Stream:	Internet Engineering Task Force (IETF)		
RFC:	9761		
Category:	Standards Track		
Published:	March 2025		
ISSN:	2070-1721		
Authors:	T. Reddy.K	D. Wing	B. Anderson
	Nokia	*Citrix*	*Cisco*

RFC 9761
Manufacturer Usage Description (MUD) for TLS and DTLS Profiles for Internet of Things (IoT) Devices

Abstract

This memo extends the Manufacturer Usage Description (MUD) specification to allow manufacturers to define TLS and DTLS profile parameters. This allows a network security service to identify unexpected (D)TLS usage, which can indicate the presence of unauthorized software, malware, or security policy-violating traffic on an endpoint.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9761.

Table of Contents

1. Introduction

Encryption is necessary to enhance the privacy of end users using Internet of Things (IoT) devices. TLS [RFC8446] and DTLS [RFC9147] are the dominant protocols (counting all (D)TLS versions) that provide encryption for IoT device traffic. Unfortunately, in conjunction with IoT applications' rise of encryption, malware authors are also using encryption that thwarts network-based analysis, such as deep packet inspection (DPI). Thus, other mechanisms are needed to help detect malware running on an IoT device.

Malware often reuses certain libraries, and there are notable differences in how malware uses encryption compared to software that is not malware. Several common patterns in the use of (D)TLS by malware include:

- Use of older and weaker cryptographic parameters.
- 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").
- Higher use of self-signed certificates compared with typical legitimate software using certificates from a certificate authority (CA) trusted by the device.
- Discrepancies in the SNI TLS extension and the DNS names in the SubjectAltName (SAN) X.509 extension in the server certificate message.
- 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.
- Lower diversity in extensions advertised by TLS clients compared to legitimate clients.
- Using privacy enhancing technologies like Tor, Psiphon, Ultrasurf (see [MALWARE-TLS]), and evasion techniques such as ClientHello randomization.
- Using an alternative DNS server (via encrypted transport) to avoid detection by malware DNS filtering services [MALWARE-DOH]. Specifically, malware may not use the Do53 or encrypted DNS server provided by the local network (DHCP, Discovery of Network-designated Resolvers (DNR) [RFC9462], or Discovery of Designated Resolvers (DDR) [RFC9462]).

If (D)TLS profile parameters are defined, the following functions that have a positive impact on the local network security are possible:

- Permit intended DTLS or TLS use, and block malicious DTLS or TLS use. This is superior to the layers 3 and 4 Access Control Lists (ACLs) of Manufacturer Usage Description Specification (MUD) [RFC8520], which are not suitable for broad communication patterns. The goal of this document is to enhance and complement the existing MUD specifications rather than undermine them.
- 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 [CRYPTO-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 (D)TLS 1.2 [RFC5246] [RFC6347], the ClientHello, ServerHello, and Certificate messages are all sent in cleartext. This check is not possible with (D)TLS 1.3, which encrypts the Certificate message and therefore hides the server identity from any intermediary. In (D)TLS 1.3, the server certificate validation functions should be executed within an on-path (D)TLS proxy if such a proxy exists.
- 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 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 (see [CLEAR-AS-MUD]). In such a case, observable (D)TLS profile parameters can be used to permit intended use and block malicious behavior from the IoT device.

The YANG module specified in Section 5.2 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 block malicious or insecure flows. Additionally, it supports network communications adherence to security policies by ensuring that TLS certificates are valid and deprecated cipher suites are avoided. The proposed technique is also suitable in deployments where decryption techniques are not ideal due to privacy concerns, non-cooperating endpoints, 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: Used for statements that apply to both Transport Layer Security [RFC8446] and Datagram Transport Layer Security [RFC6347]. Specific terms "TLS" and "DTLS" are used for any statement that applies to either protocol alone.

DoH/DoT: Refers to DNS-over-HTTPS and/or DNS-over-TLS [RFC7858].

Middlebox: A middlebox that interacts with TLS traffic can either act as a TLS proxy, intercepting and decrypting the traffic for inspection, or inspect the traffic between TLS peers without terminating the TLS session.

Endpoint Security Agent: An Endpoint Security Agent is a software installed on endpoint devices that protects them from security threats. It provides features such as malware protection, firewall, and intrusion prevention to ensure the device's security and integrity.

Network Security Service: A Network Security Service refers to a set of mechanisms designed to protect network communications and resources from attacks.

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. Endpoint security agents have deep visibility on the devices where they are installed, whereas the network has broader visibility. Installing endpoint security agents may not be a viable option on IoT devices, and network security service 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. In addition, it enforces organizational security policies, ensuring that devices comply. By monitoring (D)TLS parameters, network administrators can identify and mitigate the use of outdated TLS versions, cryptographic algorithms, and non-compliant certificates. 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 (EOL) and the IoT manufacturer will not issue a firmware or software update to the IoT device 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 manufacturer no longer supports the device. The EOL of a device, where the IoT manufacturer no longer supports it, does not necessarily mean the device is defective. Instead, it signifies that the device is no longer receiving updates, support, or security patches, which necessitates replacement and upgrading to next-generation devices to ensure continued functionality, security, and compatibility with modern networks. The network security service will have to rely on other techniques discussed in Section 9 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 [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. When using these extensions, handshake messages will be encrypted and network security services (such as a firewall) are incapable of deciphering 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, extensions in ClientHello, and the 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 a middlebox acting as a (D)TLS proxy is restricted to the Enterprise network owning and managing the IoT devices. The middlebox would have to follow the behavior detailed in Section 9.3 of [RFC8446] to act as a compliant (D)TLS 1.3 proxy.

To further increase privacy, the Encrypted Client Hello (ECH) extension [TLS-ESNI] prevents passive observation of the TLS Server Name Indication extension and other potentially sensitive fields, such as the Application-Layer Protocol Negotiation (ALPN) [RFC7301]. To effectively provide that privacy protection, the ECH extension needs to be used in conjunction with DNS encryption (e.g., DoH). A middlebox (e.g., firewall) passively inspecting the ECH extension cannot observe the encrypted SNI nor observe the encrypted DNS traffic. The middlebox acting as a (D)TLS 1.3 proxy that does not support the ECH extension will act as if it is connecting to the public name and follows the behavior discussed in Section 6.1.6 of [TLS-ESNI] to securely signal the client to disable ECH.

4.2. Encrypted DNS

A common usage pattern for certain types 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], these devices tend to require access to very few sites. Thus, all other access should be considered suspect. This technique complements MUD policy enforcement at the TLS level by ensuring that DNS queries are monitored and filtered, thereby enhancing overall security. If an IoT device is pre-configured to use a DNS resolver not signaled by the network, the MUD policy enforcement point is moved to that resolver, 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 detect any issues. Therefore, the use of a DNS resolver that is not signaled by the network is generally incompatible with MUD. A network-designated DoH/DoT server is necessary to allow MUD policy enforcement on the local network, for example, using the techniques specified in DNR [RFC9463] and DDR [RFC9462].

5. (D)TLS Profile of an IoT device

This document specifies a YANG module that represents the (D)TLS profile. This YANG module provides a means to characterize the (D)TLS traffic profile of a device. Network security services can use these profiles to permit conformant traffic or to deny traffic from devices that deviates from it. This module uses the cryptographic types defined in [RFC9640]. See [RFC7925] for (D)TLS 1.2 and [IoT-PROFILE] for DTLS 1.3 recommendations related to IoT devices, and [RFC9325] 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:

- Profile name
- (D)TLS versions supported by the IoT device.
- List of supported cipher suites (Section 11 of [RFC8446]). For (D)TLS 1.2, [RFC7925] recommends Authenticated Encryption with Associated Data (AEAD) ciphers for IoT devices.
- List of supported extension types.

- 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, blocks 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.
- List of pre-shared key exchange modes.
- List of named groups (DHE or ECDHE) supported by the client.
- List of signature algorithms the client can validate in X.509 server certificates.
- List of signature algorithms the client is willing to accept for the 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.
- List of supported application protocols (e.g., h3, h2, http/1.1 etc.).
- List of certificate compression algorithms (defined in [RFC8879]).
- 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] defines a mechanism for TLS peers to send 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 any other TLS parameters defined in future RFCs.

The (D)TLS profile does not include parameters like compression methods for data compression. [RFC9325] 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]).

5.1. Tree Structure of the (D)TLS Profile Extension to the ACL YANG Module

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". The meaning of the symbols in the YANG tree diagram are defined in [RFC8340] and it has the following tree structure:

```
module: ietf-acl-tls
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches:
    +--rw client-profiles {match-on-tls-dtls}?
       +--rw tls-dtls-profile* [name]
          +--rw name                          string
          +--rw supported-tls-version*        ianatp:tls-version
          +--rw supported-dtls-version*       ianatp:dtls-version
          +--rw cipher-suite*                 ianatp:cipher-algorithm
          +--rw extension-type*
          |       ianatp:extension-type
          +--rw accept-list-ta-cert*
          |       ct:trust-anchor-cert-cms
          +--rw psk-key-exchange-mode*
          |       ianatp:psk-key-exchange-mode
          |       {tls13 or dtls13}?
          +--rw supported-groups*
          |       ianatp:supported-group
          +--rw signature-algorithm-cert*
          |       ianatp:signature-algorithm
          |       {tls13 or dtls13}?
          +--rw signature-algorithm*
          |       ianatp:signature-algorithm
          +--rw application-protocol*
          |       ianatp:application-protocol
          +--rw cert-compression-algorithm*
          |       ianatp:cert-compression-algorithm
          |       {tls13 or dtls13}?
          +--rw certificate-authorities*
                  certificate-authority
                  {tls13 or dtls13}?
```

5.2. The (D)TLS Profile Extension to the ACL YANG Module

```
<CODE BEGINS> file "ietf-acl-tls@2025-03-10.yang"

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

  import iana-tls-profile {
    prefix ianatp;
    reference
      "RFC 9761: Manufacturer Usage Description (MUD) for TLS and
                 DTLS Profiles for Internet of Things (IoT) Devices";
  }
  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC 9640: YANG Data Types and Groupings for Cryptography";
  }
  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: Tirumaleswar Reddy.K
             kondtir@gmail.com

    ";
  description
    "This YANG module defines a component that augments the
     IETF description of an access list to allow (D)TLS profiles
     as matching criteria.

     Copyright (c) 2025 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 Revised 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 9761; see
     the RFC itself for full legal notices.";

  revision 2025-03-10 {
    description
      "Initial revision.";
    reference
      "RFC 9761: Manufacturer Usage Description (MUD) for TLS and
                 DTLS Profiles for Internet of Things (IoT) Devices";
  }

  feature tls12 {
    description
      "TLS Protocol Version 1.2 is supported.";
    reference
      "RFC 5246: The Transport Layer Security (TLS) Protocol
                 Version 1.2";
  }

  feature tls13 {
    description
      "TLS Protocol Version 1.3 is supported.";
    reference
      "RFC 8446: The Transport Layer Security (TLS) Protocol
                 Version 1.3";
  }

  feature dtls12 {
    description
      "DTLS Protocol Version 1.2 is supported.";
```

```
    reference
      "RFC 6347: Datagram Transport Layer Security
                 Version 1.2";
  }

  feature dtls13 {
    description
      "DTLS Protocol Version 1.3 is supported.";
    reference
      "RFC 9147: Datagram Transport Layer Security 1.3";
  }

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

  typedef spki-pin-set {
    type binary;
    description
      "Subject Public Key Info pin set as discussed in
       Section 2.4 of RFC 7469.";
  }

  typedef certificate-authority {
    type string;
    description
      "Distinguished Name of Certificate authority as discussed
       in Section 4.2.4 of RFC 8446.";
  }

  augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" {
    if-feature "match-on-tls-dtls";
    description
      "(D)TLS specific matches.";
    container client-profiles {
      description
        "A grouping for (D)TLS profiles.";
      list tls-dtls-profile {
        key "name";
        description
          "A list of (D)TLS version profiles supported by
           the client.";
        leaf name {
          type string {
            length "1..64";
          }
          description
            "The name of (D)TLS profile; space and special
             characters are not allowed.";
        }
        leaf-list supported-tls-version {
          type ianatp:tls-version;
          description
            "TLS versions supported by the client.";
        }
        leaf-list supported-dtls-version {
```

```
          type ianatp:dtls-version;
          description
            "DTLS versions supported by the client.";
        }
        leaf-list cipher-suite {
          type ianatp:cipher-algorithm;
          description
            "A list of Cipher Suites supported by the client.";
        }
        leaf-list extension-type {
          type ianatp:extension-type;
          description
            "A list of Extension Types supported by the client.";
        }
        leaf-list accept-list-ta-cert {
          type ct:trust-anchor-cert-cms;
          description
            "A list of trust anchor certificates used by the
             client.";
        }
        leaf-list psk-key-exchange-mode {
          if-feature "tls13 or dtls13";
          type ianatp:psk-key-exchange-mode;
          description
            "pre-shared key exchange modes.";
        }
        leaf-list supported-group {
          type ianatp:supported-group;
          description
            "A list of named groups supported by the client.";
        }
        leaf-list signature-algorithm-cert {
          if-feature "tls13 or dtls13";
          type ianatp:signature-algorithm;
          description
            "A list signature algorithms the client can validate
             in X.509 certificates.";
        }
        leaf-list signature-algorithm {
          type ianatp:signature-algorithm;
          description
            "A list signature algorithms the client can validate
             in the CertificateVerify message.";
        }
        leaf-list application-protocol {
          type ianatp:application-protocol;
          description
            "A list application protocols supported by the client.";
        }
        leaf-list cert-compression-algorithm {
          if-feature "tls13 or dtls13";
          type ianatp:cert-compression-algorithm;
          description
            "A list certificate compression algorithms
             supported by the client.";
        }
        leaf-list certificate-authorities {
          if-feature "tls13 or dtls13";
```

```
          type 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

The TLS and DTLS IANA registries are available from <https://www.iana.org/assignments/tls-parameters> and <https://www.iana.org/assignments/tls-extensiontype-values>. Changes to TLS- and DTLS-related IANA registries are discussed in [RFC8447].

The values for all the parameters in the "iana-tls-profile" YANG module are defined in the TLS and DTLS IANA registries excluding the tls-version, dtls-version, spki-pin-set, and certificate-authority parameters. The values of spki-pin-set and certificate-authority parameters will be specific to the IoT device.

The TLS and DTLS IANA registries do not maintain (D)TLS version numbers. In (D)TLS 1.2 and below, the "legacy_version" field in the ClientHello message is used for version negotiation. However, in (D)TLS 1.3, the "supported_versions" extension is used by the client to indicate which versions of (D)TLS it supports. TLS 1.3 ClientHello messages are identified as having a "legacy_version" of 0x0303 and a "supported_versions" extension present with 0x0304 as the highest version. DTLS 1.3 ClientHello messages are identified as having a "legacy_version" of 0xfefd and a "supported_versions" extension present with 0x0304 as the highest version.

In order to ease updating the "iana-tls-profile" YANG module with future (D)TLS versions, new (D)TLS version registries are defined in Section 11.3 and Section 11.4. Whenever a new (D)TLS protocol version is defined, the registry will be updated using expert review; the "iana-tls-profile" YANG module will be automatically updated by IANA.

Implementers or users of this specification must refer to the IANA-maintained "iana-tls-profile" YANG module available at <https://www.iana.org/assignments/yang-parameters>.

The initial version of the "iana-tls-profile" YANG module is defined as follows:

```
<CODE BEGINS> file "iana-tls-profile@2025-03-10.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
     Email:  iana@iana.org>";
  description
    "This module contains the YANG definition for the (D)TLS profile.

     Copyright (c) 2025 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 Revised 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).

     All revisions of IETF and IANA published modules can be found
     at the YANG Parameters registry
     (https://www.iana.org/assignments/yang-parameters).

     The initial version of this YANG module is part of RFC 9761;
     see the RFC itself for full legal notices.

     The latest version of this YANG module is available at
     https://www.iana.org/assignments/yang-parameters.";

  revision 2025-03-10 {
    description
      "Initial revision.";
    reference
      "RFC 9761: Manufacturer Usage Description (MUD) for TLS and
                 DTLS Profiles for Internet of Things (IoT) Devices";
  }

  typedef extension-type {
    type uint16;
    description
      "Extension type in the TLS ExtensionType Values registry as
       defined in Section 7 of RFC 8447.";
  }

  typedef supported-group {
    type uint16;
    description
      "Supported Group in the TLS Supported Groups registry as
       defined in Section 9 of RFC 8447.";
  }

  typedef signature-algorithm {
    type uint16;
    description
```

```
      "Signature algorithm in the TLS SignatureScheme registry as
       defined in Section 11 of RFC 8446.";
  }

  typedef psk-key-exchange-mode {
    type uint8;
    description
      "Pre-shared key exchange mode in the TLS PskKeyExchangeMode
       registry as defined in Section 11 of RFC 8446.";
  }

  typedef application-protocol {
    type string;
    description
      "Application-Layer Protocol Negotiation (ALPN) Protocol ID
       registry as defined in Section 6 of RFC 7301.";
  }

  typedef cert-compression-algorithm {
    type uint16;
    description
      "Certificate compression algorithm in TLS Certificate
       Compression Algorithm IDs registry as defined in
       Section 7.3 of RFC 8879.";
  }

  typedef cipher-algorithm {
    type uint16;
    description
      "Cipher suite in TLS Cipher Suites registry
       as discussed in Section 11 of RFC 8446.";
  }

  typedef tls-version {
    type enumeration {
      enum tls12 {
        value 1;
        description
          "TLS Protocol Version 1.2.

           TLS 1.2 ClientHello contains
           0x0303 in 'legacy_version'.";
        reference
          "RFC 5246: The Transport Layer Security (TLS) Protocol
                     Version 1.2";
      }
      enum tls13 {
        value 2;
        description
          "TLS Protocol Version 1.3.

           TLS 1.3 ClientHello contains a
           supported_versions extension with 0x0304
           contained in its body and the ClientHello contains
           0x0303 in 'legacy_version'.";
        reference
          "RFC 8446: The Transport Layer Security (TLS) Protocol
                     Version 1.3";
```

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

  typedef dtls-version {
    type enumeration {
      enum dtls12 {
        value 1;
        description
          "DTLS Protocol Version 1.2.

           DTLS 1.2 ClientHello contains
           0xfefd in 'legacy_version'.";
        reference
          "RFC 6347: Datagram Transport Layer Security 1.2";
      }
      enum dtls13 {
        value 2;
        description
          "DTLS Protocol Version 1.3.

           DTLS 1.3 ClientHello contains a
           supported_versions extension with 0x0304
           contained in its body and the ClientHello contains
           0xfefd in 'legacy_version'.";
        reference
          "RFC 9147: 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 "ietf-mud-tls@2025-03-10.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;
    reference
      "RFC 8520: Manufacturer Usage Description Specification";
  }

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

     Author: Tirumaleswar Reddy.K
             kondtir@gmail.com

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

     Copyright (c) 2025 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 Revised 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 9761; see
     the RFC itself for full legal notices.";

  revision 2025-03-10 {
    description
      "Initial revision.";
    reference
      "RFC 9761: Manufacturer Usage Description (MUD) for TLS and
                 DTLS Profiles for Internet of Things (IoT)
                 Devices";
  }

  augment "/ietf-mud:mud" {
    description
      "This adds an extension for a manufacturer
       to indicate whether the (D)TLS profile is
       supported by a device.";
    leaf is-tls-dtls-profile-supported {
      type boolean;
      default "false";
```

```
      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 so as to avoid ossification:

- 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, such as blocking the (D)TLS session or raising 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.
- 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 behavior is functionally equivalent to the compliant TLS middlebox 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. This rule also ensures that the network security service will ignore the GREASE values advertised by TLS peers and interoperate with the implementations advertising GREASE values.
- 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 new parameters in the MUD (D)TLS profile are discovered that are not recognized by the firewall, it can be updated quickly. Most importantly, if the firewall is not readily updatable, its protection efficacy to identify emerging malware will decrease with time. For example, if the cipher suite TLS_AES_128_CCM_8_SHA256 specified in the MUD (D)TLS profile is not recognized by the firewall, an alert will be triggered. Similarly, if the (D)TLS version specified in the MUD file is not recognized by the firewall, an alert will be triggered.
- If the MUD (D)TLS profile includes any parameters that are susceptible to attacks (e.g., weaker cryptographic parameters), an alert **MUST** be triggered to the firewall vendor and the IoT device owner or administrator.

7. MUD File Example

The example below contains (D)TLS profile parameters for an IoT device used to reach servers listening on port 443 using TCP transport. JSON encoding of YANG-modeled data [RFC7951] is used to illustrate the example.

```
{
   "ietf-mud:mud": {
     "mud-version": 1,
      "mud-url": "https://example.com/IoTDevice",
      "last-update": "2024-08-05T03:56:40.105+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" : [
                        {
                           "name" : "profile1",
                           "supported-tls-versions" : ["tls13"],
                           "cipher-suite" : [4865, 4866],
                           "extension-types" : [10,11,13,16,24],
```

```
                              "supported-groups" : [29]
                           }
                        ]
                     },
                     "actions": {
                        "forwarding": "accept"
                     }
                  }
               }
            ]
         }
      }
   }
}
```

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

- 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.
- If the extension type "token_binding" (code point 24) [RFC8472] in the MUD (D)TLS profile 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.
- The two-byte values assigned by IANA for the cipher suites TLS_AES_128_GCM_SHA256 and TLS_AES_256_GCM_SHA384 are represented in decimal format.

8. Software-Based ACLs and ACLs Within a (D)TLS 1.3 Proxy

While ACL technology is traditionally associated with fixed-length bit matching in hardware implementations, such as those found in Ternary Content-Addressable Memory (TCAM), the use of ACLs in software, like with iptables, allows for more flexible matching criteria, including string matching. In the context of MUD (D)TLS profiles, the ability to match binary data and strings is a deliberate choice made to leverage the capabilities of software-based ACLs. This enables more dynamic and context-sensitive access control, which is essential for the intended application of MUD. The DNS extension added to ACL in the MUD specification [RFC8520] also requires software-based ACLs.

Regarding the use of MUD (D)TLS ACL in a (D)TLS 1.3 proxy, the goal is for the proxy to intercept the (D)TLS handshake before applying any ACL rules. This implies that MUD (D)TLS ACL matching would need to occur after decrypting the encrypted TLS handshake messages within the proxy. The proxy would inspect the handshake fields according to the MUD profile. ACL matching would be performed in two stages: first, by filtering clear-text TLS handshake message and second, by filtering after decrypting the TLS handshake messages.

9. Security Considerations

Security considerations in [RFC8520] need to be taken into consideration. The middlebox **MUST** adhere to the invariants discussed in Section 9.3 of [RFC8446] to act as a compliant proxy.

Although it is challenging for malware to mimic the TLS behavior of various IoT device types and models from different manufacturers, there is still a potential for malicious agents to use similar (D)TLS profile parameters as legitimate devices to evade detection. This difficulty arises because IoT devices often have distinct (D)TLS profiles between models and especially between manufacturers. While malware may find it hard to perfectly replicate the TLS behavior across such diverse devices, it is not impossible. Malicious agents might manage to use (D)TLS profile parameters that resemble those of legitimate devices. The feasibility of this depends on the nature of the profile parameters; for instance, parameters like certificate authorities are complex to mimic, while others, such as signature algorithms, may be easier to replicate. The difficulty in mimicking these profiles correlates with the specificity of the profiles and the variability in parameters used by different devices.

Network security services should also rely on contextual network data (e.g., domain name, IP address, etc.) to detect false negatives. For example, network security services filter malicious domain names and destination IP addresses with a bad reputation score. Furthermore, 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 the 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. Network security vendors leverage TLS parameters and contextual network data to identify malware (for example, see [EVE]).

The efficacy of identifying malware in (D)TLS 1.3 flows will be significantly reduced without leveraging contextual network data or acting as a proxy, as the encryption in (D)TLS 1.3 obscures many of the handshake details that could otherwise be used for detection.

9.1. Challenges in Mimicking (D)TLS 1.2 Handshakes for IoT Devices

(D)TLS 1.2 generally does not require a proxy, as all fields in the (D)TLS profile are transmitted in cleartext during the handshake. While it is technically possible for an attacker to observe and mimic the handshake, an attacker would need to use a domain name and destination IP address with a good reputation, obtain certificates from the same CAs used by the IoT devices, and evade traffic analysis techniques (e.g., [EVE], which detects malicious patterns in encrypted traffic without decryption). This task is particularly challenging because IoT devices often have distinct (D)TLS profiles that vary between models and manufacturers. Unlike the developers of legitimate applications, malware authors are under additional constraints, such as avoiding any noticeable differences on the infected devices and the potential for take-down requests impacting their server infrastructure (e.g., certificate revocation by a CA upon reporting).

9.2. Considerations for the "iana-tls-profile" Module

This section follows the template defined in Section 3.7.1 of [YANG-GUIDELINES].

The "iana-tls-profile" YANG module specified in this document defines a schema for data can possibly be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. These network management protocols are required to use a secure transport layer and mutual authentication, e.g., SSH [RFC6242] without the "none" authentication option, Transport Layer Security (TLS) [RFC8446] with mutual X.509 authentication, and HTTPS with HTTP authentication (Section 11 of [RFC9110]).

The Network Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

This YANG module defines YANG enumerations for a public IANA- maintained registry.

YANG enumerations are not security-sensitive, as they are statically defined in the publicly-accessible YANG module. IANA **MAY** deprecate and/or obsolete enumerations over time as needed to address security issues.

This module does not define any writable-nodes, RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

9.3. Considerations for the "ietf-acl-tls" Module

This section follows the template defined in Section 3.7.1 of [YANG-GUIDELINES].

The "ietf-acl-tls" YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. These network management protocols are required to use a secure transport layer and mutual authentication, e.g., SSH [RFC6242] without the "none" authentication option, Transport Layer Security (TLS) [RFC8446] with mutual X.509 authentication, and HTTPS with HTTP authentication (Section 11 of [RFC9110]).

The Network Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Please be aware that this YANG module uses groupings from other YANG modules that define nodes that may be considered sensitive or vulnerable in network environments. Please review the Security Considerations for dependent YANG modules for information as to which nodes may be considered sensitive or vulnerable in network environments.

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, the addition or removal of references to trusted anchors, (D)TLS versions, cipher suites, etc., can dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

Some of the readable data nodes defined in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. The YANG module will provide insights into (D)TLS profiles of the IoT devices, and the privacy considerations discussed in Section 10 need to be taken into account.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

9.4. Considerations for the "ietf-mud-tls" Module

This section follows the template defined in Section 3.7.1 of [YANG-GUIDELINES].

The "ietf-mud-tls" YANG module specified in this document defines a schema for data can possibly be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. These network management protocols are required to use a secure transport layer and mutual authentication, e.g., SSH [RFC6242] without the "none" authentication option, Transport Layer Security (TLS) [RFC8446] with mutual X.509 authentication, and HTTPS with HTTP authentication (Section 11 of [RFC9110]). Note that the YANG module is not intended to be accessed via NETCONF and RESTCONF. This has already been discussed in [RFC8520], and we are reiterating it here for the sake of completeness.

The Network Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Please be aware that this YANG module uses groupings from other YANG modules that define nodes that may be considered sensitive or vulnerable in network environments. Please review the Security Considerations for dependent YANG modules for information as to which nodes may be considered sensitive or vulnerable in network environments.

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, update that the device does not support (D)TLS profile can dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

10. Privacy Considerations

Privacy considerations discussed in Section 16 of [RFC8520] to not reveal the MUD URL to an attacker need to be taken into consideration. The MUD URL can be stored in a Trusted Execution Environment (TEE) for secure operation, enhanced data security, and prevention of exposure to unauthorized software. The MUD URL **MUST** be encrypted and shared only with the authorized components in the network (see Sections 1.5 and 1.8 of [RFC8520]) so that an on-path attacker cannot read the MUD URL and identify the IoT device. Otherwise, it provides the attacker with guidance on what vulnerabilities may be present on the IoT device. Note that while protecting

the MUD URL is valuable as described above, a compromised IoT device may be susceptible to malware performing vulnerability analysis (and version mapping) of the legitimate software located in memory or on non-volatile storage (e.g., disk, NVRAM). However, the malware on the IoT device is intended to be blocked from establishing a (D)TLS connection with the C&C server to reveal this information because the connection would be blocked by the network security service supporting this specification.

Full handshake inspection (Section 4.1) requires a (D)TLS proxy device that needs to decrypt traffic between the IoT device and its server(s). There is a tradeoff between privacy of the data carried inside (D)TLS (for example, personally identifiable information and protected health information especially) and efficacy of endpoint security. The use of (D)TLS proxies is **NOT RECOMMENDED** whenever possible. For example, an enterprise firewall administrator can configure the middlebox to bypass (D)TLS proxy functionality or payload inspection for connections destined to specific well-known services. Alternatively, an IoT device could be configured to reject all sessions that involve proxy servers to specific well-known services. 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 Constrained Application Protocol (CoAP) messages by wrapping them in the CBOR Object Signing and Encryption (COSE) format [RFC9052].

11. IANA Considerations

11.1. (D)TLS Profile YANG Modules

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

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

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

IANA has created an IANA-maintained YANG module called "iana-tls-profile" based on the contents of Section 5.3, which allows for new (D)TLS parameters and (D)TLS versions to be added to "client-profile".

IANA has registered the following YANG modules in the "YANG Module Names" registry [RFC6020] of the "YANG Parameters" registry group.

Name: iana-tls-profile
Namespace: urn:ietf:params:xml:ns:yang:iana-tls-profile
Maintained by IANA: Y
Prefix: ianatp
Reference: RFC 9761

Name: ietf-acl-tls
Namespace: urn:ietf:params:xml:ns:yang:ietf-acl-tls
Maintained by IANA: N
Prefix: ietf-acl-tls
Reference: RFC 9761

Name: ietf-mud-tls
Namespace: urn:ietf:params:xml:ns:yang:ietf-mud-tls
Maintained by IANA: N
Prefix: ietf-mud-tls
Reference: RFC 9761

11.2. Considerations for the iana-tls-profile Module

IANA has created 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:

- tls-version and dtls-version values must not be directly added to the iana-tls-profile YANG module. Instead, they must be added to the "ACL TLS Version Codes" and "ACL DTLS Version Codes" registries (respectively), provided the new (D)TLS version has been standardized by the IETF. It allows a new (D)TLS version to be added to the "iana-tls-profile" YANG module.
- (D)TLS parameters must not be directly added to the iana-tls-profile YANG module. They must instead be added to the "ACL (D)TLS Parameters" registry if the new (D)TLS parameters can be used by a middlebox to identify a MUD non-compliant (D)TLS behavior. It allows new (D)TLS parameters to be added to the "iana-tls-profile" YANG module.

When a "tls-version" or "dtls-version" value is added to the "ACL TLS Version Codes" or "ACL DTLS Version Codes" registry (respectively), 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": RFC YYYY: <Title of the RFC>, where YYYY is the RFC that added the "tls-version" or "dtls-version"

When a (D)TLS parameter is added to the "ACL (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 has added this note to "ACL TLS Version Codes", "ACL DTLS Version Codes", and "ACL (D)TLS Parameters" registries:

> When this registry is modified, the YANG module "iana-tls-profile" must be updated as defined in [RFC9761].

11.3. ACL TLS Version Registry

IANA has created a new registry titled "ACL 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]. Experts must ensure that the TLS protocol version in a new registration is one that has been standardized by the IETF. It is expected that the registry will be updated infrequently, primarily when a new TLS version is standardized by the IETF.

Value	Label	Description	Reference
1	tls12	TLS Version 1.2	[RFC5246]
2	tls13	TLS Version 1.3	[RFC8446]

Table 1

11.4. ACL DTLS Version Registry

IANA has created a new registry titled "ACL 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]. Experts must ensure that the DTLS protocol version in a new registration is one that has been standardized by the IETF. It is expected that the registry will be updated infrequently, primarily when a new DTLS version is standardized by the IETF.

Value	Label	Description	Reference
1	dtls12	DTLS Version 1.2	[RFC6347]
2	dtls13	DTLS Version 1.3	[RFC9147]

Table 2

11.5. ACL (D)TLS Parameters Registry

IANA has created a new registry titled "ACL (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/ and https://www.iana.org/assignments/tls-extensiontype-values/) excluding the tls-version and dtls-version parameters. Further assignments are to be made through Expert Review [RFC8126]. Experts must ensure that the (D)TLS parameter in a new registration is one that has been standardized by the IETF. The registry is expected to be updated periodically, primarily when a new (D)TLS parameter is standardized by the IETF. The registry has been populated with the following initial parameters:

Parameter Name	YANG Type	JSON Type	Description
extension-type	uint16	Number	Extension type
supported-group	uint16	Number	Supported group
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
cipher-algorithm	uint16	Number	Cipher Suite
tls-version	enumeration	String	TLS version
dtls-version	enumeration	String	DTLS version

Table 3

11.6. MUD Extensions Registry

IANA has created a new MUD Extension Name "ietf-mud-tls" in the "MUD Extensions" IANA registry <https://www.iana.org/assignments/mud>.

12. References

12.1. Normative References

[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>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[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>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[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>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[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>.

[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>.

[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>.

[RFC8879] Ghedini, A. and V. Vasiliev, "TLS Certificate Compression", RFC 8879, DOI 10.17487/RFC8879, December 2020, <https://www.rfc-editor.org/info/rfc8879>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/info/rfc9147>.

[RFC9640] Watsen, K., "YANG Data Types and Groupings for Cryptography", RFC 9640, DOI 10.17487/RFC9640, October 2024, <https://www.rfc-editor.org/info/rfc9640>.

[X690] ITU-T, "Information technology - ASN.1 encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, July 2002, <https://www.itu.int/rec/T-REC-X.690-200207-S/en>.

12.2. Informative References

[CLEAR-AS-MUD] Hamza, A., Ranathunga, D., Gharakheili, H. H., Roughan, M., and V. Sivaraman, "Clear as MUD: Generating, Validating and Applying IoT Behaviorial Profiles (Technical Report)", arXiv:1804.04358, DOI 10.48550/arXiv.1804.04358, April 2018, <https://arxiv.org/pdf/1804.04358.pdf>.

[CRYPTO-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>.

[EVE] Cisco, "Encrypted Visibility Engine", Cisco Secure Firewall documentation, <https://secure.cisco.com/secure-firewall/docs/encrypted-visibility-engine>.

[IoT-PROFILE] Tschofenig, H., Fossati, T., and M. Richardson, "TLS/DTLS 1.3 Profiles for the Internet of Things", Work in Progress, Internet-Draft, draft-ietf-uta-tls13-iot-profile-13, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-uta-tls13-iot-profile-13>.

[MALWARE-DOH] Cimpanu, C., "First-ever malware strain spotted abusing new DoH (DNS over HTTPS) protocol", ZDNET, 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", IMC '19: Proceedings of the Internet Measurement Conference, pp. 379-392, DOI 10.1145/3355369.3355601, 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>.

[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>.

[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>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8447] Salowey, J. and S. Turner, "IANA Registry Updates for TLS and DTLS", RFC 8447, DOI 10.17487/RFC8447, August 2018, <https://www.rfc-editor.org/info/rfc8447>.

[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>.

[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>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/info/rfc9052>.

[RFC9325] Sheffer, Y., Saint-Andre, P., and T. Fossati, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 9325, DOI 10.17487/RFC9325, November 2022, <https://www.rfc-editor.org/info/rfc9325>.

[RFC9462] Pauly, T., Kinnear, E., Wood, C. A., McManus, P., and T. Jensen, "Discovery of Designated Resolvers", RFC 9462, DOI 10.17487/RFC9462, November 2023, <https://www.rfc-editor.org/info/rfc9462>.

[RFC9463] Boucadair, M., Ed., Reddy.K, T., Ed., Wing, D., Cook, N., and T. Jensen, "DHCP and Router Advertisement Options for the Discovery of Network-designated Resolvers (DNR)", RFC 9463, DOI 10.17487/RFC9463, November 2023, <https://www.rfc-editor.org/info/rfc9463>.

[SLOWLORIS] ha.ckers.org security lab, "Slowloris HTTP DoS", Wayback Machine archive, March 2015, <https://web.archive.org/web/20150315054838/http://ha.ckers.org/slowloris/>.

[TLS-ESNI] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-23, 19 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-23>.

[X501] "Information Technology - Open Systems Interconnection - The Directory: Models", ITU-T Recommendation X.501, November 1993, <https://www.itu.int/rec/T-REC-X.501-199311-S/en>.

[YANG-GUIDELINES] Bierman, A., Boucadair, M., and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc8407bis-22, 14 January 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc8407bis-22>.

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, Tom Petch, Paul Wouters, Thomas Fossati, and Nick Harper for the discussion and comments.

Thanks to Xufeng Liu for YANGDOCTOR review. Thanks to Linda Dunbar for SECDIR review. Thanks to Qin Wu for OPSDIR review. Thanks to R. Gieben for DNSDIR review.

Thanks to Roman Danyliw, Orie Steele, Éric Vyncke, Mahesh Jethanandani, Murray Kucherawy, Zaheduzzaman Sarker, and Deb Cooley for the IESG review.

Authors' Addresses

Tirumaleswar Reddy.K
Nokia
India
Email: kondtir@gmail.com

Dan Wing
Citrix Systems, Inc.
4988 Great America Pkwy
Santa Clara, CA 95054
United States of America
Email: danwing@gmail.com

Blake Anderson
Cisco Systems, Inc.
170 West Tasman Dr
San Jose, CA 95134
United States of America
Email: blake.anderson@cisco.com