OPSWG WG T. Reddy

Internet-Draft McAfee

Intended status: Standards Track D. Wing

Expires: March 3, 2021 Citrix

B. Anderson

Cisco

August 30, 2020

MUD (D)TLS Profiles for IoT Devices

draft-reddy-opsawg-mud-tls-05

Abstract

This memo extends the Manufacturer Usage Description (MUD) specification to incorporate

(D)TLS profile parameters. This allows a network element to identify

unexpected (D)TLS usage, which can used as a hint about the presence of

unauthorized software or malware on an endpoint.

Status of This Memo

This Internet-Draft is submitted in full conformance with the

provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering

Task Force (IETF). Note that other groups may also distribute

working documents as Internet-Drafts. The list of current Internet-

Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months

and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference

material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 3, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the

document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

Provisions Relating to IETF Documents

(https://trustee.ietf.org/license-info) in effect on the date of

publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of

Reddy, et al. Expires March 3, 2021 [Page 1]

Internet-Draft MUD TLS profile for IoT devices August 2020

the Trust Legal Provisions and are provided without warranty as

described in the Simplified BSD License.

Table of Contents

1. Introduction . . . . . . . . . . . . . . . . . . . . . . . . 2

2. Terminology . . . . . . . . . . . . . . . . . . . . . . . . . 4

3. Overview of MUD (D)TLS profiles for IoT devices . . . . . . . 5

4. (D)TLS 1.3 handshake . . . . . . . . . . . . . . . . . . . . 5

4.1. Full (D)TLS 1.3 handshake inspection . . . . . . . . . . 6

4.2. Encrypted SNI . . . . . . . . . . . . . . . . . . . . . . 6

5. (D)TLS profile YANG module . . . . . . . . . . . . . . . . . 7

5.1. Tree Structure . . . . . . . . . . . . . . . . . . . . . 9

5.2. YANG Module . . . . . . . . . . . . . . . . . . . . . . . 10

6. MUD File Example . . . . . . . . . . . . . . . . . . . . . . 15

7. Security Considerations . . . . . . . . . . . . . . . . . . . 17

8. Privacy Considerations . . . . . . . . . . . . . . . . . . . 17

9. IANA Considerations . . . . . . . . . . . . . . . . . . . . . 17

10. Acknowledgments . . . . . . . . . . . . . . . . . . . . . . . 17

11. References . . . . . . . . . . . . . . . . . . . . . . . . . 17

11.1. Normative References . . . . . . . . . . . . . . . . . . 18

11.2. Informative References . . . . . . . . . . . . . . . . . 19

Authors' Addresses . . . . . . . . . . . . . . . . . . . . . . . 20

1. Introduction

Encryption is necessary to enhance the privacy of end users using IoT

devices. In a network setting, TLS [RFC8446] and DTLS

[I-D.ietf-tls-dtls13] are the dominant protocols providing encryption

for IoT device traffic. Unfortunately, in conjunction with IoT

applications' rise of encryption, malware is also using encryption

which thwarts network-based analysis such as deep packet inspection

(DPI). Other mechanisms are needed to detect malware is running on

an IoT device.

Malware frequently uses its own libraries for its activities, and

those libraries are re-used much like any other software engineering

project. [malware] indicates that there are observable

differences in how malware uses encryption compared with how non-

malware uses encryption. There are several interesting findings

specific to (D)TLS which were found common to malware:

o Older and weaker cryptographic parameters (e.g.,

TLS\_RSA\_WITH\_RC4\_128\_SHA).

o TLS SNI and server certificates are composed of subjects with

characteristics of a domain generation algorithm (DGA) (e.g.,

www.33mhwt2j.net).

Reddy, et al. Expires March 3, 2021 [Page 2]

Internet-Draft MUD TLS profile for IoT devices August 2020

o Higher use of self-signed certificates compared with typical

legitimate software.

o Discrepancies in the server name indication (SNI) TLS extension in

the ClientHello message and the DNS names in the

SubjectAltName (SAN) X.509 extension in the server certificate

message.

o Discrepancies in the key exchange algorithm and the client public

key length in comparison with legitimate flows. As a reminder,

Client Key Exchange message has been removed from TLS 1.3.

o Lower diversity in TLS client advertised TLS extensions compared

to legitimate clients.

o Malware using privacy enhancing technologies like Tor, Psiphon, and

Ultrasurf (see [malware-tls]) and, evasion techniques such as

ClientHello randomization to evade detection in order to continue

exploiting the end user.

o Malware using DNS-over-HTTPS (DoH) [RFC8484] to avoid detection by

malware DNS filtering service [malware-doh]. Malware agent may

not use the DoH server provided by the local network.

If observable (D)TLS profile parameters are used, the following

functions are possible which have a positive impact on the local network

security:

o Permit intended DTLS or TLS use and block malicious DTLS or TLS

use. This is superior to the layers 3 and 4 ACLs of

Manufacturer Usage Description Specification (MUD) [RFC8520] which

are not suitable for broad communication patterns.

o Ensure TLS certificates are valid. Several TLS deployments have

been vulnerable to active Man-In-The-Middle (MITM) attacks because

of the lack of certificate validation or vulnerability in the

certificate validation function (see [cryto-vulnerability]). By

observing (D)TLS profile parameters, a network element can detect

when the TLS SNI mismatches the SubjectAltName and when the

server's certificate is invalid. In TLS 1.2, the ClientHello,

ServerHello and Certificate messages are all sent in clear-text,

however in TLS 1.3, the Certificate message is encrypted thereby

hiding the server identity from any intermediary. In TLS 1.3, the

middlebox needs to act as a TLS proxy to validate the server

certificate and to detect TLS SNI mismatch with the server

certificate.

Reddy, et al. Expires March 3, 2021 [Page 3]

Internet-Draft MUD TLS profile for IoT devices August 2020

o Support new communication patterns. An IoT device can learn a new

capability, and the new capability can change the way the IoT

device communicates with other devices located in the local

network or in the Internet. There would be an inaccurate policy if an

IoT device rapidly changes the IP addresses and domain names it

communicates with while the MUD ACLs were slower to update. In

such a case, observable (D)TLS profile parameters can be used to

permit intended use and to block malicious behaviour from the IoT

device.

This document extends MUD [RFC8520] to model observable (D)TLS

profile parameters. Using these (D)TLS profile parameters, an active

MUD-enforcing firewall can identify MUD non-compliant (D)TLS behavior

indicating outdated cryptography or malware. This detection can

prevent malware downloads, block access to malicious domains, enforce

use of strong ciphers, stop data exfiltration, etc. In addition,

organizations may have policies around acceptable ciphers and

certificates on the websites the IoT devices connect to. Examples

include no use of old and less secure versions of TLS, no use of

self-signed certificates, deny-list or accept-list of Certificate

Authorities, valid certificate expiration time, etc. These policies

can be enforced by observing the (D)TLS profile parameters.

Enterprise firewalls can use the IoT device's (D)TLS profile

parameters to identify legitimate flows by observing (D)TLS sessions,

and can make inferences to permit legitimate flows and to block

malicious or insecure flows. The proposed technique is also suitable

in deployments where decryption techniques are not ideal due to

privacy concerns, non-cooperating end-points, and expense.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",

"SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and

"OPTIONAL" in this document are to be interpreted as described in BCP

14 [RFC2119][RFC8174] when, and only when, they appear in all

capitals, as shown here.

"(D)TLS" is used for statements that apply to both Transport Layer

Security [RFC8446] and Datagram Transport Layer Security [RFC6347].

Specific terms are used for any statement that applies to either

protocol alone.

'DoH/DoT' refers to DNS-over-HTTPS and/or DNS-over-TLS.

Reddy, et al. Expires March 3, 2021 [Page 4]

Internet-Draft MUD TLS profile for IoT devices August 2020

3. Overview of MUD (D)TLS Profiles for IoT Devices

In Enterprise networks, protection and detection are typically done

both on end hosts and in the network. Host agents have deep

visibility on the devices where they are installed, whereas the

network has a broader visibility. Installing host agents may not be a

viable option on IoT devices, and network-based security is an

efficient means to protect such IoT devices.

(D)TLS profile

parameters of IoT devices can be used by middleboxes to detect and

block malware communication, while at the same time preserving the

privacy of legitimate uses of encryption. Middleboxes need not

proxy (D)TLS but can passively observe the parameters of (D)TLS

handshakes from IoT devices and gain good visibility into TLS 1.0 to

1.2 parameters and partial visibility into TLS 1.3 parameters.

Malicious agents can try to use the (D)TLS profile parameters of

legitimate agents to evade detection, but it becomes a challenge to

mimic the behavior of various IoT device types and IoT device models

from several manufacturers. In other words, malware developers will

have to develop malicious agents per IoT device type, manufacturer

and model, infect the device with the tailored malware agent and will

have keep up with updates to the device's (D)TLS profile parameters

over time. Furthermore, the malware's command and control server

certificates need to be signed by the same certifying authorities

trusted by the IoT devices. Typically, IoT devices have an

infrastructure that supports a rapid deployment of updates, and

malware agents will have a near-impossible task of similarly

deploying updates and continuing to mimic the TLS behavior of the IoT

device it has infected.

Compromised IoT devices are typically used for launching DDoS

attacks (Section 3 of [RFC8576]). Some of the DDoS attacks like

Slowloris and Transport Layer Security (TLS) re-negotiation can be

detected by observing the (D)TLS profile parameters. For example,

the victim's server certificate need not be signed by the same

certifying authorities trusted by the IoT device.

4. (D)TLS 1.3 Handshake

In (D)TLS 1.3, full (D)TLS handshake inspection is not possible since

all (D)TLS handshake messages excluding the ClientHello message are

encrypted. (D)TLS 1.3 has introduced new extensions in the handshake

record layers called Encrypted Extensions. Using these extensions

handshake messages will be encrypted and network devices (such as a

firewall) are incapable to decipher the handshake, and thus cannot view

the server certificate. However, the ClientHello and ServerHello

still have some fields visible, such as the list of supported

versions, named groups, cipher suites, signature algorithms, and

extensions in ClientHello and, chosen cipher in the ServerHello. For

Reddy, et al. Expires March 3, 2021 [Page 5]

Internet-Draft MUD TLS profile for IoT devices August 2020

instance, if the malware uses evasion techniques like ClientHello

randomization, the observable list of cipher suites and extensions

offered by the malware agent in the ClientHello message will not

match the list of cipher suites and extensions offered by the

legitimate client in the ClientHello message, and the middle-box can

block malicious flows without acting as a (D)TLS 1.3 proxy.

4.1. Full (D)TLS 1.3 Handshake Inspection

To obtain more visibility into negotiated TLS 1.3 parameters, a

middle-box can act as a (D)TLS 1.3 proxy. A middle-box can act as a

(D)TLS proxy for the IoT devices owned and managed by the IT team in

the Enterprise network and the (D)TLS proxy must meet the security

and privacy requirements of the organization. In other words, the

scope of middle-box acting as a (D)TLS proxy is restricted to

Enterprise network owning and managing the IoT devices. The middle-

box MUST follow the behaviour detailed in Section 9.3 of [RFC8446]

to act as a compliant (D)TLS 1.3 proxy.

To function as a (D)TLS proxy the middle-box has to create a signed

certificate using itself as a certificate authority. That

certificate authority has to be trusted by the (D)TLS client. The

IoT device needs to be configured with the middle-box's CA

certificate as Explicit Trust Anchor database entry to validate the

server certificate. The mechanism to configure the IoT device with

the middle-box's CA certificate is out of the scope of the document. The middle-box uses

the "supported\_versions" TLS extension (defined in TLS 1.3 to

negotiate the supported TLS versions between client and server) to

determine the TLS version. During the (D)TLS handshake, If (D)TLS

version 1.3 is used, the middle-box ((D)TLS proxy) modifies the

certificate provided by the server and signs it with the private key

from the local CA certificate. The middle-box has visibility into

further exchanges between the IoT device and server which enables it

to inspect the (D)TLS 1.3 handshake, enforce the MUD (D)TLS profile

and can inspect subsequent network traffic. The IoT device uses the

Explicit Trust Anchor database to validate the server certificate.

4.2. Encrypted SNI

To increase privacy, encrypted SNI (ESNI,

[I-D.ietf-tls-sni-encryption]) prevents passive observation of the

TLS Server Name Indication extension. To effectively provide that

privacy protection, SNI encryption needs to be used in conjunction

with DNS encryption (e.g., DoH or DoT). A middle-box (e.g., firewall) passively

inspecting an encrypted SNI (D)TLS handshake cannot observe the

encrypted SNI nor observe the encrypted DNS traffic. If an IoT

device is pre-configured to use public DoH/DoT servers, that middle-

Reddy, et al. Expires March 3, 2021 [Page 6]

Internet-Draft MUD TLS profile for IoT devices August 2020

box needs to act as a DoH/DoT proxy and replace the ECH configuration

in the "echconfig" SvcParamKey (Section 6.3 of

[I-D.ietf-dnsop-svcb-https]) with the middle box's ECH configuration.

Instead of an unappealing DoH/DoT proxy, the IoT device can be

bootstrapped to discover and authenticate DoH/DoT servers provided by

a local network by making use of one of the mechanisms described in

Section 4 of [I-D.reddy-add-enterprise]. The local DoH/DoT server

replaces the ECH configuration in the "echconfig" SvcParamKey with

the middle box's ECH configuration.

A common usage pattern for certain type of IoT devices (e.g., light

bulb) is for it to "call home" to a service that resides on the

public Internet, where that service is referenced through a domain

name (A or AAAA record). As discussed in Manufacturer Usage

Description Specification [RFC8520], because these devices tend to

require access to very few sites, all other access should be

considered suspect. If an IoT device is pre-configured to use public

DoH/DoT server, the MUD policy enforcement point is moved to that

public server, which cannot enforce the MUD policy based on domain

names (Section 8 of [RFC8520]). If the DNS query is not accessible

for inspection, it becomes quite difficult for the infrastructure to

suspect anything. Thus the use of a public DoH/DoT server is

incompatible with MUD in general. A local DoH/DoT server is

necessary to allow MUD policy enforcement on the local network.

5. (D)TLS Profile YANG Module

This document specifies a YANG module for representing (D)TLS

profile. The (D)TLS profile YANG module provides a method for

firewalls to observe the (D)TLS profile parameters in the (D)TLS

handshake to permit intended use and to block malicious behavior.

This module uses the common YANG types defined in [RFC6991], the rules

defined in [RFC8519], and the cryptographic types defined in

[I-D.ietf-netconf-crypto-types].

The (D)TLS parameters in each (D)TLS profile include

the following:

o Profile name

o (D)TLS version in ClientHello.legacy\_version

o (D)TLS versions supported by the IoT device. As a reminder,

"supported\_versions" extension defined in (D)TLS 1.3 is used by

the client to indicate which versions of (D)TLS it supports and a

client is considered to be attempting to negotiate (D)TLS 1.3 if

the ClientHello contains a "supported\_versions" extension with

0x0304 contained in its body.

Reddy, et al. Expires March 3, 2021 [Page 7]

Internet-Draft MUD TLS profile for IoT devices August 2020

o If GREASE [RFC8701] (Generate Random Extensions And Sustain

Extensibility) values are offered by the client or not.

o List of supported symmetric encryption algorithms. TLS 1.3

defines five cipher suites (Appendix B.4 of [RFC8446]), but most

clients are continuing to offer TLS 1.2 compatible cipher suites

for backwards compatibility.

o List of supported compression methods for data compression. In

TLS 1.3, only the "null" compression method is allowed

(Section 4.1.2 of [RFC8446]).

o List of supported extension types

o List of trust anchor certificates used by the IoT device. If the

server certificate is signed by one of the trust anchors, the

middle-box continues with the connection as normal. Otherwise,

the middle-box will react as if the server certificate validation

has failed and takes appropriate action (e.g., block the (D)TLS

session). An IoT device can use a private trust anchor to

validate a server's certificate (e.g., the private trust anchor

can be preloaded at manufacturing time on the IoT device and the

IoT device fetches the firmware image from the Firmware server

whose certificate is signed by the private CA).

o List of SPKI pin set pre-configured on the client to validate

self-signed server certificates or raw public keys. A SPKI pin

set is a cryptographic digest to "pin" public key information in a

manner similar to HTTP Public Key Pinning (HPKP) [RFC7469]. If

SPKI pin set is present in the (D)TLS profile of a IoT device and

the server certificate does not pass the PKIX certification path

validation, the middle-box computes the SPKI Fingerprint for the

public key found in the server's certificate (or in the raw public

key, if the server provides that instead). If a computed

fingerprint exactly matches one of the SPKI pin sets in the (D)TLS

profile, the middle-box continues with the connection as normal.

Otherwise, the middle-box will act on the SPKI validation failure

and takes appropriate action.

o Cryptographic hash algorithm used to generate the SPKI pinsets

o List of pre-shared key exchange modes

o List of named groups (DHE or ECDHE) supported by the client

o List signature algorithms the client can validate in X.509 server

certificates

Reddy, et al. Expires March 3, 2021 [Page 8]

Internet-Draft MUD TLS profile for IoT devices August 2020

o List signature algorithms the client is willing to accept for

CertificateVerify message (Section 4.2.3 of [RFC8446]). For

example, a TLS client implementation can support different sets of

algorithms for certificates and in TLS to signal the capabilities

in "signature\_algorithms\_cert" and "signature\_algorithms"

extensions.

o List of supported application protocols (e.g., h3, h2, http/1.1

etc.)

o List of certificate compression algorithms (defined in

[I-D.ietf-tls-certificate-compression])

o List of the distinguished names [X501] of acceptable certificate

authorities, represented in DER-encoded format [X690] (defined in

Section 4.2.4 of [RFC8446])

o List of client key exchange algorithms and the client public key

lengths in versions prior to (D)TLS 1.3

The (D)TLS profile parameters include the GREASE values for extension

types, named groups, signature algorithms, (D)TLS versions, pre-

shared key exchange modes and cipher suites, but normalized to the

value 0x0a to preserve ordering information. Note that the GREASE

values are random but their positions are deterministic (Section 5 of

[RFC8701]) and peers will ignore these values and interoperate.

If the (D)TLS profile parameters are not observed in a (D)TLS session

from the IoT device, the default behaviour is to block the (D)TLS

session.

Note: The TLS and DTLS IANA registries are available from

<https://www.iana.org/assignments/tls-parameters/tls-parameters.txt>.

5.1. Tree Structure

This document augments the "ietf-mud" MUD YANG module defined in

[RFC8520] for signaling the IoT device (D)TLS profile. This document

defines the YANG module "reddy-opsawg-mud-tls-profile", which has the

following tree structure:

Reddy, et al. Expires March 3, 2021 [Page 9]

Internet-Draft MUD TLS profile for IoT devices August 2020

module: reddy-opsawg-mud-tls-profile

augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches:

+--rw client-profile

+--rw tls-profiles\* [profile-name]

+--rw profile-name string

+--rw protocol-version? uint16

+--rw supported\_versions\* uint16

+--rw grease\_extension? boolean

+--rw encryption-algorithms\* encryption-algorithm

+--rw compression-methods\* compression-method

+--rw extension-types\* extension-type

+--rw acceptlist-ta-certs\* ct:trust-anchor-cert-cms

+--rw SPKI-pin-sets\* SPKI-pin-set

+--rw SPKI-hash-algorithm? iha:hash-algorithm-type

+--rw psk-key-exchange-modes\* psk-key-exchange-mode

+--rw supported-groups\* supported-group

+--rw signature-algorithms-cert\* signature-algorithm

+--rw signature-algorithms\* signature-algorithm

+--rw application-protocols\* application-protocol

+--rw cert-compression-algorithms\* cert-compression-algorithm

+--rw certificate\_authorities\* certificate\_authorities

+--rw client-public-keys

+--rw key-exchange-algorithms\* key-exchange-algorithm

+--rw client-public-key-lengths\* client-public-key-length

5.2. YANG Module

module reddy-opsawg-mud-tls-profile {

yang-version 1.1;

namespace "urn:ietf:params:xml:ns:yang:reddy-opsawg-mud-tls-profile";

prefix mud-tls-profile;

import ietf-crypto-types {

prefix ct;

reference "draft-ietf-netconf-crypto-types-01:

Common YANG Data Types for Cryptography";

}

import iana-hash-algs {

prefix iha;

reference

"RFC XXXX: Common YANG Data Types for Hash algorithms";

}

import ietf-access-control-list {

prefix acl;

reference

Reddy, et al. Expires March 3, 2021 [Page 10]

Internet-Draft MUD TLS profile for IoT devices August 2020

"RFC 8519: YANG Data Model for Network Access

Control Lists (ACLs)";

}

organization

"IETF Operations and Management Area Working Group Working Group";

contact

"Editor: Konda, Tirumaleswar Reddy

<mailto:TirumaleswarReddy\_Konda@McAfee.com>";

description

"This module contains YANG definition for the IoT device

(D)TLS profile.

Copyright (c) 2019 IETF Trust and the persons identified as

authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or

without modification, is permitted pursuant to, and subject

to the license terms contained in, the Simplified BSD License

set forth in Section 4.c of the IETF Trust's Legal Provisions

Relating to IETF Documents

(http://trustee.ietf.org/license-info).

This version of this YANG module is part of RFC XXXX; see

the RFC itself for full legal notices.";

revision 2019-06-12 {

description

"Initial revision";

}

typedef compression-method {

type uint8;

description "Compression method";

}

typedef extension-type {

type uint16;

description "Extension type";

}

typedef encryption-algorithm {

type uint16;

description "Encryption algorithm";

}

typedef supported-group {

Reddy, et al. Expires March 3, 2021 [Page 11]

Internet-Draft MUD TLS profile for IoT devices August 2020

type uint16;

description "Named group (DHE or ECDHE)";

}

typedef SPKI-pin-set {

type binary;

description "Subject Public Key Info pin set";

}

typedef signature-algorithm {

type uint16;

description "Signature algorithm";

}

typedef key-exchange-algorithm {

type uint8;

description "key exchange algorithm";

}

typedef psk-key-exchange-mode {

type uint8;

description "pre-shared key exchange mode";

}

typedef client-public-key-length {

type uint8;

description "client public key length";

}

typedef application-protocol {

type string;

description "application protocol";

}

typedef cert-compression-algorithm {

type uint8;

description "certificate compression algorithm";

}

typedef certificate\_authority {

type binary;

description "Distinguished Name of Certificate authority";

}

grouping client-profile {

description

"A grouping for (D)TLS profiles.";

container client-profile {

Reddy, et al. Expires March 3, 2021 [Page 12]

Internet-Draft MUD TLS profile for IoT devices August 2020

list tls-profiles {

key "profile-name";

description

"A list of (D)TLS version profiles supported by the client.";

leaf profile-name {

type string {

length "1..64";

}

description

"The name of (D)TLS profile; space and special

characters are not allowed.";

}

leaf protocol-version {

type uint16;

description "(D)TLS version in ClientHello.legacy\_version";

}

leaf-list supported\_versions {

type uint16;

description

"TLS versions supported by the client indicated

in the supported\_versions extension in (D)TLS 1.3.";

}

leaf grease\_extension {

type boolean;

description

"If set to 'true', Grease extension values are offered by

the client.";

}

leaf-list encryption-algorithms {

type encryption-algorithm;

description "Encryption algorithms";

}

leaf-list compression-methods {

type compression-method;

description "Compression methods";

}

leaf-list extension-types {

type extension-type;

description "Extension Types";

}

leaf-list acceptlist-ta-certs {

type ct:trust-anchor-cert-cms;

description

"A list of trust anchor certificates used by the client.";

}

leaf-list SPKI-pin-sets {

type SPKI-pin-set;

description

Reddy, et al. Expires March 3, 2021 [Page 13]

Internet-Draft MUD TLS profile for IoT devices August 2020

"A list of SPKI pin sets pre-configured on the client

to validate self-signed server certificate or

raw public key.";

}

leaf SPKI-hash-algorithm {

type iha:hash-algorithm-type;

description

"cryptographic hash algorithm used to generate the

SPKI pinset.";

}

leaf-list psk-key-exchange-modes {

type psk-key-exchange-mode;

description

"pre-shared key exchange modes";

}

leaf-list supported-groups {

type supported-group;

description

"A list of named groups supported by the client.";

}

leaf-list signature-algorithms-cert {

type signature-algorithm;

description

"A list signature algorithms the client can validate

in X.509 certificates.";

}

leaf-list signature-algorithms {

type signature-algorithm;

description

"A list signature algorithms the client can validate

in the CertificateVerify message.";

}

leaf-list application-protocols {

type application-protocol;

description

"A list application protocols supported by the client";

}

leaf-list cert-compression-algorithms {

type cert-compression-algorithm;

description

"A list certificate compression algorithms

supported by the client";

}

leaf-list certificate\_authorities {

type certificate\_authority;

description

"A list of the distinguished names of certificate authorities

acceptable to the client";

Reddy, et al. Expires March 3, 2021 [Page 14]

Internet-Draft MUD TLS profile for IoT devices August 2020

}

container client-public-keys {

leaf-list key-exchange-algorithms {

type key-exchange-algorithm;

description

"Key exchange algorithms supported by the client";

}

leaf-list client-public-key-lengths {

type client-public-key-length;

description

"client public key lengths";

}

}

}

}

}

augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" {

description

"MUD (D)TLS specific matches.";

uses client-profile;

}

}

6. MUD File Example

The example below contains (D)TLS profile parameters for a IoT

device used to reach servers listening on port 443 using TCP

transport. JSON encoding of YANG modelled data [RFC7951] is used to

illustrate the example.

{

"ietf-mud:mud": {

"mud-version": 1,

"mud-url": "https://example.com/IoTDevice",

"last-update": "2019-18-06T03:56:40.105+10:00",

"cache-validity": 100,

"is-supported": true,

"systeminfo": "IoT device name",

"from-device-policy": {

"access-lists": {

"access-list": [

{

"name": "mud-7500-profile"

}

]

}

},

"ietf-access-control-list:acls": {

Reddy, et al. Expires March 3, 2021 [Page 15]

Internet-Draft MUD TLS profile for IoT devices August 2020

"acl": [

{

"name": "mud-7500-profile",

"type": "ipv6-acl-type",

"aces": {

"ace": [

{

"name": "cl0-frdev",

"matches": {

"ipv6": {

"protocol": 6

},

"tcp": {

"ietf-mud:direction-initiated": "from-device",

"destination-port": {

"operator": "eq",

"port": 443

}

},

"reddy-opsawg-mud-tls-profile:client-profile" : {

"tls-profiles" : [

{

"protocol-version" : 771,

"supported\_versions\_ext" : "FALSE",

"encryption-algorithms" :

[31354, 4865, 4866, 4867],

"extension-types" : [10],

"supported-groups" : [29]

}

]

},

"actions": {

"forwarding": "accept"

}

}

}

]

}

}

]

}

}

}

Reddy, et al. Expires March 3, 2021 [Page 16]

Internet-Draft MUD TLS profile for IoT devices August 2020

7. Security Considerations

Security considerations in [RFC8520] need to be taken into

consideration. Although it is challenging for a malware to mimic the

TLS behavior of various IoT device types and IoT device models from

several manufacturers, malicious agents have a very low probability of

using the same (D)TLS profile parameters as legitimate agents on the

IoT device to evade detection. Network security services should also

rely on contextual network data to detect false negatives. In order

to detect such malicious flows, anomaly detection (deep learning

techniques on network data) can be used to detect malicious agents

using the same (D)TLS profile parameters as legitimate agent on the

IoT device. In anomaly detection, the main idea is to maintain

rigorous learning of "normal" behavior and where an "anomaly" (or an

attack) is identified and categorized based on the knowledge about

the normal behavior and a deviation from this normal behavior.

8. Privacy Considerations

The middle-box acting as a (D)TLS proxy must immediately delete the

decrypted data upon completing any necessary inspection functions.

TLS proxy potentially has access to a user's PII (Personally

identifiable information) and PHI (Protected Health Information).

The TLS proxy must not store, process or modify PII data. For

example, IT administrator can configure firewall to bypass payload

inspection for a connection destined to a specific service due to

privacy compliance requirements.

9. IANA Considerations

This document requests IANA to register the following URIs in the

"ns" subregistry within the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:reddy-opsawg-mud-tls-profile

Registrant Contact: The IESG.

XML: N/A; the requested URI is an XML namespace.

10. Acknowledgments

Thanks to Flemming Andreasen, Shashank Jain, Michael Richardson,

Piyush Joshi and Harsha Joshi for the discussion and comments.

11. References

Reddy, et al. Expires March 3, 2021 [Page 17]

Internet-Draft MUD TLS profile for IoT devices August 2020

11.1. Normative References

[I-D.ietf-netconf-crypto-types]

Watsen, K., "YANG Data Types and Groupings for

Cryptography", draft-ietf-netconf-crypto-types-18 (work in

progress), August 2020.

[I-D.ietf-tls-certificate-compression]

Ghedini, A. and V. Vasiliev, "TLS Certificate

Compression", draft-ietf-tls-certificate-compression-10

(work in progress), January 2020.

[I-D.ietf-tls-dtls13]

Rescorla, E., Tschofenig, H., and N. Modadugu, "The

Datagram Transport Layer Security (DTLS) Protocol Version

1.3", draft-ietf-tls-dtls13-38 (work in progress), May

2020.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate

Requirement Levels", BCP 14, RFC 2119,

DOI 10.17487/RFC2119, March 1997,

<https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,

DOI 10.17487/RFC3688, January 2004,

<https://www.rfc-editor.org/info/rfc3688>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer

Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,

January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types",

RFC 6991, DOI 10.17487/RFC6991, July 2013,

<https://www.rfc-editor.org/info/rfc6991>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC

2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,

May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol

Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018,

<https://www.rfc-editor.org/info/rfc8446>.

[RFC8519] Jethanandani, M., Agarwal, S., Huang, L., and D. Blair,

"YANG Data Model for Network Access Control Lists (ACLs)",

RFC 8519, DOI 10.17487/RFC8519, March 2019,

<https://www.rfc-editor.org/info/rfc8519>.

Reddy, et al. Expires March 3, 2021 [Page 18]

Internet-Draft MUD TLS profile for IoT devices August 2020

[RFC8701] Benjamin, D., "Applying Generate Random Extensions And

Sustain Extensibility (GREASE) to TLS Extensibility",

RFC 8701, DOI 10.17487/RFC8701, January 2020,

<https://www.rfc-editor.org/info/rfc8701>.

[X690] ITU-T, "Information technology - ASN.1 encoding Rules:

Specification of Basic Encoding Rules (BER), Canonical

Encoding Rules (CER) and Distinguished Encoding Rules

(DER)", ISO/IEC 8825-1:2002, 2002.

11.2. Informative References

[cryto-vulnerability]

Perez, B., "Exploiting the Windows CryptoAPI

Vulnerability", January 2020,

<https://media.defense.gov/2020/Jan/14/2002234275/-1/-1/0/

CSA-WINDOWS-10-CRYPT-LIB-20190114.PDF>.

[I-D.ietf-dnsop-svcb-https]

Schwartz, B., Bishop, M., and E. Nygren, "Service binding

and parameter specification via the DNS (DNS SVCB and

HTTPS RRs)", draft-ietf-dnsop-svcb-https-01 (work in

progress), July 2020.

[I-D.ietf-tls-sni-encryption]

Huitema, C. and E. Rescorla, "Issues and Requirements for

SNI Encryption in TLS", draft-ietf-tls-sni-encryption-09

(work in progress), October 2019.

[I-D.reddy-add-enterprise]

Reddy.K, T. and D. Wing, "DNS-over-HTTPS and DNS-over-TLS

Server Deployment Considerations for Enterprise Networks",

draft-reddy-add-enterprise-00 (work in progress), June

2020.

[malware] Anderson, B., Paul, S., and D. McGrew, "Deciphering

Malware's use of TLS (without Decryption)", July 2016,

<https://arxiv.org/abs/1607.01639>.

[malware-doh]

Cimpanu, C., "First-ever malware strain spotted abusing

new DoH (DNS over HTTPS) protocol", July 2019,

<https://www.zdnet.com/article/first-ever-malware-strain-

spotted-abusing-new-doh-dns-over-https-protocol/>.

Reddy, et al. Expires March 3, 2021 [Page 19]

Internet-Draft MUD TLS profile for IoT devices August 2020

[malware-tls]

Anderson, B. and D. McGrew, "TLS Beyond the Browser:

Combining End Host and Network Data to Understand

Application Behavior", October 2019,

<https://dl.acm.org/citation.cfm?id=3355601>.

[RFC7469] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning

Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April

2015, <https://www.rfc-editor.org/info/rfc7469>.

[RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D.,

and P. Hoffman, "Specification for DNS over Transport

Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May

2016, <https://www.rfc-editor.org/info/rfc7858>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG",

RFC 7951, DOI 10.17487/RFC7951, August 2016,

<https://www.rfc-editor.org/info/rfc7951>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS

(DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018,

<https://www.rfc-editor.org/info/rfc8484>.

[RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage

Description Specification", RFC 8520,

DOI 10.17487/RFC8520, March 2019,

<https://www.rfc-editor.org/info/rfc8520>.

[RFC8576] Garcia-Morchon, O., Kumar, S., and M. Sethi, "Internet of

Things (IoT) Security: State of the Art and Challenges",

RFC 8576, DOI 10.17487/RFC8576, April 2019,

<https://www.rfc-editor.org/info/rfc8576>.

[X501] "Information Technology - Open Systems Interconnection -

The Directory: Models", ITU-T X.501, 1993.

Authors' Addresses

Tirumaleswar Reddy

McAfee, Inc.

Embassy Golf Link Business Park

Bangalore, Karnataka 560071

India

Email: kondtir@gmail.com

Reddy, et al. Expires March 3, 2021 [Page 20]

Internet-Draft MUD TLS profile for IoT devices August 2020

Dan Wing

Citrix Systems, Inc.

4988 Great America Pkwy

Santa Clara, CA 95054

USA

Email: danwing@gmail.com

Blake Anderson

Cisco Systems, Inc.

170 West Tasman Dr

San Jose, CA 95134

USA

Email: blake.anderson@cisco.com

Reddy, et al. Expires March 3, 2021 [Page 21]