Stream: Internet Engineering Task Force (IETF)
RFC: 9472
Category: Standards Track
Published: October 2023
ISSN: 2070-1721
Authors: E. Lear, *Cisco Systems*; S. Rose, *NIST*

RFC 9472
A YANG Data Model for Reporting Software Bills of Materials (SBOMs) and Vulnerability Information

Abstract

To improve cybersecurity posture, automation is necessary to locate the software a device is using, whether that software has known vulnerabilities, and what, if any, recommendations suppliers may have. This memo extends the Manufacturer User Description (MUD) YANG schema to provide the locations of software bills of materials (SBOMs) and vulnerability information by introducing a transparency schema.

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/rfc9472.

Copyright Notice

Table of Contents

1. Introduction

A number of activities have taken place to improve the visibility of what software is running on a system and what vulnerabilities that software may have [EO2021].

Put simply, this memo seeks to answer two classes of questions for tens of thousands of devices and a large variety of device types. Those questions are as follows:

- Is this system susceptible to a particular vulnerability?
- Which devices in a particular environment contain vulnerabilities that require some action?

This memo doesn't specify the format of this information but rather only how to locate and retrieve these objects. That is, the model is intended to facilitate discovery and on its own provides no access to the underlying data.

Software bills of materials (SBOMs) are descriptions of what software, including versioning and dependencies, a device contains. There are different SBOM formats such as Software Package Data Exchange [SPDX] or CycloneDX [CycloneDX15].

System vulnerabilities may be similarly described using several data formats, including the aforementioned CycloneDX, the Common Vulnerability Reporting Framework [CVRF], and the Common Security Advisory Format [CSAF]. This information is typically used to report the state of any known vulnerabilities on a system to administrators.

SBOM and vulnerability information can be used in concert with other sources of vulnerability information. A network management tool could discover that a system uses a particular set of software components, searches a national vulnerability database to determine known vulnerabilities, and applies information provided by the manufacturer through this mechanism to produce a vulnerability report. That report may be used to indicate what, if any, versions of software correct that vulnerability or whether the system exercises the vulnerable code at all.

Both classes of information elements are optional under the model specified in this memo. One can provide only an SBOM, only vulnerability information, or both an SBOM and vulnerability information.

Note that SBOM formats may also carry other information, the most common being any licensing terms. Because this specification is neutral regarding content, it is left for format developers such as the Linux Foundation, OASIS, and ISO to decide what attributes they will support.

This memo does not specify how vulnerability information may be retrieved directly from the endpoint. That is because vulnerability information changes occur to software updates at different rates. However, some SBOM formats may also contain vulnerability information.

SBOMs and vulnerability information are advertised and retrieved through the use of a YANG augmentation of the Manufacturer User Description (MUD) model [RFC8520]. Note that the schema creates a grouping that can also be used independently of MUD. Moreover, other MUD features, such as access controls, needn't be present.

The mechanisms specified in this document are meant to address two use cases:

- A network-layer management system retrieving information from an Internet of Things (IoT) device as part of its ongoing life cycle. Such devices may or may not have query interfaces available.
- An application-layer management system retrieving vulnerability or SBOM information in order to evaluate the posture of an application server of some form. These application servers may themselves be containers or hypervisors. Discovery of the topology of a server is beyond the scope of this memo.

To satisfy these two key use cases, objects may be found in one of three methods:

1. on the devices themselves
2. on a website (e.g., via a URI)
3. through some form of out-of-band contact with the supplier

Using the first method, devices will have interfaces that permit direct retrieval. Examples of these interfaces might be an HTTP [RFC9110] or Constrained Application Protocol (CoAP) [RFC7252] endpoint for retrieval. There may also be private interfaces as well.

Using the second method, when a device does not have an appropriate retrieval interface, but one is directly available from the manufacturer, a URI to that information is discovered through interfaces such as MUD via DHCP or bootstrapping and ownership transfer mechanisms.

Using the third method, a supplier may wish to make an SBOM or vulnerability information available under certain circumstances and may need to individually evaluate requests. The result of that evaluation might be the SBOM, the vulnerability itself, a restricted URL, or no access.

To enable application-layer discovery, this memo defines a well-known URI [RFC8615]. Management or orchestration tools can query this well-known URI to retrieve a system's SBOM information. Further queries may be necessary based on the content and structure of the response.

1.1. Requirements Language

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.

1.2. How This Information Is Retrieved

Section 4 describes a data model to extend the MUD file format to carry SBOM and vulnerability information. Section 1.5 of [RFC8520] describes mechanisms by which devices can emit a URL to point to this file. Additionally, devices can share this URL either through documentation or within a QR code on a box. Section 2 describes a well-known URL from which an SBOM could be served from the local device.

Note that vulnerability and SBOM information are likely to change at different rates. MUD's cache-validity node provides a way for manufacturers to control how often tooling should check for those changes through the cache-validity node.

1.3. Formats

There are multiple ways to express both SBOMs and vulnerability information. When these are retrieved either from the device or from a remote web server, tools will need to observe the Content-Type header to determine precisely which format is being transmitted. Because IoT devices in particular have limited capabilities, use of a specific Accept: header in HTTP or the Accept Option in CoAP is **NOT RECOMMENDED**. Instead, backend tooling is encouraged to support all known formats and **SHOULD** silently discard SBOM information sent with a media type that is not understood.

If multiple SBOMs are intended to be supported in the same file, the media type should properly reflect that. For example, one might make use of application/{someformat}+json-seq. It is left to those supporting those formats to make the appropriate registrations in this case.

Some formats may support both vulnerability and software inventory information. When both vulnerability and software inventory information is available from the same URL, both sbom-url and members of the vuln-url list **MUST** indicate that. Network management systems **MUST** take note of when the SBOM and vulnerability information are accessible via the same resource and not retrieve the resource a second time.

2. The Well-Known Transparency Endpoint Set

A well-known endpoint is defined:

"/.well-known/sbom" retrieves an SBOM

As discussed previously, the precise format of a response is based on the Content-Type provided.

3. The mud-transparency Extension

We now formally define the mud-transparency extension; this is done in two parts.

First, the extension name "transparency" is listed in the "extensions" array of the MUD file. Note that this schema extension is intended to be used wherever it might be appropriate (e.g., not just with MUD).

Second, the "mud" container is augmented with a list of SBOM sources.

This is done as follows:

```
module: ietf-mud-transparency

  augment /mud:mud:
    +--rw transparency
       +--rw (sbom-retrieval-method)?
       |  +--:(cloud)
       |  |  +--rw sboms* [version-info]
       |  |     +--rw version-info    string
       |  |     +--rw sbom-url?       inet:uri
       |  +--:(local-well-known)
       |  |  +--rw sbom-local-well-known?   identityref
       |  +--:(sbom-contact-info)
       |     +--rw sbom-contact-uri?        inet:uri
       +--rw sbom-archive-list?             inet:uri
       +--rw (vuln-retrieval-method)?
          +--:(cloud)
          |  +--rw vuln-url*                inet:uri
          +--:(vuln-contact-info)
             +--rw vuln-contact-uri?        inet:uri
```

See [RFC8340] for a description of YANG trees.

4. The mud-sbom Augmentation to the MUD YANG Data Model

This YANG module references [RFC6991], [RFC7231], [RFC7252], [RFC8520], and [RFC9110].

```
<CODE BEGINS> file "ietf-mud-transparency@2023-10-10.yang"

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

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-mud {
    prefix mud;
    reference
      "RFC 8520: Manufacturer Usage Description Specification";
  }

  organization
```

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

     Editor: Eliot Lear <lear@cisco.com>
     Editor: Scott Rose <scott.rose@nist.gov>";
  description
    "This YANG module augments the ietf-mud model to provide for
     reporting of SBOMs and vulnerability information.

     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 (RFC 2119) (RFC 8174) when, and only when,
     they appear in all capitals, as shown here.

     Copyright (c) 2023 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
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC 9472
     (https://www.rfc-editor.org/info/rfc9472);
     see the RFC itself for full legal notices.";

  revision 2023-10-10 {
    description
      "Initial proposed standard.";
    reference
      "RFC 9472: A YANG Data Model for Reporting Software Bills
       of Materials (SBOMs) and Vulnerability Information";
  }

  identity local-type {
    description
      "Base identity for local well-known choices.";
  }

  identity http {
    base mudtx:local-type;
    description
      "Use http (RFC 7231) (insecure) to retrieve SBOM information.
        This method is NOT RECOMMENDED but may be unavoidable for
        certain classes of deployment where TLS has not or
        cannot be implemented.";
      reference
        "RFC 7231: Hypertext Transfer Protocol (HTTP/1.1):
         Semantics and Content";
  }

  identity https {
    base mudtx:local-type;
```

```
      description
        "Use https (secure) to retrieve SBOM information.  See
         RFC 9110.";
        reference
          "RFC 9110: HTTP Semantics";
    }

    identity coap {
      base mudtx:local-type;
      description
        "Use COAP (RFC 7252) (insecure) to retrieve SBOM.  This method
         is NOT RECOMMENDED, although it may be unavoidable
         for certain classes of implementations/deployments.";
        reference
          "RFC 7252: The Constrained Application Protocol (CoAP)";
    }

    identity coaps {
      base mudtx:local-type;
      description
        "Use COAPS (secure) to retrieve SBOM (RFC 7252).";
    }

    grouping transparency-extension {
      description
        "This grouping provides a means to describe the location of
         software bills of material and vulnerability descriptions.";
      container transparency {
        description
          "Container of methods to get SBOMs and vulnerability
           information.";
        choice sbom-retrieval-method {
          description
            "How to find SBOM information.";
          case cloud {
            list sboms {
              key "version-info";
              description
                "A list of SBOMs tied to different software
                 or hardware versions.";
              leaf version-info {
                type string;
                description
                  "The version to which this SBOM refers.";
              }
              leaf sbom-url {
                type inet:uri {
                  pattern '((coaps?)|(https?)):.*';
                }
                description
                  "A statically located URL.";
              }
            }
          }
          case local-well-known {
            leaf sbom-local-well-known {
              type identityref {
                base mudtx:local-type;
```

```
              }
              description
                "Which communication protocol to choose.";
            }
          }
          case sbom-contact-info {
            leaf sbom-contact-uri {
              type inet:uri {
                pattern '((mailto)|(https?)|(tel)):.*';
              }
              description
                "This MUST be a tel, an http, an https, or
                 a mailto uri schema that customers can use to
                 contact someone for SBOM information.";
            }
          }
        }
        leaf sbom-archive-list {
          type inet:uri;
          description
            "This URI returns a JSON list of URLs that consist of
             SBOMs that were previously published for this
             device.  Publication dates can be found inside
             the SBOMs.";
        }
        choice vuln-retrieval-method {
          description
            "How to find vulnerability information.";
          case cloud {
            leaf-list vuln-url {
              type inet:uri;
              description
                "List of statically located URLs that reference
                 vulnerability information.";
            }
          }
          case vuln-contact-info {
            leaf vuln-contact-uri {
              type inet:uri {
                pattern '((mailto)|(https?)|(tel)):.*';
              }
              description
                "This MUST be a tel, an http, an https, or
                 a mailto uri schema that customers can use to
                 contact someone for vulnerability information.";
            }
          }
        }
      }
    }

    augment "/mud:mud" {
      description
        "Add extension for software transparency.";
      uses transparency-extension;
    }
  }
```

```
<CODE ENDS>
```

5. Examples

In this example MUD file that uses a cloud service, the modelX presents a location of the SBOM in a URL. Note that the Access Control Lists (ACLs) in a MUD file are NOT required, although they are a very good idea for IP-based devices.

5.1. Without ACLS

This first MUD file demonstrates how to get SBOM and vulnerability information without ACLs.


```
{
 "ietf-mud:mud": {
   "mud-version": 1,
   "extensions": [
     "transparency"
   ],
   "mudtx:transparency": {
     sboms: [ {
     "version-info": "1.2",
     "sbom-url": "https://iot.example.com/info/modelX/sbom.json"
     } ],
     "vuln-url" : [
       "https://iotd.example.com/info/modelX/csaf.json"
     ]
   },
   "mud-url": "https://iot.example.com/modelX.json",
   "mud-signature": "https://iot.example.com/modelX.p7s",
   "last-update": "2022-01-05T13:29:12+00:00",
   "cache-validity": 48,
   "is-supported": true,
   "systeminfo": "retrieving vuln and SBOM info via a cloud service",
   "mfg-name": "Example, Inc.",
   "documentation": "https://iot.example.com/doc/modelX",
   "model-name": "modelX"
 }
}
```


The second example demonstrates that just SBOM information is included from the cloud.

```
{
 "ietf-mud:mud": {
   "mud-version": 1,
   "extensions": [
     "transparency"
   ],
   "mudtx:transparency": {
     sboms: [ {
     "version-info": "1.2",
     "sbom-url": "https://iot.example.com/info/modelX/sbom.json"
     } ],
   },
   "mud-url": "https://iot.example.com/modelX.json",
   "mud-signature": "https://iot.example.com/modelX.p7s",
   "last-update": "2022-01-05T13:29:12+00:00",
   "cache-validity": 48,
   "is-supported": true,
   "systeminfo": "retrieving vuln and SBOM info via a cloud service",
   "mfg-name": "Example, Inc.",
   "documentation": "https://iot.example.com/doc/modelX",
   "model-name": "modelX"
 }
}
```


5.2. SBOM Located on the Device

In the next example, the SBOM is located on the device, and there is no vulnerability information provided.


```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "extensions": [
      "transparency"
    ],
    "mudtx:transparency": {
      "sbom-local-well-known": "https"
    },
    "mud-url": "https://iot.example.com/modelX.json",
    "mud-signature": "https://iot.example.com/modelX.p7s",
    "last-update": "2022-01-05T13:29:47+00:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "retrieving SBOM info from a local source",
    "mfg-name": "Example, Inc.",
    "documentation": "https://iot.example.com/doc/modelX",
    "model-name": "modelX"
  }
}
```


In this example, the SBOM is retrieved from the device, while vulnerability information is available from the cloud. This is likely a common case because vendors may learn of vulnerability information more frequently than they update software.

```
{
 "ietf-mud:mud": {
   "mud-version": 1,
   "extensions": [
     "transparency"
   ],
   "mudtx:transparency": {
     "sbom-local-well-known": "https",
     "vuln-url" : [
       "https://iotd.example.com/info/modelX/csaf.json"
     ]
   },
   "mud-url": "https://iot-device.example.com/modelX.json",
   "mud-signature": "https://iot-device.example.com/modelX.p7s",
   "last-update": "2022-01-05T13:25:14+00:00",
   "cache-validity": 48,
   "is-supported": true,
   "systeminfo": "mixed example: SBOM on device, vuln info in cloud",
   "mfg-name": "Example, Inc.",
   "documentation": "https://iot-device.example.com/doc/modelX",
   "model-name": "modelX"
 }
}
```


5.3. Further Contact Required

In this example, the network manager must take further steps to retrieve SBOM information. Vulnerability information is still available.


```
{
"ietf-mud:mud": {
"mud-version": 1,
"extensions": [
  "transparency"
],
"mudtx:transparency": {
  "contact-info": "https://iot-device.example.com/contact-info.html",
    "vuln-url" : [
      "https://iotd.example.com/info/modelX/csaf.json"
    ]
},
"mud-url": "https://iot-device.example.com/modelX.json",
"mud-signature": "https://iot-device.example.com/modelX.p7s",
"last-update": "2021-07-09T06:16:42+00:00",
"cache-validity": 48,
"is-supported": true,
"systeminfo": "retrieving vuln and SBOM info via a cloud service",
"mfg-name": "Example, Inc.",
"documentation": "https://iot-device.example.com/doc/modelX",
"model-name": "modelX"
}
}
```

5.4. With ACLS

Finally, here is a complete example where the device provides SBOM and vulnerability information as well as access control information.


```
{
 "ietf-mud:mud": {
   "mud-version": 1,
   "extensions": [
     "transparency"
   ],
   "mudtx:transparency": {
     "sbom-local-well-known": "https",
     "vuln-url" : [
       "https://iotd.example.com/info/modelX/csaf.json"
     ]
   },
   "mud-url": "https://iot.example.com/modelX.json",
   "mud-signature": "https://iot.example.com/modelX.p7s",
   "last-update": "2022-01-05T13:30:31+00:00",
   "cache-validity": 48,
   "is-supported": true,
   "systeminfo": "retrieving vuln and SBOM info via a cloud service",
   "mfg-name": "Example, Inc.",
   "documentation": "https://iot.example.com/doc/modelX",
   "model-name": "modelX",
   "from-device-policy": {
     "access-lists": {
       "access-list": [
         {
           "name": "mud-65443-v4fr"
         }
       ]
     }
   },
   "to-device-policy": {
     "access-lists": {
       "access-list": [
         {
           "name": "mud-65443-v4to"
         }
       ]
     }
   }
 },
 "ietf-access-control-list:acls": {
   "acl": [
     {
       "name": "mud-65443-v4to",
       "type": "ipv4-acl-type",
       "aces": {
         "ace": [
           {
             "name": "cl0-todev",
             "matches": {
```

```
                "ipv4": {
                  "ietf-acldns:src-dnsname": "iotserver.example.com"
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      },
      {
        "name": "mud-65443-v4fr",
        "type": "ipv4-acl-type",
        "aces": {
          "ace": [
            {
              "name": "cl0-frdev",
              "matches": {
                "ipv4": {
                  "ietf-acldns:dst-dnsname": "iotserver.example.com"
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      }
    ]
  }
}
```


At this point, the management system can attempt to retrieve the SBOM, determine which format is in use through the Content-Type header on the response to a GET request, independently repeat the process for vulnerability information, and apply ACLs as appropriate.

6. Security Considerations

This document describes a schema for discovering the location of information relating to software transparency and does not specify the access model for the information itself. In particular, the YANG module specified in this document is not necessarily intended to be accessed via regular network management protocols, such as NETCONF [RFC6241] or RESTCONF [RFC8040], and hence the regular security considerations for such usage are not considered here.

Below, we describe protections relating to both discovery and some advice on protecting the underlying SBOM and vulnerability information.

The model specifies both encrypted and unencrypted means to retrieve information. This is a matter of pragmatism. Unencrypted communications allow for manipulation of information being retrieved. Therefore, it is **RECOMMENDED** that implementations offer a means to configure endpoints so that they may make use of TLS or DTLS.

The ietf-mud-transparency module has no operational impact on the element itself and is used to discover state information that may be available on or off the element. In as much as the module itself is made writeable, this only indicates a change in how to retrieve read-only elements. There are no means, for instance, to upload an SBOM. Additional risks are discussed below and are applicable to all nodes within the transparency container.

If an attacker modifies the elements, they may misdirect automation to retrieve a different set of URLs than was intended by the designer. This in turn leads to two specific sets of risks:

- the information retrieved would be false
- the URLs themselves point to malware

To address either of these risks or any tampering of a URL:

- test any cloud-based URL against a reputation service
- provide the administrator an opportunity to approve further processing when the authority changes to one not known to be reputable

SBOMs provide an inventory of software. Knowledge of which specific software is loaded on a system can aid an attacker in identifying an appropriate exploit for a known vulnerability or guide the development of novel exploit against this system. However, if software is available to an attacker, the attacker may already be able to derive this very same software inventory. When this information resides on the endpoint itself, the endpoint **SHOULD NOT** provide unrestricted access to the well-known URL by default.

Other servers that offer the data **MAY** restrict access to SBOM information using appropriate authorization semantics within HTTP. One way to do this would be to issue a certificate to the client for this purpose after a registration process has taken place. Another approach would involve the use of OAuth in combination. In particular, if a system attempts to retrieve an SBOM via HTTP or CoAP and the client is not authorized, the server **MUST** produce an appropriate error with instructions on how to register a particular client.

Another risk is a skew in the SBOM listing and the actual software inventory of a device/container. For example, a manufacturer may update the SBOM on its server, but an individual device has not been upgraded yet. This may result in an incorrect policy being applied to a device. A unique mapping of a device's software version and its SBOM can minimize this risk.

To further mitigate attacks against a device, manufacturers **SHOULD** recommend network access controls.

Vulnerability information is generally made available to such databases as NIST's National Vulnerability Database [NISTNVD]. It is possible that vendors may wish to release information early to some customers. We do not discuss here whether that is a good idea, but if it is employed, then appropriate access controls and authorization **SHOULD** be applied to that information.

7. IANA Considerations

7.1. MUD Extension

IANA has added "transparency" to the "MUD Extensions" registry [RFC8520] as follows:

Value: transparency
Reference: RFC 9472

7.2. YANG Registration

IANA has registered the following YANG module in the "YANG Module Names" registry [RFC6020]:

Name: ietf-mud-transparency
Namespace: urn:ietf:params:xml:ns:yang:ietf-mud-transparency
Maintained by IANA: N
Prefix: mudtx
Reference: RFC 9472

The following URI has been registered in the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-mud-transparency
Registrant Contact: IESG
XML: None. Namespace URIs do not represent an XML specification.

7.3. Well-Known Prefix

IANA has added the following URI suffix to the "Well-Known URIs" registry in accordance with [RFC8615]:

URI Suffix: sbom
Change Controller: IETF
Reference: RFC 9472
Status: permanent
Related Information: See ISO/IEC 5962:2021 and SPDX.org

8. References

8.1. 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>.

[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>.

[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>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[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>.

[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>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

[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>.

8.2. Informative References

[CSAF] Rock, L., Ed., Hagen, S., Ed., and T. Schmidt, Ed., "Common Security Advisory Framework Version 2.0", OASIS Standard, November 2022, <https://docs.oasis-open.org/csaf/csaf/v2.0/csaf-v2.0.html>.

[CVRF] Hagen, S., Ed., "CSAF Common Vulnerability Reporting Framework (CVRF) Version 1.2", Committee Specification 01, September 2017, <https://docs.oasis-open.org/csaf/csaf-cvrf/v1.2/csaf-cvrf-v1.2.pdf>.

[CycloneDX15] CycloneDX, "CycloneDX v1.5 JSON Reference", Version 1.5.0, <https://cyclonedx.org/docs/1.5/json>.

[EO2021] Biden, J., "Executive Order on Improving the Nation's Cybersecurity", EO 14028, May 2021.

[NISTNVD] NIST, "National Vulnerability Database", <https://nvd.nist.gov>.

[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>.

[SPDX] The Linux Foundation, "The Software Package Data Exchange (SPDX) Specification", Version 2.3, 2022, <https://spdx.github.io/spdx-spec/v2.3/>.

Acknowledgments

Thanks to Russ Housley, Dick Brooks, Tom Petch, and Nicolas Comstedt, who provided review comments.

Authors' Addresses

Eliot Lear
Cisco Systems
Richtistrasse 7
CH-8304 Wallisellen
Switzerland
Phone: +41 44 878 9200
Email: lear@cisco.com

Scott Rose
NIST
100 Bureau Dr.
Gaithersburg, MD 20899
United States of America
Phone: +1 301-975-8439
Email: scott.rose@nist.gov