       Manufacturer Usage Description (MUD) (D)TLS Profiles for IoT Devices
                     draft-ietf-opsawg-mud-tls-02

Abstract

   This memo extends the Manufacturer Usage Description (MUD)
   specification to incorporate (D)TLS profile parameters. This allows
   a network security service to identify unexpected (D)TLS usage, which
   can indicate 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 April 24, 2021.

Table of Contents

1. Introduction

   Encryption is necessary to enhance the privacy of end users using IoT
   devices. TLS [RFC8446] and DTLS [I-D.ietf-tls-dtls13] are the
   dominant protocols (counting all (D)TLS versions) providing
   encryption for IoT device traffic. Unfortunately, in conjunction
   with IoT applications' rise of encryption, malware authors are also
   using encryption which thwarts network-based analysis such as deep
   packet inspection (DPI). Other mechanisms are thus needed to help
   detecting malware running on an IoT device.

   Malware frequently uses proprietary libraries for its activities, and
   those libraries are reused 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 server name indication (SNI) extension [RFC6066] and server
   certificates are composed of subjects with characteristics of a
   domain generation algorithm (DGA) (e.g., 'www.33mhwt2j.net').

o Higher use of self-signed certificates compared with typical
   legitimate software.

o Discrepancies in the SNI TLS extension 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,
   the Client Key Exchange message has been removed from TLS 1.3.

o Lower diversity in TLS client advertised extensions compared to
   legitimate clients.

o Using privacy enhancing technologies like Tor, Psiphon, Ultrasurf
   (see [malware-tls]), and evasion techniques such as ClientHello
   randomization.

o Using DNS-over-HTTPS (DoH) [RFC8484] to avoid detection by malware
   DNS filtering services [malware-doh]. Specifically, malware 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.
This check is not possible with TLS 1.3, which encrypts the
Certificate message thereby hiding the server identity from any
intermediary. In TLS 1.3, the server certificate validation
functions should be executed within an on-path TLS proxy, if such
a proxy exists.

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 and 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 (see
   [clear-as-mud]). In such a case, observable (D)TLS profile
   parameters can be used to permit intended use and to block
   malicious behavior from the IoT device.

The YANG module specified in Section 5 of this document is an
extension of YANG Data Model for Network Access Control Lists (ACLs)
[RFC8519] to enhance MUD [RFC8520] to model observable (D)TLS profile
parameters. Using these (D)TLS profile parameters, an active MUD-
enforcing network security service (e.g., 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 for 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. Network security services 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.

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 security agents have deep
visibility on the devices where they are installed, whereas the
network has broader visibility. Installing host security agents may
not be a viable option on IoT devices, and network-based security is
an efficient means to protect such IoT devices. If the IoT device
supports a MUD (D)TLS profile, the (D)TLS profile parameters of the
IoT device can be used by a middlebox to detect and block malware
communication, while at the same time preserving the privacy of
legitimate uses of encryption. The middlebox need not proxy (D)TLS
but can passively observe the parameters of (D)TLS handshakes from
IoT devices and gain visibility into TLS 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. However, if the IoT device has reached end-
of-life and the IoT manufcaturer will not issue a firmware or
software update to the Thing or will not update the MUD file, the
"is-supported" attribute defined in Section 3.6 of [RFC8520] can be
used by the MUD manager to identify the IoT manufcaturer no longer
supports the device.

The end-of-life of a device does not necessarily mean that it is
defective; rather, it denotes a need to replace and upgrade the
network to next-generation devices for additional functionality. The
network security service will have to rely on other techniques

discussed in Section 8 to identify malicious connections until the device is replaced.

Compromised IoT devices are typically used for launching DDoS attacks (Section 3 of [RFC8576]). For example, DDoS attacks like Slowloris and Transport Layer Security (TLS) re-negotiation can be blocked if the victim's server certificate is 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 security services (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 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 middlebox 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 middlebox can act as a (D)TLS 1.3 proxy. A middlebox 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 middlebox acting as a (D)TLS proxy is restricted to Enterprise network owning and managing the IoT devices. The middlebox would have to follow the behaviour detailed in Section 9.3 of [RFC8446] to act as a compliant (D)TLS 1.3 proxy.

To further increase privacy, Encrypted Client Hello (ECH) extension [I-D.ietf-tls-esni] prevents passive observation of the TLS Server Name Indication extension and other potentially sensitive fields, such as the ALPN [RFC7301]. To effectively provide that privacy protection, ECH extension needs to be used in conjunction with DNS encryption (e.g., DoH). A middlebox (e.g., firewall) passively inspecting ECH extension cannot observe the encrypted SNI nor observe the encrypted DNS traffic.

4.2. Encrypted DNS

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
[I-D.reddy-add-enterprise].

5. (D)TLS Profile of a IoT device

This document specifies a YANG module for representing (D)TLS
profile. The (D)TLS profile YANG module provides a method for
network security services 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 cryptographic types defined in
[I-D.ietf-netconf-crypto-types]. See [RFC7925] for (D)TLS 1.2 and
[I-D.ietf-uta-tls13-iot-profile] for DTLS 1.3 recommendations related
to IoT devices, and [RFC7525] for additional (D)TLS 1.2
recommendations.

A companion YANG module is defined to include a collection of (D)TLS
parameters and (D)TLS versions maintained by IANA: "iana-tls-profile"
(Section 5.3).

The (D)TLS parameters in each (D)TLS profile include the following:

o   Profile name

o   (D)TLS versions supported by the IoT device.

o   List of supported cipher suites. For (D)TLS1.2, [RFC7925]
    recommends AEAD ciphers for IoT devices.

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
    middlebox continues with the connection as normal. Otherwise, the

middlebox 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). This empowers the
middlebox to reject TLS sessions to servers that the IoT device
does not trust.

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 middlebox 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 middlebox continues with the connection as normal.
   Otherwise, the middlebox 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

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])

   GREASE [RFC8701] sends random values on TLS parameters to ensure
   future extensibility of TLS extensions. Similar random values might
   be extended to other TLS parameters. Thus, the (D)TLS profile
   parameters defined in the YANG module by this document MUST NOT
   include the GREASE values for extension types, named groups,
   signature algorithms, (D)TLS versions, pre-shared key exchange modes,
   cipher suites and for any other TLS parameters defined in future
   RFCs.

   The (D)TLS profile does not include parameters like compression
   methods for data compression, [RFC7525] recommends disabling TLS-
   level compression to prevent compression-related attacks. In TLS
   1.3, only the "null" compression method is allowed (Section 4.1.2 of
   [RFC8446]).

   Note: The TLS and DTLS IANA registries are available from
   <https://www.iana.org/assignments/tls-parameters/tls-parameters.txt>
   and <https://www.iana.org/assignments/tls-extensiontype-values/tls-
   extensiontype-values.txt>. The values for all the parameters in the
   YANG module excluding the are defined in the TLS and DTLS IANA
   registries excluding the supported_tls_versions and
   supported_dtls_versions parameters. The TLS and DTLS IANA registry
   does not maintain (D)TLS version numbers.

5.1. Tree Structure of the (D)TLS profile Extension to the ACL YANG Model

   This document augments the "ietf-acl" ACL YANG module defined in
   [RFC8519] for signaling the IoT device (D)TLS profile. This document
   defines the YANG module "ietf-acl-tls", which has the following tree
   structure:

```
module: ietf-acl-tls
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches:
    +--rw client-profile {match-on-tls-dtls}?
       +--rw client-profile
          +--rw tls-dtls-profiles* [profile-name]
             +--rw profile-name                   string
             +--rw supported-tls-versions*        ianatp:tls-version
             +--rw supported-dtls-versions*       ianatp:dtls-version
             +--rw cipher-suites* [cipher aead]
             |  +--rw cipher    ianatp:cipher-algorithm
             |  +--rw aead      ianatp:aead-algorithm
             +--rw extension-types*
             |       ianatp:extension-type
             +--rw acceptlist-ta-certs*
             |       ct:trust-anchor-cert-cms
             +--rw SPKI
             |  +--rw SPKI-pin-sets*        ianatp:SPKI-pin-set
             |  +--rw SPKI-hash-algorithm?   iha:hash-algorithm-type
             +--rw psk-key-exchange-modes*
             |       ianatp:psk-key-exchange-mode
             |       {tls-1-3 or dtls-1-3}?
             +--rw supported-groups*
             |       ianatp:supported-group
             +--rw signature-algorithms-cert*
             |       ianatp:signature-algorithm
             |       {tls-1-3 or dtls-1-3}?
             +--rw signature-algorithms*
             |       ianatp:signature-algorithm
             +--rw application-protocols*
             |       ianatp:application-protocol
             +--rw cert-compression-algorithms*
             |       ianatp:cert-compression-algorithm
             |       {tls-1-3 or dtls-1-3}?
             +--rw certificate_authorities*
                     ianatp:certificate_authority
                     {tls-1-3 or dtls-1-3}?
```

5.2. The (D)TLS profile Extension to the ACL YANG Model

```
<CODE BEGINS> file "ietf-acl-tls@2020-10-07.yang"
module ietf-acl-tls {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-acl-tls";
  prefix ietf-acl-tls;

  import iana-tls-profile {
    prefix ianatp;
```

```
     reference
       "RFC XXXX: Manufacturer Usage Description (MUD) (D)TLS
                 Profiles for IoT Devices";
  }
  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC CCCC: Common YANG Data Types for Cryptography";
  }
  import iana-hash-algs {
    prefix iha;
    reference
      "RFC IIII: Common YANG Data Types for
       Hash algorithms";
  }
  import ietf-access-control-list {
    prefix acl;
    reference
      "RFC 8519: YANG Data Model for Network Access
                 Control Lists (ACLs)";
  }

  organization
    "IETF OPSAWG (Operations and Management Area Working Group)";
  contact
    "WG Web: <https://datatracker.ietf.org/wg/opsawg/>
     WG List: opsawg@ietf.org

      Author: Konda, Tirumaleswar Reddy
                   TirumaleswarReddy_Konda@McAfee.com
    ";
  description
    "This YANG moudle defines a component that augments the
      IETF description of an access list to allow (D)TLS profile
      as matching criteria.

      Copyright (c) 2020 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 2020-10-07 {
    description
      "Initial revision";
    reference
      "RFC XXXX: Manufacturer Usage Description (MUD) (D)TLS
                Profiles for IoT Devices";
  }

  feature tls-1-2 {
    description
      "TLS Protocol Version 1.2 is supported.";
    reference
      "RFC 5246: The Transport Layer Security (TLS) Protocol
                Version 1.2";
  }

  feature tls-1-3 {
    description
      "TLS Protocol Version 1.3 is supported.";
    reference
      "RFC 8446: The Transport Layer Security (TLS) Protocol
                Version 1.3";
  }

  feature dtls-1-2 {
    description
      "DTLS Protocol Version 1.2 is supported.";
    reference
      "RFC 6346: Datagram Transport Layer Security Version 1.2";
  }

  feature dtls-1-3 {
    description
      "DTLS Protocol Version 1.3 is supported.";
    reference
      "draft-ietf-tls-dtls13: Datagram Transport Layer Security 1.3";
  }

  feature match-on-tls-dtls {
    description
      "The networking device can support matching on (D)TLS parameters.";
  }

  augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" {
    if-feature "match-on-tls-dtls";
    description
      "(D)TLS specific matches.";
    container client-profile {
```

```
    description
      "A grouping for (D)TLS profiles.";
    container client-profile {
      description
        "A grouping for DTLS profiles.";
      list tls-dtls-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-list supported-tls-versions {
          type ianatp:tls-version;
          description
            "TLS versions supported by the client";
        }
        leaf-list supported-dtls-versions {
          type ianatp:dtls-version;
          description
            "DTLS versions supported by the client";
        }
        list cipher-suites {
          key "cipher aead";
          leaf cipher {
            type ianatp:cipher-algorithm;
            description
              "Cipher";
          }
          leaf aead {
            type ianatp:aead-algorithm;
            description
              "AEAD";
          }
          description
            "Cipher Suites";
        }
        leaf-list extension-types {
          type ianatp: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.";
              }
              container SPKI {
                description
                  "A grouping for SPKI.";
                leaf-list SPKI-pin-sets {
                  type ianatp:SPKI-pin-set;
                  description
                    "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 {
                if-feature "tls-1-3 or dtls-1-3";
                type ianatp:psk-key-exchange-mode;
                description
                  "pre-shared key exchange modes";
              }
              leaf-list supported-groups {
                type ianatp:supported-group;
                description
                  "A list of named groups supported by the client.";
              }
              leaf-list signature-algorithms-cert {
                if-feature "tls-1-3 or dtls-1-3";
                type ianatp:signature-algorithm;
                description
                  "A list signature algorithms the client can validate
                   in X.509 certificates.";
              }
              leaf-list signature-algorithms {
                type ianatp:signature-algorithm;
                description
                  "A list signature algorithms the client can validate
                   in the CertificateVerify message.";
              }
              leaf-list application-protocols {
                type ianatp:application-protocol;
                description
```

```
              "A list application protocols supported by the client";
          }
          leaf-list cert-compression-algorithms {
            if-feature "tls-1-3 or dtls-1-3";
            type ianatp:cert-compression-algorithm;
            description
              "A list certificate compression algorithms
               supported by the client";
          }
          leaf-list certificate_authorities {
            if-feature "tls-1-3 or dtls-1-3";
            type ianatp:certificate_authority;
            description
              "A list of the distinguished names of certificate authorities
               acceptable to the client";
          }
        }
      }
    }
  }
}
<CODE ENDS>
```

5.3. IANA (D)TLS profile YANG Module

```
   <CODE BEGINS> file "iana-tls-profile@2020-10-07.yang"

   module iana-tls-profile {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:iana-tls-profile";
     prefix ianatp;

     organization
       "IANA";
     contact
       "          Internet Assigned Numbers Authority

          Postal: ICANN
                  12025 Waterfront Drive, Suite 300
                  Los Angeles, CA  90094-2536
                  United States

          Tel:    +1 310 301 5800
          E-Mail: iana@iana.org>";
     description
       "This module contains YANG definition for the (D)TLS profile.

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

```
  revision 2020-10-07 {
    description
      "Initial revision";
    reference
      "RFC XXXX: Manufacturer Usage Description (MUD) (D)TLS Profiles
                 for IoT Devices";
  }

  typedef extension-type {
    type uint16;
    description
      "Extension type";
  }

  typedef supported-group {
    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 psk-key-exchange-mode {
    type uint8;
    description
      "pre-shared key exchange mode";
```

```
      }

      typedef application-protocol {
        type string;
        description
          "application protocol";
      }

      typedef cert-compression-algorithm {
        type uint16;
        description
          "certificate compression algorithm";
      }

      typedef certificate_authority {
        type string;
        description
          "Distinguished Name of Certificate authority";
      }

      typedef cipher-algorithm {
        type uint8;
        description
          "Cipher Algorithm";
      }

      typedef aead-algorithm {
        type uint8;
        description
          "AEAD Algorithm";
      }

      typedef tls-version {
        type enumeration {
          enum tls-1.2 {
            value 1;
            description
              "TLS Protocol Version 1.2.";
            reference
              "RFC 5246: The Transport Layer Security (TLS) Protocol
                         Version 1.2";
          }
          enum tls-1.3 {
            value 2;
            description
              "TLS Protocol Version 1.3.";
            reference
              "RFC 8446: The Transport Layer Security (TLS) Protocol
```

```
                         Version 1.3";
          }
        }
        description
          "Indicates the TLS version.";
      }

      typedef dtls-version {
        type enumeration {
          enum dtls-1.2 {
            value 1;
            description
              "DTLS Protocol Version 1.2.";
            reference
              "RFC 6346: Datagram Transport Layer Security 1.2";
          }
          enum dtls-1.3 {
            value 2;
            description
              "DTLS Protocol Version 1.3.";
            reference
              "RFC DDDD: Datagram Transport Layer Security 1.3";
          }
        }
        description
          "Indicates the DTLS version.";
      }
    }
    <CODE ENDS>
```

5.4. MUD (D)TLS Profile Extension

This document augments the "ietf-mud" MUD YANG module to indicate
whether the device supports (D)TLS profile. If the "ietf-mud-tls"
extension is supported by the device, MUD file is assumed to
implement the "match-on-tls-dtls" ACL model feature defined in this
specification. Furthermore, only "accept" or "drop" actions SHOULD
be included with the (D)TLS profile similar to the actions allowed in
Section 2 of [RFC8520].

This document defines the YANG module "ietf-mud-tls", which has the
following tree structure:

```
module: ietf-mud-tls
  augment /ietf-mud:mud:
    +--rw is-tls-dtls-profile-supported?   boolean
```

The model is defined as follows:

```
<CODE BEGINS> file "iana-tls-mud@2020-10-20.yang"

module ietf-mud-tls {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-mud-tls";
  prefix ietf-mud-tls;

  import ietf-mud {
    prefix ietf-mud;
  }

  organization
    "IETF OPSAWG (Operations and Management Area Working Group)";
  contact
    "WG Web: <https://datatracker.ietf.org/wg/opsawg/>
     WG List: opsawg@ietf.org

     Author: Konda, Tirumaleswar Reddy
             TirumaleswarReddy_Konda@McAfee.com

    ";
   description
     "Extension to a MUD module to indicate (D)TLS
      profile support.";

   revision 2020-10-19 {
     description
       "Initial revision.";
       reference
         "RFC XXXX: Manufacturer Usage Description (MUD) (D)TLS
          Profiles for IoT Devices";
   }

   augment "/ietf-mud:mud" {
     description
       "This adds a extension for a manufacturer
        to indicate whether (D)TLS profile is
        is supported by a device.";
     leaf is-tls-dtls-profile-supported {
       type boolean;
         description
           "This value will equal 'true' if a device supports
            (D)TLS profile.";
     }
   }
}
<CODE ENDS>
```

6. **Processing of the MUD (D)TLS Profile**

   The following text outlines the rules for a network security service
   (e.g., firewall) to follow to process the MUD (D)TLS Profile:

   o If the (D)TLS parameter observed in a (D)TLS session is not
      specified in the MUD (D)TLS profile and the parameter is
      recognized by the firewall, it can identify unexpected (D)TLS
      usage, which can indicate the presence of unauthorized software or
      malware on an endpoint. The firewall can take several actions
      like block the (D)TLS session or raise an alert to quarantine and
      remediate the compromised device. For example, if the cipher
      suite TLS_RSA_WITH_AES_128_CBC_SHA in the ClientHello message is
      not specified in the MUD (D)TLS profile and the cipher suite is
      recognized by the firewall, it can identify unexpected TLS usage.

   o If the (D)TLS parameter observed in a (D)TLS session is not
      specified in the MUD (D)TLS profile and the (D)TLS parameter is
      not recognized by the firewall, it can ignore the unrecognized
      parameter and the correct behavior is not to block the (D)TLS
      session. The behaviour is functionally equivalent to the
      description in Section 9.3 of [RFC8446] to ignore all unrecognized
      cipher suites, extensions, and other parameters. For example, if
      the cipher suite TLS_CHACHA20_POLY1305_SHA256 in the ClientHello
      message is not specified in the MUD (D)TLS profile and the cipher
      suite is not recognized by the firewall, it can ignore the
      unrecognized cipher suite.

   o Deployments update at different rates, so an updated MUD (D)TLS
      profile may support newer parameters. If the firewall does not
      recognize the newer parameters, an alert should be triggered to
      the firewall vendor and the IoT device owner or administrator. A
      firewall must be readily updatable, so that when ossification
      problems are discovered, they can be addressed quickly. Most
      importantly, if the firewall is not readily updatable, its
      efficacy to identify emerging malware will decrease with time.
      For example, if the cipher suite TLS_AES_128_CCM_8_SHA256 in the
      ClientHello message is specified in the MUD (D)TLS profile and the
      cipher suite is not recognized by the firewall, an alert will be
      triggered.

7. **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,
         "extensions": [
              "ietf-mud-tls"
          ],
         "ietf-mud-tls:is-tls-dtls-profile-supported": "true",
         "is-supported": true,
         "systeminfo": "IoT device name",
         "from-device-policy": {
            "access-lists": {
              "access-list": [
                {
                  "name": "mud-7500-profile"
                }
              ]
            }
          },
         "ietf-access-control-list:acls": {
           "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
                         }
                       },
                       "ietf-acl-tls:client-profile" : {
                         "tls-dtls-profiles" : [
                            {
                                "supported-tls-versions" : ["tls-1.3"],
                                "cipher-suites" : [
                                  {
                                      "cipher": 19,
```

```
                                    "aead":  1
                                  },
                                  {
                                   "cipher": 19,
                                   "aead":  2
                                  }
                                 ],
                                "extension-types" : [10,11,13,16,24],
                                "supported-groups" : [29]
                              }
                           ]
                         },
                         "actions": {
                            "forwarding": "accept"
                         }
                      }
                   }
                 ]
               }
             }
           ]
        }
      }
    }
```

   The following illustrates the example scenarios for processing the
   above profile:

   o If the extension type "encrypt_then_mac" (code point 22) [RFC7366]
      in the ClientHello message is recognized by the firewall, it can
      identify unexpected TLS usage.

   o If the extension type "token_binding" (code point 24) [RFC8472] in
      the ClientHello message is not recognized by the firewall, it can
      ignore the unrecognized extension. Because the extension type
      token_binding is specified in the profile, an alert will be
      triggered to the firewall vendor and the IoT device owner or
      administrator to notify the firewall is not up to date.

8. 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.

9. Privacy Considerations

Privacy considerations discussed in Section 16 of [RFC8520] to not
reveal the MUD URL to an atacker need to be taken into consideration.
The MUD URL can be stored in Trusted Execution Environment (TEE) for
secure operation, enhanced data security, and prevent exposure to
unauthorized software.

The middlebox 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 the middlebox to bypass
payload inspection for a connection destined to a specific service
due to privacy compliance requirements. In addition, mechanisms
based on object security can be used by IoT devices to enable end-to-
end security and the middlebox will not have any access to the packet
data. For example, Object Security for Constrained RESTful
Environments (OSCORE) [RFC8613] is a proposal that protects CoAP
messages by wrapping them in the COSE format [RFC8152].

10. IANA Considerations

10.1. (D)TLS Profile YANG Modules

Each normative YANG module MUST be registered in both the "IETF XML
Registry" [RFC3688] and the "YANG Module Names" registry [RFC6020].

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:iana-tls-profile
      Registrant Contact: The IESG.
      XML: N/A; the requested URI is an XML namespace.

      URI: urn:ietf:params:xml:ns:yang:ietf-acl-tls
      Registrant Contact: The IESG.
      XML: N/A; the requested URI is an XML namespace.

```
     URI: urn:ietf:params:xml:ns:yang:ietf-mud-tls
     Registrant Contact: The IESG.
     XML: N/A; the requested URI is an XML namespace.
```

IANA is requested to create an IANA-maintained YANG Module called
"iana-tls-profile", based on the contents of Section 5.3, which will
allow for new (D)TLS parameters and (D)TLS versions to be added to
"client-profile". The registration procedure will be Expert Review,
as defined by [RFC8126].

This document requests IANA to register the following YANG modules in
the "YANG Module Names" subregistry [RFC6020] within the "YANG
Parameters" registry.

```
     name: iana-tls-profile
     namespace: urn:ietf:params:xml:ns:yang:iana-tls-profile
     maintained by IANA: Y
     prefix: ianatp
     reference: RFC XXXX

     name: ietf-acl-tls
     namespace: urn:ietf:params:xml:ns:yang:ietf-acl-tls
     maintained by IANA: N
     prefix: ietf-acl-tls
     reference: RFC XXXX

     name: ietf-mud-tls
     namespace: urn:ietf:params:xml:ns:yang:ietf-mud-tls
     maintained by IANA: N
     prefix: ietf-mud-tls
     reference: RFC XXXX
```

IANA is requested to create an the initial version of the IANA-
maintained YANG Module called "iana-tls-profile", based on the
contents of Section 5.3, which will allow for new (D)TLS parameters
and (D)TLS versions to be added. IANA is requested to add this note:

o tls-version and dtls-version values must not be directly added to
   the iana-tls-profile YANG module. They must instead be
   respectively added to the "TLS Version Codes", and "DTLS Version
   Codes" registries.

o (D)TLS parameters must not be directly added to the iana-tls-
   profile YANG module. They must instead be added to the "(D)TLS
   Parameters" registry.

When a 'tls-version' or 'dtls-version' value is respectively added to
the "TLS Version Codes" or "DTLS Version Codes" registry, a new

"enum" statement must be added to the iana-tls-profile YANG module.
The following "enum" statement, and substatements thereof, should be
defined:

"enum":   Replicates the label from the registry.

"value":   Contains the IANA-assigned value corresponding to the
               'tls-version' or 'dtls-version'.

"description": Replicates the description from the registry.

"reference":   Replicates the reference from the registry and adds
               the title of the document.

When a (D)TLS parameter is added to "(D)TLS Parameters" registry, a
new "type" statement must be added to the iana-tls-profile YANG
module. The following "type" statement, and substatements thereof,
should be defined:

"derived type": Replicates the parameter name from the registry.

"built-in type": Contains the built-in YANG type.

"description": Replicates the description from the registry.

When the iana-tls-profile YANG module is updated, a new "revision"
statement must be added in front of the existing revision statements.

IANA is requested to add this note to "TLS Version Codes", "DTLS
Version Codes", and "(D)TLS Parameters" registries:

   When this registry is modified, the YANG module iana-tls-profile
   must be updated as defined in [RFCXXXX].

The registration procedure for "ietf-acl-tls" YANG module will be
Specification Required, as defined by [RFC8126].

10.2. TLS Version registry

IANA is requested to create a new subregistry titled "TLS Version
Codes". Codes in this registry are used as valid values of 'tls-
version' parameter. Further assignments are to be made through
Expert Review [RFC8126].

```
+-----------+------------+-------------------------+-----------+
| Value     | Label      | Description             | Reference |
|           |            |                         |           |
|           |            |                         |           |
+-----------+------------+-------------------------+-----------+
| 1         | tls-1.2    | TLS Protocol Version 1.2 | [RFC5246] |
+-----------+------------+-------------------------+-----------+
| 2         | tls-1.3    | TLS Protocol Version 1.3 | [RFC8446] |
+---------------------------+-------------------------+-----------+
```

10.3. DTLS version registry

IANA is requested to create a new subregistry titled "DTLS Version
Codes". Codes in this registry are used as valid values of 'dtls-
version' parameter. Further assignments are to be made through
Expert Review [RFC8126].

```
+-----------+------------+--------------------------
+------------------------+
| Value     | Label      | Description              |
Reference               |
|           |            |                          |
|                        |
|           |            |                          |
|                        |
+-----------+------------+--------------------------
+------------------------+
| 1         | dtls-1.2   | DTLS Protocol Version 1.2 | [RFC6346]
|
+-----------------------------------------------------
+------------------------+
| 2         | dtls-1.3   | DTLS Protocol Version 1.3 | [draft-ietf-tls-
dtls13] |
+---------------------------+-------------------------
+------------------------+
```

10.4. (D)TLS Parameters registry

IANA is requested to create a new subregistry titled "(D)TLS
parameters". The values for all the (D)TLS parameters in the
registry are defined in the TLS and DTLS IANA registries
(<https://www.iana.org/assignments/tls-parameters/tls-parameters.txt>
and <https://www.iana.org/assignments/tls-extensiontype-values/tls-
extensiontype-values.txt>) excluding the supported_tls_versions and
supported_dtls_versions parameters. Further assignments are to be
made through Expert Review [RFC8126]. The registry is initially
populated with the following values:

Parameter Name	YANG Type	JSON Type	Description
extension-type	uint16	Number	Extension type
supported-group	uint16	Number	Named group (DHE or ECDHE)
SPKI-pin-set	binary	String	Subject Public Key Info pin set
signature-algorithm	uint16	Number	Signature algorithm
psk-key-exchange-mode	uint8	Number	pre-shared key exchange mode
application-protocol	string	String	application protocol
cert-compression-algorithm	uint16	Number	certificate compression algorithm
certificate_authority	string	String	Distinguished Name of Certificate authority
cipher-algorithm	uint8	Number	Cipher Algorithm
aead-algorithm	uint8	Number	AEAD Algorithm

```
   +---------------------------+-------------+--------
+---------------------------------------------+
   | tls-version               | enumeration | String | TLS
version                                       |
   +---------------------------+-------------+--------
+---------------------------------------------+
   | dtls-version              | enumeration | String | DTLS
version                                       |
   +---------------------------+-------------+--------
+---------------------------------------------+
```

10.5. MUD Extensions registry

IANA is requested to create a new MUD Extension Name "ietf-mud-tls"
in the MUD Extensions IANA registry
<https://www.iana.org/assignments/mud/mud.xhtml>.

11. Acknowledgments

Thanks to Flemming Andreasen, Shashank Jain, Michael Richardson,
Piyush Joshi, Eliot Lear, Harsha Joshi, Qin Wu, Mohamed Boucadair,
Ben Schwartz, Eric Rescorla, Panwei William, Nick Lamb and Nick
Harper for the discussion and comments.

12. References

12.1. 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>.

   [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.

12.2. Informative References

   [clear-as-mud]
              "Clear as MUD: Generating, Validating and Applying IoT
              Behaviorial Profiles", October 2019,
              <https://arxiv.org/pdf/1804.04358.pdf>.

   [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-tls-esni]
              Rescorla, E., Oku, K., Sullivan, N., and C. Wood, "TLS
              Encrypted Client Hello", draft-ietf-tls-esni-07 (work in
              progress), June 2020.

   [I-D.ietf-uta-tls13-iot-profile]
              Tschofenig, H. and T. Fossati, "TLS/DTLS 1.3 Profiles for
              the Internet of Things", draft-ietf-uta-tls13-iot-
              profile-00 (work in progress), June 2020.

   [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/>.

[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>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for
          the Network Configuration Protocol (NETCONF)", RFC 6020,
          DOI 10.17487/RFC6020, October 2010,
          <https://www.rfc-editor.org/info/rfc6020>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS)
          Extensions: Extension Definitions", RFC 6066,
          DOI 10.17487/RFC6066, January 2011,
          <https://www.rfc-editor.org/info/rfc6066>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan,
          "Transport Layer Security (TLS) Application-Layer Protocol
          Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301,
          July 2014, <https://www.rfc-editor.org/info/rfc7301>.

[RFC7366] Gutmann, P., "Encrypt-then-MAC for Transport Layer
          Security (TLS) and Datagram Transport Layer Security
          (DTLS)", RFC 7366, DOI 10.17487/RFC7366, September 2014,
          <https://www.rfc-editor.org/info/rfc7366>.

[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>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre,
          "Recommendations for Secure Use of Transport Layer
          Security (TLS) and Datagram Transport Layer Security
          (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May
          2015, <https://www.rfc-editor.org/info/rfc7525>.

[RFC7925] Tschofenig, H., Ed. and T. Fossati, "Transport Layer
          Security (TLS) / Datagram Transport Layer Security (DTLS)
          Profiles for the Internet of Things", RFC 7925,
          DOI 10.17487/RFC7925, July 2016,
          <https://www.rfc-editor.org/info/rfc7925>.

   [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>.

   [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for
              Writing an IANA Considerations Section in RFCs", BCP 26,
              RFC 8126, DOI 10.17487/RFC8126, June 2017,
              <https://www.rfc-editor.org/info/rfc8126>.

   [RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)",
              RFC 8152, DOI 10.17487/RFC8152, July 2017,
              <https://www.rfc-editor.org/info/rfc8152>.

   [RFC8472] Popov, A., Ed., Nystroem, M., and D. Balfanz, "Transport
              Layer Security (TLS) Extension for Token Binding Protocol
              Negotiation", RFC 8472, DOI 10.17487/RFC8472, October
              2018, <https://www.rfc-editor.org/info/rfc8472>.

   [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>.

   [RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz,
              "Object Security for Constrained RESTful Environments
              (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019,
              <https://www.rfc-editor.org/info/rfc8613>.

   [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

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