL=mud-81726-v4fr.in"], "Tunnel-Type": "VLAN", "Tunnel-Medium-Type": "IEEE-802", "Tunnel-Private-Group-Id": 3 } ***MUDC [INFO][handle_get_aclname:2568]--> Got ACLs from the MUD URL **MUD manager logs for expired cache:** ***MUDC [INFO][mudc_print_request_info:2185]--> print parsed HTTP request header info ***MUDC [INFO][mudc_print_request_info:2186]--> request method: POST ***MUDC [INFO][mudc_print_request_info:2187]--> request uri: /getaclname ***MUDC [INFO][mudc_print_request_info:2188]--> local uri: /getaclname ***MUDC [INFO][mudc_print_request_info:2189]--> http version: 1.1 ***MUDC [INFO][mudc_print_request_info:2190]--> query string: (null) ***MUDC [INFO][handle_get_aclname:2506]--> Mac address <**b827ebeb6c8b**> ***MUDC [INFO][fetch_uri_from_macaddr:1702]--> found the fields <{ "_id" : { "$oid" : "5c182c7edb40218cde918776" }, "URI" : "*https://mudfileserver/ciscopi2*" }>

Test Case Field	Description

```
***MUDC [INFO][fetch_uri_from_macaddr:1711]--> =============
Returning URI:https://mudfileserver/ciscopi2

***MUDC [INFO][handle_get_aclname:2513]--> Found URI
https://mudfileserver/ciscopi2 for MAC address b827ebeb6c8b

***MUDC [INFO][validate_muduri:2373]--> uri: https://mud-
fileserver/ciscopi2
***MUDC [INFO][validate_muduri:2399]--> ip: mudfileserver,
filename: ciscopi2
***MUDC [INFO][handle_get_aclname:2558]--> Got URL from mes-
sage <https://mudfileserver/ciscopi2>

***MUDC [INFO][query_policies_by_uri:1399]--> Cache has ex-
pired
```

[Omitted for brevity]

```
***MUDC [STATUS][send_mudfs_request:2005]-->
Request URI <https://mudfileserver/ciscopi2>
</home/mudtester/mud-intermediate.pem>

*   Trying 192.168.4.5...
* TCP_NODELAY set
* Connected to mudfileserver (192.168.4.5) port 443 (#0)
* found 1 certificate in /home/mudtester/mud-intermedi-
ate.pem
* found 400 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_256_GCM_SHA384
*       server certificate verification OK
*       server certificate status verification SKIPPED
*       common name: mudfileserver (matched)
*       server certificate expiration date OK
*       server certificate activation date OK
*       certificate public key: RSA
*       certificate version: #3
*       subject: C=US,ST=Maryland,L=Rockville,O=National Cy-
bersecurity Center of Excellence - NIST,CN=mudfileserver
*       start date: Fri, 05 Oct 2018 00:00:00 GMT
*       expire date: Wed, 13 Oct 2021 12:00:00 GMT
*       issuer: C=US,O=DigiCert Inc,CN=DigiCert Test SHA2
Intermediate CA-1
*       compression: NULL
* ALPN, server did not agree to a protocol
> GET /ciscopi2 HTTP/1.1
Host: mudfileserver
Accept: */*
```

Test Case Field	Description
	[Omitted for brevity]
Overall Results	Pass

247 Test case IoT-10-v6 is identical to test case IoT-10-v4 except that IoT-10-v6 tests requirement CR-1.a.2,
248 whereas IoT-10-v4 tests requirement CR-1.a.1. Hence, as explained above, test IoT-10-v6 uses IPv6,
249 DHCPv6, and IANA code 112 instead of using IPv4, DHCPv4, and IANA code 161.

250 *2.1.2.11 Test Case IoT-11-v4*

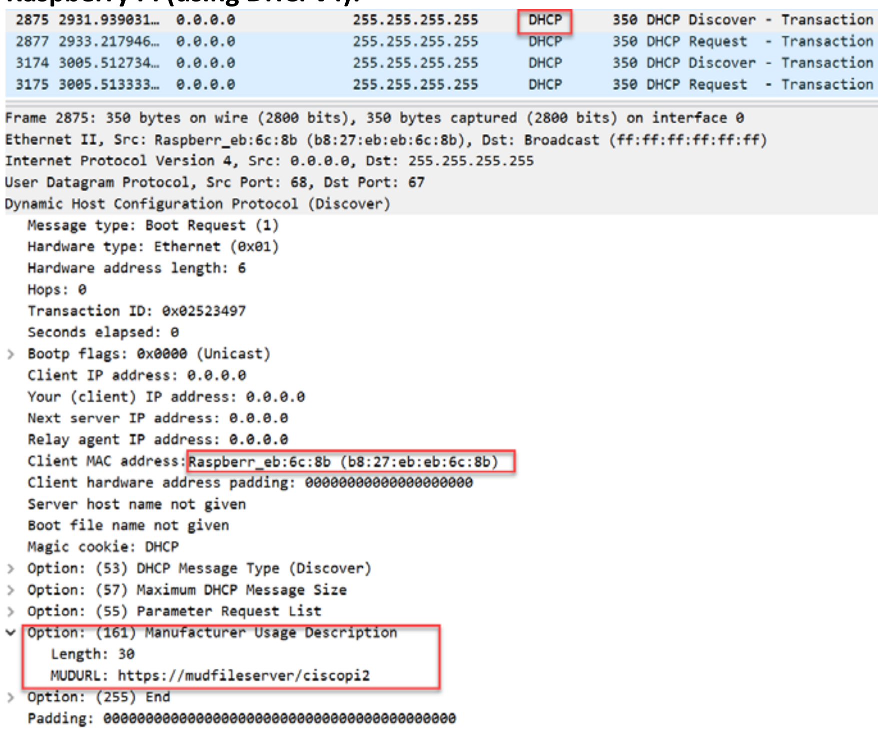
251 **Table 2-12: Test Case IoT-11-v4**

Test Case Field	Description
Parent Requirements	(CR-1) The IoT DDoS example implementation shall include a mechanism for associating a device with a MUD file URL (e.g., by having the MUD-enabled IoT device emit a MUD file URL via DHCP, Link Layer Discovery Protocol [LLDP], or X.509 or by using some other mechanism to enable the network to associate a device with a MUD file URL).
Testable Requirements	(CR-1.a) Upon initialization, the MUD-enabled IoT device shall broadcast a DHCP message on the network, including at most one MUD URL, in https scheme, within the DHCP transaction. (CR-1.a.1) The DHCP server shall be able to receive DHCPv4 DISCOVER and REQUEST with IANA code 161 (OPTION_MUD_URL_V4) from the MUD-enabled IoT device. OR (CR-1.b) Upon initialization, the MUD-enabled IoT device shall emit the MUD URL as an LLDP extension. (CR-1.b.1) The network service shall be able to process the MUD URL that is received as an LLDP extension.
Description	Shows that the IoT DDoS example implementation includes IoT devices that can emit a MUD URL via DHCP or LLDP

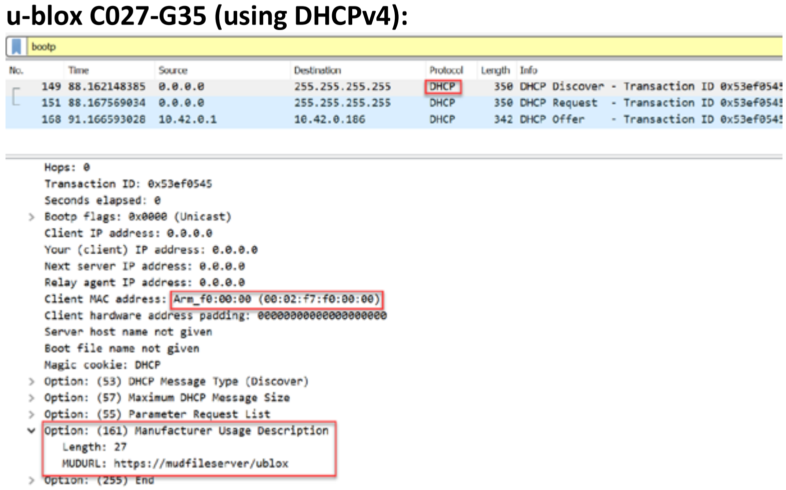
Test Case Field	Description
Associated Test Case(s)	N/A
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-1
IoT Device(s) Under Test	Raspberry Pi, Molex light engine, u-blox C027-G35
MUD File(s) Used	*Ciscopi2.json, molex.json, ublox.json*
Preconditions	Device has been developed to emit a MUD URL in a DHCP transaction
Procedure	1. Power on a device and connect it to the network. 2. Verify that the device emits a MUD URL in a DHCP transaction or LLDP message. a. Use Wireshark to capture a DHCP transaction with options present. b. Use Wireshark to capture an LLDP message with a MUD URL present in the LLDP frame.
Expected Results	DHCP transaction with MUD option 161 or LLDP TLV MUD extension enabled and MUD URL included

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

Raspberry Pi (using DHCPv4):

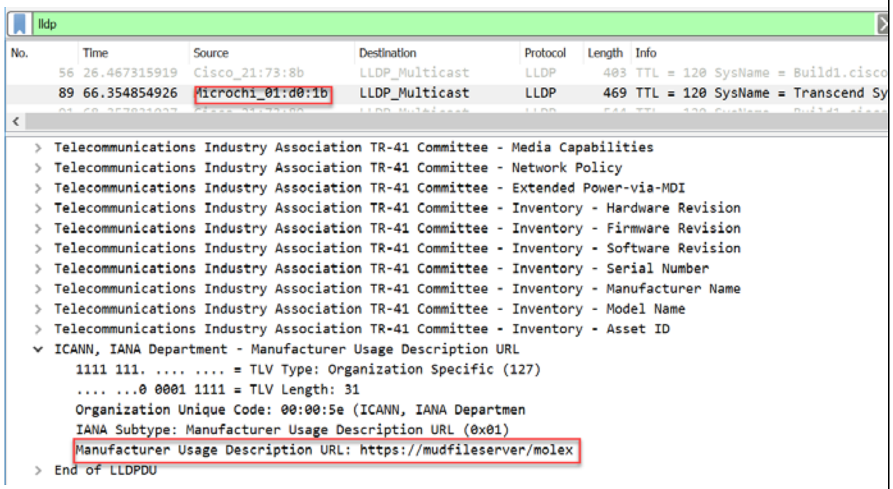


u-blox C027-G35 (using DHCPv4):



Molex light engine (using LLDP):

</td></tr>
</table>

Test Case Field	Description
	
Overall Results	Pass

2.1.3 MUD Files

This section contains the MUD files that were used in the Build 1 functional demonstration.

2.1.3.1 Ciscopi2.json

The complete Ciscopi2.json MUD file has been linked to this document. To access this MUD file please click the link below.

Ciscopi2.json

2.1.3.2 expiredcerttest.json

The complete expiredcerttest.json MUD file has been linked to this document. To access this MUD file please click the link below.

expiredcerttest.json

2.1.3.3 molex.json

The complete molex.json MUD file has been linked to this document. To access this MUD file please click the link below.

molex.json

2.1.3.4 ublox.json

The complete ublox.json MUD file has been linked to this document. To access this MUD file please click the link below.

ublox.json

2.1.3.5 dnstest.json

The complete dnstest.json MUD file has been linked to this document. To access this MUD file please click the link below.

dnstest.json

2.2 Demonstration of Non-MUD-Related Capabilities

In addition to supporting MUD, Build 1 supports capabilities with respect to device discovery, attribute identification, and monitoring. Table 2-13 lists the non-MUD-related capabilities that were demonstrated for Build 1. We use the letter “C” as a prefix for these functional capability identifiers in the table below because these capabilities are specific to Build 1, which uses Cisco equipment.

2.2.1 Non-MUD-Related Functional Capabilities Demonstrated

Table 2-13: Non-MUD-Related Functional Capabilities Demonstrated

Functional Capability	Parent Capability	Subrequirement 1	Subrequirement 2	Exercise ID
C-1	The IoT DDoS example implementation shall include a visibility component that can **detect, identify, categorize, and monitor the status of IoT devices** that are on the network.			CnMUD-13-v4, CnMUD-13-v6
C-1.a		The visibility component shall **detect and identify** the attributes and category of a newly connected IoT device.		CnMUD-13-v4, IoT-13-v6
C-1.a.1			The visibility component shall **monitor the status** of the IoT device (e.g., notice if the device goes offline).	CnMUD-13-v4, IoT-13-v6

2.2.2 Exercises to Demonstrate the Above Non-MUD-Related Capabilities

This section contains the exercises that were performed to verify that Build 1 supports the non-MUD-related capabilities listed in Table 2-13.

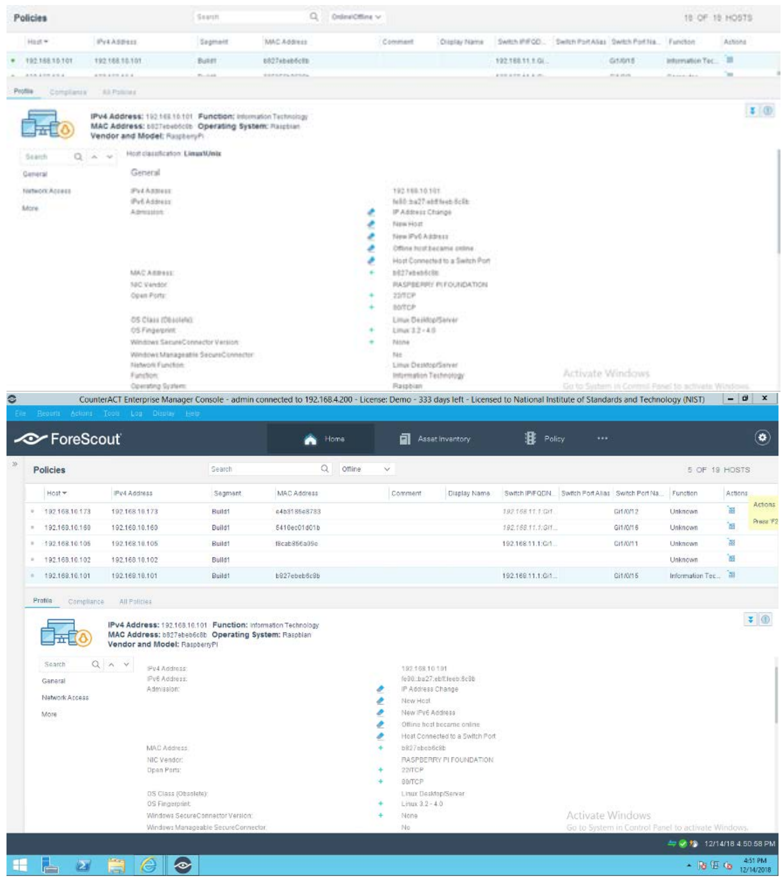
285 ### 2.2.2.1 Exercise CnMUD-13-v4

286 **Table 2-14: Exercise CnMUD-13-v4**

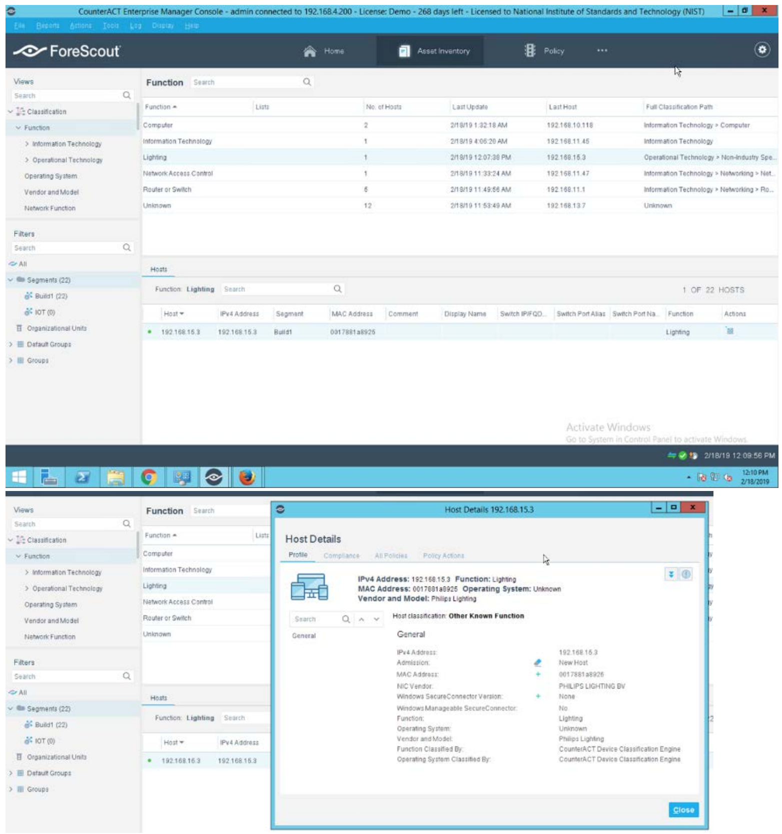
Test Case Field	Description
Parent Requirements	(C-1) The IoT DDoS example implementation shall include a visibility component that can detect, identify, categorize, and monitor the status of IoT devices that are on the network.
Testable Requirements	(C-1.a) The visibility component shall detect and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

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td></td><td>3. Turn off the device.
4. Verify that its absence from the network is detected.
5. Power the device back on.
6. Verify that its presence is detected and its features are identified correctly.
7. Disconnect the device from the network.
8. Verify that its absence from the network is detected.</td></tr>
<tr><td>Expected Results</td><td>All expectations as enumerated in items 2, 4, 6, and 8 above are observed.</td></tr>
<tr><td>Actual Results</td><td>At Power-On:
<pre>pi@raspberrypi:~ $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.10.101 netmask 255.255.255.0 broadcast
192.168.10.255
        ether b8:27:eb:eb:6c:8b txqueuelen 1000 (Ethernet)
        RX packets 9193 bytes 8208593 (7.8 MiB)
        RX errors 0 dropped 5 overruns 0 frame 0
        TX packets 7210 bytes 822414 (803.1 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 colli-
sions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 16 bytes 1467 (1.4 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 16 bytes 1467 (1.4 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 colli-
sions 0</pre>
Screenshot from Forescout:
IoT device status is indicated by green or gray light shown in the screen capture</td></tr>
</table>

Test Case Field	Description
	

Test Case Field	Description
	Categorizing IoT Device: We tested this function with a smart light bulb. See the example screen-shots below.

Test Case Field	Description
	
Overall Results	Pass

287 Test case CnMUD-13-v6 is identical to test case CnMUD-13-v4 except that test case CnMUD-13-v6 uses
288 IPv6 and DHCPv6 instead of using IPv4 and DHCPv4.

3 Build 2

Build 2 uses equipment from MasterPeace Solutions Ltd., GCA, and ThreatSTOP. The MasterPeace Solutions Yikes! router, cloud service, and mobile application are used to support MUD as well as to perform device discovery on the network and to apply additional traffic rules to both MUD-capable and non-MUD-capable devices based on device manufacturer and model. The GCA Quad9 DNS Service and the ThreatSTOP Threat MUD File Server are used to support threat signaling.

3.1 Evaluation of MUD-Related Capabilities

The functional evaluation that was conducted to verify that Build 2 conforms to the MUD specification was based on the Build 2-specific requirements listed in Table 3-1.

3.1.1 Requirements

Table 3-1: MUD Use Case Functional Requirements

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-1	The IoT DDoS example implementation shall include a mechanism for associating a device with a MUD file URL (e.g., by having the MUD-enabled **IoT device emit a MUD file URL via DHCP, LLDP, or X.509 or by using some other mechanism to enable the network to associate a device with a MUD file URL).**			IoT-1-v4, IoT-1-v6, IoT-11-v4, IoT-11-v6
CR-1.a		Upon initialization, the MUD-enabled IoT device shall broadcast a DHCP message on the network, including at most one **MUD URL, in https scheme,**		IoT-1-v4, IoT-1-v6, IoT-11-v4, IoT-11-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
		within the DHCP transaction.		
CR-1.a.1			The DHCP server shall be able to receive **DHCPv4 DISCOVER and REQUEST with IANA code 161** (OPTION_MUD_URL_V4) from the MUD-enabled IoT device.	IoT-1-v4, IoT-11-v4
CR-1.a.2			The DHCP server shall be able to receive **DHCPv6 Solicit and Request with IANA code 112** (OPTION_MUD_URL_V6) from the MUD-enabled IoT device.	IoT-1-v6, IoT-11-v6
CR-2	The IoT DDoS example implementation shall include the capability for the MUD URL **to be provided to a MUD manager.**			IoT-1-v4, IoT-1-v6
CR-2.a		The DHCP server shall **assign an IP address lease** to the MUD-enabled IoT device.		IoT-1-v4, IoT-1-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-2.a.1			The MUD-enabled IoT device shall **receive the IP address.**	IoT-1-v4, IoT-1-v6
CR-2.b		**The DHCP server shall** receive the DHCP message and **extract the MUD URL, which is then passed to the MUD manager.**		IoT-1-v4, IoT-1-v6
CR-2.b.1			**The MUD manager shall receive the MUD URL.**	IoT-1-v4, IoT-1-v6
CR-3	The IoT DDoS example implementation shall include a **MUD manager that can request a MUD file and signature from a MUD file server.**			IoT-1-v4, IoT-1-v6
CR-3.a		The MUD manager shall use the GET method (RFC 7231) to **request MUD and signature files** (per RFC 7230) from the MUD file server and can **validate the MUD file server's TLS certificate** by using the rules in RFC 2818.		IoT-1-v4, IoT-1-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-3.a.1			**The MUD file server shall receive the https request from the MUD manager.**	IoT-1-v4, IoT-1-v6
CR-3.b		**The MUD manager** shall use the GET method (RFC 7231) to request MUD and signature files (per RFC 7230) from the MUD file server, but it **cannot validate the MUD file server's TLS certificate** by using the rules in RFC 2818.		IoT-2-v4, IoT-2-v6
CR-3.b.1			**The MUD manager shall drop the connection** to the MUD file server.	IoT-2-v4, IoT-2-v6
CR-3.b.2			**The MUD manager shall send locally defined policy to the router or switch** that handles whether to allow or block traffic to and from the MUD-enabled IoT device.	IoT-2-v4, IoT-2-v6
CR-4	The IoT DDoS example implementation shall include a **MUD file server that can**			IoT-1-v4, IoT-1-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
	serve a MUD file and signature to the MUD manager.			
CR-4.a		**The MUD file server shall serve the file and signature to the MUD manager, and the MUD manager shall check to determine whether the certificate used to sign the MUD file** (signed using DER-encoded CMS [RFC 5652]) was valid at the time of signing, i.e., the **certificate had not expired.**		IoT-1-v4, IoT-1-v6
CR-4.b		**The MUD file server shall serve the file and signature to the MUD manager, and the MUD manager shall check to determine whether the certificate used to sign the MUD file** was valid at the time of signing, i.e., the **certificate had already expired when it was used to sign the MUD file.**		IoT-3-v4, IoT-3-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-4.b.1			The MUD manager shall cease to process the MUD file.	IoT-3-v4, IoT-3-v6
CR-4.b.2			The MUD manager shall send locally defined policy to the router or switch that handles whether to allow or block traffic to and from the MUD-enabled IoT device.	IoT-3-v4, IoT-3-v6
CR-5	The IoT DDoS example implementation shall include a **MUD manager** that **can translate local network configurations based on the MUD file.**			IoT-1-v4, IoT-1-v6
CR-5.a		**The MUD manager shall successfully validate the signature of the MUD file.**		IoT-1-v4, IoT-1-v6
CR-5.a.1			The MUD manager, after validation of the MUD file signature, shall **check for an existing MUD file and translate abstractions in the MUD file to router or switch configurations.**	IoT-1-v4, IoT-1-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-5.a.2			The MUD manager shall **cache** this newly received MUD file.	IoT-10-v4, IoT-10-v6
CR-5.b		The MUD manager shall attempt to validate the signature of the **MUD file,** but the **signature validation fails** (even though the certificate that had been used to create the signature had not been expired at the time of signing, i.e., the signature is invalid for a different reason).		IoT-4-v4, IoT-4-v6
CR-5.b.1			**The MUD manager shall cease processing the MUD file.**	IoT-4-v4, IoT-4-v6
CR-5.b.2			**The MUD manager shall send locally defined policy to the router or switch** that handles whether to allow or block traffic to and from the MUD-enabled IoT device.	IoT-4-v4, IoT-4-v6
CR-6	The IoT DDoS example implementation shall include a			IoT-1-v4, IoT-1-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
	MUD manager that can configure the MUD PEP, i.e., the router or switch nearest the MUD-enabled IoT device that emitted the URL.			
CR-6.a		**The MUD manager shall install a router configuration** on the router or switch nearest the MUD-enabled IoT device that emitted the URL.		IoT-1-v4, IoT-1-v6
CR-6.a.1			**The router or switch shall have been configured to enforce the route filter sent by the MUD manager.**	IoT-1-v4, IoT-1-v6
CR-7	The IoT DDoS example implementation shall **allow the MUD-enabled IoT device to communicate with approved internet services in the MUD file.**			IoT-5-v4, IoT-5-v6
CR-7.a		The MUD-enabled IoT device shall attempt to **initiate outbound traffic to approved internet services.**		IoT-5-v4, IoT-5-v6

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

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

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

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

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-10.a		The MUD-enabled IoT device shall **attempt to initiate lateral traffic to unapproved** (implicitly denied) **devices.**		IoT-6-v4, IoT-6-v6
CR-10.a.1			**The router or switch shall receive the attempt and shall deny it** based on the filters from the MUD file.	IoT-6-v4, IoT-6-v6
CR-10.b		**An unapproved** (implicitly denied) **device shall attempt to initiate a lateral connection** to the MUD-enabled IoT device.		IoT-6-v4, IoT-6-v6
CR-10.b.1			**The router or switch shall receive the attempt and shall deny it** based on the filters from the MUD file.	IoT-6-v4, IoT-6-v6
CR-11	If the IoT DDoS example implementation is such that its DHCP server does not act as a MUD manager and it forwards a MUD URL to a MUD manager, **the DHCP server must notify the MUD manager of any corresponding change to the DHCP state** of			IoT-7-v4, IoT-7-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
	the MUD-enabled IoT device, and the MUD manager should **remove the implemented policy configuration in the router/switch pertaining to that MUD-enabled IoT device.**			
CR-11.a		The MUD-enabled IoT **device shall explicitly release the IP address lease** (i.e., it sends a DHCP release message to the DHCP server).		IoT-7-v4, IoT-7-v6
CR-11.a.1			**The DHCP server shall notify the MUD manager that the device's IP address lease has been released.**	IoT-7-v4, IoT-7-v6
CR-11.a.2			**The MUD manager should remove all policies** associated with the disconnected IoT device that had been configured on the MUD PEP router/switch.	IoT-7-v4, IoT-7-v6
CR-11.b		The MUD-enabled IoT **device's IP address lease shall expire.**		IoT-8-v4, IoT-8-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-11.b.1			**The DHCP server shall notify the MUD manager that the device's IP address lease has expired.**	IoT-8-v4, IoT-8-v6
CR-11.b.2			**The MUD manager should remove all policies** associated with the affected IoT device that had been configured on the MUD PEP router/switch.	IoT-8-v4, IoT-8-v6
CR-12	The IoT DDoS example implementation shall include a **MUD manager that uses a cached MUD file rather than retrieve a new one if the cache-validity time period has not yet elapsed** for the MUD file indicated by the MUD URL**. The MUD manager should fetch a new MUD file if the cache-validity time period has already elapsed.**			IoT-10-v4, IoT-10-v6
CR-12.a		The MUD manager shall check if the file associated with the **MUD URL is present in its cache** and shall determine that it is.		IoT-10-v4, IoT-10-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-12.a.1			The MUD manager shall **check whether the amount of time that has elapsed since the cached file was retrieved is less than or equal to the number of hours in the cache-validity value for this MUD file.** If so, the MUD manager shall apply the contents of the cached MUD file.	IoT-10-v4, IoT-10-v6
CR-12.a.2			The MUD manager **shall check whether the amount of time that has elapsed since the cached file was retrieved is greater than the number of hours in the cache-validity value for this MUD file.** If so, the MUD manager may (but does not have to) fetch a new file by using the MUD URL received.	IoT-10-v4, IoT-10-v6
CR-13	The IoT DDoS example implementation shall ensure that for each rule in a MUD file that pertains to an external domain, the MUD PEP			IoT-9-v4, IoT-9-v6

Capability Requirement (CR-ID)	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
	router/switch will get configured with **all possible instantiations of that rule,** insofar as **each instantiation contains one of the IP addresses to which the domain in that MUD file rule may be resolved when queried by the MUD PEP router/switch.**			
CR-13.a		The MUD file for a device shall contain a rule involving a **domain that can resolve to multiple IP addresses** when queried by the MUD PEP router/switch. **An ACL for permitting access to each of those IP addresses will be inserted into the MUD PEP router/switch** for the device in question, and the device will be permitted to communicate with all of those IP addresses.		IoT-9-v4, IoT-9-v6
CR-13.a.1			IPv4 addressing is used on the network.	IoT-9-v4
CR-13.a.2			IPv6 addressing is used on the network.	IoT-9-v6

3.1.2 Test Cases

3.1.2.1 Test Case IoT-1-v4

This section contains the test cases that were used to verify that Build 2 met the requirements listed in Table 3-1.

Table 3-2: Test Case IoT-1-v4

Test Case Field	Description
Parent Requirements	(CR-1) The IoT DDoS example implementation shall include a mechanism for associating a device with a MUD file URL (e.g., by having the MUD-enabled IoT device emit a MUD file URL via DHCP, LLDP, or X.509 or by using some other mechanism to enable the network to associate a device with a MUD file URL). (CR-2) The IoT DDoS example implementation shall include the capability for the MUD URL to be provided to a MUD manager. (CR-3) The IoT DDoS example implementation shall include a MUD manager that can request a MUD file and signature from a MUD file server. (CR-4) The IoT DDoS example implementation shall include a MUD file server that can serve a MUD file and signature to the MUD manager. (CR-5) The IoT DDoS example implementation shall include a MUD manager that can translate local network configurations based on the MUD file. (CR-6) The IoT DDoS example implementation shall include a MUD manager that can configure the router or switch nearest the MUD-enabled IoT device that emitted the URL.
Testable Requirements	(CR-1.a) Upon initialization, the MUD-enabled IoT device shall broadcast a DHCP message on the network, including at most one MUD URL, in https scheme, within the DHCP transaction. (CR-1.a.1) The DHCP server shall be able to receive DHCPv4 DISCOVER and/or REQUEST with IANA code 161 (OPTION_MUD_URL_V4) from the MUD-enabled IoT device. (NOTE: Test IoT-1-v6 does not test this requirement; instead, it tests CR-1.a.2, which pertains to DHCPv6 rather than DHCPv4.)

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

Test Case Field	Description
Associated Test Case(s)	N/A
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-1, ID.AM-2, ID.AM-3, PR.DS-5, DE.AE-1, PR.AC-4, PR.AC-5, PR.IP-1, PR.IP-3, PR.PT-3, PR.DS-2
IoT Device(s) Under Test	Raspberry Pi (1)
MUD File(s) Used	*Yikesmain.json*
Preconditions	1. This MUD file is not currently cached at the MUD manager. 2. The device's MUD file has a valid signature that was signed by a certificate that had not yet expired, and it is being hosted on a MUD file server that has a valid TLS certificate. 3. The MUD PEP router/switch does not yet have any configuration settings pertaining to the IoT device being used in the test. 4. The MUD file for the IoT device being used in the test is identical to the MUD file provided in Section 3.1.3.
Procedure	Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. Also verify that the MUD file of the IoT device to be used is not currently cached at the MUD manager. Power on the IoT device and connect it to the test network. This should set in motion the following series of steps, which should occur automatically: 1. The IoT device automatically emits a MUD URL in a DHCPv4 message containing the device's MUD URL (IANA code 161). (Note that in the v6 version of this test, IPv6, DHCPv6, and IANA code 112 will be used.) 2. The DHCP server offers an IP address lease to the newly connected IoT device.

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td></td><td>3. The IoT device requests this IP address lease, which the DHCP server acknowledges.
4. The DHCP server receives the DHCP message containing the IoT device's MUD URL.
5. The DHCP service extracts the MUD URL.
6. The MUD URL is then provided to the MUD manager.
7. The MUD manager automatically contacts the MUD file server that is located by using the MUD URL, verifies that it has a valid TLS certificate, requests and receives the MUD file and signature from the MUD file server, validates the MUD file's signature, and translates the MUD file's contents into appropriate route filtering rules. The MUD manager installs these rules onto the MUD PEP for the IoT device in question so that this router/switch is now configured to enforce the policies specified in the MUD file.</td></tr>
<tr><td>Expected Results</td><td>The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to enforce the policies specified in the IoT device's MUD file. The expected configuration should resemble the following:<pre>
config rule
       option enabled   '1'
       option name   'mud_192.168.20.222_main-pi-
Build2_cl0-frdev'
       option target   ACCEPT
       option src   lan
       option dest   wan
       option proto   tcp
       option family   ipv4
       option src_ip   192.168.20.222
       option dest_ip   198.71.233.87
       option dest_port 443:443

config rule
       option enabled   '1'
       option name   'mud_192.168.20.222_main-pi-
Build2_cl0-todev'
       option target   ACCEPT
       option src   wan
       option dest   lan
       option proto   tcp
       option family   ipv4
</pre></td></tr>
</table>

Test Case Field	Description
	<pre> option src_ip 198.71.233.87 option dest_ip 192.168.20.222 option dest_port 443:443 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_cl1-frdev' option target ACCEPT option src lan option dest wan option proto tcp option family ipv4 option src_ip 192.168.20.222 option dest_ip 192.168.4.7 option dest_port 80:80 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_cl1-todev' option target ACCEPT option src wan option dest lan option proto tcp option family ipv4 option src_ip 192.168.4.7 option dest_ip 192.168.20.222 option dest_port 80:80 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_cl2-frdev' option target ACCEPT option src lan option dest wan option proto tcp option family ipv4 option src_ip 192.168.20.222 option dest_ip 99.84.216.69 option dest_port 443:443 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_cl2-frdev' option target ACCEPT</pre>

Test Case Field	Description
	`` option src lan option dest wan option proto tcp option family ipv4 option src_ip 192.168.20.222 option dest_ip 99.84.216.65 option dest_port 443:443 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_cl2-frdev' option target ACCEPT option src lan option dest wan option proto tcp option family ipv4 option src_ip 192.168.20.222 option dest_ip 99.84.216.79 option dest_port 443:443 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_cl2-frdev' option target ACCEPT option src lan option dest wan option proto tcp option family ipv4 option src_ip 192.168.20.222 option dest_ip 99.84.216.27 option dest_port 443:443 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_cl2-todev' option target ACCEPT option src wan option dest lan option proto tcp option family ipv4 option src_ip 99.84.216.27 option dest_ip 192.168.20.222 option dest_port 443:443 config rule ```

Test Case Field	Description
	` option enabled '1'` ` option name 'mud_192.168.20.222_main-pi-` `Build2_cl2-todev'` ` option target ACCEPT` ` option src wan` ` option dest lan` ` option proto tcp` ` option family ipv4` ` option src_ip 99.84.216.79` ` option dest_ip 192.168.20.222` ` option dest_port 443:443` `config rule` ` option enabled '1'` ` option name 'mud_192.168.20.222_main-pi-` `Build2_cl2-todev'` ` option target ACCEPT` ` option src wan` ` option dest lan` ` option proto tcp` ` option family ipv4` ` option src_ip 99.84.216.65` ` option dest_ip 192.168.20.222` ` option dest_port 443:443` `config rule` ` option enabled '1'` ` option name 'mud_192.168.20.222_main-pi-` `Build2_cl2-todev'` ` option target ACCEPT` ` option src wan` ` option dest lan` ` option proto tcp` ` option family ipv4` ` option src_ip 99.84.216.69` ` option dest_ip 192.168.20.222` ` option dest_port 443:443` `config rule` ` option enabled '1'` ` option name 'mud_192.168.20.222_main-pi-` `Build2_ent0-frdev'` ` option target ACCEPT` ` option src lan` ` option dest wan` ` option proto tcp` ` option family ipv4` ` option src_ip 192.168.20.222`

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td></td><td><pre>
        option dest_ip   172.217.164.132
        option dest_port 443:443

config rule
        option enabled   '1'
        option name   'mud_192.168.20.222_main-pi-
Build2_ent0-frdev'
        option target   ACCEPT
        option src   lan
        option dest   wan
        option proto   tcp
        option family   ipv4
        option src_ip   192.168.20.222
        option dest_ip   0.0.0.0
        option dest_port 443:443

config rule
        option enabled   '1'
        option name   'mud_192.168.20.222_main-pi-
Build2_ent0-todev'
        option target   ACCEPT
        option src   wan
        option dest   lan
        option proto   tcp
        option family   ipv4
        option src_ip   172.217.164.132
        option dest_ip   192.168.20.222
        option dest_port 443:443

config rule
        option enabled   '1'
        option name   'mud_192.168.20.222_main-pi-
Build2_ent0-todev'
        option target   ACCEPT
        option src   wan
        option dest   lan
        option proto   tcp
        option family   ipv4
        option src_ip   0.0.0.0
        option dest_ip   192.168.20.222
        option dest_port 443:443

config rule
        option enabled   '1'
        option name   'mud_192.168.20.222_main-pi-
Build2_loc0-frdev'
        option target   ACCEPT
        option src   lan
</pre></td></tr>
</table>

Test Case Field	Description
	`` option dest lan option proto tcp option family ipv4 option src_ip 192.168.20.222 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_loc0-todev' option target ACCEPT option src lan option dest lan option proto tcp option family ipv4 option src_ip any option dest_ip 192.168.20.222 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_man0-frdev-SM' option target ACCEPT option src lan option dest lan option proto tcp option family ipv4 option src_ip 192.168.20.222 option ipset www_gmail_com-SMTD option dest_port 80:80 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_man0-todev-SM' option target ACCEPT option src lan option dest lan option proto tcp option family ipv4 option ipset www_gmail_com-SMFD option dest_ip 192.168.20.222 option dest_port 80:80 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_myctl0-frdev' option target ACCEPT ```

Test Case Field	Description
	`option src lan` `option dest wan` `option proto all` `option family ipv4` `option src_ip 192.168.20.222` `option dest_ip 192.168.20.101` `config rule` `option enabled '1'` `option name 'mud_192.168.20.222_main-pi-Build2_myctl0-todev'` `option target ACCEPT` `option src wan` `option dest lan` `option proto all` `option family ipv4` `option src_ip 192.168.20.101` `option dest_ip 192.168.20.222` `config rule` `option enabled '1'` `option name 'mud_192.168.20.222_main-pi-Build2_myman0-frdev-SM'` `option target ACCEPT` `option src lan` `option dest lan` `option proto udp` `option family ipv4` `option src_ip 192.168.20.222` `option ipset mudfiles_nist_getyikes_com-SMTD` `config rule` `option enabled '1'` `option name 'mud_192.168.20.222_main-pi-Build2_myman0-todev-SM'` `option target ACCEPT` `option src lan` `option dest lan` `option proto udp` `option family ipv4` `option ipset mudfiles_nist_getyikes_com-SMFD` `option dest_ip 192.168.20.222` `config rule` `option enabled '1'` `option name 'mud_192.168.20.222_main-pi-Build2_REJECT-ALL-LOCAL-FROM'` `option target REJECT`

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

```
        option src       lan
        option dest      lan
        option proto     all
        option family    ipv4
        option src_ip     192.168.20.222

config rule
        option enabled   '1'
        option name      'mud_192.168.20.222_main-pi-
Build2_REJECT-ALL-LOCAL-TO'
        option target    REJECT
        option src       lan
        option dest      lan
        option proto     all
        option family    ipv4
        option src_ip     any
        option dest_ip    192.168.20.222

config rule
        option enabled   '1'
        option name      'mud_192.168.20.222_main-pi-
Build2_REJECT-ALL'
        option target    REJECT
        option src       lan
        option dest      wan
        option proto     all
        option family    ipv4
        option src_ip     192.168.20.222
# OSMUD end
```

All protocol exchanges described in steps 1–7 above are expected to occur and can be viewed via Wireshark if desired. If the router/switch does not get configured in accordance with the MUD file, each exchange of DHCP and MUD-related protocol traffic should be viewed on the network via Wireshark to determine which transactions did not proceed as expected, and the observed and absent protocol exchanges should be described here.</td></tr>
<tr><td>Actual Results</td><td>

Procedures 1–3:

```
pi@main-pi-Build2:~$ sudo dhclient -v -i eth0
sudo: unable to resolve host main-pi-Build2: Connection re-
fused
Internet Systems Consortium DHCP Client 4.3.5
Copyright 2004-2016 Internet Systems Consortium.
All rights reserved.
```

</td></tr>
</table>

Test Case Field	Description
	For info, please visit https://www.isc.org/software/dhcp/ RTNETLINK answers: Operation not possible due to RF-kill Listening on LPF/wlan0/b8:27:eb:be:39:de Sending on LPF/wlan0/b8:27:eb:be:39:de Listening on LPF/eth0/b8:27:eb:eb:6c:8b Sending on LPF/eth0/b8:27:eb:eb:6c:8b Sending on Socket/fallback **DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 4** **DHCPREQUEST of 192.168.20.222 on eth0 to 255.255.255.255 port 67** **DHCPOFFER of 192.168.20.222 from 192.168.20.1** **DHCPACK of 192.168.20.222 from 192.168.20.1** Too few arguments. Too few arguments. **bound to 192.168.20.222** -- renewal in 1800 seconds. **Procedures 4–5:** **dhcpmasq.txt** 2019-07-15T20:27:57Z|OLD|Wired|DHCP|-|MUD|-|-|ba:47:a1:7d:60:44|192.168.20.148|| 2019-07-15T20:28:01Z|OLD|NIST 5|DHCP|-|MUD|-|-|18:b4:30:50:98:38|192.168.20.203|| 2019-07-15T20:28:08Z|OLD|NIST 2.4|DHCP|-|MUD|-|-|d0:73:d5:28:08:2a|192.168.20.202|| 2019-07-15T20:28:11Z|OLD|Wired|DHCP|-|MUD|-|-|b8:27:eb:95:55:fe|192.168.20.232|raspberrypi| **2019-07-15T20:28:31Z|NEW|Wired|DHCP|1,28,2,3,15,6,119,12,44,47,26,121,42|MUD|**https://mudfiles.nist.getyikes.com/yikesmain.json|-|b8:27:eb:eb:6c:8b|192.168.20.222|main-pi-Build2| 2019-07-15T20:28:42Z|NEW|NIST 5|DHCP|1,28,2,121,15,6,12,40,41,42,26,119,3,121,249,33,252,42|MUD|-|-|80:00:0b:ef:81:70|192.168.20.238|| **Procedure 6:** **MUD MANAGER:** **2019-07-15 20:28:32 DEBUG::GENERAL::2019-07-15T20:28:31Z|NEW|Wired|DHCP|1,28,2,3,15,6,119,12,44,47,26,121,42|MUD|**https://mudfiles.nist.getyikes.com/yikesmain.json|-|b8:27:eb:eb:6c:8b|192.168.20.222|main-pi-Build2|

Test Case Field	Description
	2019-07-15 20:28:32 DEBUG::GENERAL::Executing on dhcpmasq info 2019-07-15 20:28:32 INFO::GENERAL::NEW Device Action: IP: 192.168.20.222, MAC: b8:27:eb:eb:6c:8b 2019-07-15 20:28:32 DEBUG::COMMUNICATION::curl_easy_perform() doing it now.... 2019-07-15 20:28:32 DEBUG::COMMUNICATION::https://mudfiles.nist.getyikes.com/yikesmain.json 2019-07-15 20:28:32 DEBUG::COMMUNICATION::Found HTTPS 2019-07-15 20:28:32 DEBUG::COMMUNICATION::in write data 2019-07-15 20:28:32 DEBUG::COMMUNICATION::curl_easy_perform() success 2019-07-15 20:28:32 DEBUG::COMMUNICATION::MUD File Server returned success state. 2019-07-15 20:28:32 DEBUG::COMMUNICATION::curl_easy_perform() doing it now.... 2019-07-15 20:28:32 DEBUG::COMMUNICATION::https://mudfiles.nist.getyikes.com/yikesmain.p7s 2019-07-15 20:28:32 DEBUG::COMMUNICATION::Found HTTPS 2019-07-15 20:28:32 DEBUG::COMMUNICATION::in write data **2019-07-15 20:28:32 DEBUG::COMMUNICATION::curl_easy_perform() success** **2019-07-15 20:28:32 DEBUG::COMMUNICATION::MUD File Server returned success state.** **2019-07-15 20:28:32 DEBUG::MUD_FILE_OPERATIONS::IN ****NEW**** MUD and SIG FILE RETRIEVED!!!** **2019-07-15 20:28:32 DEBUG::GENERAL::IN ****NEW**** validateMudFileWithSig()** **2019-07-15 20:28:32 DEBUG::GENERAL::openssl cms -verify -in /etc/osmud/state/mudfiles/yikesmain.p7s -inform DER -content /etc/osmud/state/mudfiles/yikesmain.json -purpose any > /dev/null** 2019-07-15 20:28:32 DEBUG::GENERAL::IN ****NEW**** executeMudWithDhcpContext() 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/create_mud_db_entry.sh -d /etc/osmud/state/mudfiles/mudStateFile.txt -i 192.168.20.222 -m b8:27:eb:eb:6c:8b -c main-pi-Build2 -u https://mudfiles.nist.getyikes.com/yikesmain.json -f /etc/osmud/state/mudfiles/yikesmain.json 2019-07-15 20:28:32 DEBUG::GENERAL::rm -f /tmp/osmud/* 2019-07-15 20:28:32 DEBUG::GENERAL::cp /etc/osmud/state/ipSets/* /tmp/osmud 2019-07-15 20:28:32 WARNING::DEVICE_INTERFACE::The URL in the MUD file does not match the URL used to download the MUD

Test Case Field	Description
	FILE 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/remove_ip_fw_rule.sh -i 192.168.20.222 -m b8:27:eb:eb:6c:8b -d /tmp/osmud 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/remove_from_ipset.sh -d /tmp/osmud -i 192.168.20.222 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/add_to_ipset.sh -d /tmp/osmud -a mudfiles.nist.getyikes.com -n SM -i 192.168.20.222 -c main-pi-Build2 2019-07-15 20:28:32 INFO::DEVICE_INTERFACE::Processing ACL-DNS *from* ace rule. 2019-07-15 20:28:32 DEBUG::GENERAL::Starting DNS lookup 2019-07-15 20:28:32 DEBUG::GENERAL::www.osmud.org 2019-07-15 20:28:32 DEBUG::GENERAL::198.71.233.87 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d wan -i 192.168.20.222 -a any -j 198.71.233.87 -b 443:443 -p tcp -n cl0-frdev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:32 INFO::DEVICE_INTERFACE::Processing ACL-DNS *from* ace rule. 2019-07-15 20:28:32 DEBUG::GENERAL::Starting DNS lookup 2019-07-15 20:28:32 DEBUG::GENERAL::us.dlink.com 2019-07-15 20:28:32 DEBUG::GENERAL::192.168.4.7 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d wan -i 192.168.20.222 -a any -j 192.168.4.7 -b 80:80 -p tcp -n cl1-frdev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:32 INFO::DEVICE_INTERFACE::Processing ACL-DNS *from* ace rule. 2019-07-15 20:28:32 DEBUG::GENERAL::Starting DNS lookup 2019-07-15 20:28:32 DEBUG::GENERAL::www.trytechy.com 2019-07-15 20:28:32 DEBUG::GENERAL::99.84.216.69 2019-07-15 20:28:32 DEBUG::GENERAL::99.84.216.65 2019-07-15 20:28:32 DEBUG::GENERAL::99.84.216.79 2019-07-15 20:28:32 DEBUG::GENERAL::99.84.216.27 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d wan -i 192.168.20.222 -a any -j 99.84.216.69 -b 443:443 -p tcp -n cl2-frdev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d wan -i 192.168.20.222 -a any -j 99.84.216.65 -b 443:443 -p

Test Case Field	Description
	tcp -n cl2-frdev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d wan -i 192.168.20.222 -a any -j 99.84.216.79 -b 443:443 -p tcp -n cl2-frdev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d wan -i 192.168.20.222 -a any -j 99.84.216.27 -b 443:443 -p tcp -n cl2-frdev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:32 WARNING::DEVICE_INTERFACE::Processing CONTROLLER *from* ace rule. 2019-07-15 20:28:32 DEBUG::GENERAL::Starting DNS lookup 2019-07-15 20:28:32 DEBUG::GENERAL::www.google.com 2019-07-15 20:28:32 DEBUG::GENERAL::172.217.164.132 2019-07-15 20:28:32 DEBUG::GENERAL::0.0.0.0 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d wan -i 192.168.20.222 -a any -j 172.217.164.132 -b 443:443 -p tcp -n ent0-frdev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d wan -i 192.168.20.222 -a any -j 0.0.0.0 -b 443:443 -p tcp -n ent0-frdev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:32 WARNING::DEVICE_INTERFACE::Processing MY_CONTROLLER *from* ace rule. 2019-07-15 20:28:32 DEBUG::GENERAL::Starting DNS lookup 2019-07-15 20:28:32 DEBUG::GENERAL::yikes.example.com 2019-07-15 20:28:32 DEBUG::GENERAL::192.168.20.101 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d wan -i 192.168.20.222 -a any -j 192.168.20.101 -b any -p all -n myctl0-frdev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:32 INFO::DEVICE_INTERFACE::Processing LOCAL_NETWORK *to* ace rule. 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d lan -i 192.168.20.222 -a any -j any -b any -p tcp -n loc0-frdev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:32 INFO::DEVICE_INTERFACE::Processing MANUFACTURER *from* ace rule. 2019-07-15 20:28:32

Test Case Field	Description
	DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d lan -i 192.168.20.222 -a any -e www.gmail.com-SMTD -b 80:80 -p tcp -n man0-frdev-SM -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:32 INFO::DEVICE_INTERFACE::Processing SAME_MANUFACTURER *from* THING ace rule. 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d lan -i 192.168.20.222 -a any -e mudfiles.nist.getyikes.com-SMTD -b any -p udp -n myman0-frdev-SM -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:32 INFO::DEVICE_INTERFACE::Successfully installed fromAccess rule. 2019-07-15 20:28:32 INFO::DEVICE_INTERFACE::Processing DNS-ACL *to* ace rule. 2019-07-15 20:28:32 DEBUG::GENERAL::Starting DNS lookup 2019-07-15 20:28:32 DEBUG::GENERAL::www.osmud.org 2019-07-15 20:28:32 DEBUG::GENERAL::198.71.233.87 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s wan -d lan -i 198.71.233.87 -a any -j 192.168.20.222 -b 443:443 -p tcp -n cl0-todev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:32 INFO::DEVICE_INTERFACE::Processing DNS-ACL *to* ace rule. 2019-07-15 20:28:32 DEBUG::GENERAL::Starting DNS lookup 2019-07-15 20:28:32 DEBUG::GENERAL::us.dlink.com 2019-07-15 20:28:32 DEBUG::GENERAL::192.168.4.7 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s wan -d lan -i 192.168.4.7 -a any -j 192.168.20.222 -b 80:80 -p tcp -n cl1-todev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:32 INFO::DEVICE_INTERFACE::Processing DNS-ACL *to* ace rule. 2019-07-15 20:28:32 DEBUG::GENERAL::Starting DNS lookup 2019-07-15 20:28:32 DEBUG::GENERAL::www.trytechy.com 2019-07-15 20:28:33 DEBUG::GENERAL::99.84.216.27 2019-07-15 20:28:33 DEBUG::GENERAL::99.84.216.79 2019-07-15 20:28:33 DEBUG::GENERAL::99.84.216.65 2019-07-15 20:28:33 DEBUG::GENERAL::99.84.216.69 2019-07-15 20:28:33 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s wan -d lan -i 99.84.216.27 -a any -j 192.168.20.222 -b 443:443 -p tcp -n cl2-todev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:33

Test Case Field	Description
	DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s wan -d lan -i 99.84.216.79 -a any -j 192.168.20.222 -b 443:443 -p tcp -n cl2-todev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:33 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s wan -d lan -i 99.84.216.65 -a any -j 192.168.20.222 -b 443:443 -p tcp -n cl2-todev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:33 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s wan -d lan -i 99.84.216.69 -a any -j 192.168.20.222 -b 443:443 -p tcp -n cl2-todev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:33 WARNING::DEVICE_INTERFACE::Processing CONTROLLER *to* ace rule. 2019-07-15 20:28:33 DEBUG::GENERAL::Starting DNS lookup 2019-07-15 20:28:33 DEBUG::GENERAL::www.google.com 2019-07-15 20:28:33 DEBUG::GENERAL::172.217.164.132 2019-07-15 20:28:33 DEBUG::GENERAL::0.0.0.0 2019-07-15 20:28:33 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s wan -d lan -i 172.217.164.132 -a any -j 192.168.20.222 -b 443:443 -p tcp -n ent0-todev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:33 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s wan -d lan -i 0.0.0.0 -a any -j 192.168.20.222 -b 443:443 -p tcp -n ent0-todev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:33 WARNING::DEVICE_INTERFACE::Processing MY_CONTROLLER *to* ace rule. 2019-07-15 20:28:33 DEBUG::GENERAL::Starting DNS lookup 2019-07-15 20:28:33 DEBUG::GENERAL::yikes.example.com 2019-07-15 20:28:33 DEBUG::GENERAL::192.168.20.101 2019-07-15 20:28:33 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s wan -d lan -i 192.168.20.101 -a any -j 192.168.20.222 -b any -p all -n myctl0-todev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:33 INFO::DEVICE_INTERFACE::Processing LOCAL_NETWORK *to* ace rule. 2019-07-15 20:28:33 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d lan -i any -a any -j 192.168.20.222 -b any -p tcp -n loc0-todev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:33 INFO::DEVICE_INTERFACE::Processing (TBD)

Test Case Field	Description
	MANUFACTURER *to* ace rule. 2019-07-15 20:28:33 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d lan -j 192.168.20.222 -a any -e www.gmail.com-SMFD -b 80:80 -p tcp -n man0-todev-SM -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:33 INFO::DEVICE_INTERFACE::Processing SAME_MANUFACTURER *to* THING ace rule. 2019-07-15 20:28:33 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d lan -j 192.168.20.222 -a any -e mudfiles.nist.getyikes.com-SMFD -b any -p udp -n myman0-todev-SM -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:33 INFO::DEVICE_INTERFACE::Successfully installed toAccess rule. 2019-07-15 20:28:33 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d wan -i 192.168.20.222 -a any -j any -b any -p all -n REJECT-ALL -t REJECT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:33 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d lan -i 192.168.20.222 -a any -j any -b any -p all -n REJECT-ALL-LOCAL-FROM -t REJECT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:33 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d lan -i any -a any -j 192.168.20.222 -b any -p all -n REJECT-ALL-LOCAL-TO -t REJECT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 2019-07-15 20:28:33 DEBUG::GENERAL::/etc/osmud/commit_ip_fw_rules.sh -d /etc/osmud/state/ipSets -t /tmp/osmud 2019-07-15 20:28:33 DEBUG::GENERAL::**Success returned from for transaction** **Procedure 7:** **Router/PEP:** config rule option enabled '1' option name 'mud_192.168.20.222_main-pi-Build2_cl0-frdev' option target ACCEPT option src lan option dest wan option proto tcp option family ipv4

Test Case Field	Description
	`` option src_ip 192.168.20.222 option dest_ip 198.71.233.87 option dest_port 443:443 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi-Build2_cl0-todev' option target ACCEPT option src wan option dest lan option proto tcp option family ipv4 option src_ip 198.71.233.87 option dest_ip 192.168.20.222 option dest_port 443:443 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi-Build2_cl1-frdev' option target ACCEPT option src lan option dest wan option proto tcp option family ipv4 option src_ip 192.168.20.222 option dest_ip 192.168.4.7 option dest_port 80:80 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi-Build2_cl1-todev' option target ACCEPT option src wan option dest lan option proto tcp option family ipv4 option src_ip 192.168.4.7 option dest_ip 192.168.20.222 option dest_port 80:80 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi-Build2_cl2-frdev' option target ACCEPT option src lan```

Test Case Field	Description
	`` option dest wan option proto tcp option family ipv4 option src_ip 192.168.20.222 option dest_ip 99.84.216.69 option dest_port 443:443 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_cl2-frdev' option target ACCEPT option src lan option dest wan option proto tcp option family ipv4 option src_ip 192.168.20.222 option dest_ip 99.84.216.65 option dest_port 443:443 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_cl2-frdev' option target ACCEPT option src lan option dest wan option proto tcp option family ipv4 option src_ip 192.168.20.222 option dest_ip 99.84.216.79 option dest_port 443:443 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_cl2-frdev' option target ACCEPT option src lan option dest wan option proto tcp option family ipv4 option src_ip 192.168.20.222 option dest_ip 99.84.216.27 option dest_port 443:443 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- ```

Test Case Field	Description
	``Build2_cl2-todev' option target ACCEPT option src wan option dest lan option proto tcp option family ipv4 option src_ip 99.84.216.27 option dest_ip 192.168.20.222 option dest_port 443:443 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_cl2-todev' option target ACCEPT option src wan option dest lan option proto tcp option family ipv4 option src_ip 99.84.216.79 option dest_ip 192.168.20.222 option dest_port 443:443 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_cl2-todev' option target ACCEPT option src wan option dest lan option proto tcp option family ipv4 option src_ip 99.84.216.65 option dest_ip 192.168.20.222 option dest_port 443:443 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_cl2-todev' option target ACCEPT option src wan option dest lan option proto tcp option family ipv4 option src_ip 99.84.216.69 option dest_ip 192.168.20.222 option dest_port 443:443```

Test Case Field	Description
	`` config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_ent0-frdev' option target ACCEPT option src lan option dest wan option proto tcp option family ipv4 option src_ip 192.168.20.222 option dest_ip 172.217.164.132 option dest_port 443:443 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_ent0-frdev' option target ACCEPT option src lan option dest wan option proto tcp option family ipv4 option src_ip 192.168.20.222 option dest_ip 0.0.0.0 option dest_port 443:443 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_ent0-todev' option target ACCEPT option src wan option dest lan option proto tcp option family ipv4 option src_ip 172.217.164.132 option dest_ip 192.168.20.222 option dest_port 443:443 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_ent0-todev' option target ACCEPT option src wan option dest lan option proto tcp option family ipv4 option src_ip 0.0.0.0 ```

Test Case Field	Description
	`option dest_ip    192.168.20.222` `option dest_port  443:443` `config rule` `option enabled   '1'` `option name      'mud_192.168.20.222_main-pi-Build2_loc0-frdev'` `option target    ACCEPT` `option src       lan` `option dest      lan` `option proto     tcp` `option family    ipv4` `option src_ip     192.168.20.222` `config rule` `option enabled   '1'` `option name      'mud_192.168.20.222_main-pi-Build2_loc0-todev'` `option target    ACCEPT` `option src       lan` `option dest      lan` `option proto     tcp` `option family    ipv4` `option src_ip     any` `option dest_ip    192.168.20.222` `config rule` `option enabled   '1'` `option name      'mud_192.168.20.222_main-pi-Build2_man0-frdev-SM'` `option target    ACCEPT` `option src       lan` `option dest      lan` `option proto     tcp` `option family    ipv4` `option src_ip     192.168.20.222` `option ipset      www_gmail_com-SMTD` `option dest_port  80:80` `config rule` `option enabled   '1'` `option name      'mud_192.168.20.222_main-pi-Build2_man0-todev-SM'` `option target    ACCEPT` `option src       lan` `option dest      lan` `option proto     tcp` `option family    ipv4` `option ipset      www_gmail_com-SMFD`

Test Case Field	Description
	<pre> option dest_ip 192.168.20.222 option dest_port 80:80 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_myctl0-frdev' option target ACCEPT option src lan option dest wan option proto all option family ipv4 option src_ip 192.168.20.222 option dest_ip 192.168.20.101 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_myctl0-todev' option target ACCEPT option src wan option dest lan option proto all option family ipv4 option src_ip 192.168.20.101 option dest_ip 192.168.20.222 config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_myman0-frdev-SM' option target ACCEPT option src lan option dest lan option proto udp option family ipv4 option src_ip 192.168.20.222 option ipset mudfiles_nist_getyikes_com-SMTD config rule option enabled '1' option name 'mud_192.168.20.222_main-pi- Build2_myman0-todev-SM' option target ACCEPT option src lan option dest lan option proto udp option family ipv4 option ipset mudfiles_nist_getyikes_com-SMFD</pre>

Test Case Field	Description
	`option dest_ip    192.168.20.222` `config rule` `option enabled   '1'` `option name      'mud_192.168.20.222_main-pi-Build2_REJECT-ALL-LOCAL-FROM'` `option target    REJECT` `option src       lan` `option dest      lan` `option proto     all` `option family    ipv4` `option src_ip     192.168.20.222` `config rule` `option enabled   '1'` `option name      'mud_192.168.20.222_main-pi-Build2_REJECT-ALL-LOCAL-TO'` `option target    REJECT` `option src       lan` `option dest      lan` `option proto     all` `option family    ipv4` `option src_ip     any` `option dest_ip    192.168.20.222` `config rule` `option enabled   '1'` `option name      'mud_192.168.20.222_main-pi-Build2_REJECT-ALL'` `option target    REJECT` `option src       lan` `option dest      wan` `option proto     all` `option family    ipv4` `option src_ip     192.168.20.222` `# OSMUD end`
Overall Results	Pass

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

3.1.2.2 Test Case IoT-2-v4

Table 3-3: Test Case IoT-2-v4

Test Case Field	Description
Parent Requirement	(CR-3) The IoT DDoS example implementation shall include a MUD manager that can request a MUD file and signature from a MUD file server.
Testable Requirement	(CR-3.b) The MUD manager shall use the GET method (RFC 7231) to request MUD and signature files (per RFC 7230) from the MUD file server, but it cannot validate the MUD file server's TLS certificate by using the rules in RFC 2818. (CR-3.b.1) The MUD manager shall drop the connection to the MUD file server. (CR-3.b.2) The MUD manager shall send locally defined policy to the router or switch that handles whether to allow or block traffic to and from the MUD-enabled IoT device.
Description	Shows that if a MUD manager cannot validate the TLS certificate of a MUD file server when trying to retrieve the MUD file for a specific IoT device, the MUD manager will drop the connection to the MUD file server and configure the router/switch according to locally defined policy regarding whether to allow or block traffic to the IoT device in question
Associated Test Case(s)	IoT-11-v4 (for the v6 version of this test, IoT-11-v6)
Associated Cybersecurity Framework Subcategory(ies)	PR.AC-7
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*Yikesmain.json, yikesmantest.json*
Preconditions	1. All devices have been configured to use IPv4. 2. This MUD file is not currently cached at the MUD manager. 3. The MUD file server that is hosting the MUD file of the device under test does not have a valid TLS certificate. 4. Local policy has been defined to ensure that if the MUD file for a device is located on a server with an invalid certificate, the

Test Case Field	Description
	router/switch will be configured to deny all communication to and from the device. 5. The MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings with respect to the IoT device being used in the test.
Procedure	Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. Power on the IoT device and connect it to the test network. This should set in motion the following series of steps, which should occur automatically: 1. The IoT device automatically emits a DHCPv4 message containing the device's MUD URL (IANA code 161). (Note that in the v6 version of this test, IPv6, DHCPv6, and IANA code 112 will be used.) 2. The DHCP server receives the DHCP message containing the IoT device's MUD URL. 3. The DHCP server offers an IP address lease to the newly connected IoT device. 4. The IoT device requests this IP address lease, which the DHCP server acknowledges. 5. The DHCP server sends the MUD URL to the MUD manager. 6. The MUD manager automatically contacts the MUD file server that is located by using the MUD URL, determines that it does not have a valid TLS certificate, and drops the connection to the MUD file server. 7. The MUD manager configures the router/switch that is closest to the IoT device according to locally defined policy, which in this case allows traffic to the IoT device in question.
Expected Results	The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to local policy for communication to/from the IoT device.

Test Case Field	Description
Actual Results	**Procedures 1–4:** `pi@main-pi-Build2:~$ sudo dhclient -v -i eth0` `sudo: unable to resolve host main-pi-Build2: Connection re-fused` `Internet Systems Consortium DHCP Client 4.3.5` `Copyright 2004-2016 Internet Systems Consortium.` `All rights reserved.` `For info, please visit https://www.isc.org/software/dhcp/` `RTNETLINK answers: Operation not possible due to RF-kill` `Listening on LPF/wlan0/b8:27:eb:be:39:de` `Sending on   LPF/wlan0/b8:27:eb:be:39:de` `Listening on LPF/eth0/b8:27:eb:eb:6c:8b` `Sending on   LPF/eth0/b8:27:eb:eb:6c:8b` `Sending on   Socket/fallback` **`DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 4`** **`DHCPREQUEST of 192.168.20.224 on eth0 to 255.255.255.255 port 67`** **`DHCPOFFER of 192.168.20.224 from 192.168.20.1`** **`DHCPACK of 192.168.20.224 from 192.168.20.1`** `Too few arguments.` `Too few arguments.` **`bound to 192.168.20.224`** `-- renewal in 1800 seconds.` --- **Procedure 5:** **dhcpmasq.txt** `2019-07-15T20:27:57Z|OLD|Wired|DHCP|-|MUD|-|-|ba:47:a1:7d:60:44|192.168.20.148||` `2019-07-15T20:28:01Z|OLD|NIST 5|DHCP|-|MUD|-|-|18:b4:30:50:98:38|192.168.20.203||` `2019-07-15T20:28:08Z|OLD|NIST 2.4|DHCP|-|MUD|-|-|d0:73:d5:28:08:2a|192.168.20.202||` `2019-07-15T20:28:11Z|OLD|Wired|DHCP|-|MUD|-|-|b8:27:eb:95:55:fe|192.168.20.232|raspberrypi|` **`2019-07-15T20:28:31Z|NEW|Wired|DHCP|1,28,2,3,15,6,119,12,44,47,26,121,42|MUD|`**https://mudfiles.nist.getyikes.com/yikesmain.json|-|b8:27:eb:eb:6c:8b|192.168.20.224|main-pi-Build2| `2019-07-15T20:28:42Z|NEW|NIST 5|DHCP|1,28,2,121,15,6,12,40,41,42,26,119,3,121,249,33,252,42|MUD|-|-|80:00:0b:ef:81:70|192.168.20.238||`

Test Case Field	Description
	Procedure 6: **MUD Manager:** `2019-06-18 13:59:50 INFO::GENERAL::NEW Device Action: IP: 192.168.20.224, MAC: b8:27:eb:eb:6c:8b` `2019-06-18 13:59:50 ERROR::COMMUNICATION::curl_easy_getinfo(curl, CURLINFO_RESPONSE_CODE -- http-code: 0` `2019-06-18 13:59:50 WARNING::COMMUNICATION::`**`Comm error with a mud-file-server. Retrying transaction`**`...` `2019-06-18 13:59:50 INFO::GENERAL::NEW Device Action: IP: 192.168.20.224, MAC: b8:27:eb:eb:6c:8b` `2019-06-18 13:59:51 ERROR::COMMUNICATION::curl_easy_getinfo(curl, CURLINFO_RESPONSE_CODE -- http-code: 0` `2019-06-18 13:59:51 ERROR::GENERAL::Comm error with mud-file-server. Aborting transaction after second attempt and quarantine device.` **Procedure 7:** **Router/PEP:** `# OSMUD start` `#` `# DO NOT EDIT THESE LINES. OSMUD WILL REPLACE WITH ITS CON-FIGURATION` `#` `config ipset` `    option enabled 1` `    option name mudfiles_nist_getyikes_com-SMTD` `    option match dest_ip` `    option storage hash` `    option family ipv4` `    option external mudfiles_nist_getyikes_com-SM` `config ipset` `    option enabled 1` `    option name mudfiles_nist_getyikes_com-SMFD` `    option match src_ip` `    option storage hash` `    option family ipv4` `    option external mudfiles_nist_getyikes_com-SM`

Test Case Field	Description
	config ipset option enabled 1 option name mudfileserver-SMTD option match dest_ip option storage hash option family ipv4 option external mudfileserver-SM config ipset option enabled 1 option name mudfileserver-SMFD option match src_ip option storage hash option family ipv4 option external mudfileserver-SM config ipset option enabled 1 option name www_facebook_com-SMTD option match dest_ip option storage hash option family ipv4 option external www_facebook_com-SM config ipset option enabled 1 option name www_facebook_com-SMFD option match src_ip option storage hash option family ipv4 option external www_facebook_com-SM config ipset option enabled 1 option name www_gmail_com-SMTD option match dest_ip option storage hash option family ipv4 option external www_gmail_com-SM config ipset option enabled 1 option name www_gmail_com-SMFD option match src_ip option storage hash option family ipv4 option external www_gmail_com-SM config rule

Test Case Field	Description
	`` option enabled '1' option name 'mud_192.168.20.197_same-manufac- ture-pi_cl0-frdev' option target ACCEPT option src lan option dest wan option proto tcp option family ipv4 option src_ip 192.168.20.197 option dest_ip 198.71.233.87 config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac- ture-pi_cl0-todev' option target ACCEPT option src wan option dest lan option proto tcp option family ipv4 option src_ip 198.71.233.87 option dest_ip 192.168.20.197 config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac- ture-pi_myman0-frdev-SM' option target ACCEPT option src lan option dest lan option proto tcp option family ipv4 option src_ip 192.168.20.197 option ipset www_facebook_com-SMTD option dest_port 80:80 config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac- ture-pi_myman0-todev-SM' option target ACCEPT option src lan option dest lan option proto tcp option family ipv4 option ipset www_facebook_com-SMFD option dest_ip 192.168.20.197 option dest_port 80:80 ```

Test Case Field	Description
	config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac-ture-pi_REJECT-ALL-LOCAL-FROM' option target REJECT option src lan option dest lan option proto all option family ipv4 option src_ip 192.168.20.197 config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac-ture-pi_REJECT-ALL-LOCAL-TO' option target REJECT option src lan option dest lan option proto all option family ipv4 option src_ip any option dest_ip 192.168.20.197 config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac-ture-pi_REJECT-ALL' option target REJECT option src lan option dest wan option proto all option family ipv4 option src_ip 192.168.20.197 # OSMUD end
Overall Results	Pass

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

3.1.2.3 Test Case IoT-3-v4

Table 3-4: Test Case IoT-3-v4

Test Case Field	Description
Parent Requirement	(CR-4) The IoT DDoS example implementation shall include a MUD file server that can serve a MUD file and signature to the MUD manager.
Testable Requirement	(CR-4.b) The MUD file server shall serve the file and signature to the MUD manager, and the MUD manager shall check to determine whether the certificate used to sign the MUD file was valid at the time of signing, i.e., the certificate had already expired when it was used to sign the MUD file. (CR-4.b.1) The MUD manager shall cease to process the MUD file. (CR-4.b.2) The MUD manager shall send locally defined policy to the router or switch that handles whether to allow or block traffic to and from the MUD-enabled IoT device.
Description	Shows that if a MUD file server serves a MUD file with a signature that was created with an expired certificate, the MUD manager will cease processing the MUD file
Associated Test Case(s)	IoT-11-v4 (for the v6 version of this test, IoT-11-v6)
Associated Cybersecurity Framework Subcategory(ies)	PR.DS-6
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*ExpiredCertTest.json*
Preconditions	1. This MUD file is not currently cached at the MUD manager. 2. The IoT device's MUD file is being hosted on a MUD file server that has a valid TLS certificate, but the MUD file signature was signed by a certificate that had already expired at the time of signature. 3. Local policy has been defined to ensure that if the MUD file for a device has a signature that was signed by a certificate that had already expired at the time of signature, the device's MUD PEP router/switch will be configured to deny all communication to/from the device.

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

Test Case Field	Description
	from the IoT device. The expected configuration should resemble the following. Expecting a show access session without a MUD file as seen below: `# OSMUD start` `#` `# DO NOT EDIT THESE LINES. OSMUD WILL REPLACE WITH ITS CON-FIGURATION` `#` `config ipset` `    option enabled 1` `    option name mudfiles_nist_getyikes_com-SMTD` `    option match dest_ip` `    option storage hash` `    option family ipv4` `    option external mudfiles_nist_getyikes_com-SM` `config ipset` `    option enabled 1` `    option name mudfiles_nist_getyikes_com-SMFD` `    option match src_ip` `    option storage hash` `    option family ipv4` `    option external mudfiles_nist_getyikes_com-SM` `config ipset` `    option enabled 1` `    option name mudfileserver-SMTD` `    option match dest_ip` `    option storage hash` `    option family ipv4` `    option external mudfileserver-SM` `config ipset` `    option enabled 1` `    option name mudfileserver-SMFD` `    option match src_ip` `    option storage hash` `    option family ipv4` `    option external mudfileserver-SM` `config ipset` `    option enabled 1` `    option name www_facebook_com-SMTD` `    option match dest_ip` `    option storage hash` `    option family ipv4`

Test Case Field	Description
	`    option external www_facebook_com-SM` `config ipset` `    option enabled 1` `    option name www_facebook_com-SMFD` `    option match src_ip` `    option storage hash` `    option family ipv4` `    option external www_facebook_com-SM` `config ipset` `    option enabled 1` `    option name www_gmail_com-SMTD` `    option match dest_ip` `    option storage hash` `    option family ipv4` `    option external www_gmail_com-SM` `config ipset` `    option enabled 1` `    option name www_gmail_com-SMFD` `    option match src_ip` `    option storage hash` `    option family ipv4` `    option external www_gmail_com-SM` `# OSMUD end`
Actual Results	**Procedures 1–4:** `pi@main-pi-Build2:~$ sudo dhclient -v -i eth0` `sudo: unable to resolve host main-pi-Build2: Connection re-fused` `Internet Systems Consortium DHCP Client 4.3.5` `Copyright 2004-2016 Internet Systems Consortium.` `All rights reserved.` `For info, please visit https://www.isc.org/software/dhcp/` `RTNETLINK answers: Operation not possible due to RF-kill` `Listening on LPF/wlan0/b8:27:eb:be:39:de` `Sending on   LPF/wlan0/b8:27:eb:be:39:de` `Listening on LPF/eth0/b8:27:eb:eb:6c:8b` `Sending on   LPF/eth0/b8:27:eb:eb:6c:8b` `Sending on   Socket/fallback` **`DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 4`**

Test Case Field	Description
	DHCPREQUEST of 192.168.20.226 on eth0 to 255.255.255.255 port 67 **DHCPOFFER of 192.168.20.226 from 192.168.20.1** **DHCPACK of 192.168.20.226 from 192.168.20.1** Too few arguments. Too few arguments. **bound to 192.168.20.226** -- renewal in 1800 seconds. **Procedure 5:** **dhcpmasq.txt** 2019-07-11T18:03:00Z|OLD|Wired|DHCP|-|MUD|-|-|ba:47:a1:7d:41:bb|192.168.20.160|| 2019-07-11T18:03:05Z|OLD|NIST 5|DHCP|-|MUD|-|-|18:b4:30:50:E2:01|192.168.20.143|| 2019-07-11T18:03:12Z|DEL|Wired|DHCP|-|MUD|-||b8:27:eb:95:55:fe|192.168.20.233|raspberrypi| 2019-07-**11T18:03:25Z|NEW|Wired|DHCP|1,28,2,3,15,6,119,12,44,47,26,121,42|MUD|https://mudfiles.nist.getyikes.com/ExpiredCert-Test.json|-|b8:27:eb:eb:6c:8b|192.168.20.226|main-pi-Build2|** **Procedure 7:** **MUD Manager:** 2019-07-11 18:03:26 DEBUG::GENERAL::**2019-07-11T18:03:25Z|NEW|Wired|DHCP|1,28,2,3,15,6,119,12,44,47,26,121,42|MUD|https://mudfiles.nist.getyikes.com/ExpiredCert-Test.json|-|b8:27:eb:eb:6c:8b|192.168.20.226|main-pi-Build2|** 2019-07-11 18:03:26 DEBUG::GENERAL::Executing on dhcpmasq info 2019-07-11 18:03:26 INFO::GENERAL::NEW Device Action: IP: 192.168.20.226, MAC: b8:27:eb:eb:6c:8b 2019-07-11 18:03:26 DEBUG::COMMUNICATION::curl_easy_per-form() doing it now.... 2019-07-11 18:03:26 DEBUG::COMMUNICATION::https://mud-files.nist.getyikes.com/ExpiredCertTest.json 2019-07-11 18:03:26 DEBUG::COMMUNICATION::Found HTTPS 2019-07-11 18:03:26 DEBUG::COMMUNICATION::in write data 2019-07-11 18:03:26 DEBUG::COMMUNICATION::curl_easy_per-form() success 2019-07-11 18:03:26 DEBUG::COMMUNICATION::MUD File Server returned success state. 2019-07-11 18:03:26 DEBUG::COMMUNICATION::curl_easy_per-form() doing it now.... 2019-07-11 18:03:26 DEBUG::COMMUNICATION::https://mud-files.nist.getyikes.com/ExpiredCertTest.p7s 2019-07-11 18:03:26 DEBUG::COMMUNICATION::Found HTTPS 2019-07-11 18:03:27 DEBUG::COMMUNICATION::in write data

Test Case Field	Description
	`2019-07-11 18:03:27 DEBUG::COMMUNICATION::curl_easy_per-form() success` `2019-07-11 18:03:27 DEBUG::COMMUNICATION::MUD File Server returned success state.` `2019-07-11 18:03:27 DEBUG::MUD_FILE_OPERATIONS::IN ****NEW**** MUD and SIG FILE RETRIEVED!!!` `2019-07-11 18:03:27 DEBUG::GENERAL::IN ****NEW**** vali-dateMudFileWithSig()` `2019-07-11 18:03:27 DEBUG::GENERAL::openssl cms -verify -in /etc/osmud/state/mudfiles/ExpiredCertTest.p7s -inform DER -content /etc/osmud/state/mudfiles/ExpiredCertTest.json -pur-pose any > /dev/null` `2019-07-11 18:03:27 ERROR::DEVICE_INTERFACE::openssl cms -verify -in /etc/osmud/state/mudfiles/ExpiredCertTest.p7s -inform DER -content /etc/osmud/state/mudfiles/ExpiredCert-Test.json -purpose any > /dev/null` **`2019-07-11 18:03:27 ERROR::MUD_FILE_OPERATIONS::Could not validate the MUD File signature using openssl cms verify. Abort mud file processing and quarantine device.`** `2019-07-11 18:03:27 DEBUG::GENERAL::/etc/osmud/cre-ate_ip_fw_rule.sh -s lan -d wan -i 192.168.20.226 -a any -j any -b any -p all -n REJECT-ALL -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.226` `2019-07-11 18:03:27 DEBUG::GENERAL::/etc/osmud/cre-ate_ip_fw_rule.sh -s lan -d lan -i 192.168.20.226 -a any -j any -b any -p all -n REJECT-ALL-LOCAL-FROM -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.226` `2019-07-11 18:03:27 DEBUG::GENERAL::/etc/osmud/cre-ate_ip_fw_rule.sh -s lan -d lan -i any -a any -j 192.168.20.226 -b any -p all -n REJECT-ALL-LOCAL-TO -t AC-CEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.226` **Router/PEP:** `# OSMUD start` `#` `# DO NOT EDIT THESE LINES. OSMUD WILL REPLACE WITH ITS CON-FIGURATION` `#` `config ipset` `    option enabled 1` `    option name mudfiles_nist_getyikes_com-SMTD` `    option match dest_ip` `    option storage hash` `    option family ipv4` `    option external mudfiles_nist_getyikes_com-SM` `config ipset`

Test Case Field	Description
	`` option enabled 1 option name mudfiles_nist_getyikes_com-SMFD option match src_ip option storage hash option family ipv4 option external mudfiles_nist_getyikes_com-SM config ipset option enabled 1 option name mudfileserver-SMTD option match dest_ip option storage hash option family ipv4 option external mudfileserver-SM config ipset option enabled 1 option name mudfileserver-SMFD option match src_ip option storage hash option family ipv4 option external mudfileserver-SM config ipset option enabled 1 option name www_facebook_com-SMTD option match dest_ip option storage hash option family ipv4 option external www_facebook_com-SM config ipset option enabled 1 option name www_facebook_com-SMFD option match src_ip option storage hash option family ipv4 option external www_facebook_com-SM config ipset option enabled 1 option name www_gmail_com-SMTD option match dest_ip option storage hash option family ipv4 option external www_gmail_com-SM config ipset option enabled 1 ```

Test Case Field	Description
	`` option name www_gmail_com-SMFD option match src_ip option storage hash option family ipv4 option external www_gmail_com-SM config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac- ture-pi_cl0-frdev' option target ACCEPT option src lan option dest wan option proto tcp option family ipv4 option src_ip 192.168.20.197 option dest_ip 198.71.233.87 config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac- ture-pi_cl0-todev' option target ACCEPT option src wan option dest lan option proto tcp option family ipv4 option src_ip 198.71.233.87 option dest_ip 192.168.20.197 config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac- ture-pi_myman0-frdev-SM' option target ACCEPT option src lan option dest lan option proto tcp option family ipv4 option src_ip 192.168.20.197 option ipset www_facebook_com-SMTD option dest_port 80:80 config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac- ture-pi_myman0-todev-SM' option target ACCEPT option src lan ```

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td></td><td><pre>
        option dest   lan
        option proto   tcp
        option family   ipv4
        option ipset   www_facebook_com-SMFD
        option dest_ip   192.168.20.197
        option dest_port 80:80

config rule
        option enabled   '1'
        option name   'mud_192.168.20.197_same-manufac-
ture-pi_REJECT-ALL-LOCAL-FROM'
        option target   REJECT
        option src   lan
        option dest   lan
        option proto   all
        option family   ipv4
        option src_ip   192.168.20.197

config rule
        option enabled   '1'
        option name   'mud_192.168.20.197_same-manufac-
ture-pi_REJECT-ALL-LOCAL-TO'
        option target   REJECT
        option src   lan
        option dest   lan
        option proto   all
        option family   ipv4
        option src_ip   any
        option dest_ip   192.168.20.197

config rule
        option enabled   '1'
        option name   'mud_192.168.20.197_same-manufac-
ture-pi_REJECT-ALL'
        option target   REJECT
        option src   lan
        option dest   wan
        option proto   all
        option family   ipv4
        option src_ip   192.168.20.197
OSMUD end
</pre></td></tr>
<tr><td>Overall Results</td><td>Pass</td></tr>
</table>

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

316 ### 3.1.2.4 Test Case IoT-4-v4

317 **Table 3-5: Test Case IoT-4-v4**

Test Case Field	Description
Parent Requirement	(CR-5) The IoT DDoS example implementation shall include a MUD manager that can translate local network configurations based on the MUD file.
Testable Requirement	(CR-5.b) The MUD manager shall attempt to validate the signature of the MUD file, but the signature validation fails (even though the certificate that had been used to create the signature had not been expired at the time of signing, i.e., the signature is invalid for a different reason). (CR-5.b.1) The MUD manager shall cease processing the MUD file. (CR-5.b.2) The MUD manager shall send locally defined policy to the router or switch that handles whether to allow or block traffic to and from the MUD-enabled IoT device.
Description	Shows that if the MUD manager determines that the signature on the MUD file it receives from the MUD file server is invalid, it will cease processing the MUD file and configure the router/switch according to locally defined policy regarding whether to allow or block traffic to the IoT device in question
Associated Test Case(s)	IoT-11-v4 (for the v6 version of this test, IoT-11-v6)
Associated Cybersecurity Framework Subcategory(ies)	PR.DS-6
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*cr-5b.json*
Preconditions	1. This MUD file is not currently cached at the MUD manager. 2. The MUD file that is served from the MUD file server to the MUD manager has a signature that is invalid, even though it was signed by a certificate that had not expired at the time of signing.

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

Test Case Field	Description
	the IoT device. The expected configuration should resemble the following: Expecting a show access session without a MUD file as seen below: `# OSMUD start` `#` `# DO NOT EDIT THESE LINES. OSMUD WILL REPLACE WITH ITS CON-FIGURATION` `#` `config ipset` `    option enabled 1` `    option name mudfiles_nist_getyikes_com-SMTD` `    option match dest_ip` `    option storage hash` `    option family ipv4` `    option external mudfiles_nist_getyikes_com-SM` `config ipset` `    option enabled 1` `    option name mudfiles_nist_getyikes_com-SMFD` `    option match src_ip` `    option storage hash` `    option family ipv4` `    option external mudfiles_nist_getyikes_com-SM` `config ipset` `    option enabled 1` `    option name mudfileserver-SMTD` `    option match dest_ip` `    option storage hash` `    option family ipv4` `    option external mudfileserver-SM` `config ipset` `    option enabled 1` `    option name mudfileserver-SMFD` `    option match src_ip` `    option storage hash` `    option family ipv4` `    option external mudfileserver-SM` `config ipset` `    option enabled 1` `    option name www_facebook_com-SMTD` `    option match dest_ip` `    option storage hash`

Test Case Field	Description
	` option family ipv4` ` option external www_facebook_com-SM` `config ipset` ` option enabled 1` ` option name www_facebook_com-SMFD` ` option match src_ip` ` option storage hash` ` option family ipv4` ` option external www_facebook_com-SM` `config ipset` ` option enabled 1` ` option name www_gmail_com-SMTD` ` option match dest_ip` ` option storage hash` ` option family ipv4` ` option external www_gmail_com-SM` `config ipset` ` option enabled 1` ` option name www_gmail_com-SMFD` ` option match src_ip` ` option storage hash` ` option family ipv4` ` option external www_gmail_com-SM` `# OSMUD end`
Actual Results	**Procedures 1-5:** Excluded for sake of length. **Procedure 6:** **MUD MANAGER:** `2019-07-11 18:10:30 DEBUG::GENERAL::2019-07-`**`11T18:10:24Z|NEW|Wired|DHCP|1,28,2,3,15,6,119,12,44,47,26,121,42|MUD|https://mudfiles.nist.getyikes.com/cr-5b.json|-|b8:27:eb:eb:6c:8b|192.168.20.226|main-pi-Build2|`** `2019-07-11 18:10:30 DEBUG::GENERAL::Executing on dhcpmasq info` `2019-07-11 18:10:30 INFO::GENERAL::NEW Device Action: IP: 192.168.20.226, MAC: b8:27:eb:eb:6c:8b` `2019-07-11 18:10:30 DEBUG::COMMUNICATION::curl_easy_per-form() doing it now....`

Test Case Field	Description
	2019-07-11 18:10:30 DEBUG::COMMUNICATION::https://mud-files.nist.getyikes.com/cr-5b.json 2019-07-11 18:10:30 DEBUG::COMMUNICATION::Found HTTPS 2019-07-11 18:10:31 DEBUG::COMMUNICATION::in write data 2019-07-11 18:10:31 DEBUG::COMMUNICATION::curl_easy_per-form() success 2019-07-11 18:10:31 DEBUG::COMMUNICATION::MUD File Server returned success state. 2019-07-11 18:10:31 DEBUG::COMMUNICATION::curl_easy_per-form() doing it now.... 2019-07-11 18:10:31 DEBUG::COMMUNICATION::https://mud-files.nist.getyikes.com/cr-5b.p7s 2019-07-11 18:10:31 DEBUG::COMMUNICATION::Found HTTPS 2019-07-11 18:10:31 DEBUG::COMMUNICATION::in write data 2019-07-11 18:10:31 DEBUG::COMMUNICATION::curl_easy_per-form() success 2019-07-11 18:10:31 DEBUG::COMMUNICATION::MUD File Server returned success state. 2019-07-11 18:10:31 DEBUG::MUD_FILE_OPERATIONS::IN ****NEW**** MUD and SIG FILE RETRIEVED!!! 2019-07-11 18:10:31 DEBUG::GENERAL::IN ****NEW**** vali-dateMudFileWithSig() 2019-07-11 18:10:31 DEBUG::GENERAL::openssl cms -verify -in /etc/osmud/state/mudfiles/cr-5b.p7s -inform DER -content /etc/osmud/state/mudfiles/cr-5b.json -purpose any > /dev/null 2019-07-11 18:10:31 ERROR::DEVICE_INTERFACE::openssl cms -verify -in /etc/osmud/state/mudfiles/cr-5b.p7s -inform DER -content /etc/osmud/state/mudfiles/cr-5b.json -purpose any > /dev/null **2019-07-11 18:10:31 ERROR::MUD_FILE_OPERATIONS::Could not validate the MUD File signature using openssl cms verify. Abort mud file processing and quarantine device.** 2019-07-11 18:10:31 DEBUG::GENERAL::/etc/osmud/cre-ate_ip_fw_rule.sh -s lan -d wan -i 192.168.20.226 -a any -j any -b any -p all -n REJECT-ALL -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.226 2019-07-11 18:10:31 DEBUG::GENERAL::/etc/osmud/cre-ate_ip_fw_rule.sh -s lan -d lan -i 192.168.20.226 -a any -j any -b any -p all -n REJECT-ALL-LOCAL-FROM -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.226 2019-07-11 18:10:31 DEBUG::GENERAL::/etc/osmud/cre-ate_ip_fw_rule.sh -s lan -d lan -i any -a any -j

Test Case Field	Description
	192.168.20.226 -b any -p all -n REJECT-ALL-LOCAL-TO -t AC-CEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.226 **Procedure 7:** **Router/PEP:** # OSMUD start # # DO NOT EDIT THESE LINES. OSMUD WILL REPLACE WITH ITS CON-FIGURATION # config ipset option enabled 1 option name mudfiles_nist_getyikes_com-SMTD option match dest_ip option storage hash option family ipv4 option external mudfiles_nist_getyikes_com-SM config ipset option enabled 1 option name mudfiles_nist_getyikes_com-SMFD option match src_ip option storage hash option family ipv4 option external mudfiles_nist_getyikes_com-SM config ipset option enabled 1 option name mudfileserver-SMTD option match dest_ip option storage hash option family ipv4 option external mudfileserver-SM config ipset option enabled 1 option name mudfileserver-SMFD option match src_ip option storage hash option family ipv4 option external mudfileserver-SM config ipset option enabled 1 option name www_facebook_com-SMTD

Test Case Field	Description
	`` option match dest_ip option storage hash option family ipv4 option external www_facebook_com-SM config ipset option enabled 1 option name www_facebook_com-SMFD option match src_ip option storage hash option family ipv4 option external www_facebook_com-SM config ipset option enabled 1 option name www_gmail_com-SMTD option match dest_ip option storage hash option family ipv4 option external www_gmail_com-SM config ipset option enabled 1 option name www_gmail_com-SMFD option match src_ip option storage hash option family ipv4 option external www_gmail_com-SM config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac- ture-pi_cl0-frdev' option target ACCEPT option src lan option dest wan option proto tcp option family ipv4 option src_ip 192.168.20.197 option dest_ip 198.71.233.87 config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac- ture-pi_cl0-todev' option target ACCEPT option src wan option dest lan option proto tcp ```

Test Case Field	Description
	`` option family ipv4``` ``` option src_ip 198.71.233.87``` ``` option dest_ip 192.168.20.197``` ```config rule``` ``` option enabled '1'``` ``` option name 'mud_192.168.20.197_same-manufac-``` ```ture-pi_myman0-frdev-SM'``` ``` option target ACCEPT``` ``` option src lan``` ``` option dest lan``` ``` option proto tcp``` ``` option family ipv4``` ``` option src_ip 192.168.20.197``` ``` option ipset www_facebook_com-SMTD``` ``` option dest_port 80:80``` ```config rule``` ``` option enabled '1'``` ``` option name 'mud_192.168.20.197_same-manufac-``` ```ture-pi_myman0-todev-SM'``` ``` option target ACCEPT``` ``` option src lan``` ``` option dest lan``` ``` option proto tcp``` ``` option family ipv4``` ``` option ipset www_facebook_com-SMFD``` ``` option dest_ip 192.168.20.197``` ``` option dest_port 80:80``` ```config rule``` ``` option enabled '1'``` ``` option name 'mud_192.168.20.197_same-manufac-``` ```ture-pi_REJECT-ALL-LOCAL-FROM'``` ``` option target REJECT``` ``` option src lan``` ``` option dest lan``` ``` option proto all``` ``` option family ipv4``` ``` option src_ip 192.168.20.197``` ```config rule``` ``` option enabled '1'``` ``` option name 'mud_192.168.20.197_same-manufac-``` ```ture-pi_REJECT-ALL-LOCAL-TO'``` ``` option target REJECT``` ``` option src lan``` ``` option dest lan``` ``` option proto all```

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td></td><td><pre>
        option family   ipv4
        option src_ip   any
        option dest_ip   192.168.20.197

config rule
        option enabled   '1'
        option name   'mud_192.168.20.197_same-manufac-
ture-pi_REJECT-ALL'
        option target   REJECT
        option src   lan
        option dest   wan
        option proto   all
        option family   ipv4
        option src_ip   192.168.20.197
OSMUD end
</pre></td></tr>
<tr><td>Overall Results</td><td>Pass</td></tr>
</table>

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

3.1.2.5 Test Case IoT-5-v4

Table 3-6: Test Case IoT-5-v4

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

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

Test Case Field	Description
Associated Test Case(s)	IoT-1-v4 (for the v6 version of this test, IoT-1-v6)
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-3, PR.DS-5, PR.IP-1, PR.PT-3
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*Yikesmain.json*
Preconditions	Test IoT-1-v4 (or IoT-1-v6) has run successfully, meaning that the MUD PEP router/switch has been configured to enforce the following policies for the IoT device in question (as defined in the MUD file in Section 3.1.3): Note: Procedure steps with strikethrough are not tested due to network address translation (NAT). ~~a) Explicitly permit *https://yes-permit-from.com* to initiate communications with the IoT device.~~ b) Explicitly permit the IoT device to initiate communications with *https://yes-permit-to.com*. c) Implicitly deny all other communications with the internet, including denying i) the IoT device to initiate communications with *https://yes-permit-from.com* ~~ii) *https://yes-permit-to.com* to initiate communications with the IoT device~~ iii) communication between the IoT device and all other internet locations, such as *https://unnamed-to.com* (by not mentioning this or any other URLs in the MUD file)
Procedure	Note: Procedure steps with strikethrough are not tested due to NAT. 1. As stipulated in the preconditions, right before this test, test IoT-1-v4 (or IoT-1-v6) must have been run successfully.

Test Case Field	Description
	2. Initiate communications from the IoT device to *https://yes-permit-to.com* and verify that this traffic is received at *https://yes-permit-to.com*. (egress) ~~3. Initiate communications to the IoT device from *https://yes-permit-to.com* and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at the IoT device. (ingress)~~ ~~4. Initiate communications to the IoT device from *https://yes-permit-from.com* and verify that this traffic is received at the IoT device. (ingress)~~ ~~5. Initiate communications from the IoT device to *https://yes-permit-from.com* and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at *https://yes-permit-from.com*. (ingress)~~ 6. Initiate communications from the IoT device to *https://unnamed.com* and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at *https://unnamed.com*. (egress) ~~7. Initiate communications to the IoT device from *https://unnamed.com* and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at the IoT device. (ingress)~~
Expected Results	Each of the results that is listed as needing to be verified in procedure steps above occurs as expected.
Actual Results	**Procedure 1:** Excluded for length's sake **Procedure 2:** https://www.google.com (approved): `--2019-07-11 18:23:38--  https://www.google.com/` `Resolving www.google.com (www.google.com)... 172.217.164.132, 2607:f8b0:4004:814::2004`

Test Case Field	Description
	Connecting to www.google.com (www.google.com)|172.217.164.132|:443... connected. HTTP request sent, awaiting response... 200 OK Length: unspecified [text/html] Saving to: 'index.html.6' 0K .... 15.7M=0.001s 2019-07-11 18:23:38 (15.7 MB/s) - 'index.html.6' saved [11449] https://www.osmud.org (approved): --2019-07-11 18:23:04-- https://www.osmud.org/ Resolving www.osmud.org (www.osmud.org)... 198.71.233.87 **Connecting to www.osmud.org (www.osmud.org)|198.71.233.87|:443... connected.** HTTP request sent, awaiting response... 301 Moved Permanently Location: https://osmud.org/ [following] --2019-07-11 18:23:04-- https://osmud.org/ Resolving osmud.org (osmud.org)... 198.71.233.87 **Connecting to osmud.org (osmud.org)|198.71.233.87|:443... connected.** **HTTP request sent, awaiting response... 200 OK** Length: unspecified [text/html] Saving to: 'index.html.4' 0K .... 3.40M=0.007s

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td></td><td>
<pre>2019-07-11 18:23:05 (3.40 MB/s) - ‘index.html.4’ saved
[24697]</pre>
<hr>
https://www.trytechy.com (approved):
<pre>--2019-07-11 18:23:24-- https://www.trytechy.com/

Resolving www.trytechy.com (www.trytechy.com)...
99.84.181.77, 99.84.181.123, 99.84.181.11, ...

Connecting to www.trytechy.com
(www.trytechy.com)|99.84.181.77|:443... connected.

HTTP request sent, awaiting response... 200 OK

Length: unspecified [text/html]

Saving to: ‘index.html.5’

    0K ....
13.1M=0.001s

2019-07-11 18:23:24 (13.1 MB/s) - ‘index.html.5’ saved
[16529]</pre>
<hr>
Procedure 6:
https://www.facebook.com (unapproved):
<pre>--2019-07-11 18:23:55-- https://www.facebook.com/

Resolving www.facebook.com (www.facebook.com)...
31.13.71.36, 2a03:2880:f103:83:face:b00c:0:25de

Connecting to www.facebook.com
(www.facebook.com)|31.13.71.36|:443... failed: Connection
refused.

Connecting to www.facebook.com
(www.facebook.com)|2a03:2880:f103:83:face:b00c:0:25de|:443..
. failed: Network is unreachable.</pre>
<hr>
https://www.twitter.com (unapproved):
</td></tr>
</table>

Test Case Field	Description
	`--2019-07-11 18:24:07--  https://www.twitter.com/` `Resolving www.twitter.com (www.twitter.com)... 104.244.42.1, 104.244.42.65` **`Connecting to www.twitter.com (www.twitter.com)|104.244.42.1|:443... failed: Connection refused.`** `Connecting to www.twitter.com (www.twitter.com)|104.244.42.65|:443... failed: Connection refused.`
Overall Results	Pass (for testable procedures, ingress cannot be tested due to NAT)

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

3.1.2.6 Test Case IoT-6-v4

Table 3-7: Test Case IoT-6-v4

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

Test Case Field	Description
	(CR-9.b.1) The router or switch shall receive the attempt and shall allow it to pass based on the filters from the MUD file. (CR-10.a) The MUD-enabled IoT device shall attempt to initiate lateral traffic to unapproved (implicitly denied) devices. (CR-10.a.1) The router or switch shall receive the attempt and shall deny it based on the filters from the MUD file. (CR-10.b) An unapproved (implicitly denied) device shall attempt to initiate a lateral connection to the MUD-enabled IoT device. (CR-10.b.1) The router or switch shall receive the attempt and shall deny it based on the filters from the MUD file.
Description	Shows that, upon connection to the network, a MUD-enabled IoT device used in the IoT DDoS example implementation has its MUD PEP router/switch automatically configured to enforce the route filtering that is described in the device's MUD file with respect to communication with lateral devices. Further shows that the policies that are configured on the MUD PEP router/switch with respect to communication with lateral devices will be enforced as expected, with communications that are configured as denied being blocked and communications that are configured as permitted being allowed.
Associated Test Case(s)	IoT-1-v4 (for the v6 version of this test, IoT-1-v6)
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-3, PR.DS-5, PR.AC-5, PR.IP-1, PR.PT-3, PR.IP-3, PR.DS-3
IoT Device(s) Under Test	Raspberry Pi (3)
MUD File(s) Used	*Fe-localnetwork.json, Fe-my-controller.json, Fe-controller.json, Fe-manufacturer1.json, Fe-manufacturer2.json, Fe-samemanufacturer.json, Fe-localnetwork-to2.json, Fe-localnetwork-from2.json, Fe-samemanufacturer-from2.json, Fe-samemanufacturer-to2.json*
Preconditions	Test IoT-1-v4 (or IoT-1-v6) has run successfully, meaning that the MUD PEP router/switch has been configured to enforce the following policies

Test Case Field	Description
	for the IoT device in question with respect to local communications (as defined in the MUD files in Section 3.1.3): a) Local-network class—Explicitly permit **local communication to and from the IoT device and any local hosts** (including the specific local hosts *anyhost-to* and *anyhost-from*) **for specific services,** as specified in the MUD file by source port: any; destination port: 80; and protocol: TCP, and which party initiates the connection. b) Manufacturer class—Explicitly permit **local communication to and from the IoT device and other classes of IoT devices, as identified by their MUD URL (*www.devicetype.com*), and further constrained** by source port: any; destination port: 80; and protocol: TCP. c) Same-manufacturer class—Explicitly permit **local communication to and from IoT devices of the same manufacturer as the IoT device in question (the domain in the MUD URLs (mudfileserver) of the other IoT devices is the same as the domain in the MUD URL (mudfileserver) of the IoT device in question),** and further constrained by source port: any; destination port: 80; and protocol: TCP. d) Implicitly deny all other local communication that is not explicitly permitted in the MUD file, including denying i) ***anyhost-to*** **to initiate communications** with the IoT device ii) the **IoT device to initiate communications** with *anyhost-to* by **using a source port, destination port, or protocol (TCP or UDP) that is not explicitly permitted** iii) the **IoT device to initiate communications with *anyhost-from*** iv) ***anyhost-from*** **to initiate communications** with the IoT device by **using a source port, destination port, or protocol (TCP or UDP) that is not explicitly permitted** v) communications between the IoT device and all lateral hosts (including *unnamed-host*) whose **MUD URLs are not explicitly mentioned** as being permissible in the MUD file

Test Case Field	Description
	vi) communications between the IoT device and all lateral hosts whose **MUD URLs are explicitly mentioned** as being permissible **but using a source port, destination port, or protocol (TCP or UDP) that is not explicitly permitted** vii) communications between the IoT device and all lateral hosts that are **not from the same manufacturer** as the IoT device in question viii) communications between the IoT device and a lateral host that **is from the same manufacturer but using a source port, destination port, or protocol (TCP or UDP) that is not explicitly permitted**
Procedure	1. As stipulated in the preconditions, right before this test, test IoT-1-v4 (or IoT-1-v6) must have been run successfully. 2. Local-network (ingress): Initiate communications to the IoT device from *anyhost-from* **for specific permitted service,** and verify that this traffic is received at the IoT device. 3. Local-network (egress): **Initiate communications from the IoT device to *anyhost-from*** for specific permitted service, and verify that this traffic is received at the MUD PEP, but it **is not forwarded** by the MUD PEP, nor is it received at *anyhost-from*. 4. Local-network, controller, my-controller, manufacturer class (egress): Initiate communications from the IoT device to *anyhost-to* **for specific permitted service,** and verify that this traffic **is received** at *anyhost-to.* 5. Local-network, controller, my-controller, manufacturer class (ingress): **Initiate communications to the IoT device from *anyhost-to*** for specific permitted service, and verify that this traffic is received at the MUD PEP, but it **is not forwarded** by the MUD PEP, nor is it received at the IoT device. 6. No associated class (egress): Initiate communications **from** the IoT device to *unnamed-host* (where *unnamed-host* is a host that is not from the same manufacturer as the IoT device in question and whose **MUD URL is not explicitly mentioned in the MUD file as being permitted),** and verify that this traffic is received at the MUD

Test Case Field	Description
	PEP, but it **is not forwarded** by the MUD PEP, nor is it received at *unnamed-host*. 7. No associated class (ingress): Initiate communications **to** the IoT device from *unnamed-host* (where *unnamed-host* is a host that is not from the same manufacturer as the IoT device in question and whose **MUD URL is not explicitly mentioned in the MUD file as being permitted),** and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at the IoT device. 8. Same-manufacturer class (egress): Initiate communications **from** the IoT device to *same-manufacturer-host* (where *same-manufacturer-host* is **a host that is from the same manufacturer as the IoT device** in question) and verify that this traffic **is received** at *same-manufacturer-host*. 9. Same-manufacturer class (egress): Initiate communications **from** the IoT device to *same-manufacturer-host* (where *same-manufacturer-host* is **a host that is from the same manufacturer as the IoT device** in question) **but using a port or protocol that is not specified,** and verify that this traffic is received at the MUD PEP, but it **is not forwarded** by the MUD PEP, nor is it received at *same-manufacturer-host*.
Expected Results	Each of the results that is listed as needing to be verified in the procedure steps above occurs as expected.
Actual Results	**Local-Network:** Procedure 2 (from laptop to pi): *http://192.168.20.222* `[mud@localhost ~]$ wget 192.168.20.222` `--2019-07-24 15:30:01--  http://192.168.20.222/` `Connecting to 192.168.20.222:80... connected.` `HTTP request sent, awaiting response... 200 OK` `Length: 10701 (10K) [text/html]` `Saving to: ‘index.html’` `100%[=====================================>]`

Test Case Field	Description
	`10,701      --.-K/s   in 0s` `2019-07-24 15:30:01 (139 MB/s) - 'index.html' saved [10701/10701]`
	Procedure 3 (from pi to laptop): *http://192.168.20.238/* (unapproved): `--2019-07-10 17:37:09--  `*`http://192.168.20.238/`* **`Connecting to 192.168.20.238:80... failed: Connection refused.`**
	Procedure 4 (from pi to local hosts): *http://192.168.20.110:443/* (approved): `--2019-07-10 19:02:34--  `*`http://192.168.20.110:443/`* **`Connecting to 192.168.20.110:443... connected.`** **`HTTP request sent, awaiting response... 200 OK`** `Length: 10701 (10K) [text/html]` `Saving to: 'index.html.28'` `     0K .......... 100% 11.2M=0.001s` `2019-07-10 19:02:34 (11.2 MB/s) - 'index.html.28' saved [10701/10701]`
	http://192.168.20.232/ (approved): `--2019-07-10 19:00:10--  `*`http://192.168.20.232/`* **`Connecting to 192.168.20.232:80... connected.`** **`HTTP request sent, awaiting response... 200 OK`** `Length: 277` `Saving to: 'index.html.14'`

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td></td><td>
<pre>
    0K   100%
10.9M=0s

2019-07-10 19:00:10 (10.9 MB/s) - ‘index.html.14’ saved
[277/277]
</pre>
<hr>
http://192.168.20.117/ (approved):
<pre>
--2019-07-10 18:59:40-- http://192.168.20.117/

Connecting to 192.168.20.117:80... connected.

HTTP request sent, awaiting response... 200 OK

Length: 10701 (10K) [text/html]

Saving to: ‘index.html.12’

    0K ....
100% 6.05M=0.002s

2019-07-10 18:59:40 (6.05 MB/s) - ‘index.html.12’ saved
[10701/10701]
</pre>
<hr>
http://192.168.20.197/ (approved):
<pre>
--2019-07-10 18:55:39-- http://192.168.20.197/

Connecting to 192.168.20.197:80... connected.

HTTP request sent, awaiting response... 200 OK

Length: 10701 (10K) [text/html]

Saving to: ‘index.html.8’

    0K ....
100% 2.03M=0.005s

2019-07-10 18:55:40 (2.03 MB/s) - ‘index.html.8’ saved
[10701/10701]
</pre>
<hr>
http://192.168.20.183/ (approved):
<pre>
 --2019-07-10 18:59:21-- http://192.168.20.183/
</pre>
</td></tr>
</table>

Test Case Field	Description
	Connecting to 192.168.20.183:80... connected. **HTTP request sent, awaiting response... 200 OK** Length: 10701 (10K) [text/html] Saving to: 'index.html.10' 0K .... 100% 17.6M=0.001s 2019-07-10 18:59:21 (17.6 MB/s) - 'index.html.10' saved [10701/10701] Procedure 5 (from laptop to pi): [mud@localhost ~]$ wget 192.168.20.222 --2019-07-10 19:03:17-- *http://192.168.20.222/* **Connecting to 192.168.20.222:80... failed: Connection refused.** Procedure 6 (from device): http://www.facebook.com (unapproved): --2019-07-10 19:17:39-- https://www.facebook.com/ Resolving www.facebook.com (www.facebook.com)... 31.13.71.36, 2a03:2880:f112:83:face:b00c:0:25de Connecting to www.facebook.com (www.facebook.com)|31.13.71.36|:443... **failed: Connection refused.** Connecting to www.facebook.com (www.facebook.com)|2a03:2880:f112:83:face:b00c:0:25de|:443... failed: Network is unreachable. Procedure 7 (from laptop to Pi): [mud@localhost ~]$ wget 192.168.20.222 --2019-07-10 19:20:06-- *http://192.168.20.222/* **Connecting to 192.168.20.222:80... failed: Connection refused.**

Test Case Field	Description
	Controller: Procedure 4 (from Pi to controller): https://www.trytechy.com/ (approved): `--2019-07-10 17:29:55--  https://www.trytechy.com/` `Resolving www.trytechy.com (www.trytechy.com)... 54.230.193.215, 54.230.193.99, 54.230.193.140, ...` **`Connecting to www.trytechy.com (www.trytechy.com)|54.230.193.215|:443... connected.`** **`HTTP request sent, awaiting response... 200 OK`** `Length: unspecified [text/html]` `Saving to: 'index.html'` `    0K .......... ...... 1.80M=0.009s` `2019-07-10 17:29:55 (1.80 MB/s) - 'index.html' saved [16529]` Procedure 5 (from laptop to pi): `[mud@localhost ~]$ wget 192.168.20.222` `--2019-07-10 17:30:04--  `*`http://192.168.20.222/`* **`Connecting to 192.168.20.222:80... failed: Connection refused.`** Procedure 6 (from pi to local hosts): *http://192.168.20.232/* (unapproved): `--2019-07-10 17:37:09--  `*`http://192.168.20.232/`* **`Connecting to 192.168.20.232:80... failed: Connection refused.`** *http://192.168.20.110/* (unapproved): `--2019-07-10 17:38:49--  `*`http://192.168.20.110/`*

Test Case Field	Description
	Connecting to 192.168.20.110:80... failed: Connection refused. *http://192.168.20.183/* (unapproved): `--2019-07-10 17:46:38--` *http://192.168.20.183/* **Connecting to 192.168.20.183:80... failed: Connection refused.** *http://192.168.20.142/* (unapproved): `--2019-07-10 17:36:38--` *http://192.168.20.142/* **Connecting to 192.168.20.142:80... failed: Connection refused.** *http://192.168.20.117/* (unapproved): `--2019-07-10 17:36:55--` *http://192.168.20.117/* **Connecting to 192.168.20.117:80... failed: Connection refused.** *http://192.168.20.171/* (unapproved): `--2019-07-10 17:47:18--` *http://192.168.20.171/* **Connecting to 192.168.20.171:80... failed: Connection refused.** *http://192.168.20.181/* (unapproved): `--2019-07-10 17:47:49--` *http://192.168.20.181/* **Connecting to 192.168.20.181:80... failed: Connection refused.** *http://192.168.20.247/* (unapproved): `--2019-07-10 17:48:13--` *http://192.168.20.247/* **Connecting to 192.168.20.247:80... failed: Connection refused.**

Test Case Field	Description
	Procedure 7 (from laptop to Pi): `[mud@localhost ~]$ wget 192.168.20.222` `--2019-07-10 17:50:22--  http://192.168.20.222/` **`Connecting to 192.168.20.222:80... failed: Connection refused.`** **My Controller:** Procedure 4 (from device): https://www.google.com (approved): `--2019-07-10 18:13:12--  https://www.google.com/` `Resolving www.google.com (www.google.com)... 172.217.164.132, 2607:f8b0:4004:814::2004` **`Connecting to www.google.com (www.google.com)|172.217.164.132|:443... connected.`** `HTTP request sent, awaiting response... 200 OK` `Length: unspecified [text/html]` `Saving to: ‘index.html.1’` `    0K .......... ..` `14.9M=0.001s` `2019-07-10 18:13:12 (14.9 MB/s) - ‘index.html.1’ saved [12327]` Procedure 5 (from laptop to pi): `[mud@localhost ~]$ wget 192.168.20.222` `--2019-07-24 18:22:48--  http://192.168.20.222/` **`Connecting to 192.168.20.222:80... failed: Connection refused.`** Procedure 6 (from device): *http://192.168.20.110/* (unapproved): `--2019-07-10 18:29:42--  http://192.168.20.110/` **`Connecting to 192.168.20.110:80... failed: Connection refused.`** *http://192.168.20.117/* (unapproved): `--2019-07-10 18:29:34--  http://192.168.20.117/`

Test Case Field	Description
	Connecting to 192.168.20.117:80... failed: Connection refused.
	http://192.168.20.142/ (unapproved): --2019-07-10 18:30:26-- *http://192.168.20.142/* **Connecting to 192.168.20.142:80... failed: Connection refused.**
	http://192.168.20.171/ (unapproved): --2019-07-10 18:29:55-- *http://192.168.20.171/* **Connecting to 192.168.20.171:80... failed: Connection refused.**
	http://192.168.20.181/ (unapproved): --2019-07-10 18:29:08-- *http://192.168.20.181/* **Connecting to 192.168.20.181:80... failed: Connection refused.**
	http://192.168.20.183/ (unapproved): --2019-07-10 18:29:23-- *http://192.168.20.183/* **Connecting to 192.168.20.183:80... failed: Connection refused.**
	http://192.168.20.197/ (unapproved): --2019-07-10 18:28:32-- *http://192.168.20.197/* **Connecting to 192.168.20.197:80... failed: Connection refused.**
	http://192.168.20.232/ (unapproved): --2019-07-10 18:30:36-- *http://192.168.20.232/* **Connecting to 192.168.20.232:80... failed: Connection refused.**
	http://192.168.20.247/ (unapproved): --2019-07-10 18:28:45-- *http://192.168.20.247/* **Connecting to 192.168.20.247:80... failed: Connection refused.**
	Procedure 7 (from laptop to Pi): [mud@localhost ~]$ wget 192.168.20.222 --2019-07-10 18:29:13-- http://192.168.20.222/

Test Case Field	Description
	`Connecting to 192.168.20.222:80... failed: Connection refused.`
	Same Manufacturer 1 (.197): Procedure 4 (from device): *http://192.168.20.222/* (approved): `--2019-07-12 16:04:46--  http://192.168.20.222/` **`Connecting to 192.168.20.222:80... connected.`** **`HTTP request sent, awaiting response... 200 OK`** `Length: 10701 (10K) [text/html]` `Saving to: 'index.html.9'` `    0K ..........` `100%  104K=0.1s` `2019-07-12 16:04:46 (104 KB/s) - 'index.html.9' saved [10701/10701]`
	Procedure 5 (from laptop to pi): `[mud@localhost ~]$ wget 192.168.20.222` `--2019-07-12 16:08:28--` http://192.168.20.222/ **`Connecting to 192.168.20.222:80... failed: Connection refused.`**
	Procedure 6 (from device): http://192.168.20.232/ (unapproved): `--2019-07-12 16:06:35--  http://192.168.20.232/` **`Connecting to 192.168.20.232:80... failed: Connection refused.`**
	http://192.168.20.110:443/ (unapproved): `--2019-07-12 16:06:16--  http://192.168.20.110:443/` **`Connecting to 192.168.20.110:443... failed: Connection refused.`**
	http://192.168.20.117/ (unapproved): `--2019-07-12 16:06:01--  http://192.168.20.117/` **`Connecting to 192.168.20.117:80... failed: Connection refused.`**

Test Case Field	Description
	http://192.168.20.181/ (unapproved): `--2019-07-12 16:05:39--  http://192.168.20.181/` **`Connecting to 192.168.20.181:80... failed: Connection refused.`** *http://192.168.20.183/* (unapproved): `--2019-07-12 16:05:11--  http://192.168.20.183/` **`Connecting to 192.168.20.183:80... failed: Connection refused.`** Procedure 7 (from laptop to Pi): `[mud@localhost ~]$ wget 192.168.20.222` `--2019-07-12 16:12:03--` http://192.168.20.222/ **`Connecting to 192.168.20.222:80... failed: Connection refused.`** **Manufacturer:** Procedure 4 (from device): *http://192.168.20.183/* (approved): `--2019-07-12 15:57:00--  http://192.168.20.183/` **`Connecting to 192.168.20.183:80... connected.`** **`HTTP request sent, awaiting response... 200 OK`** `Length: 10701 (10K) [text/html]` `Saving to: 'index.html.21'` `    0K ..........` `100% 26.9M=0s` `2019-07-12 15:57:00 (26.9 MB/s) - 'index.html.21' saved [10701/10701]` Procedure 5 (from laptop to pi): `[mud@localhost ~]$ wget 192.168.20.222` `--2019-07-12 15:59:31--` http://192.168.20.222/ **`Connecting to 192.168.20.222:80... failed: Connection refused.`**

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td></td><td>
Procedure 6 (from device):
http://192.168.20.110:443/ (unapproved):
<pre>
--2019-07-12 15:58:13-- http://192.168.20.110:443/
Connecting to 192.168.20.110:443... failed: Connection refused.
</pre>
http://192.168.20.117/ (unapproved):
<pre>
--2019-07-12 15:57:19-- http://192.168.20.117/
Connecting to 192.168.20.117:80... failed: Connection refused.
</pre>
http://192.168.20.232/ (unapproved):
<pre>
--2019-07-12 15:57:29-- http://192.168.20.232/
Connecting to 192.168.20.232:80... failed: Connection refused.
</pre>
http://192.168.20.197 (unapproved):
<pre>
--2019-07-12 15:58:35-- http://192.168.20.197/
Connecting to 192.168.20.197:80... failed: Connection refused.
</pre>
Procedure 7 (from laptop to Pi):
<pre>
[mud@localhost ~]$ wget 192.168.20.222
--2019-07-12 15:59:31-- http://192.168.20.222/
Connecting to 192.168.20.222:80... failed: Connection refused.
</pre>
Same Manufacturer:
Procedure 8 (from device):
http://192.168.20.197/ (approved):
<pre>
--2019-07-12 16:27:24-- http://192.168.20.197/
Connecting to 192.168.20.197:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10701 (10K) [text/html]
Saving to: ‘index.html.43’
     0K ....
100% 3.75M=0.003s
</pre>
</td></tr>
</table>

Test Case Field	Description
	`2019-07-12 16:27:24 (3.75 MB/s) - 'index.html.43' saved [10701/10701]` Procedure 6 (from device): *http://192.168.20.183/* (unapproved): `--2019-07-12 16:27:36--  http://192.168.20.183/` **`Connecting to 192.168.20.183:80... failed: Connection refused.`** *http://192.168.20.181/* (unapproved): `--2019-07-12 16:28:11--  http://192.168.20.181/` **`Connecting to 192.168.20.181:80... failed: Connection refused.`** *http://192.168.20.142/* (unapproved): `--2019-07-12 16:27:48--  http://192.168.20.142/` **`Connecting to 192.168.20.142:80... failed: Connection refused.`** *http://192.168.20.117/* (unapproved): `--2019-07-12 16:28:20--  http://192.168.20.117/` **`Connecting to 192.168.20.117:80... failed: Connection refused.`** *http://192.168.20.110:443/* (unapproved): `--2019-07-12 16:27:59--  http://192.168.20.110:443/` **`Connecting to 192.168.20.110:443... failed: Connection refused.`** **Procedure 9:** `pi@same-manufacture-pi:~ $ wget 192.168.20.222` `--2019-07-24 20:49:51--` http://192.168.20.222/ **`Connecting to 192.168.20.222:80... failed: Connection refused.`**
Overall Results	Pass

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

3.1.2.7 Test Case IoT-7-v4

Table 3-8: Test Case IoT-7-v4

Test Case Field	Description
Parent Requirement	(CR-11) If the IoT DDoS example implementation is such that its DHCP server does not act as a MUD manager and it forwards a MUD URL to a MUD manager, the DHCP server must notify the MUD manager of any corresponding change to the DHCP state of the MUD-enabled IoT device, and the MUD manager should remove the implemented policy configuration in the router/switch pertaining to that MUD-enabled IoT device.
Testable Requirement	(CR-11.a) The MUD-enabled IoT device shall explicitly release the IP address lease (i.e., it sends a DHCP release message to the DHCP server). (CR-11.a.1) The DHCP server shall notify the MUD manager that the device's IP address lease has been released. (CR-11.a.2) The MUD manager should remove all policies associated with the disconnected IoT device that had been configured on the MUD PEP router/switch.
Description	Shows that when a MUD-enabled IoT device explicitly releases its IP address lease, the MUD-related configuration for that IoT device will be removed from its MUD PEP router/switch
Associated Test Case(s)	IoT-1-v4 (or IoT-1-v6 when IPv6 addressing is used)
Associated Cybersecurity Framework Subcategory(ies)	PR.IP-3, PR.DS-3
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*Fe-samemanufacturer.json*

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td>Preconditions</td><td>Test IoT-1-v4 (or IoT-1-v6) has run successfully, meaning that the MUD PEP router/switch has been configured to enforce the policies defined in the MUD file in Section 3.1.3 for the IoT device in question.</td></tr>
<tr><td>Procedure</td><td>1. As stipulated in the preconditions, right before this test, test IoT-1-v4 (or IoT-1-v6) must have been run successfully. Verify that the MUD PEP router/switch for the IoT device has been configured to enforce the policies listed in the preconditions section above for the IoT device in question.
2. Cause a DHCP release of the IoT device in question.
3. Check the log file for the MUD manager to verify that it was notified of the change of DHCP state.
4. Verify that all the configuration rules listed above have been removed from the MUD PEP router/switch for the IoT device in question.</td></tr>
<tr><td>Expected Results</td><td>All of the configuration rules listed above have been removed from the MUD PEP router/switch for the IoT device in question.</td></tr>
<tr><td>Actual Results</td><td>Procedure 2:
<code>pi@main-pi-Build2:~ $ sudo dhclient -r</code>
<hr>Procedure 3:
<code>MUD Manager:</code>
<code>2019-07-11 18:57:30 DEBUG::GENERAL::2019-07-11T18:57:29Z|DEL|Wired|DHCP|-|MUD|-|-|b8:27:eb:eb:6c:8b|192.168.20.226|main-pi-Build2|</code>
<code>2019-07-11 18:57:30 DEBUG::GENERAL::Executing on dhcpmasq info</code>
<code>2019-07-11 18:57:30 INFO::GENERAL::DEL Device Action: IP: 192.168.20.226, MAC: b8:27:eb:eb:6c:8b</code>
<code>2019-07-11 18:57:30 DEBUG::GENERAL::/etc/osmud/find_device_in_db.sh -d /etc/osmud/state/mudfiles/mudStateFile.txt -m b8:27:eb:eb:6c:8b -i 192.168.20.226 -s /etc/osmud/state/ipSets -a DELETE -u NONE</code>
<code>2019-07-11 18:57:30 DEBUG::GENERAL::Return: 4864.</code>
<code>2019-07-11 18:57:30 DEBUG::GENERAL::FinalReturn: 19.</code></td></tr>
</table>

Test Case Field	Description
	`2019-07-11 18:57:30 ERROR::DEVICE_INTERFACE::FinalReturn: 19.` **`2019-07-11 18:57:30 DEBUG::CONTROLLER::MUD Controller: A delete event associated with a MUD file is being processed. IP: 192.168.20.226.`** **`2019-07-11 18:57:30 DEBUG::GENERAL::rm -f /tmp/osmud/*`** **`2019-07-11 18:57:30 DEBUG::GENERAL::cp /etc/osmud/state/ipSets/* /tmp/osmud`** **`2019-07-11 18:57:30 DEBUG::GENERAL::/etc/osmud/remove_ip_fw_rule.sh -i 192.168.20.226 -m b8:27:eb:eb:6c:8b -d /tmp/osmud`** **`2019-07-11 18:57:30 DEBUG::GENERAL::/etc/osmud/remove_from_ipset.sh -d /tmp/osmud -i 192.168.20.226`** **`2019-07-11 18:57:30 DEBUG::GENERAL::/etc/osmud/commit_ip_fw_rules.sh -d /etc/osmud/state/ipSets -t /tmp/osmud`** **`2019-07-11 18:57:30 DEBUG::GENERAL::/etc/osmud/remove_mud_db_entry.sh -d /etc/osmud/state/mudfiles/mudStateFile.txt -i 192.168.20.226 -m b8:27:eb:eb:6c:8b`** **`2019-07-11 18:57:30 DEBUG::GENERAL::Success returned from for transaction`** **Procedure 4:** **ROUTER/PEP:** `# OSMUD start` `#` `# DO NOT EDIT THESE LINES. OSMUD WILL REPLACE WITH ITS CONFIGURATION` `#` `config ipset` `    option enabled 1` `    option name mudfiles_nist_getyikes_com-SMTD` `    option match dest_ip` `    option storage hash` `    option family ipv4` `    option external mudfiles_nist_getyikes_com-SM` `config ipset` `    option enabled 1` `    option name mudfiles_nist_getyikes_com-SMFD` `    option match src_ip` `    option storage hash` `    option family ipv4` `    option external mudfiles_nist_getyikes_com-SM` `config ipset` `    option enabled 1`

Test Case Field	Description
	`` option name mudfileserver-SMTD option match dest_ip option storage hash option family ipv4 option external mudfileserver-SM config ipset option enabled 1 option name mudfileserver-SMFD option match src_ip option storage hash option family ipv4 option external mudfileserver-SM config ipset option enabled 1 option name www_facebook_com-SMTD option match dest_ip option storage hash option family ipv4 option external www_facebook_com-SM config ipset option enabled 1 option name www_facebook_com-SMFD option match src_ip option storage hash option family ipv4 option external www_facebook_com-SM config ipset option enabled 1 option name www_gmail_com-SMTD option match dest_ip option storage hash option family ipv4 option external www_gmail_com-SM config ipset option enabled 1 option name www_gmail_com-SMFD option match src_ip option storage hash option family ipv4 option external www_gmail_com-SM config rule option enabled '1' ```

Test Case Field	Description
	`` option name 'mud_192.168.20.197_same- manufacture-pi_cl0-frdev' option target ACCEPT option src lan option dest wan option proto tcp option family ipv4 option src_ip 192.168.20.197 option dest_ip 198.71.233.87 config rule option enabled '1' option name 'mud_192.168.20.197_same- manufacture-pi_cl0-todev' option target ACCEPT option src wan option dest lan option proto tcp option family ipv4 option src_ip 198.71.233.87 option dest_ip 192.168.20.197 config rule option enabled '1' option name 'mud_192.168.20.197_same- manufacture-pi_myman0-frdev-SM' option target ACCEPT option src lan option dest lan option proto tcp option family ipv4 option src_ip 192.168.20.197 option ipset www_facebook_com-SMTD option dest_port 80:80 config rule option enabled '1' option name 'mud_192.168.20.197_same- manufacture-pi_myman0-todev-SM' option target ACCEPT option src lan option dest lan option proto tcp option family ipv4 option ipset www_facebook_com-SMFD option dest_ip 192.168.20.197 option dest_port 80:80 ```

Test Case Field	Description
	`config rule` `        option enabled   '1'` `        option name      'mud_192.168.20.197_same-` `manufacture-pi_REJECT-ALL-LOCAL-FROM'` `        option target    REJECT` `        option src       lan` `        option dest      lan` `        option proto     all` `        option family    ipv4` `        option src_ip     192.168.20.197` `config rule` `        option enabled   '1'` `        option name      'mud_192.168.20.197_same-` `manufacture-pi_REJECT-ALL-LOCAL-TO'` `        option target    REJECT` `        option src       lan` `        option dest      lan` `        option proto     all` `        option family    ipv4` `        option src_ip     any` `        option dest_ip    192.168.20.197` `config rule` `        option enabled   '1'` `        option name      'mud_192.168.20.197_same-` `manufacture-pi_REJECT-ALL'` `        option target    REJECT` `        option src       lan` `        option dest      wan` `        option proto     all` `        option family    ipv4` `        option src_ip     192.168.20.197` `# OSMUD end`
Overall Results	Pass

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

3.1.2.8 Test Case IoT-8-v4

Table 3-9: Test Case IoT-8-v4

Test Case Field	Description
Parent Requirement	(CR-11) If the IoT DDoS example implementation is such that its DHCP server does not act as a MUD manager and it forwards a MUD URL to a MUD manager, the DHCP server must notify the MUD manager of any corresponding change to the DHCP state of the MUD-enabled IoT device, and the MUD manager should remove the implemented policy configuration in the router/switch pertaining to that MUD-enabled IoT device.
Testable Requirement	(CR-11.b) The MUD-enabled IoT device's IP address lease shall expire. (CR-11.b.1) The DHCP server shall notify the MUD manager that the device's IP address lease has expired. (CR-11.b.2) The MUD manager should remove all policies associated with the affected IoT device that had been configured on the MUD PEP router/switch.
Description	Shows that when a MUD-enabled IoT device's IP address lease expires, the MUD-related configuration for that IoT device will be removed from its MUD PEP router/switch
Associated Test Case(s)	IoT-1-v4 (or IoT-1-v6 when IPv6 addressing is used)
Associated Cybersecurity Framework Subcategory(ies)	PR.IP-3, PR.DS-3
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*Fe-manufacturer1.json*
Preconditions	Test IoT-1-v4 (or IoT-1-v6) has run successfully, meaning that the MUD PEP router/switch has been configured to enforce the policies defined in the MUD file in Section 3.1.3 for the IoT device in question.
Procedure	1. Configure the DHCP server to have a DHCP lease time of 60 minutes. 2. Run test IoT-1-v4 (or IoT-1-v6).

Test Case Field	Description
	3. Verify that the MUD PEP router/switch for the IoT device has been configured to enforce the policies listed above for the IoT device in question. 4. Disconnect the IoT device in question from the network. 5. After 60 minutes have elapsed, (1) look at the log file for the MUD manager to verify that it has received notice of the change of DHCP state, and (2) verify that all of the configuration rules listed above have been removed from the MUD PEP router/switch for the IoT device in question.
Expected Results	Once 60 minutes have elapsed after disconnecting the IoT device from the network, all of the configuration rules listed above have been removed from the MUD PEP router/switch for the IoT device in question.
Actual Results	**Procedures 1–4:** **Completed; excluded for brevity** **Procedure 5:** 1. MUD MANAGER: **2019-07-12 17:34:49 DEBUG::GENERAL::2019-07-12T17:34:49Z|DEL|Wired|DHCP|-|MUD|-|-|b8:27:eb:a2:88:f3|192.168.20.184|manufacturer-pi|** 2019-07-12 17:34:49 DEBUG::GENERAL::Executing on dhcpmasq info 2019-07-12 17:34:49 INFO::GENERAL::DEL Device Action: IP: 192.168.20.184, MAC: b8:27:eb:a2:88:f3 2019-07-12 17:34:49 DEBUG::GENERAL::/etc/osmud/find_device_in_db.sh -d /etc/osmud/state/mudfiles/mudStateFile.txt -m b8:27:eb:a2:88:f3 -i 192.168.20.184 -s /etc/osmud/state/ipSets -a DELETE -u NONE 2019-07-12 17:34:49 DEBUG::GENERAL::Return: 3328. 2019-07-12 17:34:49 DEBUG::GENERAL::FinalReturn: 13. 2019-07-12 17:34:49 ERROR::DEVICE_INTERFACE::FinalReturn: 13. **2019-07-12 17:34:49 DEBUG::CONTROLLER::MUD Controller: A delete event associated with a MUD file is being processed. IP: 192.168.20.184.**2019-07-12 17:34:49 DEBUG::GENERAL::rm -f /tmp/osmud/*

Test Case Field	Description
	2019-07-12 17:34:49 DEBUG::GENERAL::cp /etc/osmud/state/ipSets/* /tmp/osmud 2019-07-12 17:34:49 DEBUG::GENERAL::/etc/osmud/**remove**_ip_fw_rule.sh -i 192.168.20.184 -m b8:27:eb:a2:88:f3 -d /tmp/osmud 2019-07-12 17:34:49 DEBUG::GENERAL::/etc/osmud/**remove**_from_ipset.sh -d /tmp/osmud -i 192.168.20.184 2019-07-12 17:34:49 DEBUG::GENERAL::/etc/osmud/commit_ip_fw_rules.sh -d /etc/osmud/state/ipSets -t /tmp/osmud 2019-07-12 17:34:50 DEBUG::GENERAL::/etc/osmud/**remove**_mud_db_entry.sh -d /etc/osmud/state/mudfiles/mudStateFile.txt -i 192.168.20.184 -m b8:27:eb:a2:88:f3 **2019-07-12 17:34:50 DEBUG::GENERAL::Success returned from for transaction** 2. Router/PEP: # OSMUD start # # DO NOT EDIT THESE LINES. OSMUD WILL REPLACE WITH ITS CON-FIGURATION # config ipset option enabled 1 option name mudfiles_nist_getyikes_com-SMTD option match dest_ip option storage hash option family ipv4 option external mudfiles_nist_getyikes_com-SM config ipset option enabled 1 option name mudfiles_nist_getyikes_com-SMFD option match src_ip option storage hash option family ipv4 option external mudfiles_nist_getyikes_com-SM config ipset option enabled 1 option name mudfileserver-SMTD option match dest_ip option storage hash option family ipv4 option external mudfileserver-SM

Test Case Field	Description
	`` config ipset option enabled 1 option name mudfileserver-SMFD option match src_ip option storage hash option family ipv4 option external mudfileserver-SM config ipset option enabled 1 option name www_facebook_com-SMTD option match dest_ip option storage hash option family ipv4 option external www_facebook_com-SM config ipset option enabled 1 option name www_facebook_com-SMFD option match src_ip option storage hash option family ipv4 option external www_facebook_com-SM config ipset option enabled 1 option name www_gmail_com-SMTD option match dest_ip option storage hash option family ipv4 option external www_gmail_com-SM config ipset option enabled 1 option name www_gmail_com-SMFD option match src_ip option storage hash option family ipv4 option external www_gmail_com-SM config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac- ture-pi_cl0-frdev' option target ACCEPT option src lan option dest wan option proto tcp ```

Test Case Field	Description
	option family ipv4 option src_ip 192.168.20.197 option dest_ip 198.71.233.87 config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac-ture-pi_cl0-todev' option target ACCEPT option src wan option dest lan option proto tcp option family ipv4 option src_ip 198.71.233.87 option dest_ip 192.168.20.197 config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac-ture-pi_myman0-frdev-SM' option target ACCEPT option src lan option dest lan option proto tcp option family ipv4 option src_ip 192.168.20.197 option ipset www_facebook_com-SMTD option dest_port 80:80 config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac-ture-pi_myman0-todev-SM' option target ACCEPT option src lan option dest lan option proto tcp option family ipv4 option ipset www_facebook_com-SMFD option dest_ip 192.168.20.197 option dest_port 80:80 config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac-ture-pi_REJECT-ALL-LOCAL-FROM' option target REJECT option src lan option dest lan

Test Case Field	Description
	option proto all option family ipv4 option src_ip 192.168.20.197 config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac-ture-pi_REJECT-ALL-LOCAL-TO' option target REJECT option src lan option dest lan option proto all option family ipv4 option src_ip any option dest_ip 192.168.20.197 config rule option enabled '1' option name 'mud_192.168.20.197_same-manufac-ture-pi_REJECT-ALL' option target REJECT option src lan option dest wan option proto all option family ipv4 option src_ip 192.168.20.197 # OSMUD end
Overall Results	Pass

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

3.1.2.9 Test Case IoT-9-v4

Table 3-10: Test Case IoT-9-v4

Test Case Field	Description
Parent Requirements	(CR-13) The IoT DDoS example implementation shall ensure that for each rule in a MUD file that pertains to an external domain, the MUD PEP router/switch will get configured with all possible instantiations of that rule, insofar as each instantiation contains one of the IP addresses

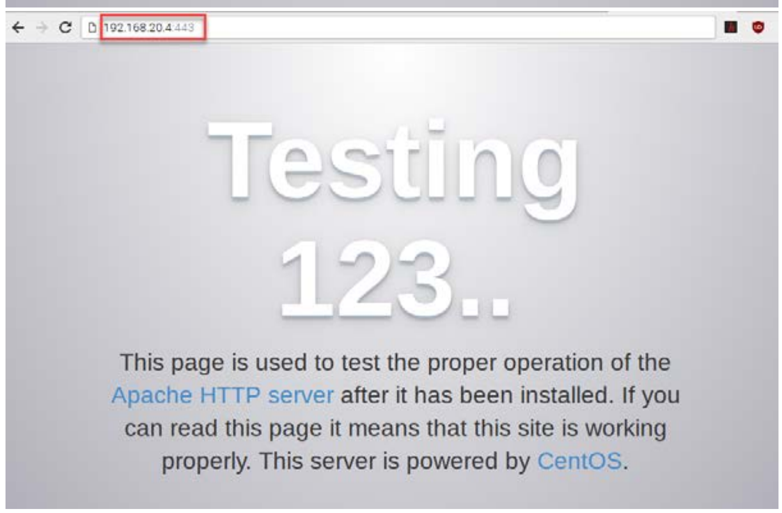
Test Case Field	Description
	to which the domain in that MUD file rule may be resolved when queried by the MUD PEP router/switch.
Testable Requirements	(CR-13.a) The MUD file for a device shall contain a rule involving an external **domain that can resolve** to multiple IP addresses when queried by the MUD PEP router/switch. An ACL for permitting access to each of those IP addresses will be inserted into the MUD PEP router/switch for the device in question, and the device will be permitted to communicate with all of those IP addresses.
Description	Shows that if a domain in a MUD file rule resolves to multiple IP addresses when the address resolution is queried by the network gateway, then 1. ACLs instantiating that MUD file rule corresponding to each of these IP addresses will be configured in the gateway for the IoT device associated with the MUD file, and 2. the IoT device associated with the MUD file will be permitted to communicate with all of the IP addresses to which that domain resolves
Associated Test Case(s)	N/A
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-1, ID.AM-2, ID.AM-3, PR.DS-5, DE.AE-1, PR.AC-4, PR.AC-5, PR.IP-1, PR.IP-3, PR.DS-2
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*Yikesmain.json*
Preconditions	1. The MUD PEP router/switch does not yet have any configuration settings pertaining to the IoT device being used in the test. 2. The MUD file for the IoT device being used in the test is identical to the MUD file provided in Section 3.1.3. (Therefore, the MUD file used in the test permits the device to send data to *www.updateserver.com*.)

Test Case Field	Description
	3. The tester has access to a DNS server that will be used by the MUD PEP router/switch and can configure it so that it will resolve the domain *www.updateserver.com* to any of these addresses when queried by the MUD PEP router/switch: x1.x1.x1.x1, y1.y1.y1.y1, and z1.z1.z1.z1. 4. There is an update server running at each of these three IP addresses.
Procedure	1. Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. 2. Run test IoT-1-v4 (or IoT-1-v6). The result should be that the MUD PEP router/switch has been configured to explicitly permit the IoT device to initiate communication with *www.updateserver.com*. 3. Verify that the MUD PEP router/switch has been configured with ACLs that permit the IoT device to send data to IP addresses x1.x1.x1.x1, y1.y1.y1.y1, and z1.z1.z1.z1. 4. Have the device in question attempt to connect to x1.x1.x1.x1, y1.y1.y1.y1, and z1.z1.z1.z1.
Expected Results	The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to permit the IoT device to send data to IP addresses x1.x1.x1.x1, y1.y1.y1.y1, and z1.z1.z1.z1. The IoT device is permitted to send data to each of the update servers at these addresses.
Actual Results	**Procedures 1–2:** **Completed; excluded for brevity** **Procedure 3:** **MUD MANAGER:** **`2019-07-15 20:28:32 DEBUG::GENERAL::2019-07-15T20:28:31Z|NEW|Wired|DHCP|1,28,2,3,15,6,119,12,44,47,26,121,42|MUD|https://mudfiles.nist.getyikes.com/yikesmain.json|-|b8:27:eb:eb:6c:8b|192.168.20.222|main-pi-Build2|`** `2019-07-15 20:28:32 DEBUG::GENERAL::Executing on dhcpmasq info` `2019-07-15 20:28:32 INFO::GENERAL::NEW Device Action: IP: 192.168.20.222, MAC: b8:27:eb:eb:6c:8b`

Test Case Field	Description
	2019-07-15 20:28:32 DEBUG::COMMUNICATION::curl_easy_perform() doing it now.... 2019-07-15 20:28:32 DEBUG::COMMUNICATION::https://mudfiles.nist.getyikes.com/yikesmain.json 2019-07-15 20:28:32 DEBUG::COMMUNICATION::Found HTTPS 2019-07-15 20:28:32 DEBUG::COMMUNICATION::in write data 2019-07-15 20:28:32 DEBUG::COMMUNICATION::curl_easy_perform() success 2019-07-15 20:28:32 DEBUG::COMMUNICATION::MUD File Server returned success state. 2019-07-15 20:28:32 DEBUG::COMMUNICATION::curl_easy_perform() doing it now.... 2019-07-15 20:28:32 DEBUG::COMMUNICATION::https://mudfiles.nist.getyikes.com/yikesmain.p7s 2019-07-15 20:28:32 DEBUG::COMMUNICATION::Found HTTPS 2019-07-15 20:28:32 DEBUG::COMMUNICATION::in write data **2019-07-15 20:28:32 DEBUG::COMMUNICATION::curl_easy_perform() success** **2019-07-15 20:28:32 DEBUG::COMMUNICATION::MUD File Server returned success state.** **2019-07-15 20:28:32 DEBUG::MUD_FILE_OPERATIONS::IN ****NEW**** MUD and SIG FILE RETRIEVED!!!** **2019-07-15 20:28:32 DEBUG::GENERAL::IN ****NEW**** validateMudFileWithSig()** **2019-07-15 20:28:32 DEBUG::GENERAL::openssl cms -verify -in /etc/osmud/state/mudfiles/yikesmain.p7s -inform DER -content /etc/osmud/state/mudfiles/yikesmain.json -purpose any > /dev/null** 2019-07-15 20:28:32 DEBUG::GENERAL::IN ****NEW**** executeMudWithDhcpContext() 2019-07-15 20:28:32 DEBUG::GENERAL::/etc/osmud/create_mud_db_entry.sh -d /etc/osmud/state/mudfiles/mudStateFile.txt -i 192.168.20.222 -m b8:27:eb:eb:6c:8b -c main-pi-Build2 -u https://mudfiles.nist.getyikes.com/yikesmain.json -f /etc/osmud/state/mudfiles/yikesmain.json **[Logs omitted for brevity]** **2019-07-15 20:28:32 DEBUG::GENERAL::***www.updateserver.com* **2019-07-15 20:28:33 DEBUG::GENERAL::192.168.20.4** **2019-07-15 20:28:33 DEBUG::GENERAL::192.168.20.238** **2019-07-15 20:28:33 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d wan -i 192.168.20.222 -a any -j 192.168.20.4 -b 443:443 -p**

Test Case Field	Description
	tcp -n cl2-frdev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222
	2019-07-15 20:28:33 DEBUG::GENERAL::/etc/osmud/create_ip_fw_rule.sh -s lan -d wan -i 192.168.20.222 -a any -j 192.168.20.238 -b 443:443 -p tcp -n cl2-frdev -t ACCEPT -f all -c main-pi-Build2 -k /tmp/osmud -r 192.168.20.222 **[Logs omitted for brevity]** 2019-07-15 20:28:33 DEBUG::GENERAL::**Success returned from for transaction**
	Router/PEP: `config rule` `option enabled '1'` `option name 'mud_192.168.20.222_main-pi-Build2_cl2-frdev'` `option target ACCEPT` `option src lan` `option dest wan` `option proto tcp` `option family ipv4` **`option src_ip 192.168.20.222`** **`option dest_ip 192.168.20.4`** `option dest_port 443:443` `config rule` `option enabled '1'` `option name 'mud_192.168.20.222_main-pi-Build2_cl2-frdev'` `option target ACCEPT` `option src lan` `option dest wan` `option proto tcp` `option family ipv4` **`option src_ip 192.168.20.222`** **`option dest_ip 192.168.20.238`** `option dest_port 443:443`
	Procedure 4:

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

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

338 Test case IoT-9-v6 is identical to test case IoT-9-v4 except that IoT-9-v6 uses IPv6 addresses rather than
339 IPv4 addresses.

3.1.2.10 Test Case IoT-10-v4

Table 3-111: Test Case IoT-10-v4

Test Case Field	Description
Parent Requirements	(CR-12) The IoT DDoS example implementation shall include a MUD manager that uses a cached MUD file rather than retrieve a new one if the cache-validity time period has not yet elapsed for the MUD file indicated by the MUD URL**.** The MUD manager should fetch a new MUD file if the cache-validity time period has already elapsed.
Testable Requirements	(CR-12.a) The MUD manager shall check if the file associated with the MUD URL is present in its cache and shall determine that it is. (CR-12.a.1) The MUD manager shall check whether the amount of time that has elapsed since the cached file was retrieved is less than or equal to the number of hours in the cache-validity value for this MUD file. If so, the MUD manager shall apply the contents of the cached MUD file. (CR-12.a.2) The MUD manager shall check whether the amount of time that has elapsed since the cached file was retrieved is greater than the number of hours in the cache-validity value for this MUD file. If so, the MUD manager may (but does not have to) fetch a new file by using the MUD URL received.
Description	Shows that, upon connection to the network, a MUD-enabled IoT device used in the IoT DDoS example implementation has its MUD PEP router/switch automatically configured to enforce the route filtering that is described in the cached MUD file for that device's MUD URL, assuming that the amount of time that has elapsed since the cached MUD file was retrieved is less than or equal to the number of hours in the file's cache-validity value. If the cache validity has expired for the respective file, the MUD manager should fetch a new MUD file from the MUD file server.
Associated Test Case(s)	N/A
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-1, ID.AM-2, ID.AM-3, PR.DS-5, DE.AE-1, PR.AC-4, PR.AC-5, PR.IP-1, PR.IP-3, PR.DS-2, PR.PT-3

Test Case Field	Description
IoT Device(s) Under Test	To be determined (TBD) (Not testable in Build 2's preproduction of Yikes!)
MUD File(s) Used	TBD (Not testable in Build 2's preproduction of Yikes!)
Preconditions	1. The MUD PEP router/switch does not yet have any configuration settings pertaining to the IoT device being used in the test. 2. The MUD file for the IoT device being used in the test is identical to the MUD file provided in Section 3.1.3.
Procedure	Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. 1. Run test IoT-1-v4 (or IoT-1-v6). 2. Within 24 hours (i.e., within the cache-validity period for the MUD file) of running test IoT-1-v4 (or IoT-1-v6), verify that the IoT device that was connected during test IoT-1-v4 (or IoT-1-v6) is still up and running on the network. Power on a second IoT device that has been configured to emit the same MUD URL as the device that was connected during test IoT-1-v4 (or IoT-1-v6), and connect it to the test network. This should set in motion the following series of steps, which should occur automatically. 3. The IoT device automatically emits a DHCPv4 message containing the device's MUD URL (IANA code 161). (Note that in the v6 version of this test, IPv6, DHCPv6, and IANA code 112 will be used.) 4. The DHCP server receives the DHCPv4 message containing the IoT device's MUD URL. 5. The DHCP server offers an IP address lease to the newly connected IoT device. 6. The IoT device requests this IP address lease, which the DHCP server acknowledges. 7. The DHCP server sends the MUD URL to the MUD manager. 8. The MUD manager determines that it has this MUD file cached and checks that the amount of time that has elapsed since the cached

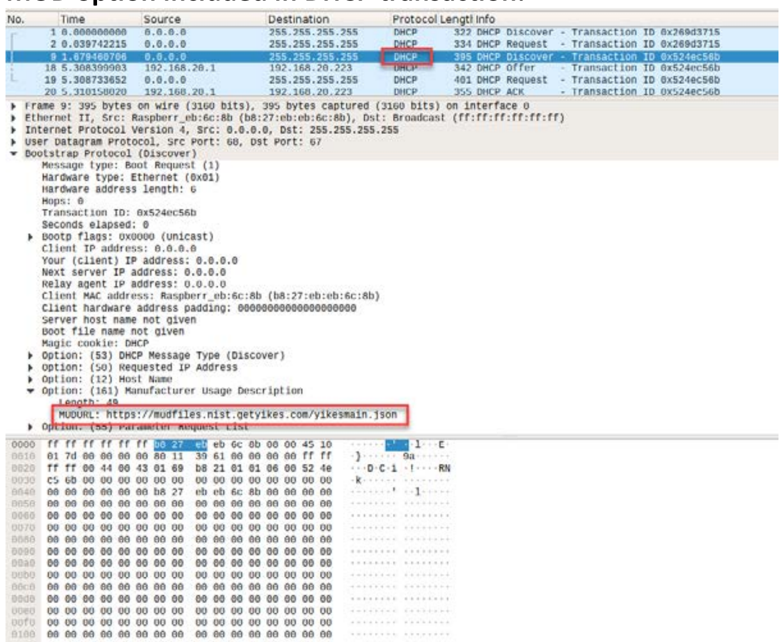
Test Case Field	Description
	file was retrieved is less than or equal to the number of hours in the cache-validity value for this MUD file. If the cache validity has been exceeded, the MUD manager will fetch a new MUD file. (Run the test both ways—with a cache-validity period that has expired and with one that has not.) 9. The MUD manager translates the MUD file's contents into appropriate route filtering rules and installs these rules onto the MUD PEP for the IoT device in question so that this router/switch is now configured to enforce the policies specified in the MUD file.
Expected Results	The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to enforce the policies specified in the IoT device's MUD file. The expected configuration should resemble the following. **Cache is valid** (the MUD manager does NOT retrieve the MUD file from the MUD file server)**:** TBD (Not testable in Build 2's preproduction of Yikes!) **Cache is not valid** (the MUD manager does retrieve the MUD file from the MUD file server)**:** TBD (Not testable in Build 2's preproduction of Yikes!) All protocol exchanges described in steps 1–9 above are expected to occur and can be viewed via Wireshark if desired. If the router/switch does not get configured in accordance with the MUD file, each exchange of DHCP and MUD-related protocol traffic should be viewed on the network via Wireshark to determine which transactions did not proceed as expected, and the observed and absent protocol exchanges should be described here.
Actual Results	TBD (Not testable in Build 2's preproduction of Yikes!)
Overall Results	TBD (Not testable in Build 2's preproduction of Yikes!)

Test case IoT-10-v6 is identical to test case IoT-10-v4 except that IoT-10-v6 tests requirement CR-1.a.2, whereas IoT-10-v4 tests requirement CR-1.a.1. Hence, as explained above, test IoT-10-v6 uses IPv6, DHCPv6, and IANA code 112 instead of using IPv4, DHCPv4, and IANA code 161.

3.1.2.11 Test Case IoT-11-v4

Table 3-12: Test Case IoT-11-v4

Test Case Field	Description
Parent Requirements	(CR-1) The IoT DDoS example implementation shall include a mechanism for associating a device with a MUD file URL (e.g., by having the MUD-enabled IoT device emit a MUD file URL via DHCP, LLDP, or X.509 or by using some other mechanism to enable the network to associate a device with a MUD file URL).
Testable Requirements	(CR-1.a) Upon initialization, the MUD-enabled IoT device shall broadcast a DHCP message on the network, including at most one MUD URL, in https scheme, within the DHCP transaction. (CR-1.a.1) The DHCP server shall be able to receive DHCPv4 DISCOVER and REQUEST with IANA code 161 (OPTION_MUD_URL_V4) from the MUD-enabled IoT device.
Description	Shows that the IoT DDoS example implementation includes IoT devices that can emit a MUD URL via DHCP
Associated Test Case(s)	N/A
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-1
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*Yikesmain.json*
Preconditions	Device has been developed to emit MUD URL in DHCP transaction

Test Case Field	Description
Procedure	1. Power on a device and connect it to the network. 2. Verify that the device emits a MUD URL in a DHCP transaction. (Use Wireshark to capture the DHCP transaction with options present.)
Expected Results	DHCP transaction with MUD option 161 enabled and MUD URL included
Actual Results	**MUD option included in DHCP transaction:** 
Overall Results	Pass

347 ## 3.1.3 MUD Files

348 This section contains the MUD files that were used in the Build 2 functional demonstration.

349 ### *3.1.3.1 Fe-controller.json*

350 The complete Fe-controller.json MUD file has been linked to this document. To access this MUD file
351 please click the link below.

352 Fe-controller.json

3.1.3.2 Fe-localnetwork-from2.json

The complete Fe-localnetwork-from2.json MUD file has been linked to this document. To access this MUD file please click the link below.

Fe-localnetwork-from2.json

3.1.3.3 Fe-localnetwork-to2.json

The complete fe-localnetwork-to2.json MUD file has been linked to this document. To access this MUD file please click the link below.

Fe-localnetwork-to2.json

3.1.3.4 Fe-manufacturer1.json

The complete Fe-manufacturer1.json MUD file has been linked to this document. To access this MUD file please click the link below.
Fe-manufacturer1.json

3.1.3.5 Fe-manufacturer2.json

The complete Fe-manufacturer2.json MUD file has been linked to this document. To access this MUD file please click the link below.

Fe-manufacturer2.json

3.1.3.6 Fe-mycontroller.json

The complete Fe-mycontroller.json MUD file has been linked to this document. To access this MUD file please click the link below.

Fe-mycontroller.json

3.1.3.7 Fe-samemanufacturer-from2.json

The complete Fe-samemanufacturer-from2.json MUD file has been linked to this document. To access this MUD file please click the link below.

Fe-samemanufacturer-from2.json

3.1.3.8 Fe-samemanufacturer-to2.json

The complete Fe-samemanufacturer-to2.json MUD file has been linked to this document. To access this MUD file please click the link below.

Fe-samemanufacturer-to2.json

3.1.3.9 Yikesmain.json

The complete Yikesmain.json MUD file has been linked to this document. To access this MUD file please click the link below.

Yikesmain.json

3.2 Demonstration of Non-MUD-Related Capabilities

In addition to supporting MUD, Build 2 supports capabilities with respect to device discovery, identification, categorization, and application of traffic rules based on device make and model. Table 3-13 lists the non-MUD-related capabilities that were demonstrated for Build 2. Before examining these capabilities, however, it is instructive to define terminology and provide an overview of Build 2's non-MUD-related capabilities.

3.2.1 Terminology

The terminology that is used to describe non-MUD capabilities is not standardized. To avoid confusion, we offer the following definitions for use in this section:

- Device discovery—detection that a device is on the network
- Device identity—an identifier that a build assigns to the device and uses to keep track of the device. In Build 2, when a device is discovered, it is assigned a unique identity.
- Device identification—determination of the device's make (i.e., manufacturer) and model. In Build 2, each make and model combination may be associated with internet traffic rules that, if present, will be applied to all devices having that same make and model.
- Category—a predefined class to which devices are assigned based on their make and model. Each category is associated with traffic rules (for both local traffic and internet traffic) that will be applied to all devices in that category.
- Device categorization—determination of which of the build's predefined categories to which to assign the device. The device's make and model determine its category, e.g., if the device is determined to be a Samsung Galaxy S8, it is placed in the phone category.
- Traffic policy—a set of traffic rules that may be associated with a category of devices or a set of devices having the same make and model; the traffic policy determines to what other local devices and remote domains these devices are permitted to initiate communication.

3.2.2 General Overview of Build 2's Non-MUD Functionality

Once Build 2 discovers a device on the network, it applies the following non-MUD capabilities to it:

- automatic (if possible) identification of the device's make (i.e., manufacturer) and model

- categorization of the device based on its make and model
- association of the device category with a traffic policy that indicates what communication devices in that category are permitted to initiate. This policy consists of rules that apply to both local and internet communications. The rules in this policy can be viewed using the Yikes! User Interface (UI). By selecting the specific category (e.g., “cellphone” or “computer”) on the UI Categories page, one can see two categories of rules, Local Network and Internet:
    - Internet rules that may be set to either
        - Allow All Internet Traffic, which indicates that all devices in this category are permitted to initiate communications to all internet domains

        or

        - IoT Specific Sites, which indicates that there may be additional rules configured on the router that apply to specific makes and models of devices in this category and that restrict the internet sites to which those devices are permitted to initiate communications. (These per-make-and-model rules are stored in the cloud and viewed using the Yikes! UI. The IoT Devices tab displays the list of domain names to which communications may be initiated. For this version of the Yikes! cloud, these rules were set manually based on Build 2 test cases.)
    - Local Network rules that may be set to either
        - Allow All, which, if set, indicates that devices in this category are permitted to initiate communications to all other devices on the local network

        or

        - any combination of other categories (cell phones, printers, tablets, printers, etc.) These indicate the other categories of devices on the local network to which devices in this category are permitted to initiate communications.

3.2.3 Non-MUD-Related Functional Capabilities

Table 3-13 lists the non-MUD-related capabilities that were demonstrated for Build 2. We use the letter “Y” as a prefix for these functional capability identifiers in the table below because these capabilities are specific to Build 2, which uses Yikes! equipment.

Table 3-133: Non-MUD-Related Functional Capabilities Demonstrated

Functional Capability	Parent Capability	Subrequirement 1	Subrequirement 2	Exercise ID
Y-1	**Device Identification–**The device is detected, and its make and model are identified upon connection to the network.			
Y-1.a		The non-MUD-capable device's **make and model are correctly identified** based on some combination of information such as the device's media access control (MAC) address, DHCP header information, and lookup in repositories.		YnMUD-1-v4, Yn-MUD-1-v6
Y-1.b		The non-MUD-capable **device's make and model cannot be identified.**		YnMUD-1-v4, Yn-MUD-2-v6
Y-1.c		The non-MUD-capable **device's make and model can be assigned manually.**		YnMUD-2-v4, Yn-MUD-3-v6
Y-2	**Device Categorization–**The device is correctly categorized according to its type (e.g., phone, printer, computer, watch)			

Functional Capability	Parent Capability	Subrequirement 1	Subrequirement 2	Exercise ID
	upon connection to the network.			
Y-2.a		The non-MUD-capable **device is correctly categorized based on its make and model.**	The device make and model were determined using some combination of MAC address, DHCP header information, and lookup in repositories.	YnMUD-1-v4, Yn-MUD-1-v6
Y-2.b		The **make and model of the non-MUD-capable device cannot be determined.**	The non-MUD-capable **device is designated as uncategorized.**	YnMUD-1-v4, Yn-MUD-1-v6
Y-2.c		The non-MUD-capable **device's category can be assigned manually.**		YnMUD-2-v4, Yn-MUD-3-v6
Y-3	**Rules regarding initiation of (south-north) communications to internet sites** by the non-MUD-capable device **are enforced according to** rules associated with **the device's category and, possibly, its make and model.**			

Functional Capability	Parent Capability	Subrequirement 1	Subrequirement 2	Exercise ID
Y-3.a		The device's category has **the Allow All Internet Traffic rule set** (i.e., the IoT Specific Sites rule is not set).	The device will be **permitted to connect to any internet location.**	YnMUD-3-v4, YnMUD-3-v6
Y-3.b		The device's category has **the IoT Specific Sites rule set,** indicating that there may be **rules associated with specific makes and models of devices in this category** that further restrict the internet locations to which those devices are able to initiate communications.		
Y-3.b.1			There **are (south to north) rules associated with the device's make and model,** so the device will be **allowed to initiate communications with the internet sites permitted by those rules but prohibited from initiating communications to all other internet sites.**	YnMUD-3-v4, YnMUD-3-v6
Y-3.b.2			There are **no (south to north) rules associated with a device's make and model,** so that device will be **allowed to**	YnMUD-3-v4, YnMUD-3-v6

Functional Capability	Parent Capability	Subrequirement 1	Subrequirement 2	Exercise ID
			initiate communications with all internet sites.	
Y-3.c			There **are (north to south) rules associated with a device's make and model,** so that device will be **allowed to receive communications from the internet sites permitted by the rules but prohibited from receiving communications from all other internet sites.**	**N/A for IPv4** due to NAT
Y-3.d			There are **no (north to south) rules associated with a device's make and model,** so that device will be **allowed to receive communications from all internet sites.**	**N/A for IPv4** due to NAT
Y-4	**Lateral (east-west) communications** of the non-MUD-capable device to other devices on the local network are **enforced according to the policy associated with**			

Functional Capability	Parent Capability	Subrequirement 1	Subrequirement 2	Exercise ID
	the device's category.			
Y-4.a		**A rule** associated with the device's category **permits the device to initiate communications with local devices in category X, but** there is **no such rule that permits the device to initiate communications with local devices in category Y.**		YnMUD-4-v4, YnMUD-4-v6
Y-4.a.1			The device will be allowed to **initiate communications to** any local device that is in **category X.**	YnMUD-4-v4, YnMUD-4-v6
Y-4.a.2			The device will be **prohibited from initiating communications to** any local device that is in **category Y.**	YnMUD-4-v4, YnMUD-4-v6
Y-5	**In response to threat information, all devices on the local network are prohibited from visiting specific domains and IP addresses.**			

Functional Capability	Parent Capability	Subrequirement 1	Subrequirement 2	Exercise ID
Y-5.a		Threat intelligence indicates a **specific internet domain that should not be trusted.**	**Devices are prohibited from initiating communications to the internet domain listed in the threat intelligence. In addition, they are prohibited from initiating communications to any other domains and IP addresses that are associated with the same threat campaign as this domain.**	YnMUD-5-v4, YnMUD-5-v6
Y-5.b		Threat intelligence indicates a **specific IP address that should not be trusted.**	**Devices are prohibited from initiating communications to the IP address listed in the threat intelligence. In addition, they are prohibited from initiating communications to any other IP addresses and domains that are associated with the same threat campaign as this IP address.**	YnMUD-6-v4, YnMUD-6-v6
Y-5.c		Threat intelligence was received more than 24 hours prior, indicating domains and IP addresses that should not be trusted, and those domains and IP addresses were blocked by ACLs installed on the router.	**After 24 hours, these ACLs are no longer configured in the router.**	YnMUD-7-v4, YnMUD-7-v6

3.2.4 Exercises to Demonstrate the Above Non-MUD-Related Capabilities

This section contains the exercises that were performed to verify that Build 2 supports the non-MUD-related capabilities listed in Table 3-13.

To support these tests, the following domains must be available on the internet (i.e., outside the local network):

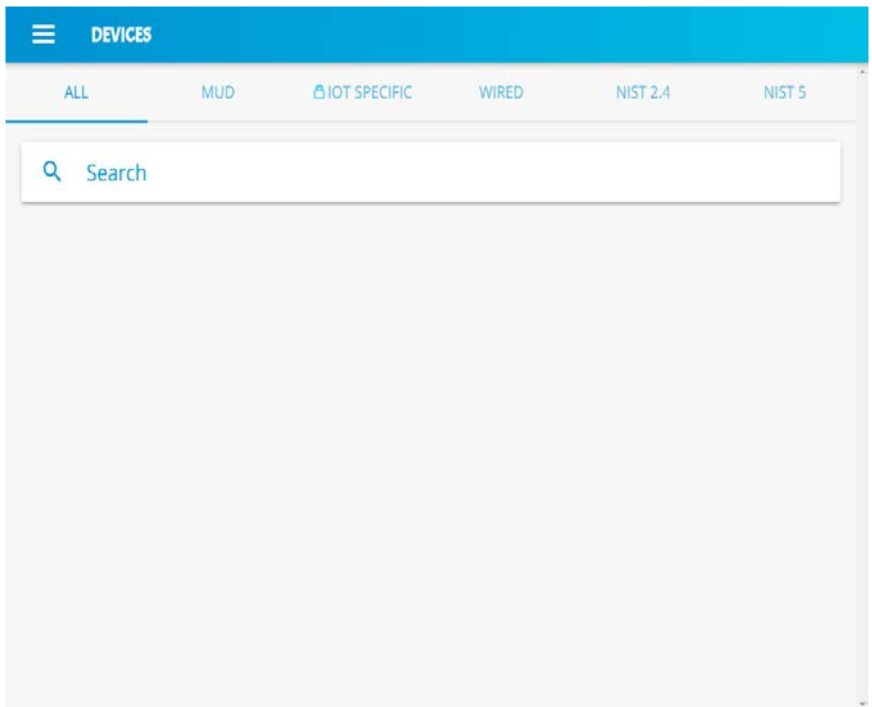
- www.google.com
- www.osmud.org
- www.trytechy.com

3.2.4.1 *Exercise YnMUD-1-v4*

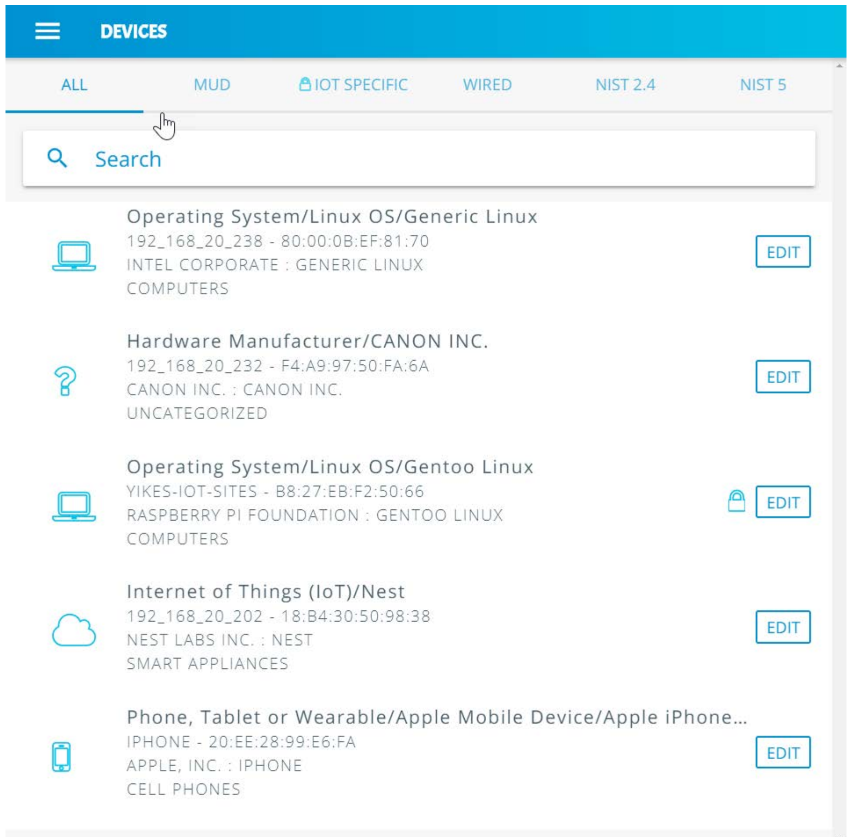
Table 3-144: Exercise YnMUD-1-v4

Exercise Field	Description
Parent Capability	(Y-1) Device Identification–The device is detected, and its make and model are identified upon connection to the network. (Y-2) Device Categorization–The device is correctly categorized according to its type (e.g., phone, printer, computer, watch) upon connection to the network.
Subrequirement(s) of Parent Capability to Be Demonstrated	(Y-1.a) The non-MUD-capable device's make and model are correctly identified based on some combination of information such as the device's MAC address, DHCP header information, and lookup in repositories. (Y-2.a) The non-MUD-capable device is correctly categorized based on its make and model. The device make and model were determined using some combination of MAC address, DHCP header information, and lookup in repositories. (Y-1.b) The non-MUD-capable device's make and model cannot be identified. (Y-2.b) The make and model of the non-MUD-capable device cannot be determined. The non-MUD-capable device is designated as uncategorized.
Description	Verify that upon detection, when possible, the make (i.e., manufacturer) and model of a non-MUD-capable device are identified correctly based on some combination of its MAC address, DHCP header info, and lookup

Exercise Field	Description
	through the Yikes! cloud service; the device is assigned to the correct category; and it is assigned a unique identity. In addition, verify that a non-MUD-capable device whose make and model cannot be determined will be assigned to the "uncategorized" category.
Associated Exercises	N/A
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-1, ID.AM-2, ID.AM-3, DE.AE-1, DE.CM-1
IoT Device(s) Used	- Laptop–with network-scanning software loaded - Cell phone–with network-scanning application loaded - Printer - Nest Camera to serve as an actual IoT device - Raspberry PI emulating an IoT device
Policy Used	N/A
Preconditions	The Yikes! router is installed on the local network and connected to the internet. The Yikes! account is set up and available to the user at https://nist.getyikes.com. The IoT devices listed above are available to be connected to the local network.
Procedure	1. Use the Yikes! UI to determine whether any devices are present (either active or inactive) on the network. 2. If any devices are present, they are to be deleted. Then verify that no devices are present (either active or inactive) on the network. 3. Connect each of the five devices above to the local network. 4. Validate that each device has appeared in Yikes! UI.
Demonstrated Results	Access the Yikes! UI, go to the Devices page, click the ALL tab, and verify that the following information is present, showing that each device has been given a unique identifier (not necessarily ID_X), has had its make

Exercise Field	Description
	and model correctly identified (if possible), and has been categorized appropriately: **Procedures 1–2:**  **Procedures 3–4:**

Exercise Field	Description



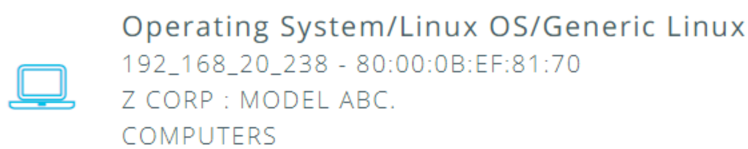
Device	Device ID	Make	Model	Category
Laptop	ID_1	Dell	E6540	Computer
Cell Phone	ID_2	Apple	iPhone 7	Cell Phone
Printer	ID_3	Canon	MX922	Uncategorized
Camera	ID_4	Nest	Indoor Cam	Smart Device
Test-PI	ID_5	Raspberry	Pi B+	Computer

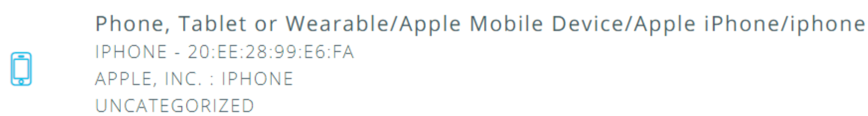
Exercise YnMUD-1-v6 is identical to exercise YnMUD-1-v4 except that it uses IPv6 instead of IPv4.

3.2.4.2 Exercise YnMUD-2-v4

Table 3-15: Exercise YnMUD-2-v4

Exercise Field	Description
Parent Capability	(Y-1) Device Identification–The device is detected, and its make and model are identified upon connection to the network. (Y-2) Device Categorization–The device is correctly categorized according to its type (e.g., phone, printer, computer, watch) upon connection to the network.
Subrequirement(s) of Parent Capability to Be Demonstrated	(Y-1.c) The non-MUD-capable device's make and model can be assigned manually. (Y-2.c) The non-MUD-capable device's category can be assigned manually.
Description	Verify that a non-MUD-capable device can have its make, model, or category assigned manually.
Associated Exercises	YnMUD-1-v4
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-1, ID.AM-3
IoT Device(s) Used	Same as for exercise YnMUD-1-v4
Policy Used	N/A
Preconditions	Same as for exercise YnMUD-1-v4
Procedure	1. Run exercise YnMUD-1-v4. 2. Use the Yikes! UI to modify the make (i.e., manufacturer) of Device X to be Z Corp. 3. Use the Yikes! UI to modify the model of Device X to be Model ABC. 4. Use the Yikes! UI to modify the category of the cell phone to be Uncategorized.

<table>
<tr><th>Exercise Field</th><th>Description</th></tr>
<tr><td>Demonstrated Results</td><td>
Access the Yikes! UI, go to the Device tab, and verify that the following information is present:
Procedure 1: Completed; excluded for brevity
Procedures 2–3:


Procedure 4:


<table>
<tr><th>Device</th><th>Device ID</th><th>Make</th><th>Model</th><th>Category</th></tr>
<tr><td>Laptop</td><td>ID_1</td><td>Dell</td><td>E6540</td><td>Computer</td></tr>
<tr><td>Cell Phone</td><td>ID_2</td><td>Apple</td><td>iPhone7</td><td>Cell phone</td></tr>
<tr><td>Printer</td><td>ID_3</td><td>Canon</td><td>MX922</td><td>Uncategorized</td></tr>
<tr><td>Camera</td><td>ID_4</td><td>Nest</td><td>Indoor Cam</td><td>Smart Device</td></tr>
<tr><td>Test-PI</td><td>ID_5</td><td>Raspberry</td><td>Pi B+</td><td>Computer</td></tr>
</table>
</td></tr>
</table>

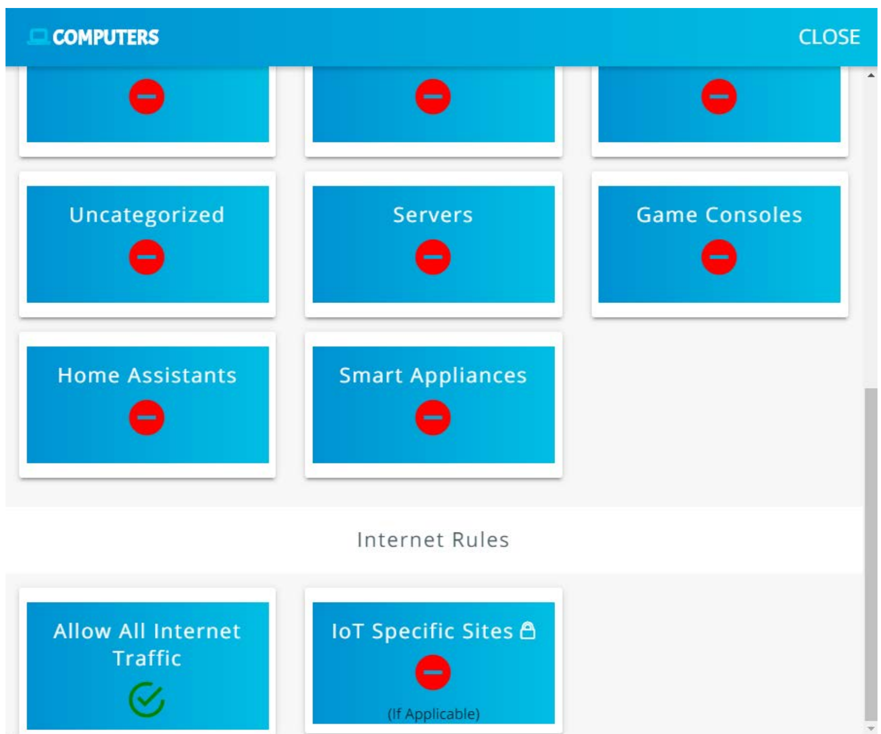
Exercise YnMUD-2-v6 is identical to exercise YnMUD-2-v4 except that it uses IPv6 instead of IPv4.

455 ### 3.2.4.3 Exercise YnMUD-3-v4

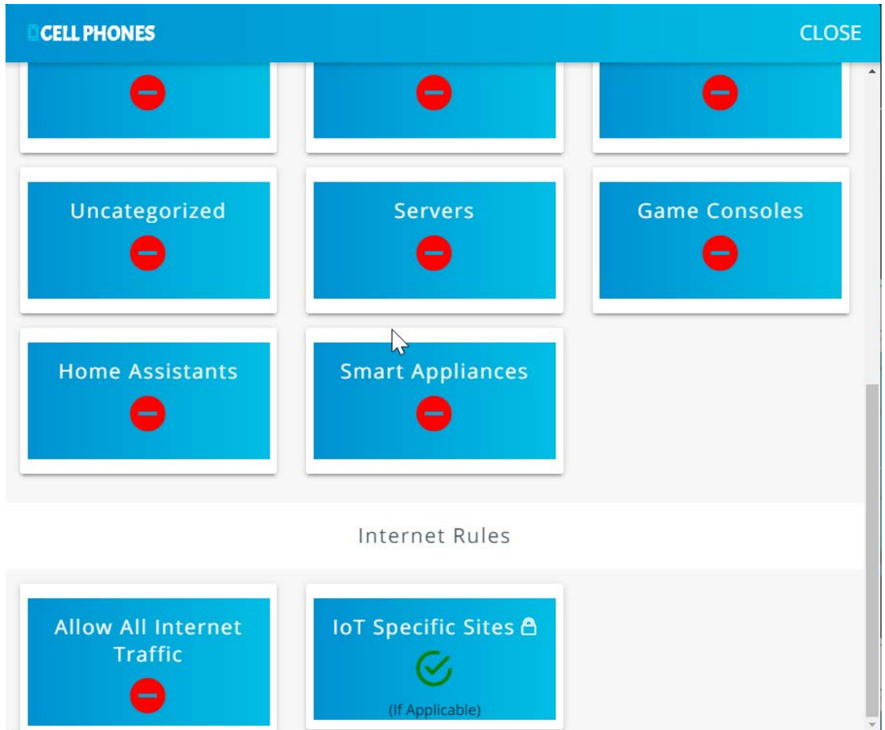
456 **Table 3-16: Exercise YnMUD-3-v4**

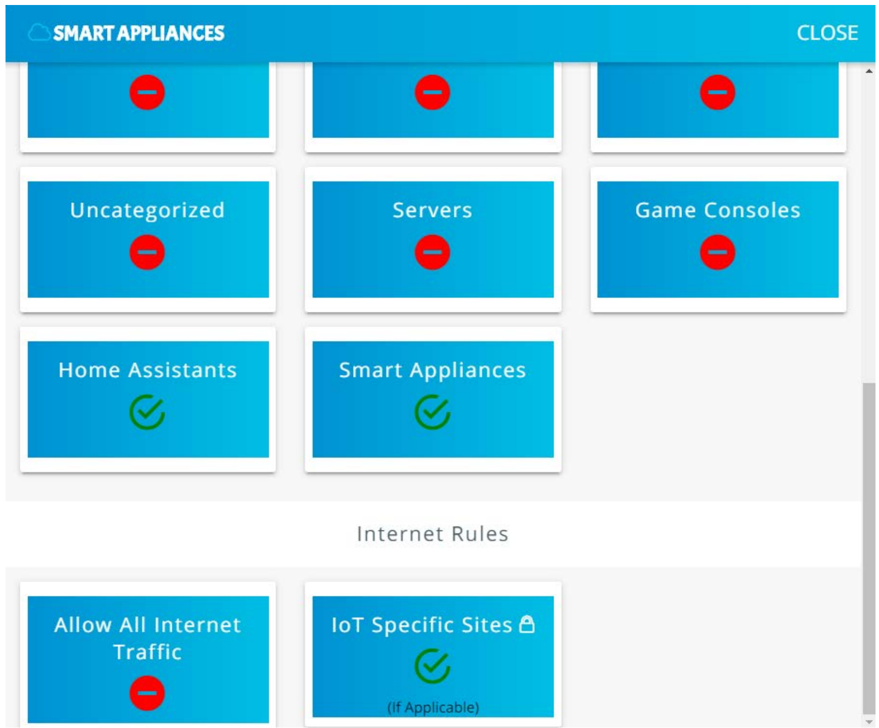
Exercise Field	Description
Parent Capability	(Y-3) Rules regarding initiation of (south-north) communications to internet sites by the non-MUD-capable device are enforced according to rules associated with the device's category and, possibly, its make and model.
Subrequirement(s) of Parent Capability to Be Demonstrated	(Y-3.a) The device's category has the Allow All Internet Traffic rule set (i.e., the IoT Specific Sites rule is not set). The device will be permitted to connect to any internet location. (Y-3.b) The device's category has the IoT Specific Sites rule set, indicating that there may be rules associated with specific makes and models of devices in this category that further restrict the internet locations to which those devices are able to initiate communications. (Y-3.b.1) There are (south to north) rules associated with the device's make and model, so the device will be allowed to initiate communications with the internet sites permitted by those rules but prohibited from initiating communications to all other internet sites. (Y-3.b.2) There are no (south to north) rules associated with a device's make and model, so that device will be allowed to initiate communications with all internet sites.
Description	Verify that once a device has been categorized, the device will be able to initiate communications to internet sites as constrained by any south-to-north rules that may be in place on the router that pertain to the device's make and model. In particular: - If the IoT Specific Sites rule is not set for the device's category, the device will be permitted to initiate communication with all internet sites. - If the IoT Specific Sites rule is set for this device's category and there are south-to-north rules on the router that apply to the device's make and model, the device will be restricted to initiating communications to only those internet sites permitted by those rules on the router.

Exercise Field	Description
	- If the IoT Specific Sites rule is set for this device's category but there are no south-to-north rules on the router that apply to the device's make and model, the device will not be permitted to initiate communication with any internet sites.
Associated Exercises	N/A
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-3, ID.AM-4, PR.AC-1, PR.AC-3, PR.AC-4, PR.AC-5
IoT Device(s) Used	- Laptop - iPhone 7 cell phone - Raspberry Pi
Policy Used	In the Yikes! UI, the Smart Appliances and Cell Phone internet rule is set to IoT Specific Sites. On the router, one ACL rule applies to the Raspberry Pi that permits it to visit www.getyikes.com and www.osmud.org, but there are no device-specific rules that apply to cell phones. On the router, there are no rules that apply to iPhone 7 devices. In the Yikes! UI, the Computer internet rule is set to Allow All Internet Traffic rather than to IoT Specific Sites.
Preconditions	The Smart Appliance, Cell Phone, and Computer category rules in the Yikes! UI and the ACL rules on the router are configured as described in the policy row above. (The presence of the Smart Appliances, Cell Phone, and Computer category rules can be verified by accessing the Yikes! UI. Using the UI, we should also be able to see the fully qualified domain names (FQDNs) of the sites that the rules permit each make and model of smart appliance and cell phone to access if any exist. The presence of the ACL rules can be verified only by logging in to the router.)
Procedure	1. Validate Yikes! UI configuration for Smart Appliances, Cell Phone, and Computer categories. 2. Connect the iPhone 7, Raspberry Pi, and laptop to the network. 3. Validate that the Raspberry Pi can browse to www.osmud.org and www.getyikes.com but not to www.google.com.

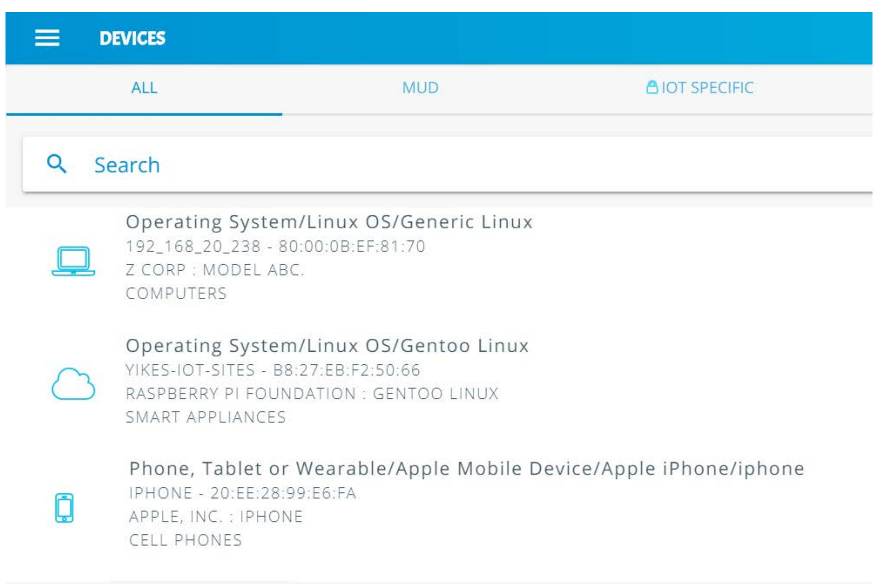
<table>
<tr><th>Exercise Field</th><th>Description</th></tr>
<tr><td></td><td>
4. Validate that the iPhone 7 cannot browse to www.google.com, www.osmud.org, and www.getyikes.com.
5. Validate that a computer on the network can browse to www.google.com, www.osmud.org, and www.getyikes.com.
6. Log in to the router to validate that the appropriate ACL rules are in place.
</td></tr>
<tr><td>Demonstrated Results</td><td>
Cell phone access is permitted and prohibited as expected in the procedure steps above. Computer access is permitted as expected.
Procedure 1:
Computers


Cell Phones
</td></tr>
</table>

Exercise Field	Description
	 **Smart Appliances**

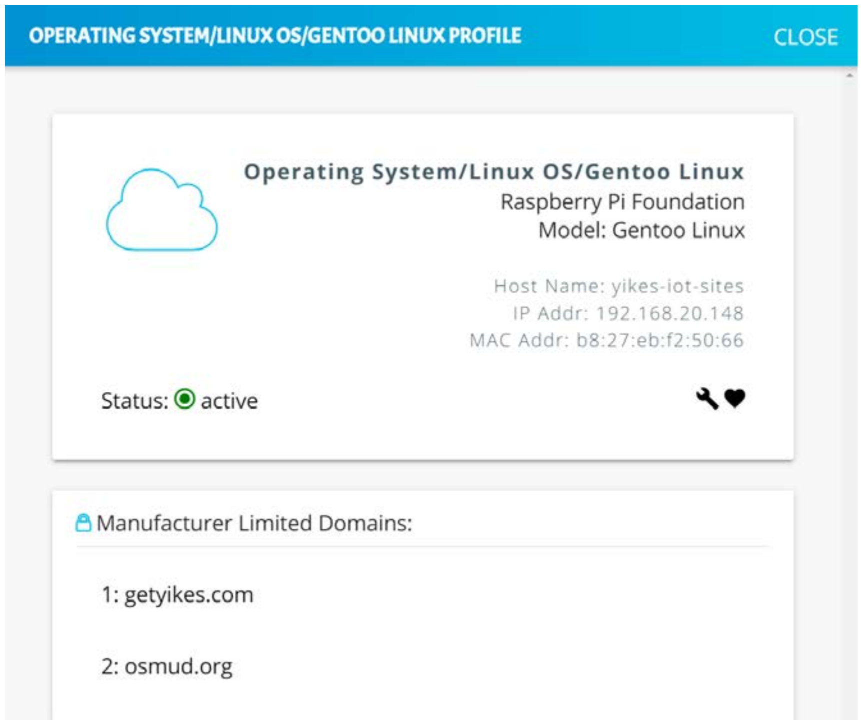
<table>
<tr><th>Exercise Field</th><th>Description</th></tr>
<tr><td></td><td>


Procedure 2:



</td></tr>
</table>

Exercise Field	Description
	Procedure 3: **Smart Appliance**



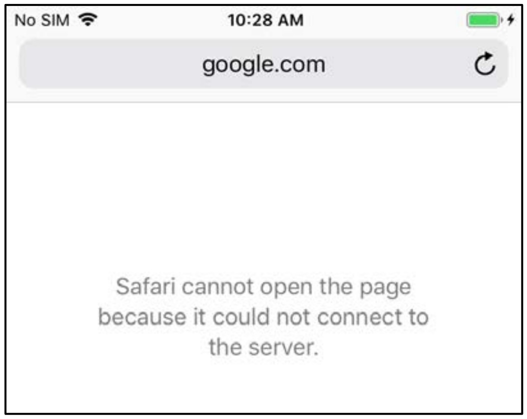
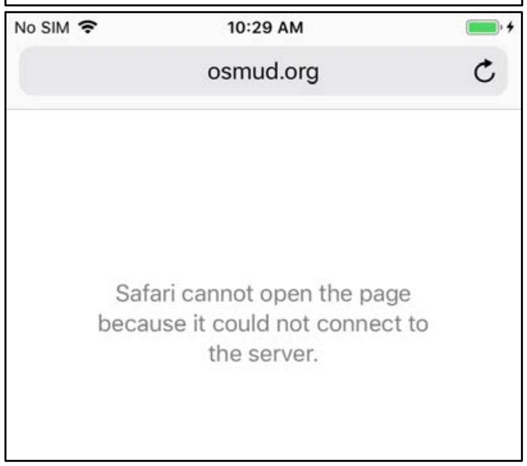
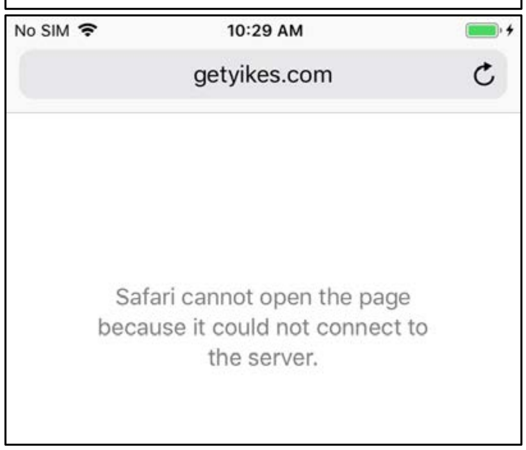
Yikes! approved communication:

```
pi@yikes-iot-sites:~ $ wget https://osmud.org
--2019-07-29 10:28:56--  https://osmud.org/
Resolving osmud.org (osmud.org)... 198.71.233.87
Connecting to osmud.org
(osmud.org)|198.71.233.87|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: ‘index.html.1’

index.html.1          [ <=>              ]  24.12K  -
-.-KB/s    in 0.02s

2019-07-29 10:28:58 (1.30 MB/s) - ‘index.html.1’
saved [24697]
```

Exercise Field	Description
	pi@yikes-iot-sites:~ $ wget https://getyikes.com --2019-07-29 10:29:05-- https://getyikes.com/ Resolving getyikes.com (getyikes.com)... 54.213.16.153 Connecting to getyikes.com (getyikes.com)|54.213.16.153|:443... connected. HTTP request sent, awaiting response... 200 OK Length: 15759 (15K) [text/html] Saving to: 'index.html.2' index.html.2 100%[===================>] 15.39K --.-KB/s in 0.1s 2019-07-29 10:29:06 (119 KB/s) - 'index.html.2' saved [15759/15759] **Yikes! unapproved communication:** pi@yikes-iot-sites:~ $ wget https://www.google.com --2019-07-29 10:29:29-- https://www.google.com/ Resolving www.google.com (www.google.com)... 74.125.136.99, 74.125.136.103, 74.125.136.106, ... Connecting to www.google.com (www.google.com)|74.125.136.99|:443... failed: Connection refused. Connecting to www.google.com (www.google.com)|74.125.136.103|:443... failed: Connection refused. Connecting to www.google.com (www.google.com)|74.125.136.106|:443... failed: Connection refused. Connecting to www.google.com (www.google.com)|74.125.136.147|:443... failed: Connection refused. Connecting to www.google.com (www.google.com)|74.125.136.105|:443... failed: Connection refused. Connecting to www.google.com (www.google.com)|74.125.136.104|:443... failed: Connection refused. Connecting to www.google.com (www.google.com)|2607:f8b0:4002:c06::6a|:443... failed: Network is unreachable.

	Procedure 4: **Cell Phone**   
	Procedure 5: **Computers**

Exercise Field	Description
	[mud@localhost ~]$ wget www.google.com --2019-07-23 14:47:52-- http://www.google.com/ Resolving www.google.com (www.google.com)... 172.217.164.68, 2607:f8b0:4002:c08::67 Connecting to www.google.com (www.google.com)|172.217.164.68|:80... connected. HTTP request sent, awaiting response... 200 OK Length: unspecified [text/html] Saving to: 'index.html.13' [<=>] 11,492 --.-K/s in 0.005s 2019-07-23 14:47:53 (2.30 MB/s) - 'index.html.13' saved [11492] [mud@localhost ~]$ wget osmud.org --2019-07-23 14:48:11-- http://osmud.org/ Resolving osmud.org (osmud.org)... 198.71.233.87 Connecting to osmud.org (osmud.org)|198.71.233.87|:80... connected. HTTP request sent, awaiting response... 301 Moved Permanently Location: https://osmud.org/ [following] --2019-07-23 14:48:11-- https://osmud.org/ Connecting to osmud.org (osmud.org)|198.71.233.87|:443... connected. HTTP request sent, awaiting response... 200 OK Length: unspecified [text/html] Saving to: 'index.html.14' [<=>] 24,697 --.-K/s in 0.009s 2019-07-23 14:48:11 (2.73 MB/s) - 'index.html.14' saved [24697] [mud@localhost ~]$ wget getyikes.com --2019-07-23 14:48:36-- http://getyikes.com/ Resolving getyikes.com (getyikes.com)... 54.213.16.153 Connecting to getyikes.com (getyikes.com)|54.213.16.153|:80... connected. HTTP request sent, awaiting response... 301 Moved Permanently Location: https://getyikes.com/ [following] --2019-07-23 14:48:36-- https://getyikes.com/ Connecting to getyikes.com (getyikes.com)|54.213.16.153|:443... connected. HTTP request sent, awaiting response... 200 OK

Exercise Field	Description
	`Length: 15759 (15K) [text/html]` `Saving to: ‘index.html.15’` `100%[====================================>] 15,759 --.-K/s in 0.09s` `2019-07-23 14:48:37 (180 KB/s) - ‘index.html.15’ saved [15759/15759]`

As explained above, exercise YnMUD-3-v6 is identical to exercise YnMUD-3-v4 except that it uses IPv6 instead of IPv4.

3.2.4.4 Exercise YnMUD-4-v4

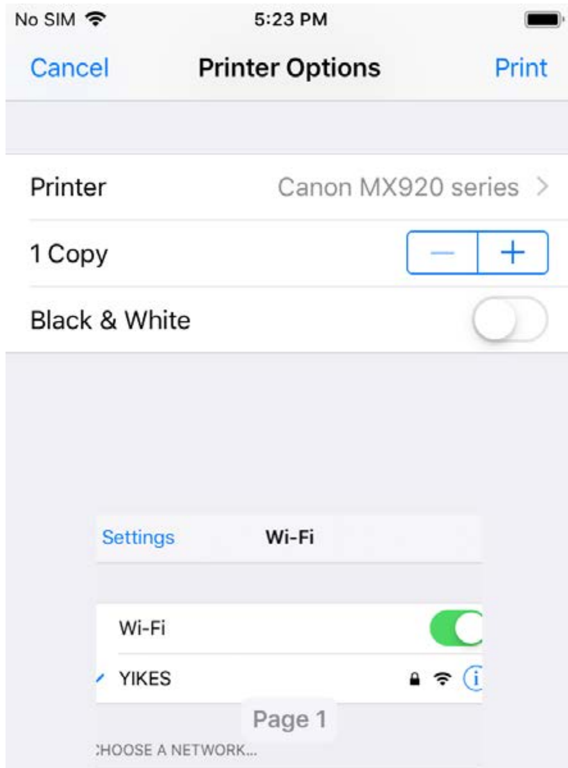
Table 3-17: Exercise YnMUD-4-v4

Exercise Field	Description
Parent Capability	(Y-4) Lateral (east-west) communications of the non-MUD-capable device to other devices on the local network are enforced according to the policy associated with the device’s category.
Subrequirement(s) of Parent Capability to Be Demonstrated	(Y-4.a) A rule associated with the device’s category permits the device to initiate communications with local devices in category X, but there is no such rule that permits the device to initiate communications with local devices in category Y. (Y-4.a.1) The device will be allowed to initiate communications to any local device that is in category X. (Y-4.a.2) The device will be prohibited from initiating communications to any local device that is in category Y.
Description	Verify that once a device has been identified and categorized, the communications that it initiates to other devices on the local network will be restricted according to the local network (east-west) rules in place for the device’s category.
Associated Exercises	YnMUD-1-v4

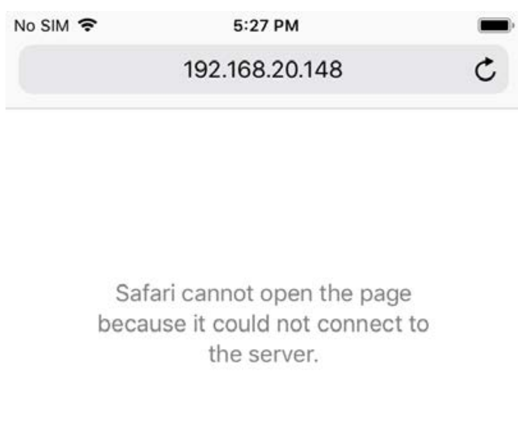
Exercise Field	Description
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-3, ID.AM-4, PR.AC-1, PR.AC-3, PR.AC-4, PR.AC-5
IoT Device(s) Used	Same as for exercise YnMUD-1-v4
Policy Used	In the Yikes! UI: - The Cell Phone local rules are set to allow cell phones to initiate communications to printers but not to any other category of devices. - The Computer local rules are set to allow computers to initiate communications to all other devices. - The Printer local rules are set to deny printers from initiating communications to all other devices.
Preconditions	Same as for exercise YnMUD-1-v4. In addition, the device category rules are as described in the policy row above (the presence of these rules can be verified by accessing the Yikes! UI). Add several devices to the Printer and Laptop categories.
Procedure	1. Execute the procedures defined in exercise YnMUD-1-v4 and verify that the exercise has achieved the expected results (all IoT devices have had their make and model identified, if possible, and they have all been categorized correctly). 2. Verify that the cell phone can print a file successfully. 3. Verify that the cell phone cannot communicate with the smart appliance. 4. Recategorize a Raspberry Pi as a printer. 5. Verify that the Raspberry Pi cannot communicate with the laptop. 6. Verify that the laptop can send traffic to each of the other devices.
Demonstrated Results	When using the scanning software on the phone and laptop, only the devices that we expected to see in the procedural steps above could be seen. **Procedure 1: Completed; excluded for brevity**

<table>
<tr><th>Exercise Field</th><th>Description</th></tr>
<tr><td></td><td>

Procedure 2:



Procedure 3:



</td></tr>
</table>

Exercise Field	Description

Procedure 4:

Operating System/Linux OS/Gentoo Linux
MY-CONTROLLER-PI - B8:27:EB:2B:39:B1
RASPBERRY PI FOUNDATION : GENTOO LINUX
PRINTERS

Procedure 5:

```
pi@my-controller-pi:~ $ wget 192.168.20.238

--2019-07-24 18:13:12--  http://192.168.20.238/

Connecting to 192.168.20.238:80... failed: Connection
refused.
```

Procedure 6:

Laptop to printer

```
[mud@localhost ~]$ wget 192.168.20.232
--2019-07-24 13:44:14--  http://192.168.20.232/
Connecting to 192.168.20.232:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 277
Saving to: 'index.html.17'

100%[======================================>] 277        --
.-K/s   in 0s

2019-07-24 13:44:14 (39.8 MB/s) - 'index.html.17' saved
[277/277]
```

Laptop to Pi categorized as printer

```
[mud@localhost ~]$ wget 192.168.20.117
--2019-07-24 14:03:29--  http://192.168.20.117/
Connecting to 192.168.20.117:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10701 (10K) [text/html]
Saving to: 'index.html.18'

100%[======================================>] 10,701     --
.-K/s   in 0.001s

2019-07-24 14:03:29 (8.95 MB/s) - 'index.html.18' saved
[10701/10701]
```

As explained above, exercise YnMUD-4-v6 is identical to exercise YnMUD-4-v4 except that it uses IPv6 instead of IPv4.

3.2.4.5 Exercise YnMUD-5-v4

Table 3-18: Exercise YnMUD-5-v4

Exercise Field	Description
Parent Capability	(Y-5) In response to threat information, all devices on the local network are prohibited from visiting specific domains and IP addresses.
Subrequirement(s) of Parent Capability to Be Demonstrated	(Y-5.a) Threat intelligence indicates a specific internet domain that should not be trusted. Devices are prohibited from initiating communications to the internet domain listed in the threat intelligence. In addition, they are prohibited from initiating communications to any other domains and IP addresses that are associated with the same threat campaign as this domain.
Description	Verify that when threat signaling information indicates that a specific domain is not safe, all devices on the local network will be restricted from initiating communications to that domain as well as to all other domains and IP addresses that are associated with the same threat campaign as this domain.
Associated Exercises	YnMUD-3-v4
Associated Cybersecurity Framework Subcategory(ies)	ID.RA-2, ID.RA-3, PR.AC-3, PR.AC-4, PR.AC-5
IoT Device(s) Used	Use the same non-MUD-capable devices as for exercise YnMUD-3-v4: - laptop - Samsung Galaxy S8 cell phone - iPhone 7 cell phone
Policy Used	Use the same (non-MUD) Yikes! router policy as for exercise YnMUD-3-v4, specifically:

Exercise Field	Description
	In the Yikes! UI, the Computer internet rule is set to Allow All Internet Traffic rather than to IoT Specific Sites.
Preconditions	Threat signaling is enabled. Threat signaling intelligence indicates that internet domain *www.dangerousSite.org* is dangerous and devices shall be prohibited from visiting it. It also associates *www.dangerousSite1.org* with the same threat campaign as *www.dangerousSite.org*, and these domains are associated with IP addresses XX.XX.XX.XX and YY.YY.YY.YY. In addition, the other preconditions are the same as for exercise Yn-MUD-3-v4, specifically: The Computer category internet rule in the Yikes! UI is set to Allow All Internet Traffic rather than to IoT Specific Sites. Therefore, the ACL rules on the router are configured to permit the laptop to send traffic to any site.
Procedure	1. Log in to the router and verify that there is no ACL that prohibits visiting *www.dangerousSite.org*, *www.dangerousSite1.org*, or IP addresses XX.XX.XX.XX or YY.YY.YY.YY. 2. Run exercise YnMUD-3-v4 and verify that it has the expected results, i.e., verify that the laptop can browse to www.google.com, www.osmud.org, and www.getyikes.com. 3. At this point, the test has verified that the Yikes! router rules are being enforced as expected. Now test the threat signaling capability by using the laptop to try to browse to a site that is prohibited by the threat signaling information: *www.dangerousSite.org*. 4. Verify that the laptop is not permitted to connect to this site. 5. Verify that firewall rules corresponding to the threat response have been installed on the router, prohibiting communication with *www.dangerousSite.org*, *www.dangerousSite1.org*, and IP addresses XX.XX.XX.XX and YY.YY.YY.YY.
Demonstrated Results	With threat signaling enabled, the laptop is prohibited from initiating communications to domains flagged by threat signaling. **Procedure 1:** `config defaults`

Exercise Field	Description
	`option syn_flood 1` `option input ACCEPT` `option output ACCEPT` `option forward REJECT` `# Uncomment this line to disable ipv6 rules` `# option disable_ipv6 1` `config zone` `option name lan` `list   network 'lan'` `option input ACCEPT` `option output ACCEPT` `        option log '1'` `config zone` `option name wan` `list   network 'wan'` `list   network 'wan6'` `option input REJECT` `option output ACCEPT` `option forward REJECT` `option masq 1` `option mtu_fix 1` `        option log '1'` `config forwarding` `option src lan` `option dest wan` `# We need to accept udp packets on port 68,` `# see https://dev.openwrt.org/ticket/4108` `config rule` `option name Allow-DHCP-Renew` `option src wan` `option proto udp` `option dest_port 68` `option target ACCEPT` `option family ipv4` `# Allow IPv4 ping` `config rule` `option name Allow-Ping` `option src wan` `option proto icmp` `option icmp_type echo-request` `option family ipv4` `option target ACCEPT` `config rule` `option name Allow-IGMP` `option src wan` `option proto igmp`

Exercise Field	Description
	option family ipv4 option target ACCEPT # Allow DHCPv6 replies # see https://dev.openwrt.org/ticket/10381 config rule option name Allow-DHCPv6 option src wan option proto udp option src_ip fc00::/6 option dest_ip fc00::/6 option dest_port 546 option family ipv6 option target ACCEPT config rule option name Allow-MLD option src wan option proto icmp option src_ip fe80::/10 list icmp_type '130/0' list icmp_type '131/0' list icmp_type '132/0' list icmp_type '143/0' option family ipv6 option target ACCEPT # Allow essential incoming IPv6 ICMP traffic config rule option name Allow-ICMPv6-Input option src wan option proto icmp list icmp_type echo-request list icmp_type echo-reply list icmp_type destination-unreachable list icmp_type packet-too-big list icmp_type time-exceeded list icmp_type bad-header list icmp_type unknown-header-type list icmp_type router-solicitation list icmp_type neighbour-solicitation list icmp_type router-advertisement list icmp_type neighbour-advertisement option limit 1000/sec option family ipv6 option target ACCEPT # Allow essential forwarded IPv6 ICMP traffic config rule option name Allow-ICMPv6-Forward option src wan option dest *

Exercise Field	Description
	`option proto icmp` `list icmp_type echo-request` `list icmp_type echo-reply` `list icmp_type destination-unreachable` `list icmp_type packet-too-big` `list icmp_type time-exceeded` `list icmp_type bad-header` `list icmp_type unknown-header-type` `option limit 1000/sec` `option family ipv6` `option target ACCEPT` `config rule` `option name Allow-IPSec-ESP` `option src wan` `option dest lan` `option proto esp` `option target ACCEPT` `config rule` `option name Allow-ISAKMP` `option src wan` `option dest lan` `option dest_port 500` `option proto udp` `option target ACCEPT` `# include a file with users custom iptables rules` `config include` `option path /etc/firewall.user` `### EXAMPLE CONFIG SECTIONS` **[Omitted for brevity]** `config rule` `        option enabled '1'` `        option target 'ACCEPT'` `        option src 'wan'` `        option proto 'tcp'` `        option dest_port '80'` `        option name 'AllowYikesAdminRemoteWeb'` `config rule` `        option enabled '1'` `        option target 'ACCEPT'` `        option src 'wan'` `        option proto 'tcp'` `        option dest_port '22'` `        option name 'AllowYikesAdminRemoteSsh'`

<table>
<tr><th>Exercise Field</th><th>Description</th></tr>
<tr><td></td><td>

```
#
# Base OpenWRT firewall rules to force the local router to
be the only DNS server allowed.
#      NOTE: This needs /etc/config/dhcp update to added the
router IP address as the primary DNS server
#            See dhcp.q9sample.conf for an example of this
configuration
#
config rule
        option target 'ACCEPT'
        option dest_port '53'
        option name 'Quad9 DNS Allow'
        option src 'lan'
        option dest_ip '9.9.9.9'
        option proto 'tcp udp'
        option dest 'wan'
        option family 'ipv4'

config rule
        option enabled '1'
        option src 'lan'
        option name 'DNS BLOCK OTHER SERVERS'
        option dest_port '53'
        option target 'REJECT'
        option proto 'tcp udp'
        option dest 'wan'

# OSMUD start
#
# DO NOT EDIT THESE LINES. OSMUD WILL REPLACE WITH ITS CON-
FIGURATION
#
```

[Omitted for brevity]

```
# OSMUD end
# AYIKES start
#
# DO NOT EDIT THESE LINES. AYIKES WILL REPLACE WITH ITS CON-
FIGURATION
#

# Begin YIKES ipset firewall declarations
```

[Omitted for brevity]

Procedure 2:

`--2019-07-24 10:50:53--` http://www.google.com/
</td></tr>
</table>

Exercise Field	Description
	`Resolving www.google.com (www.google.com)...` `172.217.164.132, 2607:f8b0:4004:815::2004` `Connecting to www.google.com` `(www.google.com)|172.217.164.132|:80... connected.` `HTTP request sent, awaiting response... 200 OK` `Length: unspecified [text/html]` `Saving to: ‘index.html’` `    0K .......... .` `45.5M=0s` `2019-07-24 10:50:53 (45.5 MB/s) - ‘index.html’ saved [11462]` `--2019-07-24 10:55:51--` https://osmud.org/ `Resolving osmud.org (osmud.org)... 198.71.233.87` `Connecting to osmud.org (osmud.org)|198.71.233.87|:443...` `connected.` `HTTP request sent, awaiting response... 200 OK` `Length: unspecified [text/html]` `Saving to: ‘index.html’` `    0K .......... .......... ....` `2.58M=0.009s` `2019-07-24 10:55:51 (2.58 MB/s) - ‘index.html’ saved [24697]`
	Procedures 3–4: `$ ping` *`www.dangerousSite.org`* `ping: cannot resolve` www.dangerousSite.org`: Unknown host` `$ ping` www.dangerousSite.org `PING` www.dangerousSite.org`(127.0.0.1): 56 data bytes` `64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.049 ms` `64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.073 ms` `64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.082 ms` `64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.139 ms` `64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.079 ms` `64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.072 ms` `64 bytes from 127.0.0.1: icmp_seq=6 ttl=64 time=0.123 ms` `64 bytes from 127.0.0.1: icmp_seq=7 ttl=64 time=0.073 ms` `ç64 bytes from 127.0.0.1: icmp_seq=8 ttl=64 time=0.066 ms` `^C` `---` www.dangerousSite.org `ping statistics ---` `9 packets transmitted, 9 packets received, 0.0% packet loss` `round-trip min/avg/max/stddev = 0.049/0.084/0.139/0.027 ms`
	`$ ping` *`www.dangerousSite1.org`* `ping: cannot resolve` *`www.dangerousSite1.org`*`: Unknown host`

Exercise Field	Description
	`$ ping www.dangerousSite1.org` `PING www.dangerousSite1.org(127.0.0.1): 56 data bytes` `64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.052 ms` `64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.073 ms` `64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.109 ms` `64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.064 ms` `64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.089 ms` `^C` `--- www.dangerousSite1.org ping statistics ---` `5 packets transmitted, 5 packets received, 0.0% packet loss` `round-trip min/avg/max/stddev = 0.052/0.077/0.109/0.022 ms`
	Procedure 5: `# Q9THREATRULES start` `#` `# DO NOT EDIT THESE LINES. Q9THRT WILL REPLACE WITH ITS CON-FIGURATION` `#` `config ipset` `    option enabled 1` `    option name Q9TS-joyheat_comFD` `    option match dest_ip` `    option storage hash` `    option family ipv4` `    option external Q9TS-joyheat_comFD` `config ipset` `    option enabled 1` `    option name Q9TS-joyheat_comTD` `    option match src_ip` `    option storage hash` `    option family ipv4` `    option external Q9TS-joyheat_comTD` `config rule` `        option enabled   '1'` `        option name      'Q9TS-joyheat_comFD'` `        option target    REJECT` `        option src       lan` `        option dest      wan` `        option proto     all` `        option family    ipv4` `        option ipset     Q9TS-joyheat_comFD` `        option src_ip    any` `config rule` `        option enabled   '1'` `        option name      'Q9TS-joyheat_comTD'`

Exercise Field	Description
	`        option target    REJECT` `        option src       wan` `        option dest      lan` `        option proto     all` `        option family    ipv4` `        option ipset     Q9TS-joyheat_comTD` `        option dest_ip    any` `# Q9THREATRULES end`

As explained above, exercise YnMUD-5-v6 is identical to exercise YnMUD-5-v4 except that it uses IPv6 instead of IPv4.

3.2.4.6 Exercise YnMUD-6-v4

Table 3-19: Exercise YnMUD-6-v4

Exercise Field	Description
Parent Capability	(Y-5) In response to threat information, all devices on the local network are prohibited from visiting specific domains and IP addresses.
Subrequirement(s) of Parent Capability to Be Demonstrated	(Y-5.b) Threat intelligence indicates a specific IP address that should not be trusted. Devices are prohibited from initiating communications to the IP address listed in the threat intelligence. In addition, they are prohibited from initiating communications to any other IP addresses and domains that are associated with the same threat campaign as this IP address.
Description	Verify that when threat signaling information indicates that a specific IP address (as opposed to domain) is not safe, all devices on the local network will be restricted from initiating communications to that IP address as well as to all other IP addresses and domains that are associated with the same threat campaign as this IP address.
Associated Exercises	YnMUD-3-v4
Associated Cybersecurity Framework Subcategory(ies)	ID.RA-2, ID.RA-3, PR.AC-3, PR.AC-4, PR.AC-5

Exercise Field	Description
IoT Device(s) Used	Use the same non-MUD-capable devices as for exercise YnMUD-3-v4: - laptop - Samsung Galaxy S8 cell phone - iPhone 7 cell phone
Policy Used	Use the same (non-MUD) Yikes! router policy as for exercise YnMUD-3-v4, specifically: In the Yikes! UI, the Computer internet rule is set to Allow All Internet Traffic rather than to IoT Specific Sites.
Preconditions	Threat signaling is enabled. Threat signaling intelligence indicates that IP address XX.XX.XX.XX is dangerous, and devices shall be prohibited from visiting it. It also associates IP address YY.YY.YY.YY with the same threat campaign as IP address XX.XX.XX.XX and these IP addresses are associated with domains *www.dangerousSite.org* and *www.dangerousSite1.org*. In addition, the other preconditions are the same as for exercise YnMUD-3-v4, specifically: The Computer category internet rule in the Yikes! UI is set to Allow All Internet Traffic rather than to IoT Specific Sites. Therefore, the firewall rules on the router are configured to permit the laptop to send traffic to any site.
Procedure	1. Log in to the router and verify that there is no ACL that prohibits visiting IP address XX.XX.XX.XX, IP address YY.YY.YY.YY, *www.dangerousSite.org*, or *www.dangerousSite1.org* (where IP address XX.XX.XX.XX is an address that is associated with the same threat as *www.dangerousSite.org*). 2. Run exercise YnMUD-3-v4 and verify that it has the expected results, i.e., verify that the laptop can browse to www.google.com, www.osmud.org, and www.trytechy.com. 3. At this point, the test has verified that the Yikes! router rules are being enforced as expected. 4. Run exercise YnMUD-5-v4. As a result, there should now be firewall rules on the router that prohibit all devices on the network from

Exercise Field	Description
	communicating with all domains and IP addresses that are associated with the same threat as the domain *www.dangerousSite.org*. 5. Use the laptop to try to browse to one of the IP addresses that is associated with the same threat as *www.dangerousSite.org*: IP address XX.XX.XX.XX. 6. Verify that the laptop is not permitted to connect to this site. 7. Verify that firewall rule corresponding to the threat response has been installed on the router, prohibiting communication with *www.dangerousSite.org*, *www.dangerousSite1.org*, and IP addresses XX.XX.XX.XX and YY.YY.YY.YY.
Demonstrated Results	With threat signaling enabled, the laptop is prohibited from initiating communications to IP addresses flagged by threat signaling intelligence. **Procedures 1–3:** **Completed; excluded for brevity** **Procedure 4:** Laptop ping *www.dangerousSite.org* `NCCoEs-MBP:results nccoe$ ping wwww.dangerousSite.org` `PING www.dangerousSite.org(127.0.0.1): 56 data bytes` `64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.039 ms` `64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.136 ms` `64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.063 ms` `64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.141 ms` `64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.071 ms` `^C` `--- www.dangerousSite.org ping statistics ---` `5 packets transmitted, 5 packets received, 0.0% packet loss` `round-trip min/avg/max/stddev = 0.039/0.090/0.141/0.041 ms` `NCCoEs-MBP:results nccoe$` `NCCoEs-MBP:results nccoe$ ping 192.60.252.130` `PING 192.60.252.130 (192.60.252.130): 56 data bytes` `Request timeout for icmp_seq 0` `Request timeout for icmp_seq 1` `Request timeout for icmp_seq 2` `Request timeout for icmp_seq 3` `^C` `--- 192.60.252.130 ping statistics ---` `5 packets transmitted, 0 packets received, 100.0% packet loss`

<table>
<tr><th>Exercise Field</th><th>Description</th></tr>
<tr><td></td><td>
<pre>NCCoEs-MBP:results nccoe$</pre>
<hr>
Procedure 5:
<pre>
Q9THREATRULES start
#
DO NOT EDIT THESE LINES. Q9THRT WILL REPLACE WITH ITS CON-
FIGURATION
#

config ipset
    option enabled 1
    option name Q9TS-joyheat_comFD
    option match dest_ip
    option storage hash
    option family ipv4
    option external Q9TS-joyheat_comFD

config ipset
    option enabled 1
    option name Q9TS-joyheat_comTD
    option match src_ip
    option storage hash
    option family ipv4
    option external Q9TS-joyheat_comTD

config rule
        option enabled   '1'
        option name   'Q9TS-joyheat_comFD'
        option target   REJECT
        option src   lan
        option dest   wan
        option proto   all
        option family   ipv4
        option ipset   Q9TS-joyheat_comFD
        option src_ip   any

config rule
        option enabled   '1'
        option name   'Q9TS-joyheat_comTD'
        option target   REJECT
        option src   wan
        option dest   lan
        option proto   all
        option family   ipv4
        option ipset   Q9TS-joyheat_comTD
        option dest_ip   any
Q9THREATRULES end
OSMUD start
</pre>
</td></tr>
</table>

As explained above, exercise YnMUD-6-v6 is identical to exercise YnMUD-6-v4 except that it uses IPv6 instead of IPv4.

471 ### 3.2.4.7 Exercise YnMUD-7-v4

472 **Table 3-20: Exercise YnMUD-7-v4**

Exercise Field	Description
Parent Capability	(Y-5) In response to threat information, all devices on the local network are prohibited from visiting specific domains and IP addresses.
Subrequirement(s) of Parent Capability to Be Demonstrated	(Y-5.c) Threat intelligence was received more than 24 hours prior, indicating domains and IP addresses that should not be trusted, and those domains and IP addresses were blocked by ACLs installed on the router. After 24 hours, these ACLs have been removed from the router.
Description	Verify that 24 or more hours after ACLs have been installed on the router as a result of threat signaling intelligence, those ACLs will be removed.
Associated Exercises	YnMUD-5-v4 and YnMUD-6-v4
Associated Cybersecurity Framework Subcategory(ies)	ID.RA-2, ID.RA-3, PR.AC-3, PR.AC-4, PR.AC-5
IoT Device(s) Used	Same as for tests YnMUD-5-v4 and YnMUD-6-v4
Policy Used	Same as the policy used for tests YnMUD-3-v4, YnMUD-5-v4, and YnMUD-6-v4
Preconditions	Threat signaling is enabled. Threat signaling intelligence indicates that *www.dangerousSite.org*, *www.dangerousSite1.org*, and IP addresses XX.XX.XX.XX and YY.YY.YY.YY are dangerous, and devices shall be prohibited from visiting them.
Procedure	1. Run test YnMUD-5-v4 and verify that the laptop is not permitted to access *www.dangerousSite.org*, *www.dangerousSite1.org*, and IP addresses XX.XX.XX.XX and YY.YY.YY.YY.

Exercise Field	Description
	2. Log on to the router and verify that ACLs have been installed on it prohibiting communication with *www.dangerousSite.org*, *www.dangerousSite1.org*, and IP addresses XX.XX.XX.XX and YY.YY.YY.YY. 3. Let 24 hours elapse. 4. Log on to the router and verify that the ACLs that had prohibited communication with *www.dangerousSite.org*, *www.dangerousSite1.org*, and IP addresses XX.XX.XX.XX and YY.YY.YY.YY are no longer there.
Demonstrated Results	ACL rules that had been installed as a result of threat signaling intelligence were removed after 24 hours. **Procedure 1:** **Completed; see YnMUD-6-v4** **Procedure 2:** `# Q9THREATRULES start` `#` `# DO NOT EDIT THESE LINES. Q9THRT WILL REPLACE WITH ITS CONFIGURATION` `#` `config ipset` `    option enabled 1` `    option name Q9TS-joyheat_comFD` `    option match dest_ip` `    option storage hash` `    option family ipv4` `    option external Q9TS-joyheat_comFD` `config ipset` `    option enabled 1` `    option name Q9TS-joyheat_comTD` `    option match src_ip` `    option storage hash` `    option family ipv4` `    option external Q9TS-joyheat_comTD` `config rule` `        option enabled   '1'` `        option name      'Q9TS-joyheat_comFD'` `        option target    REJECT` `        option src       lan` `        option dest      wan` `        option proto     all` `        option family    ipv4` `        option ipset     Q9TS-joyheat_comFD`

Exercise Field	Description
	<pre> option src_ip any config rule option enabled '1' option name 'Q9TS-joyheat_comTD' option target REJECT option src wan option dest lan option proto all option family ipv4 option ipset Q9TS-joyheat_comTD option dest_ip any # Q9THREATRULES end # OSMUD start</pre> **Procedure 4:** <pre>root@OpenWrt:~# cat /etc/config/firewall config defaults option syn_flood 1 option input ACCEPT option output ACCEPT option forward REJECT # Uncomment this line to disable ipv6 rules # option disable_ipv6 1 config zone option name lan list network 'lan' option input ACCEPT option output ACCEPT option log '1' config zone option name wan list network 'wan' list network 'wan6' option input REJECT option output ACCEPT option forward REJECT option masq 1 option mtu_fix 1 option log '1' config forwarding option src lan option dest wan # We need to accept udp packets on port 68, # see https://dev.openwrt.org/ticket/4108 config rule</pre>

Exercise Field	Description
	`option name Allow-DHCP-Renew` `option src wan` `option proto udp` `option dest_port 68` `option target ACCEPT` `option family ipv4` `# Allow IPv4 ping` `config rule` `option name Allow-Ping` `option src wan` `option proto icmp` `option icmp_type echo-request` `option family ipv4` `option target ACCEPT` `config rule` `option name Allow-IGMP` `option src wan` `option proto igmp` `option family ipv4` `option target ACCEPT` **[Omitted for brevity]** **`# Q9THREATRULES start`** **`#`** **`# DO NOT EDIT THESE LINES. Q9THRT WILL REPLACE WITH ITS CON-FIGURATION`** **`#`** **`# Q9THREATRULES end`** `# OSMUD start` `#` `# DO NOT EDIT THESE LINES. OSMUD WILL REPLACE WITH ITS CON-FIGURATION` `#` **[Omitted for brevity]** `# OSMUD end` `# AYIKES start` `#` `# DO NOT EDIT THESE LINES. AYIKES WILL REPLACE WITH ITS CON-FIGURATION` `#` `# Begin YIKES ipset firewall declarations` **[Omitted for brevity]** `# AYIKES end`

As explained above, exercise YnMUD-7-v6 is identical to exercise YnMUD-7-v4 except that it uses IPv6 instead of IPv4.

4 Build 3

Build 3 is still under development by CableLabs. Therefore, it has not yet been fully demonstrated. Documentation of Build 3's functional evaluation and demonstration is planned for inclusion in the next phase of this project.

5 Build 4

Build 4 uses software developed at the NIST Advanced Networking Technologies laboratory. This software provides support for MUD and is intended to serve as a working prototype of the MUD RFC to demonstrate feasibility and scalability.

5.1 Evaluation of MUD-Related Capabilities

The functional evaluation that was conducted to verify that Build 4 conforms to the MUD specification was based on the Build 4-specific requirements listed in Table 5-1.

5.1.1 Requirements

Table 5-1: MUD Use Case Functional Requirements

Capability Requirement (CR)-ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-1	The IoT DDoS example implementation shall include a mechanism for associating a device with a MUD file URL (e.g., by having the MUD-enabled **IoT device emit a MUD file URL via DHCP, LLDP, or X.509 or by using some other mechanism to enable the network to associate a device with a MUD file URL).**			IoT-1-v4, IoT-11-v4

Capability Requirement (CR)-ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-1.a		Upon initialization, the MUD-enabled IoT device shall broadcast a DHCP message on the network, including at most one **MUD URL, in https scheme, within the DHCP transaction.**		IoT-1-v4, IoT-11-v4
CR-1.a.1			The DHCP server shall be able to receive **DHCPv4 DISCOVER and REQUEST with IANA code 161** (OPTION_MUD_URL_V4) from the MUD-enabled IoT device.	IoT-1-v4, IoT-11-v4
CR-2	The IoT DDoS example implementation shall include the capability for the extracted MUD URL **to be provided to a MUD manager.**			IoT-1-v4
CR-2.a		The DHCP server shall **assign an IP address lease** to the MUD-enabled IoT device.		IoT-1-v4
CR-2.a.1			The MUD-enabled IoT device shall **receive the IP address.**	IoT-1-v4

Capability Requirement (CR)-ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-2.b		**The MUD manager** shall receive the DHCP message and **extract the MUD URL.**		IoT-1-v4
CR-2.b.1			**The MUD manager shall receive the MUD URL.**	IoT-1-v4
CR-3	The IoT DDoS example implementation shall include a **MUD manager that can request a MUD file and signature from a MUD file server.**			IoT-1-v4
CR-3.a		The MUD manager shall use the GET method (RFC 7231) to **request MUD and signature files** (per RFC 7230) from the MUD file server and can **validate the MUD file server's TLS certificate** by using the rules in RFC 2818.		IoT-1-v4
CR-3.a.1			**The MUD file server shall receive the https request from the MUD manager.**	IoT-1-v4
CR-3.b		**The MUD manager** shall use the GET method (RFC 7231) to		IoT-2-v4

Capability Requirement (CR)-ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
		request MUD and signature files (per RFC 7230) from the MUD file server, but it **cannot validate the MUD file server's TLS certificate** by using the rules in RFC 2818.		
CR-3.b.1			**The MUD manager shall drop the connection** to the MUD file server.	IoT-2-v4
CR-3.b.2			**The MUD manager shall send locally defined policy to the router or switch** that handles whether to allow or block traffic to and from the MUD-enabled IoT device.	IoT-2-v4
CR-4	The IoT DDoS example implementation shall include a **MUD file server that can serve a MUD file and signature to the MUD manager.**			IoT-1-v4
CR-4.a		**The MUD file server shall serve the file and signature to the MUD manager, and the MUD manager**		IoT-1-v4

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

Capability Requirement (CR)-ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
			handles whether to allow or block traffic to and from the MUD-enabled IoT device.	
CR-5	The IoT DDoS example implementation shall include a **MUD manager** that **can translate local network configurations based on the MUD file.**			IoT-1-v4
CR-5.a		**The MUD manager shall successfully validate the signature of the MUD file.**		IoT-1-v4
CR-5.a.1			The MUD manager, after validation of the MUD file signature, shall **check for an existing MUD file, - and translate abstractions in the MUD file to router or switch configurations**.	IoT-1-v4
CR-5.a.2			The MUD manager shall **cache** this newly received MUD file.	IoT-10-v4
CR-5.b		The MUD manager shall attempt to validate the signature of		IoT-4-v4

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

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

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

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

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

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

Capability Requirement (CR)-ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-10.a.1			**The router or switch shall receive the attempt and shall deny it** based on the filters from the MUD file.	IoT-6-v4
CR-10.b		**An unapproved** (implicitly denied) **device shall attempt to initiate a lateral connection** to the MUD-enabled IoT device.		IoT-6-v4
CR-10.b.1			**The router or switch shall receive the attempt and shall deny it** based on the filters from the MUD file.	IoT-6-v4
CR-11	If the IoT DDoS example implementation is such that its DHCP server does not act as a MUD manager and it forwards a MUD URL to a MUD manager, **the DHCP server must notify the MUD manager of any corresponding change to the DHCP state** of the MUD-enabled IoT device, and the MUD manager should **remove the implemented policy configuration in the router/switch pertaining to that MUD-enabled IoT device.**			No test needed because the DHCP server does not forward the MUD URL to the MUD manager, as intended.

Capability Requirement (CR)-ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
CR-11.a		The MUD-enabled IoT **device shall explicitly release the IP address lease** (i.e., it sends a DHCP release message to the DHCP server).		N/A
CR-11.a.1			**The DHCP server shall notify the MUD manager that the device's IP address lease has been released.**	N/A
CR-11.a.2			**The MUD manager should remove all policies** associated with the disconnected IoT device that had been configured on the MUD PEP router/switch.	N/A
CR-11.b		The MUD-enabled IoT **device's IP address lease shall expire.**		N/A
CR-11.b.1			**The DHCP server shall notify the MUD manager that the device's IP address lease has expired.**	N/A
CR-11.b.2			**The MUD manager should remove all**	N/A

Capability Requirement (CR)-ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
			policies associated with the affected IoT device that had been configured on the MUD PEP router/switch.	
CR-12	The IoT DDoS example implementation shall include a **MUD manager that uses a cached MUD file rather than retrieve a new one if the cache-validity time period has not yet elapsed** for the MUD file indicated by the MUD URL**. The MUD manager should fetch a new MUD file if the cache-validity time period has already elapsed.**			IoT-10-v4
CR-12.a		The MUD manager shall check if the file associated with the **MUD URL is present in its cache** and shall determine that it is.		IoT-10-v4
CR-12.a.1			The MUD manager shall **check whether the amount of time that has elapsed since the cached file was retrieved is less than or equal to the**	IoT-10-v4

Capability Requirement (CR)-ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
			number of hours in the cache-validity value for this MUD file. If so, the MUD manager shall apply the contents of the cached MUD file.	
CR-12.a.2			The MUD manager **shall check whether the amount of time that has elapsed since the cached file was retrieved is greater than the number of hours in the cache-validity value for this MUD file.** If so, the MUD manager may (but does not have to) fetch a new file by using the MUD URL received.	IoT-10-v4
CR-13	The IoT DDoS example implementation shall ensure that for each rule in a MUD file that pertains to an external domain, the MUD PEP router/switch will get configured with **all possible instantiations of that rule,** insofar as **each instantiation contains one of the IP addresses**			IoT-9-v4

Capability Requirement (CR)-ID	Parent Requirement	Subrequirement 1	Subrequirement 2	Test Case
	to which the domain in that MUD file rule may be resolved when queried by the MUD PEP router/switch.			
CR-13.a		The MUD file for a device shall contain a rule involving a **domain that can resolve to multiple IP addresses** when queried by the MUD PEP router/switch. **Flow rules for permitting access to each of those IP addresses will be inserted into the MUD PEP router/switch** for the device in question, and the device will be permitted to communicate with all of those IP addresses.		IoT-9-v4
CR-13.a.1			IPv4 addressing is used on the network.	IoT-9-v4

488 ### 5.1.2 Test Cases

489 This section contains the test cases that were used to verify that Build 4 met the requirements listed in
490 Table 5-1.

491 The test setup consists of five Raspberry Pis. Two of these are designated as having MUD Uniform Re-
492 source Identifiers (URIs) *sensor.nist.local* and one is designated *otherman.nist.local*. MUD files for "sen-
493 sor" and "otherman" were generated using mudmaker. The Software Defined Network (SDN) enabled

wireless router/NAT maps these fake hosts to test servers that are on the public side of the NAT. They are given fake 203.0.113.x addresses for name resolution. One of the Raspberry Pis is designated as a controller, and the last Raspberry Pi is designated as a host on the "local network."

The SDN switch is an unmodified Northbound Networks wireless SDN switch.

The controller host address and the DNS/DHCP host address are configured statically in the SDN controller by using the standard URIs for these entities. The controller URIs for the devices are likewise configured. dhclient is used to issue DHCP requests with MUD URLs embedded for Raspberry Pis 1, 2, and 3. The MUD URIs for 1 and 2 are identical and set to *https://sensor.nist.local/nistmud1*, while the MUD URI for Pi 3 is set to *https://otherman.nist.local/nistmud2*.

The controller host maps the fake host names in these URIs to 127.0.0.1 and runs a manufacturer https server. The server logs access to verify if file caching is properly working on the MUD manager.

Before the tests are conducted, the MUD files are signed using the NCCoE-supplied DigiCert key, and the trusted certificate is installed in the Java virtual machine trust store.

Accessibility testing is done using simple scripts and command line utilities that test whether permissible access works and whether forbidden access is blocked by the MUD-enabled SDN switch. The MUD files have access control entries that enable testing interactions with the hosts and web servers.

5.1.2.1 Test Case IoT-1-v4

Table 5-2: Test Case IoT-1-v4

Test Case Field	Description
Parent Requirements	(CR-1) The IoT DDoS example implementation shall include a mechanism for associating a device with a MUD file URL (e.g., by having the MUD-enabled IoT device emit a MUD file URL via DHCP, LLDP, or X.509 or by using some other mechanism to enable the network to associate a device with a MUD file URL). (CR-2) The IoT DDoS example implementation shall include the capability for the MUD URL to be provided to a MUD manager. (CR-3) The IoT DDoS example implementation shall include a MUD manager that can request a MUD file and signature from a MUD file server. (CR-4) The IoT DDoS example implementation shall include a MUD file server that can serve a MUD file and signature to the MUD manager.

Test Case Field	Description
	(CR-5) The IoT DDoS example implementation shall include a MUD manager that can translate local network configurations based on the MUD file. (CR-6) The IoT DDoS example implementation shall include a MUD manager that can configure the router or switch nearest the MUD-enabled IoT device that emitted the URL.
Testable Requirements	(CR-1.a) Upon initialization, the MUD-enabled IoT device shall broadcast a DHCP message on the network, including at most one MUD URL, in https scheme, within the DHCP transaction. (CR-1.a.1) The DHCP server shall be able to receive DHCPv4 DISCOVER and REQUEST with IANA code 161 (OPTION_MUD_URL_V4) from the MUD-enabled IoT device. (CR-2.a) The DHCP server shall assign an IP address lease to the MUD-enabled IoT device. (CR-2.a.1) The MUD-enabled IoT device shall receive the IP address. (CR-2.b) The MUD manager shall receive the DHCP message and extract the MUD URL. (CR-2.b.1) The MUD manager shall receive the MUD URL. (CR-3.a) The MUD manager shall use the GET method (RFC 7231) to request MUD and signature files (per RFC 7230) from the MUD file server and can validate the MUD file server's TLS certificate by using the rules in RFC 2818. (CR-3.a.1) The MUD file server shall receive the https request from the MUD manager. (CR-4.a) The MUD file server shall serve the file and signature to the MUD manager, and the MUD manager shall check to determine whether the certificate used to sign the MUD file (signed using DER-encoded CMS [RFC 5652]) was valid at the time of signing, i.e., the certificate had not expired. (CR-5.a) The MUD manager shall successfully validate the signature of the MUD file. (CR-5.a.1) The MUD manager, after validation of the MUD file signature, shall check for an existing MUD file and translate abstractions in the MUD file to router or switch configurations.

Test Case Field	Description
	(CR-6.a) The MUD manager shall install a router configuration on the router or switch nearest the MUD-enabled IoT device that emitted the URL. (CR-6.a.1) The router or switch shall have been configured to enforce the route filter sent by the MUD manager.
Description	Shows that, upon connection to the network, a MUD-enabled IoT device used in the IoT DDoS example implementation has its MUD PEP router/switch automatically configured to enforce the route filtering that is described in the device's MUD file, assuming the MUD file has a valid signature and is served from a MUD file server that has a valid TLS certificate
Associated Test Case(s)	N/A
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-1, ID.AM-2, ID.AM-3, PR.DS-5, DE.AE-1, PR.AC-4, PR.AC-5, PR.IP-1, PR.IP-3, PR.PT-3, PR.DS-2
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*mudfile-sensor.json*
Preconditions	1. All devices have been configured to use IPv4. 2. This MUD file is not currently cached at the MUD manager. 3. The device's MUD file has a valid signature that was signed by a certificate that had not yet expired, and it is being hosted on a MUD file server that has a valid TLS certificate. 4. The MUD PEP router/switch does not yet have any configuration settings pertaining to the IoT device being used in the test. 5. The MUD file for the IoT device being used in the test is identical to the MUD file provided in Section 5.1.3.
Procedure	Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. Also verify that the MUD

Test Case Field	Description
	file of the IoT device to be used is not currently cached at the MUD manager. 1. Power on the IoT device and connect it to the test network. 2. On the IoT device, using the dhclient application with appropriate configuration file, manually send a DHCPv4 message containing the device's MUD URL (IANA code 161). 3. The DHCP server receives the DHCP message containing the IoT device's MUD URL. 4. The MUD manager snoops the DHCP request through the switch and extracts the MUD URL from the DHCP request. 5. The MUD manager automatically contacts the MUD file server that is located by using the MUD URL, verifies that it has a valid TLS certificate, requests and receives the MUD file and signature from the MUD file server, validates the MUD file's signature, and translates the MUD file's contents into appropriate route filtering rules. It then installs these rules onto the MUD PEP for the IoT device in question so that this router/switch is now configured to enforce the policies specified in the MUD file. 6. The DHCP server offers an IP address lease to the newly connected IoT device. 7. The IoT device requests this IP address lease, which the DHCP server acknowledges.
Expected Results	The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to enforce the policies specified in the IoT device's MUD file. Flow rules on the switch are updated to reflect MUD filtering rules. The flow rules in the MUD flow rules table should reflect the ACLs in the MUD file.
Actual Results	**<u>Flow rules on router/switch</u>:** As seen below, tables zero and one classify the packets based on source and destination address, and tables two and three implement the MUD rules filtering. Tables four and five are pass and drop tables respectively. Additionally, to simplify, this test is successful when flows other than the default flows are viewed on the MUD PEP router/switch.

Test Case Field	Description
	OFPST_FLOW reply (OF1.3) (xid=0x2): cookie=0x995ac, duration=38.664s, table=0, n_packets=12, n_bytes=996, idle_timeout=120, hard_timeout=240, prior-ity=40,ip,dl_src=00:13:ef:20:1d:14 ac-tions=write_metadata:0x100300300000000/0x7ffffff00000000,goto_table:1 cookie=0x995ac, duration=38.148s, table=0, n_packets=12, n_bytes=996, idle_timeout=120, hard_timeout=240, prior-ity=40,ip,dl_src=00:13:ef:70:47:66 ac-tions=write_metadata:0x100300300000000/0x7ffffff00000000,goto_table:1 cookie=0x995ac, duration=37.655s, table=0, n_packets=13, n_bytes=1081, idle_timeout=120, hard_timeout=240, prior-ity=40,ip,dl_src=74:da:38:56:10:66 ac-tions=write_metadata:0x100300300000000/0x7ffffff00000000,goto_table:1 cookie=0x995ac, duration=37.149s, table=0, n_packets=16, n_bytes=1324, idle_timeout=120, hard_timeout=240, prior-ity=40,ip,dl_src=b8:27:eb:ac:45:76 ac-tions=write_metadata:0x300300000000/0x7ffffff00000000,goto_table:1 cookie=0x995ac, duration=33.630s, table=0, n_packets=58, n_bytes=4806, idle_timeout=120, hard_timeout=240, prior-ity=40,ip,dl_src=70:b3:d5:6c:db:92 ac-tions=write_metadata:0x300300000000/0x7ffffff00000000,goto_table:1 cookie=0x995ac, duration=23.550s, table=0, n_packets=8, n_bytes=664, idle_timeout=120, hard_timeout=240, prior-ity=40,ip,dl_src=b8:27:eb:3d:65:78 ac-tions=write_metadata:0x400500000000/0x7ffffff00000000,goto_table:1 cookie=0xca8bf, duration=82.206s, table=0, n_packets=25, n_bytes=2073, priority=31,ip actions=CONTROL-LER:65535,write_metadata:0x200200000000/0xffffff00000000 cookie=0xf6736, duration=88.641s, table=0, n_packets=272, n_bytes=20928, priority=30 ac-tions=write_metadata:0xf6736,goto_table:1 cookie=0xe809d, duration=38.641s, table=1, n_packets=60, n_bytes=4976, idle_timeout=120, hard_timeout=240, prior-ity=40,ip,dl_dst=70:b3:d5:6c:db:92 ac-tions=write_metadata:0x3003/0x7ffffff,goto_table:2 cookie=0xe809d, duration=33.105s, table=1, n_packets=10, n_bytes=826, idle_timeout=120, hard_timeout=240, prior-ity=40,ip,dl_dst=00:13:ef:20:1d:14 ac-tions=write_metadata:0x1003003/0x7ffffff,goto_table:2

Test Case Field	Description
	cookie=0xe809d, duration=32.411s, table=1, n_packets=10, n_bytes=826, idle_timeout=120, hard_timeout=240, priority=40,ip,dl_dst=00:13:ef:70:47:66 actions=write_metadata:0x1003003/0x7ffffff,goto_table:2 cookie=0xe809d, duration=31.916s, table=1, n_packets=12, n_bytes=996, idle_timeout=120, hard_timeout=240, priority=40,ip,dl_dst=74:da:38:56:10:66 actions=write_metadata:0x1003003/0x7ffffff,goto_table:2 cookie=0xe809d, duration=31.417s, table=1, n_packets=15, n_bytes=1239, idle_timeout=120, hard_timeout=240, priority=40,ip,dl_dst=b8:27:eb:ac:45:76 actions=write_metadata:0x3003/0x7ffffff,goto_table:2 cookie=0xe809d, duration=18.337s, table=1, n_packets=7, n_bytes=583, idle_timeout=120, hard_timeout=240, priority=40,ip,dl_dst=b8:27:eb:3d:65:78 actions=write_metadata:0x4005/0x7ffffff,goto_table:2 cookie=0xca8bf, duration=81.689s, table=1, n_packets=11, n_bytes=1324, priority=31,ip actions=CONTROLLER:65535,write_metadata:0x2002/0xffffff cookie=0xf6736, duration=88.335s, table=1, n_packets=272, n_bytes=20928, priority=30 actions=write_metadata:0xf6736,goto_table:2 cookie=0xea237, duration=78.043s, table=2, n_packets=3, n_bytes=1050, priority=55,udp,tp_src=68,tp_dst=67 actions=CONTROLLER:65535,goto_table:4 cookie=0x99f4d, duration=78.043s, table=2, n_packets=3, n_bytes=1031, priority=55,udp,tp_src=67,tp_dst=68 actions=CONTROLLER:65535,goto_table:4 cookie=0x90f01, duration=77.133s, table=2, n_packets=126, n_bytes=10454, priority=55,udp,nw_dst=10.0.41.1,tp_dst=53 actions=CONTROLLER:65535,goto_table:4 cookie=0x90f01, duration=77.132s, table=2, n_packets=0, n_bytes=0, priority=55,tcp,nw_dst=10.0.41.1,tp_dst=53 actions=CONTROLLER:65535,goto_table:4 cookie=0x4d67b, duration=77.133s, table=2, n_packets=117, n_bytes=9693, priority=55,udp,nw_src=10.0.41.1,tp_src=53 actions=CONTROLLER:65535,goto_table:4 cookie=0x4d67b, duration=77.132s, table=2, n_packets=0, n_bytes=0, priority=55,tcp,nw_src=10.0.41.1,tp_src=53 actions=CONTROLLER:65535,goto_table:4 cookie=0xf751b, duration=78.044s, table=2, n_packets=0, n_bytes=0, priority=45,ip,metadata=0x400000000000000/0x400000000000000 actions=goto_table:5 cookie=0x6d8f, duration=41.556s, table=2, n_packets=0, n_bytes=0, priority=41,tcp,metadata=0x400001000000/0xfff00001000000,tp_dst=80,tcp_flags=-fin+syn-rst-psh-ack-urg-ece-cwr actions=CON-

Test Case Field	Description
	TROL-LER:65535,write_metadata:0x400001000000/0xfff00001000000,goto_table:5 cookie=0x6d8f, duration=40.764s, table=2, n_packets=0, n_bytes=0, priority=41,tcp,metadata=0x100000000004000/0x100000000fff000,tp_dst=888,tcp_flags=-fin+syn-rst-psh-ack-urg-ece-cwr actions=CONTROL-LER:65535,write_metadata:0x100000000004000/0x100000000fff000,goto_table:5 cookie=0x6d8f, duration=40.627s, table=2, n_packets=0, n_bytes=0, priority=41,tcp,metadata=0x400004000/0xfff00fff000,tp_dst=800,tcp_flags=-fin+syn-rst-psh-ack-urg-ece-cwr actions=CONTROL-LER:65535,write_metadata:0x400004000/0xfff00fff000,goto_table:5 cookie=0x6d587, duration=41.634s, table=2, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x400001000000/0xfff00001000000,tp_dst=80 actions=write_metadata:0xffffffffffffffff/0,goto_table:3 cookie=0x6d587, duration=41.520s, table=2, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x400001000000/0xfff00001000000,tp_dst=888 actions=write_metadata:0xffffffffffffffff/0,goto_table:3 cookie=0x95d11, duration=41.961s, table=2, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x400000000000/0xfff00000000000,nw_dst=203.0.113.13,tp_dst=443 actions=write_metadata:0xffffffffffffffff/0,goto_table:3 cookie=0x43f0b, duration=41.889s, table=2, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x400000000000/0xfff00000000000,nw_dst=10.0.41.225,tp_dst=8080 actions=write_metadata:0xffffffffffffffff/0,goto_table:3 cookie=0xde7f1, duration=41.742s, table=2, n_packets=0, n_bytes=0, priority=40,udp,metadata=0x400000000000/0xfff00000000000,nw_dst=10.0.41.225,tp_dst=4000 actions=write_metadata:0xffffffffffffffff/0,goto_table:3 cookie=0x6d587, duration=41.676s, table=2, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x400001000000/0xfff00001000000,tp_src=80 actions=write_metadata:0xffffffffffffffff/0,goto_table:3 cookie=0x6d587, duration=41.486s, table=2, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x400001000000/0xfff00001000000,tp_src=888 actions=write_metadata:0xffffffffffffffff/0,goto_table:3

Test Case Field	Description
	cookie=0xd0bd1, duration=41.415s, table=2, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x400000000004/0xfff00000000fff,tp_src=800 actions=write_metadata:0xffffffffffffffff/0,goto_table:3 cookie=0xecf6, duration=41.334s, table=2, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x400000000005/0xfff00000000fff,tp_src=8888 actions=write_metadata:0xffffffffffffffff/0,goto_table:3 cookie=0xd0bd1, duration=41.436s, table=2, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x400000000004/0xfff00000000fff,tp_dst=800 actions=write_metadata:0xffffffffffffffff/0,goto_table:3 cookie=0xecf6, duration=41.360s, table=2, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x400000000005/0xfff00000000fff,tp_dst=8888 actions=write_metadata:0xffffffffffffffff/0,goto_table:3 cookie=0x26ef, duration=42.432s, table=2, n_packets=0, n_bytes=0, priority=35,metadata=0x400000000000/0xfff00000000000 actions=write_metadata:0xffffffffffffffff/0,goto_table:5 cookie=0x29a94, duration=81.184s, table=2, n_packets=282, n_bytes=22446, priority=30 actions=write_metadata:0x29a94,goto_table:3 cookie=0xd5afc, duration=78.045s, table=3, n_packets=0, n_bytes=0, priority=45,ip,metadata=0x4000000/0x4000000 actions=goto_table:5 cookie=0x6d8f, duration=41.094s, table=3, n_packets=0, n_bytes=0, priority=41,tcp,metadata=0x4000/0xfff000,nw_src=203.0.113.13,tp_src=443,tcp_flags=-fin+syn-rst-psh-ack-urg-ece-cwr actions=CONTROLLER:65535,write_metadata:0x4000/0xfff000,goto_table:5 cookie=0x6d8f, duration=41.001s, table=3, n_packets=0, n_bytes=0, priority=41,tcp,metadata=0x4000/0xfff000,nw_src=10.0.41.225,tp_src=8080,tcp_flags=-fin+syn-rst-psh-ack-urg-ece-cwr actions=CONTROLLER:65535,write_metadata:0x4000/0xfff000,goto_table:5 cookie=0x95d11, duration=41.138s, table=3, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x4000/0xfff000,nw_src=203.0.113.13,tp_src=443 actions=write_metadata:0xffffffffffffffff/0,goto_table:4 cookie=0x43f0b, duration=41.052s, table=3, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x4000/0xfff000,nw_src=10.0.41.225,tp_src=8080 actions=write_metadata:0xffffffffffffffff/0,goto_table:4

Test Case Field	Description
	cookie=0xde7f1, duration=40.921s, table=3, n_packets=0, n_bytes=0, priority=40,udp,metadata=0x4000/0xfff000,nw_src=10.0.41.225,tp_src=4000 actions=write_metadata:0xffffffffffffffff/0,goto_table:4 cookie=0x6d587, duration=40.896s, table=3, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x100000000004000/0x100000000fff000,tp_dst=80 actions=write_metadata:0xffffffffffffffff/0,goto_table:4 cookie=0x6d587, duration=40.799s, table=3, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x100000000004000/0x100000000fff000,tp_dst=888 actions=write_metadata:0xffffffffffffffff/0,goto_table:4 cookie=0x6d587, duration=40.852s, table=3, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x100000000004000/0x100000000fff000,tp_src=80 actions=write_metadata:0xffffffffffffffff/0,goto_table:4 cookie=0x6d587, duration=40.825s, table=3, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x100000000004000/0x100000000fff000,tp_src=888 actions=write_metadata:0xffffffffffffffff/0,goto_table:4 cookie=0xd0bd1, duration=40.729s, table=3, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x400004000/0xfff00fff000,tp_src=800 actions=write_metadata:0xffffffffffffffff/0,goto_table:4 cookie=0xecf6, duration=40.565s, table=3, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x500004000/0xfff00fff000,tp_src=8888 actions=write_metadata:0xffffffffffffffff/0,goto_table:4 cookie=0xd0bd1, duration=40.663s, table=3, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x400004000/0xfff00fff000,tp_dst=800 actions=write_metadata:0xffffffffffffffff/0,goto_table:4 cookie=0xecf6, duration=40.543s, table=3, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x500004000/0xfff00fff000,tp_dst=8888 actions=write_metadata:0xffffffffffffffff/0,goto_table:4 cookie=0x26ef, duration=42.418s, table=3, n_packets=0, n_bytes=0, priority=35,metadata=0x4000/0xfff000 actions=write_metadata:0xffffffffffffffff/0,goto_table:5 cookie=0x29a94, duration=80.685s, table=3, n_packets=282, n_bytes=22446, priority=30 actions=write_metadata:0x29a94,goto_table:4 cookie=0x64f19, duration=79.686s, table=4, n_packets=281, n_bytes=24670, priority=41 actions=NORMAL,IN_PORT

Test Case Field	Description
	cookie=0x1c2bd, duration=79.184s, table=5, n_packets=0, n_bytes=0, priority=30 actions=drop **debug-mudtables-sensor.json:** The following maps the flow rules above to the associated MUD file rules. This is for debug purposes only to verify that the MUD rules have been applied appropriately. `{` `    "input": {` `        "mud-url": "https://sensor.nist.local/nistmud1",` `        "switch-id": "openflow:123917682138002"` `    }` `}` `{` `    "output": {` `        "flow-rule": [` `            {` `                "flow-id": "https://sensor.nist.local/nist-mud1/NO_FROM_DEV_ACE_MATCH_DROP",` `                "byte-count": 1602,` `                "table-id": 2,` `                "priority": 35,` `                "src-model": "https://sensor.nist.local/nist-mud1",` `                "flow-name": "metadataMatchGoToTable(5)",` `                "packet-count": 9` `            },` `            {` `                "flow-id": "https://sensor.nist.local/nist-mud1/mud-31931-v4fr/loc1-frdev/2",` `                "byte-count": 0,` `                "table-id": 2,` `                "dst-local-networks-flag": true,` `                "priority": 40,` `                "src-model": "https://sensor.nist.local/nist-mud1",` `                "flow-name": "MetadaProtocolAndSrcDstPortMatchGoToTable(protocol=6,srcPort=888,dstPort=-1,targetTable=3)",` `                "packet-count": 0`

Test Case Field	Description
	}, { "flow-id": "https://sensor.nist.local/nist-mud1/mud-31931-v4fr/myctl0-frdev", "byte-count": 0, "table-id": 2, "priority": 40, "src-model": "https://sensor.nist.local/nist-mud1", "flow-name": "metadataDestIpAndPortMatchGoToNext(destIp=10.0.41.225,srcPort=-1,destPort=4000,protocol=17,sendToController=false)", "packet-count": 0 }, { "flow-id": "https://sensor.nist.local/nist-mud1/mud-31931-v4fr/myman0-frdev/1", "dst-manufacturer": "sensor.nist.local", "byte-count": 0, "table-id": 2, "priority": 40, "src-model": "https://sensor.nist.local/nist-mud1", "flow-name": "MetadaProtocolAndSrcDstPortMatchGoToTable(protocol=6,srcPort=-1,dstPort=8888,targetTable=3)", "packet-count": 0 }, { "flow-id": "https://sensor.nist.local/nist-mud1/mud-31931-v4fr/myman0-frdev/2", "dst-manufacturer": "sensor.nist.local", "byte-count": 0, "table-id": 2, "priority": 40, "src-model": "https://sensor.nist.local/nist-mud1", "flow-name": "MetadaProtocolAndSrcDstPortMatchGoToTable(protocol=6,srcPort=8888,dstPort=-1,targetTable=3)", "packet-count": 0 }, {

Test Case Field	Description
	"flow-id": "https://sensor.nist.local/nist-mud1/mud-31931-v4fr/loc1-frdev/1", "byte-count": 0, "table-id": 2, "dst-local-networks-flag": true, "priority": 40, "src-model": "https://sensor.nist.local/nist-mud1", "flow-name": "MetadaProtocolAndSrcDstPortMatchGoToTable(protocol=6,srcPort=-1,dstPort=888,targetTable=3)", "packet-count": 0 }, { "flow-id": "https://sensor.nist.local/nist-mud1/mud-31931-v4fr/ent0-frdev", "byte-count": 0, "table-id": 2, "priority": 40, "src-model": "https://sensor.nist.local/nist-mud1", "flow-name": "metadataDestIpAndPortMatchGoToNext(destIp=10.0.41.225,srcPort=-1,destPort=8080,protocol=6,sendToController=false)", "packet-count": 0 }, { "flow-id": "https://sensor.nist.local/nist-mud1/mud-31931-v4fr/man0-frdev/1", "dst-manufacturer": "otherman.nist.local", "byte-count": 0, "table-id": 2, "priority": 40, "src-model": "https://sensor.nist.local/nist-mud1", "flow-name": "MetadaProtocolAndSrcDstPortMatchGoToTable(protocol=6,srcPort=-1,dstPort=800,targetTable=3)", "packet-count": 0 }, { "flow-id": "https://sensor.nist.local/nist-mud1/mud-31931-v4fr/cl0-frdev",

Test Case Field	Description
	"byte-count": 0, "table-id": 2, "priority": 40, "src-model": "*https://sensor.nist.local/nist-mud1*", "flow-name": "metadataDestIpAndPortMatchGo-ToNext(destIp=203.0.113.13,srcPort=-1,destPort=443,proto-col=6,sendToController=false)", "packet-count": 0 }, { "flow-id": "*https://sensor.nist.local/nist-mud1/mud-31931-v4fr/man0-frdev/2*", "dst-manufacturer": "*otherman.nist.local*", "byte-count": 0, "table-id": 2, "priority": 40, "src-model": "*https://sensor.nist.local/nist-mud1*", "flow-name": "MetadaPro-tocolAndSrcDstPortMatchGoToTable(proto-col=6,srcPort=800,dstPort=-1,targetTable=3)", "packet-count": 0 }, { "flow-id": "*https://sensor.nist.local/nist-mud1/mud-31931-v4fr/loc0-frdev/2*", "byte-count": 0, "table-id": 2, "dst-local-networks-flag": true, "priority": 40, "src-model": "*https://sensor.nist.local/nist-mud1*", "flow-name": "MetadaPro-tocolAndSrcDstPortMatchGoToTable(protocol=6,srcPort=-1,dstPort=80,targetTable=3)", "packet-count": 0 }, { "flow-id": "*https://sensor.nist.local/nist-mud1/mud-31931-v4fr/loc0-frdev/1*", "byte-count": 0, "table-id": 2,

Test Case Field	Description
	"dst-local-networks-flag": true, "priority": 40, "src-model": "*https://sensor.nist.local/nist-mud1*", "flow-name": "MetadaProtocolAndSrcDstPortMatchGoToTable(protocol=6,srcPort=80,dstPort=-1,targetTable=3)", "packet-count": 0 }, { "flow-id": "*https://sensor.nist.local/nist-mud1/mud-31931-v4to/man0-todev/TCP_DIRECTION_CHECK*", "byte-count": 0, "table-id": 2, "dst-model": "*https://sensor.nist.local/nist-mud1*", "priority": 41, "src-manufacturer": "*otherman.nist.local*", "flow-name": "MetadataTcpSynSrcIpAndPortMatchToToNextTableFlow(srcPort=-1,dstPort=800,targetTable=5)", "packet-count": 0 }, { "flow-id": "*https://sensor.nist.local/nist-mud1/mud-31931-v4fr/loc0-frdev/TCP_DIRECTION_CHECK*", "byte-count": 0, "table-id": 2, "dst-local-networks-flag": true, "priority": 41, "src-model": "*https://sensor.nist.local/nist-mud1*", "flow-name": "MetadataTcpSynSrcIpAndPortMatchToToNextTableFlow(srcPort=-1,dstPort=80,targetTable=5)", "packet-count": 0 }, { "flow-id": "*https://sensor.nist.local/nist-mud1/mud-31931-v4to/loc1-todev/TCP_DIRECTION_CHECK*", "src-local-networks-flag": true, "byte-count": 0, "table-id": 2, "dst-model": "*https://sensor.nist.local/nist-mud1*",

Test Case Field	Description
	"priority": 41, "flow-name": "MetadataTcpSynSrcIpAndPortMatch-ToToNextTableFlow(srcPort=-1,dstPort=888,targetTable=5)", "packet-count": 0 }, { "flow-id": "*https://sensor.nist.local/nist-mud1/NO_TO_DEV_ACE_MATCH_DROP*", "byte-count": 0, "table-id": 3, "dst-model": "*https://sensor.nist.local/nist-mud1*", "priority": 35, "flow-name": "metadataMatchGoToTable(5)", "packet-count": 0 }, { "flow-id": "*https://sensor.nist.local/nist-mud1/mud-31931-v4to/myman0-todev/1*", "byte-count": 0, "table-id": 3, "dst-model": "*https://sensor.nist.local/nist-mud1*", "priority": 40, "src-manufacturer": "*sensor.nist.local*", "flow-name": "MetadaPro-tocolAndSrcDstPortMatchGoToTable(proto-col=6,srcPort=8888,dstPort=-1,targetTable=4)", "packet-count": 0 }, { "flow-id": "*https://sensor.nist.local/nist-mud1/mud-31931-v4to/loc1-todev/1*", "src-local-networks-flag": true, "byte-count": 0, "table-id": 3, "dst-model": "*https://sensor.nist.local/nist-mud1*", "priority": 40, "flow-name": "MetadaPro-tocolAndSrcDstPortMatchGoToTable(proto-col=6,srcPort=888,dstPort=-1,targetTable=4)", "packet-count": 0

Test Case Field	Description
	}, { "flow-id": "*https://sensor.nist.local/nist-mud1/mud-31931-v4to/man0-todev/1*", "byte-count": 0, "table-id": 3, "dst-model": "*https://sensor.nist.local/nist-mud1*", "priority": 40, "src-manufacturer": "*otherman.nist.local*", "flow-name": "MetadaProtocolAndSrcDstPortMatchGoToTable(protocol=6,srcPort=800,dstPort=-1,targetTable=4)", "packet-count": 0 }, { "flow-id": "*https://sensor.nist.local/nist-mud1/mud-31931-v4to/cl0-todev*", "byte-count": 0, "table-id": 3, "dst-model": "*https://sensor.nist.local/nist-mud1*", "priority": 40, "flow-name": "metadataSrcIpAndPortMatchGoTo(srcAddress =203.0.113.13,srcPort = 443,dstPort -1,protocol=6,targetTable=4)", "packet-count": 0 }, { "flow-id": "*https://sensor.nist.local/nist-mud1/mud-31931-v4to/myctl0-todev*", "byte-count": 0, "table-id": 3, "dst-model": "*https://sensor.nist.local/nist-mud1*", "priority": 40, "flow-name": "metadataSrcIpAndPortMatchGoTo(srcAddress =10.0.41.225,srcPort = 4000,dstPort -1,protocol=17,targetTable=4)", "packet-count": 0 }, {

Test Case Field	Description
	"flow-id": "https://sensor.nist.local/nist-mud1/mud-31931-v4to/ent0-todev", "byte-count": 0, "table-id": 3, "dst-model": "https://sensor.nist.local/nist-mud1", "priority": 40, "flow-name": "metadataSrcIpAndPortMatch-GoTo(srcAddress =10.0.41.225,srcPort = 8080,dstPort -1,protocol=6,targetTable=4)", "packet-count": 0 }, { "flow-id": "https://sensor.nist.local/nist-mud1/mud-31931-v4to/man0-todev/2", "byte-count": 0, "table-id": 3, "dst-model": "https://sensor.nist.local/nist-mud1", "priority": 40, "src-manufacturer": "otherman.nist.local", "flow-name": "MetadaProtocolAndSrcDstPortMatchGoToTable(protocol=6,srcPort=-1,dstPort=800,targetTable=4)", "packet-count": 0 }, { "flow-id": "https://sensor.nist.local/nist-mud1/mud-31931-v4to/myman0-todev/2", "byte-count": 0, "table-id": 3, "dst-model": "https://sensor.nist.local/nist-mud1", "priority": 40, "src-manufacturer": "sensor.nist.local", "flow-name": "MetadaProtocolAndSrcDstPortMatchGoToTable(protocol=6,srcPort=-1,dstPort=8888,targetTable=4)", "packet-count": 0 }, { "flow-id": "https://sensor.nist.local/nist-mud1/mud-31931-v4to/loc0-todev/2",

Test Case Field	Description
	"src-local-networks-flag": true, "byte-count": 0, "table-id": 3, "dst-model": "https://sensor.nist.local/nist-mud1", "priority": 40, "flow-name": "MetadaProtocolAndSrcDstPortMatchGoToTable(protocol=6,srcPort=80,dstPort=-1,targetTable=4)", "packet-count": 0 }, { "flow-id": "https://sensor.nist.local/nist-mud1/mud-31931-v4to/loc1-todev/2", "src-local-networks-flag": true, "byte-count": 0, "table-id": 3, "dst-model": "https://sensor.nist.local/nist-mud1", "priority": 40, "flow-name": "MetadaProtocolAndSrcDstPortMatchGoToTable(protocol=6,srcPort=-1,dstPort=888,targetTable=4)", "packet-count": 0 }, { "flow-id": "https://sensor.nist.local/nist-mud1/mud-31931-v4to/loc0-todev/1", "src-local-networks-flag": true, "byte-count": 0, "table-id": 3, "dst-model": "https://sensor.nist.local/nist-mud1", "priority": 40, "flow-name": "MetadaProtocolAndSrcDstPortMatchGoToTable(protocol=6,srcPort=-1,dstPort=80,targetTable=4)", "packet-count": 0 }, { "flow-id": "https://sensor.nist.local/nist-mud1/mud-31931-v4to/cl0-todev/TCP_DIRECTION_CHECK", "byte-count": 0,

Test Case Field	Description
	"table-id": 3, "dst-model": "*https://sensor.nist.local/nist-mud1*", "priority": 41, "flow-name": "MetadataTcpSynSrcIpAndPortMatch-ToToNextTableFlow (srcIp=203.0.113.13,srcPort=443,dstIp=null,dstPort=-1,tar-getTable=5)", "packet-count": 0 }, { "flow-id": "*https://sensor.nist.local/nist-mud1/mud-31931-v4to/ent0-todev/TCP_DIRECTION_CHECK*", "byte-count": 0, "table-id": 3, "dst-model": "*https://sensor.nist.local/nist-mud1*", "priority": 41, "flow-name": "MetadataTcpSynSrcIpAndPortMatch-ToToNextTableFlow (srcIp=10.0.41.225,srcPort=8080,dstIp=null,dstPort=-1,tar-getTable=5)", "packet-count": 0 }] } }
Overall Results	Pass

512 IPv6 is not supported in this implementation.

513 *5.1.2.2 Test Case IoT-2-v4*

514 **Table 5-3: Test Case IoT-2-v4**

Test Case Field	Description
Parent Requirement	(CR-3) The IoT DDoS example implementation shall include a MUD manager that can request a MUD file and signature from a MUD file server.

Test Case Field	Description
Testable Requirement	(CR-3.b) The MUD manager shall use the GET method (RFC 7231) to request MUD and signature files (per RFC 7230) from the MUD file server, but it cannot validate the MUD file server's TLS certificate by using the rules in RFC 2818. (CR-3.b.1) The MUD manager shall drop the connection to the MUD file server. (CR-3.b.2) The MUD manager shall send locally defined policy to the router or switch that handles whether to allow or block traffic to and from the MUD-enabled IoT device.
Description	Shows that if a MUD manager cannot validate the TLS certificate of a MUD file server when trying to retrieve the MUD file for a specific IoT device, the MUD manager will drop the connection to the MUD file server and configure the router/switch according to locally defined policy regarding whether to allow or block traffic to the IoT device in question.
Associated Test Case(s)	IoT-11-v4
Associated Cybersecurity Framework Subcategory(ies)	PR.AC-7
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*mudfile-sensor.json*
Preconditions	1. All devices have been configured to use IPv4. 2. This MUD file is not currently cached at the MUD manager. 3. The MUD file server that is hosting the MUD file of the device under test does not have a valid TLS certificate. 4. Local policy has been defined to ensure that if the MUD file for a device is located on a server with an invalid certificate, the router/switch will be configured to deny all communication to and from the IoT device except standard network services (DHCP, DNS, network time protocol [NTP]).

Test Case Field	Description
	5. The MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings with respect to the IoT device being used in the test.
Procedure	Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. 1. Power on the IoT device and connect it to the test network. 2. On the IoT device, using the dhclient application with appropriate configuration file, manually emit a DHCPv4 message containing the device's MUD URL (IANA code 161). 3. The MUD manager snoops the DHCP request through the switch and extracts the MUD URL from the DHCP request. 4. The DHCP server receives the DHCP message containing the IoT device's MUD URL. 5. The DHCP server offers an IP address lease to the newly connected IoT device. 6. The IoT device requests this IP address lease, which the DHCP server acknowledges. 7. The MUD manager automatically contacts the MUD file server that is located by using the MUD URL, determines that it does not have a valid TLS certificate, and drops the connection to the MUD file server. 8. The MUD manager configures the router/switch that is closest to the IoT device so that it denies all communications to and from the IoT device except for standard network services (DHCP, DNS, NTP).
Expected Results	The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to local policy for communication to/from the IoT device. Only standard network services are to be allowed (DHCP, DNS, NTP)—this is the standard policy on MUD file verification failures.
Actual Results	**IoT device before DHCP request:** `python get-src-mac-metadata.py -m 00:13:EF:20:1D:6B` `{`

Test Case Field	Description
	`"input": {` `"mac-address": "00:13:EF:20:1D:6B"` `}` `}` `{` `"output": {` `"src-local-networks-flag": true,` `"src-quarantine-flag": false,` **`"src-blocked-flag": false,`** `"src-model": "UNCLASSIFIED",` `"src-manufacturer": "UNCLASSIFIED",` `"metadata": "100300300000000"` `}` `}` **<u>MUD manager logs—exception when there is an issue with MUD file</u>:** **`MudfileFetcher: fetchAndInstall : MUD URL =`** ***`https://sensor.nist.local/nistmud1`*** **`2019-09-03 14:41:34,114 | ERROR | n-dispatcher-232 | MudFileFetcher | 93 - gov.nist.antd.sdnmud-impl - 0.1.0 | Error fetching MUD file -- not installing`** `org.apache.http.conn.HttpHostConnectException: Connect to` *`sensor.nist.local`*`:443 [`*`sensor.nist.local`*`/127.0.0.1] failed: Connection refused (Connection refused)` `at org.apache.http.impl.conn.DefaultHttpClientConnectionOperator.connect(DefaultHttpClientConnectionOperator.java:159)[379:wrap_file__home_mudmanager_nist-mud_sdnmud-aggregator_karaf_target_assembly_system_org_apache_httpcomponents_httpclient_4.5.5_httpclient-4.5.5.jar:0.0.0]` `at org.apache.http.impl.conn.PoolingHttpClientConnectionManager.connect(PoolingHttpClientConnectionManager.java:373)[379:wrap_file__home_mudmanager_nist-mud_sdnmud-aggregator_karaf_target_assembly_system_org_apache_httpcomponents_httpclient_4.5.5_httpclient-4.5.5.jar:0.0.0]` `at org.apache.http.impl.execchain.MainClientExec.establishRoute(MainClientExec.java:381)[379:wrap_file__home_mudmanager_nist-mud_sdnmud-aggregator_karaf_target_assembly_system_org_apache_httpcomponents_httpclient_4.5.5_httpclient-4.5.5.jar:0.0.0]` `at org.apache.http.impl.execchain.MainClientExec.execute(MainClientExec.java:237)[379:wrap_file__home_mudmanager_nist-mud_sdnmud-aggregator_karaf_target_assembly_system_org_apache_httpcomponents_httpclient_4.5.5_httpclient-4.5.5.jar:0.0.0]` `at org.apache.http.impl.execchain.ProtocolExec.execute(ProtocolExec.java:185)[379:wrap_file__home_mudman-`

Test Case Field	Description
	ager_nist-mud_sdnmud-aggregator_karaf_target_assembly_sys- tem_org_apache_httpcomponents_httpclient_4.5.5_httpclient- 4.5.5.jar:0.0.0] at org.apache.http.impl.execchain.RetryExec.exe- cute(RetryExec.java:89)[379:wrap_file__home_mudmanager_nist- mud_sdnmud-agg **<u>IoT device after DHCP request</u>:** python get-src-mac-metadata.py -m 00:13:EF:20:1D:6B { "input": { "mac-address": "00:13:EF:20:1D:6B" } } { "output": { "src-local-networks-flag": true, "src-quarantine-flag": false, **"src-blocked-flag": true,** "src-model": "UNCLASSIFIED", "src-manufacturer": "UNCLASSIFIED", "metadata": "500300300000000" } }
Overall Results	Pass

IPv6 is not supported in this implementation.

5.1.2.3 Test Case IoT-3-v4

Table 5-4: Test Case IoT-3-v4

Test Case Field	Description
Parent Requirement	(CR-4) The IoT DDoS example implementation shall include a MUD file server that can serve a MUD file and signature to the MUD manager.
Testable Requirement	(CR-4.b) The MUD file server shall serve the file and signature to the MUD manager, and the MUD manager shall check to determine whether the certificate used to sign the MUD file was valid at the time of signing,

Test Case Field	Description
	i.e., the certificate had already expired when it was used to sign the MUD file. (CR-4.b.1) The MUD manager shall cease to process the MUD file. (CR-4.b.2) The MUD manager shall send locally defined policy to the router or switch that handles whether to allow or block traffic to and from the MUD-enabled IoT device.
Description	Shows that if a MUD file server serves a MUD file with a signature that was created with an expired certificate, the MUD manager will cease processing the MUD file.
Associated Test Case(s)	IoT-11-v4
Associated Cybersecurity Framework Subcategory(ies)	PR.DS-6
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*mudfile-sensor.json*
Preconditions	1. All devices have been configured to use IPv4. 2. This MUD file is not currently cached at the MUD manager. 3. The IoT device's MUD file is being hosted on a MUD file server that has a valid TLS certificate, but the MUD file signature was signed by a certificate that had already expired at the time of signature. 4. Local policy has been defined to ensure that if the MUD file for a device has a signature that was signed by a certificate that had already expired at the time of signature, the device's MUD PEP router/switch will be configured to deny all communication to/from the device. 5. The MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings with respect to the IoT device being used in the test.

Test Case Field	Description
Procedure	Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. 1. Power on the IoT device and connect it to the test network. 2. On the IoT device, using the dhclient application with appropriate configuration file, manually emit a DHCPv4 message containing the device's MUD URL (IANA code 161). 3. The DHCP server receives the DHCP message containing the IoT device's MUD URL. 4. The DHCP server offers an IP address lease to the newly connected IoT device. 5. The IoT device requests this IP address lease, which the DHCP server acknowledges. 6. The DHCP server sends the MUD URL to the MUD manager. 7. The MUD manager automatically contacts the MUD file server that is located by using the MUD URL, verifies that it has a valid TLS certificate, and requests the MUD file and signature from the MUD file server. 8. The MUD file server serves the MUD file and signature to the MUD manager, and the MUD manager detects that the MUD file's signature was created by using a certificate that had already expired at the time of signing. 9. The MUD manager configures the router/switch that is closest to the IoT device so that it denies all communications to and from the IoT device.
Expected Results	The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to local policy for communication to/from the IoT device. Only standard network services are to be allowed (DHCP, DNS, NTP)—this is the standard policy on MUD file verification failures.
Actual Results	**<u>IoT device before DHCP request</u>:** `python get-src-mac-metadata.py -m 00:13:EF:20:1D:6B` `{` `    "input": {`

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td></td><td>
<pre>
        "mac-address": "00:13:EF:20:1D:6B"
    }
}
{
    "output": {
        "src-local-networks-flag": true,
        "src-quarantine-flag": false,
        "src-blocked-flag": false,
        "src-model": "UNCLASSIFIED",
        "src-manufacturer": "UNCLASSIFIED",
        "metadata": "100300300000000"
    }
}
</pre>

<u>MUD manager logs—exception when there is an issue with MUD file</u>:

<pre>
MudfileFetcher: fetchAndInstall : MUD URL = https://sen-
sor.nist.local/nistmud1
2019-09-03 14:41:34,114 | ERROR | n-dispatcher-232 | Mud-
FileFetcher | 93 - gov.nist.antd.sdnmud-impl
- 0.1.0 | Error fetching MUD file -- not installing
org.apache.http.conn.HttpHostConnectException: Connect to
sensor.nist.local:443 [sensor.nist.local/127.0.0.1] failed:
Connection refused (Connection refused)
       at org.apache.http.impl.conn.DefaultHttpClientConnec-
tionOperator.connect(DefaultHttpClientConnectionOpera-
tor.java:159)[379:wrap_file__home_mudmanager_nist-
mud_sdnmud-aggregator_karaf_target_assembly_sys-
tem_org_apache_httpcomponents_httpclient_4.5.5_httpclient-
4.5.5.jar:0.0.0]
       at org.apache.http.impl.conn.PoolingHttpClientConnec-
tionManager.connect(PoolingHttpClientConnectionMan-
ager.java:373)[379:wrap_file__home_mudmanager_nist-
mud_sdnmud-aggregator_karaf_target_assembly_sys-
tem_org_apache_httpcomponents_httpclient_4.5.5_httpclient-
4.5.5.jar:0.0.0]
       at org.apache.http.impl.execchain.MainClientExec.es-
tablishRoute(MainClien-
tExec.java:381)[379:wrap_file__home_mudmanager_nist-
mud_sdnmud-aggregator_karaf_target_assembly_sys-
tem_org_apache_httpcomponents_httpclient_4.5.5_httpclient-
4.5.5.jar:0.0.0]
       at org.apache.http.impl.execchain.MainClientExec.exe-
cute(MainClientExec.java:237)[379:wrap_file__home_mudman-
ager_nist-mud_sdnmud-aggregator_karaf_target_assembly_sys-
tem_org_apache_httpcomponents_httpclient_4.5.5_httpclient-
4.5.5.jar:0.0.0]
       at org.apache.http.impl.execchain.ProtocolExec.exe-
cute(ProtocolExec.java:185)[379:wrap_file__home_mudman-
</pre>
</td></tr>
</table>

Test Case Field	Description
	`ager_nist-mud_sdnmud-aggregator_karaf_target_assembly_sys-tem_org_apache_httpcomponents_httpclient_4.5.5_httpclient-4.5.5.jar:0.0.0]` `       at org.apache.http.impl.execchain.RetryExec.exe-cute(RetryExec.java:89)[379:wrap_file__home_mudmanager_nist-mud_sdnmud-agg` **<u>IoT device after DHCP request</u>:** `python get-src-mac-metadata.py -m 00:13:EF:20:1D:6B` `{` `    "input": {` `        "mac-address": "00:13:EF:20:1D:6B"` `    }` `}` `{` `    "output": {` `        "src-local-networks-flag": true,` `        "src-quarantine-flag": false,` **`        "src-blocked-flag": true,`** `        "src-model": "UNCLASSIFIED",` `        "src-manufacturer": "UNCLASSIFIED",` `        "metadata": "500300300000000"` `    }` `}`
Overall Results	Pass

518 IPv6 is not supported in this implementation.

519 *5.1.2.4 Test Case IoT-4-v4*

520 **Table 5-5: Test Case IoT-4-v4**

Test Case Field	Description
Parent Requirement	(CR-5) The IoT DDoS example implementation shall include a MUD manager that can translate local network configurations based on the MUD file.
Testable Requirement	(CR-5.b) The MUD manager shall attempt to validate the signature of the MUD file, but the signature validation fails (even though the certificate

Test Case Field	Description
	that had been used to create the signature had not been expired at the time of signing, i.e., the signature is invalid for a different reason). (CR-5.b.1) The MUD manager shall cease processing the MUD file. (CR-5.b.2) The MUD manager shall send locally defined policy to the router or switch that handles whether to allow or block traffic to and from the MUD-enabled IoT device.
Description	Shows that if the MUD manager determines that the signature on the MUD file it receives from the MUD file server is invalid, it will cease processing the MUD file and configure the router/switch according to locally defined policy regarding whether to allow or block traffic to the IoT device in question.
Associated Test Case(s)	IoT-11-v4
Associated Cybersecurity Framework Subcategory(ies)	PR.DS-6
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*mudfile-sensor.json*
Preconditions	1. All devices have been configured to use IPv4. 2. This MUD file is not currently cached at the MUD manager. 3. The MUD file that is served from the MUD file server to the MUD manager has a signature that is invalid, even though it was signed by a certificate that had not expired at the time of signing. 4. Local policy has been defined to ensure that if the MUD file for a device has an invalid signature, the device's MUD PEP router/switch will be configured to deny all communications to/from the device except for standard network services (DHCP, DNS, NTP). 5. The MUD PEP router/switch does not yet have any configuration settings with respect to the IoT device being used in the test.

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td>Procedure</td><td>Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test.

1. Power on the IoT device and connect it to the test network.
2. On the IoT device, using the dhclient application with appropriate configuration file, manually emit a DHCPv4 message containing the device’s MUD URL (IANA code 161).
3. The MUD manager snoops the DHCP request through the switch and extracts the MUD URL from the DHCP request.
4. The DHCP server receives the DHCP message containing the IoT device’s MUD URL.
5. The DHCP server offers an IP address lease to the newly connected IoT device.
6. The IoT device requests this IP address lease, which the DHCP server acknowledges.
7. The MUD manager automatically contacts the MUD file server that is located by using the MUD URL, verifies that it has a valid TLS certificate, and requests the MUD file and signature from the MUD file server.
8. The MUD file server sends the MUD file, and the MUD manager detects that the MUD file’s signature is invalid.
9. The MUD manager configures the router/switch that is closest to the IoT device so that it denies all communications to and from the IoT device except standard network services (DHCP, DNS, NTP).</td></tr>
<tr><td>Expected Results</td><td>The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to local policy for communication to/from the IoT device. Only standard network services are to be allowed (DHCP, DNS, NTP)—this is the standard policy on MUD file verification failures.</td></tr>
<tr><td>Actual Results</td><td><u>IoT device before DHCP request</u>:<pre>python get-src-mac-metadata.py -m 00:13:EF:20:1D:6B
{
    "input": {
        "mac-address": "00:13:EF:20:1D:6B"</pre></td></tr>
</table>

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

<pre>
    }
}
{
    "output": {
        "src-local-networks-flag": true,
        "src-quarantine-flag": false,
        "src-blocked-flag": false,
        "src-model": "UNCLASSIFIED",
        "src-manufacturer": "UNCLASSIFIED",
        "metadata": "100300300000000"
    }
}
</pre>

<u>MUD manager logs—exception when there is an issue with MUD file</u>:
<pre>
 MudfileFetcher: fetchAndInstall : MUD URL = https://sen-
sor.nist.local/nistmud1
2019-09-03 14:41:34,114 | ERROR | n-dispatcher-232 | Mud-
FileFetcher | 93 - gov.nist.antd.sdnmud-impl
- 0.1.0 | Error fetching MUD file -- not installing
org.apache.http.conn.HttpHostConnectException: Connect to
sensor.nist.local:443 [sensor.nist.local/127.0.0.1] failed:
Connection refused (Connection refused)
       at org.apache.http.impl.conn.DefaultHttpClientConnec-
tionOperator.connect(DefaultHttpClientConnectionOpera-
tor.java:159)[379:wrap_file__home_mudmanager_nist-
mud_sdnmud-aggregator_karaf_target_assembly_sys-
tem_org_apache_httpcomponents_httpclient_4.5.5_httpclient-
4.5.5.jar:0.0.0]
       at org.apache.http.impl.conn.PoolingHttpClientConnec-
tionManager.connect(PoolingHttpClientConnectionMan-
ager.java:373)[379:wrap_file__home_mudmanager_nist-
mud_sdnmud-aggregator_karaf_target_assembly_sys-
tem_org_apache_httpcomponents_httpclient_4.5.5_httpclient-
4.5.5.jar:0.0.0]
       at org.apache.http.impl.execchain.MainClientExec.es-
tablishRoute(MainClien-
tExec.java:381)[379:wrap_file__home_mudmanager_nist-
mud_sdnmud-aggregator_karaf_target_assembly_sys-
tem_org_apache_httpcomponents_httpclient_4.5.5_httpclient-
4.5.5.jar:0.0.0]
       at org.apache.http.impl.execchain.MainClientExec.exe-
cute(MainClientExec.java:237)[379:wrap_file__home_mudman-
ager_nist-mud_sdnmud-aggregator_karaf_target_assembly_sys-
tem_org_apache_httpcomponents_httpclient_4.5.5_httpclient-
4.5.5.jar:0.0.0]
       at org.apache.http.impl.execchain.ProtocolExec.exe-
cute(ProtocolExec.java:185)[379:wrap_file__home_mudman-
ager_nist-mud_sdnmud-aggregator_karaf_target_assembly_sys-
tem_org_apache_httpcomponents_httpclient_4.5.5_httpclient-
4.5.5.jar:0.0.0]
</pre>
</td></tr>
</table>

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td></td><td><pre>   at org.apache.http.impl.execchain.RetryExec.exe-
cute(RetryExec.java:89)[379:wrap_file__home_mudmanager_nist-
mud_sdnmud-agg</pre>
<u>IoT device after DHCP request</u>:
<pre>python get-src-mac-metadata.py -m 00:13:EF:20:1D:6B
{
   "input": {
      "mac-address": "00:13:EF:20:1D:6B"
   }
}
{
   "output": {
      "src-local-networks-flag": true,
      "src-quarantine-flag": false,
      "src-blocked-flag": true,
      "src-model": "UNCLASSIFIED",
      "src-manufacturer": "UNCLASSIFIED",
      "metadata": "500300300000000"
   }
}</pre></td></tr>
<tr><td>Overall Results</td><td>Pass</td></tr>
</table>

521 IPv6 is not supported in this implementation.

522 *5.1.2.5 Test Case IoT-5-v4*

523 **Table 5-6: Test Case IoT-5-v4**

Test Case Field	Description
Parent Requirement	(CR-7) The IoT DDoS example implementation shall allow the MUD-enabled IoT device to communicate with approved internet services in the MUD file. (CR-8) The IoT DDoS example implementation shall deny communications from a MUD-enabled IoT device to unapproved internet services (i.e., services that are implicitly denied by virtue of not being explicitly approved).

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

Test Case Field	Description
	with internet services will be enforced as expected, with communications that are configured as denied being blocked, and communications that are configured as permitted being allowed.
Associated Test Case(s)	IoT-1-v4
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-3, PR.DS-5, PR.IP-1, PR.PT-3
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*mudfile-sensor.json, mudfile-otherman.json*
Preconditions	Test IoT-1-v4 has run successfully, meaning that the MUD PEP router/switch has been configured to enforce the following policies for the IoT device in question (as defined in the MUD file in Section 5.1.3): a) Explicitly permit *https://yes-permit-from.com* to initiate communications with the IoT device. b) Explicitly permit the IoT device to initiate communications with *https://yes-permit-to.com*. c) Implicitly deny all other communications with the internet, including denying: i) the IoT device to initiate communications with *https://yes-permit-from.com* ii) *https://yes-permit-to.com* to initiate communications with the IoT device iii) communication between the IoT device and all other internet locations, such as *https://unnamed-to.com* (by not mentioning this or any other URLs in the MUD file)
Procedure	Note: Procedure steps with strikethrough are not tested due to NAT. 1. As stipulated in the preconditions, right before this test, test IoT-1-v4 must have been run successfully.

Test Case Field	Description
	2. Initiate communications from the IoT device to *https://yes-permit-to.com* and verify that this traffic is received at *https://yes-permit-to.com*. (egress) ~~3. Initiate communications to the IoT device from *https://yes-permit-to.com* and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at the IoT device. (ingress)~~ ~~4. Initiate communications to the IoT device from *https://yes-permit-from.com* and verify that this traffic is received at the IoT device. (ingress)~~ 5. Initiate communications from the IoT device to *https://yes-permit-from.com* and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at *https://yes-permit-from.com*. (ingress) 6. Initiate communications from the IoT device to *https://unnamed.com* and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at *https://unnamed.com*. (egress) ~~7. Initiate communications to the IoT device from *https://unnamed.com* and verify that this traffic is received at the MUD PEP, but it is not forwarded by the MUD PEP, nor is it received at the IoT device. (ingress)~~
Expected Results	Each of the results that is listed as needing to be verified in procedure steps above occurs as expected.
Actual Results	Procedure 2: Connection to approved server (*www.nist.local* port 443) successfully initiated by IoT device: `sensor ] wget www.nist.local:443` `--2019-07-04 05:09:29--  http://www.nist.local:443/` `Resolving www.nist.local (www.nist.local)... 203.0.113.13` `Connecting to www.nist.local (www.nist.local)|203.0.113.13|:443... connected.` `HTTP request sent, awaiting response... 200 OK` `Length: 116855 (114K) [text/html]` `Saving to: 'index.html.51'`

Test Case Field	Description
	`index.html.51` `100%[=========================================================` `=============================>] 114.12K   414KB/s    in 0.3s` `2019-07-04 05:09:30 (414 KB/s) - ‘index.html.51’ saved [116855/116855]` Procedure 5: Connection from device (another manufacturer) to server (*www.nist.local* port 443) fails: `anotherman ] ` **`wget www.nist.local:443 --timeout 30 --tries 2`** `--2019-05-02 12:14:32--  http://www.nist.local:443/` `Resolving www.nist.local (www.nist.local)... 203.0.113.13` `Connecting to www.nist.local (www.nist.local)|203.0.113.13|:443... failed: Connection timed out.` `Retrying.` `--2019-05-02 12:15:03--  (try: 2)  http://www.nist.local:443/` `Connecting to www.nist.local (www.nist.local)|203.0.113.13|:443... failed: Connection timed out.` `Giving up.` Procedure 6: IoT device failed to connect to unapproved server (*www.antd.local* any port): `sensor ] ` **`wget www.antd.local --timeout 30 --tries 2`** `--2019-07-04 05:14:57--  http://www.antd.local/` `Resolving www.antd.local (www.antd.local)... 203.0.113.14` `Connecting to www.antd.local (www.antd.local)|203.0.113.14|:80... failed: Connection timed out.` `Retrying.` `--2019-07-04 05:15:28--  (try: 2)  http://www.antd.local/` `Connecting to www.antd.local (www.antd.local)|203.0.113.14|:80... failed: Connection timed out.` `Giving up.`
Overall Results	Pass

524 IPv6 is not supported in this implementation.

5.1.2.6 Test Case IoT-6-v4

Table 5-7: Test Case IoT-6-v4

Test Case Field	Description
Parent Requirement	(CR-9) The IoT DDoS example implementation shall allow the MUD-enabled IoT device to communicate laterally with devices that are approved in the MUD file. (CR-10) The IoT DDoS example implementation shall deny lateral communications from a MUD-enabled IoT device to devices that are not approved in the MUD file (i.e., devices that are implicitly denied by virtue of not being explicitly approved).
Testable Requirement	(CR-9.a) The MUD-enabled IoT device shall attempt to initiate lateral traffic to approved devices. (CR-9.a.1) The router or switch shall receive the attempt and shall allow it to pass based on the filters from the MUD file. (CR-9.b) An approved device shall attempt to initiate a lateral connection to the MUD-enabled IoT device. (CR-9.b.1) The router or switch shall receive the attempt and shall allow it to pass based on the filters from the MUD file. (CR-10.a) The MUD-enabled IoT device shall attempt to initiate lateral traffic to unapproved (implicitly denied) devices. (CR-10.a.1) The router or switch shall receive the attempt and shall deny it based on the filters from the MUD file. (CR-10.b) An unapproved (implicitly denied) device shall attempt to initiate a lateral connection to the MUD-enabled IoT device. (CR-10.b.1) The router or switch shall receive the attempt and shall deny it based on the filters from the MUD file.
Description	Shows that, upon connection to the network, a MUD-enabled IoT device used in the IoT DDoS example implementation has its MUD PEP router/switch automatically configured to enforce the route filtering that is described in the device's MUD file with respect to communication with lateral devices. Further shows that the policies that are configured on the MUD PEP router/switch with respect to communication with lateral devices will be enforced as expected, with communications that are

Test Case Field	Description
	configured as denied being blocked and communications that are configured as permitted being allowed.
Associated Test Case(s)	IoT-1-v4
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-3, PR.DS-5, PR.AC-5, PR.IP-1, PR.PT-3, PR.IP-3, PR.DS-3
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*mudfile-sensor.json*
Preconditions	Test IoT-1-v4 has run successfully, meaning that the MUD PEP router/switch has been configured to enforce the following policies for the IoT device in question with respect to local communications (as defined in the MUD files in Section 5.1.3): a) Local-network class—Explicitly permit **local communication to and from the IoT device and any local hosts** (including the specific local hosts *anyhost-to* and *anyhost-from*) **for specific services,** as specified in the MUD file by source port: any; destination port: 80; and protocol: TCP, and which party initiates the connection. b) Manufacturer class—Explicitly permit **local communication to and from the IoT device and other classes of IoT devices, as identified by their MUD URL (*www.devicetype.com*), and further constrained** by source port: any; destination port: 80; and protocol: TCP. c) Same-manufacturer class—Explicitly permit **local communication to and from IoT devices of the same manufacturer as the IoT device in question (the domain in the MUD URLs [mudfileserver] of the other IoT devices is the same as the domain in the MUD URL [mudfileserver] of the IoT device in question),** and further constrained by source port: any; destination port: 80; and protocol: TCP.

Test Case Field	Description
	d) Implicitly deny all other local communication that is not explicitly permitted in the MUD file, including denying i) ***anyhost-to* to initiate communications** with the IoT device ii) the **IoT device to initiate communications** with *anyhost-to* by **using a source port, destination port, or protocol (TCP or UDP) that is not explicitly permitted** iii) the **IoT device to initiate communications with *anyhost-from*** iv) ***anyhost-from* to initiate communications** with the IoT device by **using a source port, destination port, or protocol (TCP or UDP) that is not explicitly permitted** v) communications between the IoT device and all lateral hosts (including *unnamed-host*) whose **MUD URLs are not explicitly mentioned** as being permissible in the MUD file vi) communications between the IoT device and all lateral hosts whose **MUD URLs are explicitly mentioned** as being permissible **but using a source port, destination port, or protocol (TCP or UDP) that is not explicitly permitted** vii) communications between the IoT device and all lateral hosts that are **not from the same manufacturer** as the IoT device in question viii) communications between the IoT device and a lateral host that **is from the same manufacturer but using a source port, destination port, or protocol (TCP or UDP) that is not explicitly permitted**
Procedure	1. As stipulated in the preconditions, right before this test, test IoT-1-v4 must have been run successfully. 2. Local-network (ingress): Initiate communications to the IoT device from *anyhost-from* **for specific permitted service,** and verify that this traffic is received at the IoT device. 3. Local-network (egress): **Initiate communications from the IoT device to *anyhost-from*** for specific permitted service, and verify that this traffic is received at the MUD PEP, but it **is not forwarded** by the MUD PEP, nor is it received at *anyhost-from*.

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

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td>Expected Results</td><td>Each of the results that is listed as needing to be verified in the procedure steps above occurs as expected.</td></tr>
<tr><td>Actual Results</td><td>2. Local-network (ingress)—allowed:<pre>
    laptop] wget sensor:80
    --2019-05-07 10:21:03-- http://sensor/
    Resolving sensor (sensor)... 10.0.41.190
    Connecting to sensor (sensor)|10.0.41.190|:80... con-
    nected.
    HTTP request sent, awaiting response... 200 OK
    Length: 116344 (114K) [text/html]
    Saving to: ‘index.html.3’

    index.html.3
    100%[===
    =====================================>] 113.62K
    389KB/s   in 0.3s

2019-05-07 10:21:04 (389 KB/s) - ‘index.html.3’ saved
[116344/116344]
</pre>
<hr>
3. Local-network (egress)—blocked:<pre>
    sensor] wget laptop:80 --tries 2 --timeout 30
    --2019-07-14 03:24:07-- http://laptop/
    Resolving laptop (laptop)... 10.0.41.135
    Connecting to laptop (laptop)|10.0.41.135|:80...
    failed: Connection timed out.
    Retrying.

    --2019-07-14 03:24:38-- (try: 2) http://laptop/
    Connecting to laptop (laptop)|10.0.41.135|:80...
    failed: Connection timed out.
Giving up.
</pre>
<hr>
4. Local-network, controller, my-controller, manufacturer class (egress)—allowed:
Local-network:<pre>
    sensor] wget laptop:888
    --2019-07-17 00:45:37-- http://laptop:888/
    Resolving laptop (laptop)... 10.0.41.135
    Connecting to laptop (laptop)|10.0.41.135|:888...
    connected.
</pre></td></tr>
</table>

Test Case Field	Description
	HTTP request sent, awaiting response... 200 OK Length: 116344 (114K) [text/html] Saving to: 'index.html.7' index.html.7 100%[== ==================================>] 113.62K 703KB/s in 0.2s 2019-07-17 00:45:38 (703 KB/s) - 'index.html.7' saved [116344/116344] Controller: sensor] **wget laptop2:8080** --2019-07-14 03:27:43-- http://laptop2:8080/ Resolving laptop2 (laptop2)... 10.0.41.225 Connecting to laptop2 (laptop2)|10.0.41.225|:8080... connected. HTTP request sent, awaiting response... 200 OK Length: 116344 (114K) [text/html] Saving to: 'index.html.53' index.html.53 100%[== ==================================>] 113.62K 548KB/s in 0.2s 2019-07-14 03:27:43 (548 KB/s) - 'index.html.53' saved [116344/116344] My-controller: sensor] **python udpping.py --client --npings 6 --host laptop2 --port 4000** start ... Namespace(bind=False, client=True, host='laptop2', npings=6, port=4000, quiet=False, server=False, timeout=False) PING 1 03:31:59 RTT = 1.24670505524 PING 2 03:32:00 RTT = 0.812637805939 PING 3 03:32:01 RTT = 0.652308940887 PING 4 03:32:02

Test Case Field	Description
	`RTT = 0.784868001938` `PING 5 03:32:02` `RTT = 0.573136806488` `PING 6 03:32:03` `RTT = 0.481912136078` `[rc=6]` Manufacturer: `sensor ]` **`wget anotherman:800`** `--2019-07-21 05:23:07--  http://anotherman:800/` `Resolving anotherman (anotherman)... 10.0.41.245` `Connecting to anotherman (another-man)|10.0.41.245|:800... connected.` `HTTP request sent, awaiting response... 200 OK` `Length: 116855 (114K) [text/html]` `Saving to: ‘index.html.1’` `index.html.1` `100%[=====================================================================================>] 114.12K  --.-KB/s    in 0.1s` `2019-07-21 05:23:08 (816 KB/s) - ‘index.html.1’ saved [116855/116855]` 5. Local-network, controller, my-controller, manufacturer class (in-gress)—blocked: Local-network: `laptop ]` **`wget sensor:888`** `--2019-05-10 07:47:18--  http://sensor:888/` `Resolving sensor (sensor)... 10.0.41.190` `Connecting to sensor (sensor)|10.0.41.190|:888... ^C` `laptop ] wget sensor:888 --timeout 30 --tries 2` `--2019-05-10 07:47:29--  http://sensor:888/` `Resolving sensor (sensor)... 10.0.41.190` `Connecting to sensor (sensor)|10.0.41.190|:888... failed: Connection timed out.` `Retrying.` `--2019-05-10 07:48:00--  (try: 2)  http://sensor:888/` `Connecting to sensor (sensor)|10.0.41.190|:888... failed: Connection timed out.` `Giving up.`

Test Case Field	Description
	Controller: `laptop2 ] `**`wget sensor:8080 --tries 2 --timeout 30`** `--2019-07-13 18:42:31--  http://sensor:8080/` `Resolving sensor (sensor)... 10.0.41.190` `Connecting to sensor (sensor)|10.0.41.190|:8080... failed: Connection timed out.` `Retrying.` `--2019-07-13 18:43:02--  (try: 2)  http://sensor:8080/` `Connecting to sensor (sensor)|10.0.41.190|:8080... failed: Connection timed out.` `Giving up.` My-controller: `laptop2 ] `**`python udpping.py --client --npings 6 --host sensor --port 4000`** `start ...` `Namespace(bind=False, client=True, host='sensor', npings=10, port=4000, quiet=False, server=False, timeout=False)` `PING 1 18:43:49` `UDPPING FAILED` `PING 2 18:43:50` `UDPPING FAILED` `PING 3 18:43:51` `UDPPING FAILED` `PING 4 18:43:52` `UDPPING FAILED` `PING 5 18:43:53` `UDPPING FAILED` `PING 6 18:43:54` `[rc=0]` Manufacturer: `anotherman ]`**`wget sensor:800 --timeout 30 --tries 2`** `--2019-05-20 05:55:48--  http://sensor:800/` `Resolving sensor (sensor)... 10.0.41.190` `Connecting to sensor (sensor)|10.0.41.190|:800... failed: Connection timed out.` `Retrying.`

Test Case Field	Description
	`--2019-05-20 05:56:19-- (try: 2) http://sensor:800/` `Connecting to sensor (sensor)|10.0.41.190|:800... failed: Connection timed out.` `Giving up.`
	6. No associated class (egress)—blocked: `sensor ]` **`ping laptop -c 10`** `PING laptop (10.0.41.135) 56(84) bytes of data.` `--- laptop ping statistics ---` `10 packets transmitted, 0 received, 100% packet loss, time 9355ms`
	7. No associated class (ingress)—blocked: `laptop ]` **`ping sensor -c 10`** `PING sensor (10.0.41.190) 56(84) bytes of data.` `--- sensor ping statistics ---` `10 packets transmitted, 0 received, 100% packet loss, time 9337ms`
	8. Same-manufacturer class (egress)—allowed: `sensor ]` **`wget sameman:8888`** `--2019-07-17 01:19:08-- http://sameman:8888/` `Resolving sameman (sameman)... 10.0.41.220` `Connecting to sameman (sameman)|10.0.41.220|:8888... connected.` `HTTP request sent, awaiting response... 200 OK` `Length: 116855 (114K) [text/html]` `Saving to: ‘index.html.8’` `index.html.8` `100%[=====================================================================================>] 114.12K 705KB/s in 0.2s` `2019-07-17 01:19:08 (705 KB/s) - ‘index.html.8’ saved [116855/116855]`
	9. Same-manufacturer class (egress)—blocked: `sensor ]` **`ping sameman -c 10`** `PING sameman (10.0.41.220) 56(84) bytes of data.`

Test Case Field	Description
	`--- sameman ping statistics ---` `10 packets transmitted, 0 received, 100% packet loss, time 9383ms`
Overall Results	Pass

IPv6 is not supported in this implementation.

5.1.2.7 Test Case IoT-9-v4

Table 5-8: Test Case IoT-9-v4

Test Case Field	Description
Parent Requirements	(CR-13) The IoT DDoS example implementation shall ensure that for each rule in a MUD file that pertains to an external domain, the MUD PEP router/switch will get configured with all possible instantiations of that rule, insofar as each instantiation contains one of the IP addresses to which the domain in that MUD file rule may be resolved when queried by the SDN-capable switch.
Testable Requirements	(CR-13.a) The MUD file for a device shall contain a rule involving a domain that can resolve to multiple IP addresses when queried by the SDN-capable switch. Flow rules for permitting access to each of those IP addresses will be inserted into the SDN-capable switch, for the device in question, and the device will be permitted to communicate with all of those IP addresses.
Description	Shows that if a domain in a MUD file rule resolves to multiple IP addresses when the address resolution is requested by the router/switch, then 1. flow rules instantiating that MUD file rule corresponding to each of these IP addresses will be configured in the switch for the IoT device associated with the MUD file, and

Test Case Field	Description
	2. the IoT device associated with the MUD file will be permitted to communicate with all the IP addresses to which that domain resolves
Associated Test Case(s)	N/A
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-1, ID.AM-2, ID.AM-3, PR.DS-5, DE.AE-1, PR.AC-4, PR.AC-5, PR.IP-1, PR.IP-3, PR.DS-2
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*mudfile-sensor.json*
Preconditions	1. The SDN-capable switch on the home/small-business network does not yet have any flow rules pertaining to the IoT device being used in the test. 2. The MUD file for the IoT device being used in the test is identical to the MUD file provided in Section 5.1.3. (Therefore, the MUD file used in the test permits the device to send data to *www.updateserver.com*.) 3. The DNS server that the switch uses resolves the domain *www.updateserver.com* to only one IP address. 4. The tester has access to a DNS server that will be used by the SDN-capable switch and can configure it so that it will resolve the domain *www.updateserver.com* to any of these addresses when queried by the SDN-capable switch: x1.x1.x1.x1, y1.y1.y1.y1, and z1.z1.z1.z1. 5. There is a server running at each of these three IP addresses.
Procedure	1. Verify that the SDN-capable switch on the home/small-business network does not yet have any flow rules installed with respect to the IoT device being used in the test. 2. Run test IoT-1-v4. The result should be that the SDN-capable switch on the home/small-business network has been configured to explicitly permit the IoT device to initiate communication with *www.updateserver.com*.

<table>
<tr><th>Test Case Field</th><th>Description</th></tr>
<tr><td></td><td>3. Attempt to reach www.updateserver.com on the device, and see that the SDN-capable switch is then configured with flow rules that permit the IoT device to send data to IP addresses x1.x1.x1.x1, y1.y1.y1.y1, and z1.z1.z1.z1.
4. Have the device in question attempt to connect to x1.x1.x1.x1, y1.y1.y1.y1, and z1.z1.z1.z1.</td></tr>
<tr><td>Expected Results</td><td>The SDN-capable switch has had its configuration changed, i.e., it has been configured with flow rules that permit the IoT device to send data to multiple IP addresses (i.e., x1.x1.x1.x1, y1.y1.y1.y1, and z1.z1.z1.z1).
The IoT device is permitted to send data to each of the servers at these addresses.</td></tr>
<tr><td>Actual Results</td><td>In this test, www.nist.local (an allowed internet interaction) resolved to two addresses (203.0.113.13 and 203.0.113.15). When the device attempted to reach www.nist.local, both IP addresses were allowed by the flows as intended.
The flow rules relating to this interaction are shown below:
<code>cookie=0x95d11, duration=365.237s, table=2, n_packets=1, n_bytes=74, priority=40,tcp,metadata=0x400000000000/0xfff00000000000,nw_dst=203.0.113.13,tp_dst=443 actions=wr</code>

<code>cookie=0x95d11, duration=365.141s, table=2, n_packets=6, n_bytes=493, priority=40,tcp,metadata=0x400000000000/0xfff00000000000,nw_dst=203.0.113.15,tp_dst=443 actions=w</code>

<code>cookie=0x95d11, duration=365.220s, table=3, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x4000/0xfff000,nw_src=203.0.113.13, tp_src=443 actions=write_metadata:0xff</code>

<code>cookie=0x95d11, duration=365.125s, table=3, n_packets=0, n_bytes=0, priority=40,tcp,metadata=0x4000/0xfff000,nw_src=203.0.113.15, tp_src=443 actions=write_metadata:0xff</code></td></tr>
<tr><td>Overall Result</td><td>Pass</td></tr>
</table>

530 IPv6 is not supported in this implementation.

531 #### *5.1.2.8 Test Case IoT-10-v4*

532 **Table 5-9: Test Case IoT-10-v4**

Test Case Field	Description
Parent Requirements	(CR-12) The IoT DDoS example implementation shall include a MUD manager that uses a cached MUD file rather than retrieve a new one if the cache-validity time period has not yet elapsed for the MUD file indicated by the MUD URL**.** The MUD manager should fetch a new MUD file if the cache-validity time period has already elapsed.
Testable Requirements	(CR-12.a) The MUD manager shall check if the file associated with the MUD URL is present in its cache and shall determine that it is. (CR-12.a.1) The MUD manager shall check whether the amount of time that has elapsed since the cached file was retrieved is less than or equal to the number of hours in the cache-validity value for this MUD file. If so, the MUD manager shall apply the contents of the cached MUD file. (CR-12.a.2) The MUD manager shall check whether the amount of time that has elapsed since the cached file was retrieved is greater than the number of hours in the cache-validity value for this MUD file. If so, the MUD manager may (but does not have to) fetch a new file by using the MUD URL received.
Description	Shows that, upon connection to the network, a MUD-enabled IoT device used in the IoT DDoS example implementation has its MUD PEP router/switch automatically configured to enforce the route filtering that is described in the cached MUD file for that device's MUD URL, assuming that the amount of time that has elapsed since the cached MUD file was retrieved is less than or equal to the number of hours in the file's cache-validity value. If the cache validity has expired for the respective file, the MUD manager should fetch a new MUD file from the MUD file server.
Associated Test Case(s)	N/A

Test Case Field	Description
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-1, ID.AM-2, ID.AM-3, PR.DS-5, DE.AE-1, PR.AC-4, PR.AC-5, PR.IP-1, PR.IP-3, PR.DS-2, PR.PT-3
IoT Device(s) Under Test	Raspberry Pi
MUD File(s) Used	*mudfile-sensor.json*
Preconditions	1. All devices have been configured to use IPv4. 2. The MUD PEP router/switch does not yet have any configuration settings pertaining to the IoT device being used in the test. 3. The MUD file for the IoT device being used in the test is identical to the MUD file provided in Section 5.1.3.
Procedure	Verify that the MUD PEP router/switch for the IoT device to be used in the test does not yet have any configuration settings installed with respect to the IoT device being used in the test. 1. Run test IoT-1-v4. 2. Within 24 hours (i.e., within the cache-validity period for the MUD file) of running test IoT-1-v4, verify that the IoT device that was connected during test IoT-1-v4 is still up and running on the network. Power on a second IoT device that has been configured to emit the same MUD URL as the device that was connected during test IoT-1-v4, and connect it to the test network. 3. On the IoT device, emit a DHCPv4 message containing the device's MUD URL (IANA code 161). 4. The MUD manager snoops the DHCP request through the switch and extracts the MUD URL from the DHCP request. 5. The DHCP server receives the DHCPv4 message containing the IoT device's MUD URL. 6. The DHCP server offers an IP address lease to the newly connected IoT device. 7. The IoT device requests this IP address lease, which the DHCP server acknowledges.

Test Case Field	Description
	8. The MUD manager determines that it has this MUD file cached and checks that the amount of time that has elapsed since the cached file was retrieved is less than or equal to the number of hours in the cache-validity value for this MUD file. If the cache validity has been exceeded, the MUD manager will fetch a new MUD file. 9. The MUD manager translates the MUD file's contents into appropriate route filtering rules and installs these rules onto the MUD PEP for the IoT device in question so that this router/switch is now configured to enforce the policies specified in the MUD file.
Expected Results	The MUD PEP router/switch for the IoT device has had its configuration changed, i.e., it has been configured to enforce the policies specified in the IoT device's MUD file. The expected configuration should resemble the following details: **Cache is valid** (the MUD manager does NOT retrieve the MUD file from the MUD file server)**:** Observing the MUD file server logs, notice that only the first DHCP request for a device goes out to the MUD file server. Within the next 24 hours, any additional DHCP requests will not go to the MUD file server to fetch a new MUD file. **Cache is not valid** (the MUD manager does retrieve the MUD file from the MUD file server)**:** Observing the MUD file server logs, notice that the MUD manager fetches a new copy of the MUD file and signature when the cache does not contain the MUD file of interest.
Actual Results	<u>**IoT device initial DHCP event**</u>**:** `For the first DHCLient request:` `sensor ] date` `Tue Sep  3 15:01:16 EDT 2019` ` sensor ] alias dhc` `alias dhc='sudo rm /var/lib/dhcp/dhclient.leases; sudo ifconfig wlan0 0.0.0.0;  sudo dhclient -v wlan0 -cf /etc/dhcp/dhclient.conf.toaster'` ` sensor ] dhc` `Internet Systems Consortium DHCP Client 4.3.5` `Copyright 2004-2016 Internet Systems Consortium.`

Test Case Field	Description
	All rights reserved. For info, please visit https://www.isc.org/software/dhcp/ Listening on LPF/wlan0/00:13:ef:20:1d:6b Sending on LPF/wlan0/00:13:ef:20:1d:6b Sending on Socket/fallback DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 6 DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 7 DHCPREQUEST of 10.0.41.182 on wlan0 to 255.255.255.255 port 67 DHCPOFFER of 10.0.41.182 from 10.0.41.1 DHCPACK of 10.0.41.182 from 10.0.41.1 bound to 10.0.41.182 -- renewal in 17153 seconds. **MUD file server—log of initial fetch:** sudo -E python mudfile-server.py DoGET /nistmud1 127.0.0.1 - - **[03/Sep/2019 15:02:53] "GET /nistmud1 HTTP/1.1" 200 -** Read 9548 chars DoGET /nistmud1/mudfile-sensor.p7s 127.0.0.1 - - [03/Sep/2019 15:02:55] "GET /nistmud1/mudfile-sensor.p7s HTTP/1.1" 200 - Read 3494 chars **MUD manager log file showing MUD file caching:** 2019-09-03 15:02:56,702 | INFO | on-dispatcher-99 | Mud-FileFetcher | 93 - gov.nist.antd.sdnmud-impl - 0.1.0 | verification success 2019-09-03 15:02:56,709 | INFO | on-dispatcher-99 | Mud-FileFetcher | 93 - gov.nist.antd.sdnmud-impl - 0.1.0 | Write to Cache here **2019-09-03 15:02:56,738 | INFO | on-dispatcher-99 | Mud-CacheDataStoreListener | 93 - gov.nist.antd.sdnmud-impl - 0.1.0 | Writing MUD Cache {"mud-cache-entries":[{"cache-timeout":48,"cached-mudfile-name":"*sensor.nist.local*_nistmud1","retrieval-time":1567537376711,"mud-url":"*https://sensor.nist.local/nistmud1*"}]}** 2019-09-03 15:02:56,739 | INFO | on-dispatcher-99 | Datas-toreUpdater | 93 - gov.nist.antd.sdnmud-impl - 0.1.0 | jsonData = {"mud-cache-entries":[{"cache-timeout":48,"cached-mudfile-name":"*sensor.nist.local*_nistmud1","retrieval-time":1567537376711,"mud-url":"*https://sensor.nist.local/nistmud1*"}]} **IoT device—second DHCP request:**

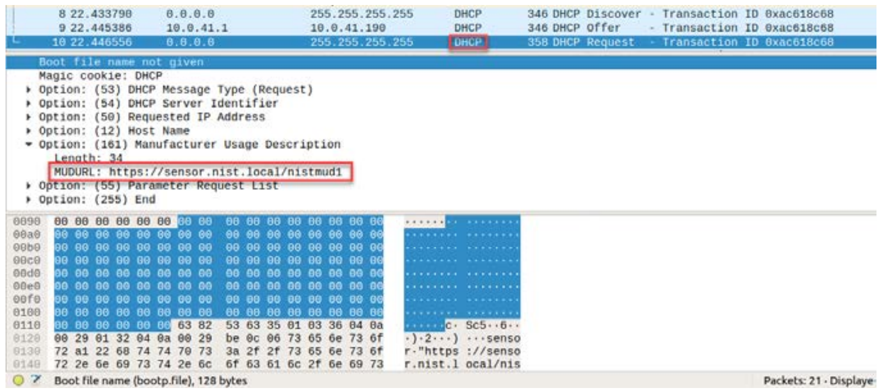
Test Case Field	Description
	` sensor ] date` **`Tue Sep  3 15:03:10 EDT 2019`** ` sensor ] dhc` `Internet Systems Consortium DHCP Client 4.3.5` `Copyright 2004-2016 Internet Systems Consortium.` `All rights reserved.` `For info, please visit https://www.isc.org/software/dhcp/` `Listening on LPF/wlan0/00:13:ef:20:1d:6b` `Sending on   LPF/wlan0/00:13:ef:20:1d:6b` `Sending on   Socket/fallback` `DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 8` `DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 19` `DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 12` `DHCPREQUEST of 10.0.41.182 on wlan0 to 255.255.255.255 port 67` `DHCPOFFER of 10.0.41.182 from 10.0.41.1` `DHCPACK of 10.0.41.182 from 10.0.41.1` `bound to 10.0.41.182 -- renewal in 17132 seconds.` **<u>MUD manager—log file showing cached file in use</u>:** **`2019-09-03 15:03:51`**`,666 | INFO  | on-dispatcher-99 | Mud-FileFetcher              | 93 - gov.nist.antd.sdnmud-impl - 0.1.0 | `**`Found file in mud cache length = 9548`** `2019-09-03 15:03:51,666 | INFO  | on-dispatcher-99 | Mud-FileFetcher              | 93 - gov.nist.antd.sdnmud-impl - 0.1.0 | read 9548 characters` **<u>MUD file server—log after second fetch (no change in output)</u>:** `sudo -E python mudfile-server.py` `DoGET /nistmud1` `127.0.0.1 - - [03/Sep/2019 15:02:53] "GET /nistmud1 HTTP/1.1" 200 -` `Read 9548 chars` `DoGET /nistmud1/mudfile-sensor.p7s` `127.0.0.1 - - [03/Sep/2019 15:02:55] "GET /nistmud1/mudfile-sensor.p7s HTTP/1.1" 200 -` `Read 3494 chars`
Overall Results	Pass

533 IPv6 is not supported in this implementation.

534 ### *5.1.2.9 Test Case IoT-11-v4*

535 **Table 5-10: Test Case IoT-11-v4**

Test Case Field	Description
Parent Requirements	(CR-1) The IoT DDoS example implementation shall include a mechanism for associating a device with a MUD file URL (e.g., by having the MUD-enabled IoT device emit a MUD file URL via DHCP, LLDP, or X.509 or by using some other mechanism to enable the network to associate a device with a MUD file URL).
Testable Requirements	(CR-1.a) Upon initialization, the MUD-enabled IoT device shall broadcast a DHCP message on the network, including at most one MUD URL, in https scheme, within the DHCP transaction. (CR-1.a.1) The DHCP server shall be able to receive DHCPv4 DISCOVER and REQUEST with IANA code 161 (OPTION_MUD_URL_V4) from the MUD-enabled IoT device.
Description	Shows that the IoT DDoS example implementation includes IoT devices that can emit a MUD URL via DHCP.
Associated Test Case(s)	N/A
Associated Cybersecurity Framework Subcategory(ies)	ID.AM-1
IoT Device(s) Under Test	Raspberry Pi 1
MUD File(s) Used	*nistmud1.json*
Preconditions	Device has been developed to emit MUD URL in DHCP transaction.
Procedure	1. Power on a device and connect it to the network. 2. Verify that the device emits a MUD URL in a DHCP transaction. (Use Wireshark to capture the DHCP transaction with options present.)
Expected Results	DHCP transaction with MUD option 161 enabled and MUD URL included

Test Case Field	Description
Actual Results	
Overall Results	Pass

5.1.3 MUD Files

This section contains the MUD files that were used in the Build 4 functional demonstration.

5.1.3.1 mudfile-sensor.json

The complete mudfile-sensor.json MUD file has been linked to this document. To access this MUD file please click the link below.

mudfile-sensor.json

5.1.3.2 mudfile-otherman.json

The complete mudfile-otherman.json MUD file has been linked to this document. To access this MUD file please click the link below.

mudfile-otherman.json